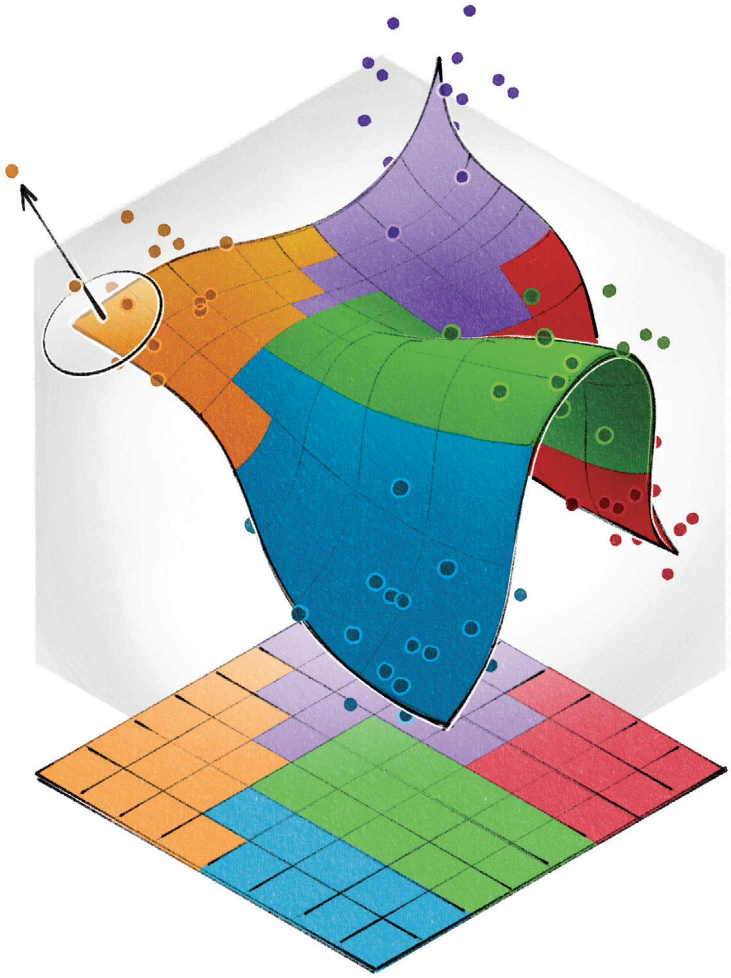
ENHANCED DECISION MAPS FOR EXPLORING CLASSIFICATION MODELS



YU WANG

Cover: A decision map visualized in both the data space and the 2D image space.

Enhanced Decision Maps for Exploring Classification Models

Yu Wang
PhD Thesis

The research for this dissertation was conducted at the Visualization and Graphics (VIG) group, part of the Department of Information and Computing Sciences (ICS), Faculty of Science, Utrecht University, the Netherlands.

DOI: https://doi.org/10.33540/3023

Enhanced Decision Maps for Exploring Classification Models

Verbeterde beslissingskaarten voor het verkennen van classificatiemodellen

(met een samenvatting in het Nederlands)

Proefschrift

ter verkrijging van de graad van doctor aan de
Universiteit Utrecht
op gezag van de
rector magnificus, prof. dr. ir. W. Hazeleger,
ingevolge het besluit van het college voor promoties
in het openbaar te verdedigen op

woensdag 16 juli 2025 des ochtends te 10.15 uur

door

Yu Wang

geboren op 9 Mei 1997
te Hebei, China

ABSTRACT

High-dimensional data is a key study object for both machine learning (ML) and information visualization. In the field of visualization, dimensionality reduction (DR) methods, also known as projections, are one of the most frequently used classes of techniques for visually exploring large and high-dimensional datasets. In ML, high-dimensional data is generated and processed by classifiers and regressors, which increasingly require visualization for explanation and exploration.

This thesis focuses on a recent visualization technique called *decision maps*. A decision map is a 2D image that visualizes the decision boundaries of a classifier in the data space, and can be used to explain and improve the behavior of ML classifiers. Constructing decision maps essentially involves a DR method and its inverse process (inverse projection). As such, ML techniques can help to create decision maps by providing improved (inverse) projections.

We begin with a case study applying decision maps to explain the classification of mineral deposit genesis. Our findings show that decision maps provide extra insights into the mineral classification model, aiding geologists in interpreting the model. However, we also identified gaps in current decision map techniques that present opportunities for improvement.

Following this case study, we conducted a comprehensive evaluation of three notable decision map techniques. Our evaluation shows that each technique has unique advantages and disadvantages. Our results can guide users in selecting the most suitable technique for specific tasks. A particularly salient finding of this evaluation was that all tested decision maps exhibit a surface-like behavior when applied to a 3D dataset.

We explored the aforementioned surface-like behavior of decision maps across more scenarios. By estimating the intrinsic dimensionality of the maps, we found that existing decision map methods cover only a small portion of the intrinsic dimensionality of high-dimensional data spaces. This finding highlights fundamental limitations in all current approaches to constructing decision maps.

To address these limitations, we propose a novel approach for computing inverse projections with the support of ML. Our method allows users to interactively control the location of the inversely projected points, thus also of visualizations like decision maps, within the high-dimensional space. In this way, users can explore larger parts of the data space, bypassing the practical limitations imposed by the aforementioned surface-like behavior of decision maps. We demonstrate the effectiveness of our approach through its application to a style transfer task.

Finally, we introduce an accelerated computation method for decision maps. Our method significantly reduces the computation time for both basic decision maps and enhanced variations thereof such as gradient maps. The acceleration facilitates the further deployment of such visualizations in interactive visual analytics workflows for classifier engineering.

SAMENVATTING

Hoog-dimensionale data is een belangrijk studieobject voor zowel machine learning (ML) als informatievisualisatie. In het vakgebied van visualisatie zijn dimensionaliteitsreductie (DR) methoden, ook wel projecties genoemd, een van de meest gebruikte klassen van technieken voor het visueel verkennen van grote en hoogdimensionale datasets. In ML wordt hoogdimensionale data gegenereerd en verwerkt door classifiers en regressoren, die steeds vaker visualisatie vereisen voor uitleg en verkenning.

Dit proefschrift richt zich op een recente visualisatietechniek genaamd beslissingskaarten. Een beslissingskaart is een 2D-afbeelding die de beslissingsgrenzen van een classifier in de gegevensruimte visualiseert en kan worden gebruikt om het gedrag van ML-classifiers te verklaren en te verbeteren. Het construeren van beslissingskaarten omvat in essentie een DR-methode en het inverse proces daarvan (inverse projectie). ML-technieken kunnen bijdragen aan het maken van beslissingskaarten door verbeterde (inverse) projecties te leveren.

We beginnen met een casestudy waarin beslissingskaarten worden toegepast om de classificatie van mineraalafzettingen te verklaren. Onze bevindingen tonen aan dat beslissingskaarten extra inzichten bieden in het mineralenclassificatiemodel, wat geologen helpt bij de interpretatie van het model. We identificeren ook tekortkomingen van de huidige technieken voor beslissingskaarten – die mogelijkheden voor verbetering bieden.

Na deze casestudy voeren wij een uitgebreide evaluatie uit van drie prominente technieken voor beslissingskaarten. Onze evaluatie toont aan dat elke techniek unieke voor- en nadelen heeft. Onze resultaten kunnen gebruikers helpen bij het kiezen van de meest geschikte techniek voor specifieke taken. Een opvallende bevinding van deze evaluatie was dat alle geteste beslissingskaarten een oppervlakachtig gedrag vertonen wanneer ze worden toegepast op een 3D-dataset.

We onderzoeken dit oppervlakachtige gedrag van beslissingskaarten verder in meerdere scenario's. Door de intrinsieke dimensionaliteit van de kaarten te schatten, ontdekten wij dat bestaande methoden slechts een klein deel van de intrinsieke dimensionaliteit van hoogdimensionale ruimtes dekken. Deze bevinding benadrukt fundamentele beperkingen in alle huidige benaderingen voor het construeren van beslissingskaarten.

Om deze beperkingen aan te pakken, stellen we een nieuwe benadering voor om inverse projecties te berekenen met ondersteuning van ML. Onze methode stelt gebruikers in staat om interactief de locatie van de invers geprojecteerde punten te controleren, en daarmee ook visualisaties zoals beslissingskaarten, binnen de hoog-dimensionale ruimte. Op deze manier kunnen gebruikers grotere delen van de dataruimte verkennen zonder de praktische beperkingen die worden opgelegd door het eerder genoemde oppervlakgedrag van beslissingskaarten. We demonstreren de effectiviteit van onze aanpak door de toepassing ervan op een stijltransfertaak.

Tot slot introduceren we een versnelde berekeningsmethode voor beslissingskaarten. Onze methode vermindert de rekentijd aanzienlijk voor zowel basisbeslissingskaarten als geavanceerde varianten zoals gradiëntkaarten. Deze versnelling vergemakkelijkt de verdere inzet van dergelijke visualisaties in interactieve visuele analyseworkflows voor classifier-engineering.

PUBLICATIONS

This thesis is a result of the following publications:

- Y. Wang, A. Machado, and A. C. Telea. Quantitative and qualitative comparison of decision-map techniques for explaining classification models. *Algorithms*, 2023.
- Y. Wang and A. C. Telea. Fundamental limitations of inverse projections and decision maps. In *Proc. International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (IVAPP)*, 2024. **(Best Student Paper Award)**
- A. C. Telea, A. Machado, and Y. Wang. Seeing is learning in high dimensions: The synergy between dimensionality reduction and machine learning. *SN Computer Science*, 2024.
- C. Grosu, Y. Wang, and A. C. Telea. Computing fast and accurate decision boundary maps. In *Proc. EuroVis Workshop on Visual Analytics (EuroVA)*, 2024.
- Y. Wang, K. Qiu, A. C. Telea, Z. Hou, T. Zhou, Y. Cai, Z. Ding, H. Yu, and J. Deng. Interpreting mineral deposit genesis classification with decision maps: A case study using pyrite trace elements. *American Mineralogist*, 2024.
- Y. Wang, C. Grosu, and A. C. Telea. Computing fast and accurate maps for explaining classification models. *Computers & Graphics*, 2025.
- Y. Wang and A. C. Telea. Investigating desirable properties of inverse projections and decision maps. *Communications in Computer and Information Science*, 2025. (In press)
- Y. Wang, F. Dennig, M. Behrisch, and A. C. Telea. LCIP: Loss-controlled inverse projection of high-dimensional data. 2025. (Submitted to *IEEE Transactions on Visualization and Computer Graphics*)

The relation between the abovementioned papers and the chapters of this thesis is explained in a footnote at the beginning of every chapter which is based on these publications.

During the development of this thesis, other contributions were also achieved:

- D. Blumberg, Y. Wang, A. C. Telea, D. A. Keim, and F. L. Dennig. Inverting multidimensional scaling projections using data point multilateration. In *Proc. EuroVis Workshop on Visual Analytics (EuroVA)*, 2024.
- D. Blumberg, Y. Wang, A. C. Telea, D. A. Keim, and F. L. Dennig. MultiInv: Inverting multidimensional scaling projections and computing classifier maps by multilateration. *Computers & Graphics*, 2025.

CONTENTS

1 INTRODUCTION

1.1 MACHINE LEARNING

In the last decades, Machine Learning (ML) has exponentially grown to be a key technology that assists a wide, and increasing, range of application domains in science, engineering, technology, and everyday life. Whether serving scientists who aim to design specific chemical molecules for effectively combatting widespread diseases – like the by now infamous COVID (Kowalewski and Ray, 2020); engineers who work to optimize specific machinery configurations to reduce costs (Geng and Wang, 2020); geologists who aim to understand which factors have led to the appearance of specific mineral deposits (Petrelli and Perugini, 2016; Gregory et al., 2019); biologists aiming to automatically classify parasites based on images thereof to increase public health efficiency (Suwannaphong et al., 2023); or consumers asking for recommendations on buying specific products (Ren et al., 2020), ML technologies have become a fundamental part of all such solutions and applications.

ML comes in many guises and variants. At a high level, and without loss of generality, we can distinguish so-called *supervised* and *unsupervised* ML approaches. In the unsupervised case, ML algorithms are used to 'find structure' in a large amount of data samples (also called observations or data points) where each sample typically has tens up to thousands of so-called dimensions (also called variables or attributes). When successfully identified, structural elements such as clusters of samples (including their spread, density, and shape); distinctions between noisy samples and samples following a given pattern; the fit of samples to a given distribution; and the presence of outlier samples (which are far away from dense sample clusters) can further help practitioners in reasoning about the underlying phenomena that have created the data under study. In the supervised case, ML algorithms aim to construct a so-called *model* that best fits a set of predefined so-called *training set* of samples. The aim hereof is to next extrapolate the constructed model to *predict*, or *infer*, behavior on subsequent, so-called unseen, samples. Supervised learning next takes various forms such as *classification* and *regression*. In classification, the model aims to predict a so-called *class label* based on the values of other available sample dimensions. In regression, the model aims to predict one or several, typically continuous, so-called dependent dimensions based on the available, so-called independent, data dimensions.

1.2 CHALLENGES OF ML ENGINEERING

Developments in the ML field in the last decade have shown results which were earlier hard to imagine. In particular, the advent of Deep Learning (LeCun et al., 2015), which proposes models architected as dense neural networks that mimic (up to an extent) the functioning of the human brain, coupled with increasing computational power delivered by Graphics Processing Units (GPUs), have led to practical solutions for training highly complex (and large) models that can perform sophisticated tasks such as face detection and recognition; text and image analysis and synthesis; and steering automotive devices to navigate real-world complex landscapes over land, sea, and air. Names such as DALL-E, ChatGPT, or self-driving cars are well-known proofs of the above advances.

However, such spectacular developments have also generated additional *challenges*. Apart from those concerning the availability of large amounts of high-quality (and, for classification purposes, labeled) data, and the increasing costs of running huge models that consume large amounts of electrical power, a predominant challenge concerns the *black-box* nature of most modern ML models. Putting it simply, such models are trained by feeding them with data they have to fit – the hope being that, given sufficient such data, the model will learn the 'essence' of the phenomena that the data have been sampled from. After training, a similar hope exists – namely that the trained models will infer plausible results which actually match the said phenomena. However, during both training and inference, ML engineers (mainly involved during the training phase) and actual model users (mainly involved in the inference phase) have very little *understanding* of how such models operate.

1.3 EXPLAINABLE AI

The black-box nature of many ML models causes several important challenges to various types of stakeholders. For ML engineers, it is not clear what one can do to increase the performance of a given model or, in some cases, even how to architect and train a model when not sufficient (labeled) data is available (Benato et al., 2021). When models respond incorrectly – that is, yield wrong predictions on their so-called *testing sets* – there is no simple, out-of-the-box, solution to fix the issue. Equally, if not even more importantly, users of trained ML models increasingly want to know *why* a given model emits certain predictions. If this can be shown, users will gain *trust* in the respective solution, which is a key element for its practical acceptance (van den Elzen et al., 2023).

Following the above, the last decade has witnessed the growth of a specific research field called Explainable Artificial Intelligence (XAI) or interpretable machine learning (Ribeiro et al., 2016; Gilpin et al., 2018; Kaur et al., 2020; Molnar, 2020; Chatzimparmpas et al., 2024). As its name says, the main goal of XAI is to *explain* the working of ML models throughout their entire lifecycle, that is, from their architecting, following their training and testing, and ending with their deployment in the field. Key questions that XAI aims to address are: *Why* has a model yielded a given prediction; *what* has a model learned from its training data; and *how* has the model arrived at emitting a given prediction for a given input. When such questions can be (partially) answered, one can next fine-tune the design and training of the models in effective ways and also embed the models in complex real-life situations where non-specialist users operate side-by-side with the model to solve actual tasks.

1.4 VISUALIZATION, VISUAL ANALYTICS, AND XAI

Data visualization (VIS) is a separate discipline that has originally developed independent on ML. At its origin, visualization's key goals were to depict large amounts of data, mainly coming from numerical simulations and physical measurements, so that scientists and engineers could get a holistic, high-level, understanding of the measured or simulated phenomena. Following this, the respective (original) branch of visualization has been known under the name *scientific visualization* (scivis) (Hansen and Johnson, 2005; Telea, 2014). Examples of domains targeted by scivis include flow (fluid) simulation, mechanical engineering, material science, geo- and Earth sciences, and medical science (in particular, radiology and surgery). Following the increase of digital sources of information, and most notably the explosion of the internet, many additional data types and data sources entered the scope of visualization. Examples of such data types include networks (*e.g.*, social, transportation), source code repositories, large text document and image archives, and databases. This expanded the scope of visualization to consider contexts where data would not come from a physical phenomenon embedded in (2D or 3D) Euclidean space and governed by (well known) physical laws. Such data was addressed by a new subfield of visualization called *information visualization* (infovis) (Tufte, 2001; Munzner, 2014).

In parallel with the development of infovis, the goals addressed by visualization users would shift from the mere *depiction* of data to the *exploration* of data and, even further, to obtaining so-called 'actionable insights' from it. In other words, the key added value associated with visualization would increasingly correlate with the ability of its users to solve complex, often ill-posed, problems pertaining to a multitude of phenomena that underlie the data at hand. To address this, a new field, called Visual Analytics (VA) emerged (Andrienko et al., 2020; Cook and Thomas, 2005; Keim et al., 2008). Since its inception, around the turn of the millennium, VA proposed a wealth of interaction, navigation, and visual depiction techniques which, when applied jointly and iteratively, enable the so-called 'sensemaking loop', *i.e.*, empower users to incrementally extract higher-level knowledge from raw data – knowledge which is next directly usable for problem solving.

Given the above, it is not surprising that VA and ML have been increasingly getting closer to each other. We summarize their interaction as two 'flows' wherein one domain helps the other one, as follows:

ML4VIS: In this context, ML algorithms, *e.g.*, classifiers and regressors, are used to improve visualization algorithms and techniques. Examples hereof include the acceleration of algorithms due to the highly-parallelizable nature of ML, in particular DL, models (Espadoto et al., 2020); simplified operation of visualization techniques by using trained ML models to execute their tasks (Espadoto et al., 2019c); and the design of novel visualization metaphors using the generative power of modern ML models (Dibia and Demiralp, 2019).

VIS4ML: In this context, the roles are reversed: Visualization (mainly infovis and VA) techniques and tools are used to support the entire workflow involved in ML pipelines (Sacha et al., 2019; Yuan et al., 2021). This directly maps to the earlier-introduced XAI goals: Indeed, since visualization can effectively depict large and complex data (and underlying phenomena), it is a prime candidate for depicting how ML models operate during their training and/or inference. Moreover, the interactive exploration proposed by VA directly supports 'what if' exploration that enable users to probe complex models to enrich their understanding of the model operation. As very suitably put, visualization is a very effective instrument for 'opening the black box' of ML models (Tzeng and Ma, 2005; Alicioglu and Sun, 2022; Chatzimparmpas et al., 2020a, 2024).

This thesis contributes to both ML4VIS and VIS4ML, as we elaborate next.

1.5 DECISION MAPS FOR CLASSIFIER ENGINEERING

We further choose to focus our work on VIS4ML and ML4VIS on one of the two main applications of ML indicated in Sec. 1.1, namely *classification*. We motivate our choice as follows. Classification is a relatively simpler problem than general-purpose regression. Indeed, the output of a classifier is relatively simple: For a given input sample, a class label (defined over a usually small set of categorical values) is to be output, optionally accompanied by a continuous confidence value. In contrast, regressors output one or several continuous values for every given input. As such, designing VIS4ML solutions for general-purpose regressors is a significantly more complex task. Given this, and since we believe that VIS4ML has still unexplored potential for the classification context, we choose to focus our work on classification models.

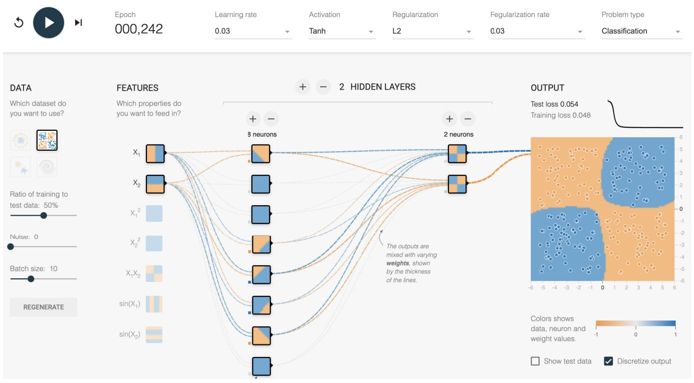

Epoch 000,242
Learning rate 0.03
Activation Tanh
Regularization L2
Fegularization rate 0.03
Problem type Classification
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
Test loss 0.054
Training loss 0.048
☐ Show test data
☑ Discretize output
DATA
Which dataset do you want to use?
REGENERATE
FEATURES
Which properties do you want to feed in?
X_1
X_2
X_1^2
X_2^2
X_1X_2
$\sin(X_1)$
$\sin(X_2)$
2 HIDDEN LAYERS
8 neurons
2 neurons
The outputs are mixed with varying weights, shown by the thickness of the lines.
OUTPUT
Colors shows data, neuron and weight values.
-1
0
1

Figure 1.1: Decision map for a simple neural network model for a 2-class problem. Two-dimensional samples enter a single hidden layer (8 neurons) followed by a 2-neuron classification layer. The sample space can be directly mapped to an image (since two-dimensional) yielding the decision map shown in the right image. Visualization constructed with Playground Tensorflow (Smilkov and Carter, 2024).

Further on, we focus on a specific visualization technique for classifier explanation, namely *decision maps*, also called by some authors decision boundary maps (DBMs). In a nutshell, decision maps are 2D images that aim to capture the behavior of a trained ML model in a *dense* way. To explain their operation, consider Fig. 1.1 (constructed using Playground Tensorflow (Smilkov and Carter, 2024)). A simple 3-layer neural network is created to classify two-dimensional samples in two classes. The network's input has two units since the data is two-dimensional. The output layer has two units, one responsible for each class. Since the data space is two-dimensional, we can map it one-to-one to a 2D image (see Fig. 1.1 right). If we color each image pixel by the label assigned to a sample having the respective x and y coordinates, we effectively obtain a decision map for our classifier. For this example, we see how, after training, the decision map consists of two disconnected blue decision zones and one, larger, orange decision zone. We also see how the training samples (white-outlined dots overlaid atop of the map) of the two classes perfectly fit the two decision zones.

Visualizing decision boundaries or decision maps is a powerful tool for understanding how a classifier operates. For example, Fig. 1.2, from the well-known machine learning library scikit-learn (Pedregosa et al., 2011), shows decision boundaries for four Support Vector Machine (SVM) classifiers with different kernels on the Iris dataset (Fisher, 1988). Since our target visual space is two-dimensional, only two features of the dataset's four original features are used in this simple example. The decision maps of these classifiers help get an intuitive understanding of their respective expressive power. In this toy example, one can see how a linear classifier (top row images) can only separate the data with a straight line; in contrast, using more complex kernels for the SVM classifier can separate the data with a more complex curve (bottom row images).

However, classification models have to handle data whose dimensionality is *far* larger than two. One way to construct decision maps for such cases is to first use *projections* to reduce such data dimensionality to two. Projection algorithms, also known as dimensionality reduction (DR) methods, well known in infovis, aim to place samples whose dimensions are similar close to each other in the (2D) projection space (Nonato and Aupetit, 2018; Espadoto et al., 2019a). Figure 1.3 (left) shows a projection of a 2310-sample, 19-dimensional, 7-class dataset, with labels mapped to categorical colors. We see clusters of same-color (same-label) points appearing in the projection since, indicating that many similar-class

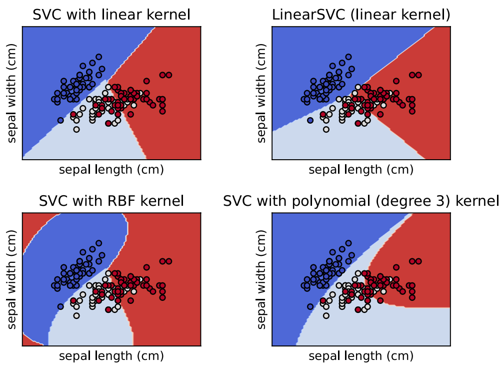


Figure 1.2: Plotting the decision boundaries for four SVM classifiers with different kernels on the Iris dataset. Only the first 2 features are used (Scikit-learn developers, 2024).

samples also have similar data values. Conversely, projection areas where such point clusters mix are a potential indication of classification difficulties, since some similar-valued samples have different classes. However, this visualization only shows what the trained model does for the *fixed* set of 2310 samples the projection was given to depict. How the model *generalizes* for other samples is left unexplained.

Decision maps aim to precisely fill the above-mentioned gap which is left unexplained by projection visualizations. More specifically, they aim to literally show how a classifier operates in the empty areas in a projection – that is, for samples which would project in the white space between projection points. To illustrate this, consider the dataset shown in Fig. 1.3 (left) on which we train a simple logistic regression model to infer the classes. Figure 1.3 (right image) shows a decision map for this classifier, constructed by an early technique (Rodrigues et al., 2018). All image pixels are now assigned a class label, that of a data sample which would project at that location. Finding such samples is done by using a so-called *inverse projection* technique which aims to reverse the effects of a user-chosen projection technique. Such visualizations extend the insights provided by simple projections in several directions, as follows:

- They show a *dense* image where every pixel conveys information on the explored classification model. The amount of depicted information does not depend on the size of the training (or test) set fed to the projection but only on the resolution of this image. Thus, increasing this resolution

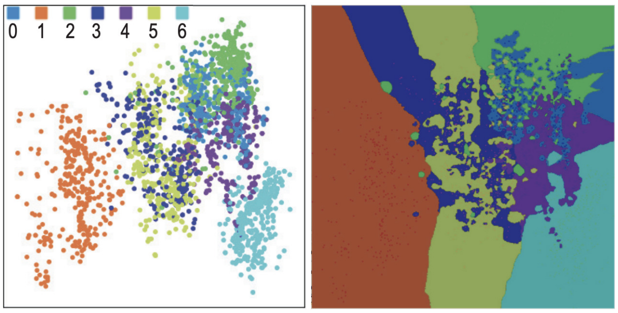


Figure 1.3: How decision boundary maps (DBMs) work. The left image shows a projection of a 19-dimensional, 2310-sample, 7-class dataset. Points map data samples and their class labels. While one can see how similar points (close in the projection) tend to have similar labels, this image does not tell us what happens in the white areas between projected points. The right image shows a DBM for the same dataset and model which fills such gaps. All image points are now colored to show the label assigned by the model to a sample which would project at that location. Compact same-colored areas show the model's *decision zones*; pixels on the boundaries of these areas show the model's *decision boundaries*. Images taken from (Rodrigues et al., 2018).

shows increasingly more insights into the classifier's behavior. Conversely, one can use (in theory at least) as many or as few samples as one wants to construct the projection.

- Decision maps show how the classifier behaves *between* the training (or test) set samples. The available 2D space is effectively partitioned into compact single-color areas which indicate the classifier's *decision zones*. Borders of these areas, where two or more different colors meet, indicate the classifier's *decision boundaries*.

- Decision zones and boundaries help get insights into a model's *behavior*. Analyzing how decision boundaries 'go through' a set of training samples effectively tells how a model *generalizes* to unseen samples. The jaggedness (or, conversely, smoothness) of decision boundaries gives insights into how well the model can fit its given training data. Additionally, if one marks misclassified samples on a decision map, one can see how close these are to decision boundaries and eventually perform training fine-tuning to alleviate such problems.

Given all above, we argue – in line with previous authors – that decision maps are useful tools for classifier engineering tasks such as assessing the actual fit of a model with what one expects from its known theoretical behavior (Rodrigues et al., 2019); getting insight on where, in data space, the classifier performs well or not (Rodrigues, 2020; Schulz et al., 2015); finding types of data samples for which the classifier is brittle (Machado et al., 2024); and guiding semi-supervised annotation to enrich labeled datasets for classifier training (Benato et al., 2024). In the same time, just as for all other XAI techniques (based on visual analytics or not), decision maps are only one of the tools in a wider arsenal offered to AI specialists to complete their various tasks related to model design, improvement, deployment, and usage. In particular, decision maps do not directly handle regressors but only classifiers; they show where, in data space, the decision boundaries are drawn, but do not offer direct mechanisms to 'tune' these boundaries, apart from suggesting the addition of extra (labeled) samples to certain zones; and they cannot directly link specific sample properties to the formation of decision boundaries in a model. Nevertheless, our conclusion is that, within these more general limitations, decision maps are a useful tool in the abovementioned arsenal and, as such, deserve being further refined to better accomplish the goals they have been originally introduced for.

1.6 RESEARCH QUESTIONS

In the last decade, several decision map techniques have been proposed (Schulz et al., 2015; Rodrigues et al., 2018; Espadoto et al., 2019c; Schulz et al., 2020; Rodrigues, 2020; Rodrigues et al., 2019; Oliveira et al., 2022, 2023a). Globally put, these techniques can handle generically any classification method, data of any dimensionality, and are relatively simple to set up and use in practice. Deep-learning-based variants of such methods additionally bring in extra computational performance (Oliveira et al., 2022) and are also robust to small-scale noise present in the input data (Oliveira et al., 2023b). Recent extensions allow enriching a plain DBM image with various additional information depicting *e.g.* the model's confidence, support of its decisions, and nonlinear deformations performed during the data-to-image mapping (Machado et al., 2024).

However, many aspects of decision maps remain unsolved to date and lead to corresponding research questions:

Quality (RQ1): While several decision map construction methods have been proposed, as outlined above, there have been formally no comparisons of their quality apart from the side-by-side display of decision maps produced by them for a few hand-picked classification models and datasets. This is not surprising: To perform such comparisons, we would need in the first place to have formal quality *metrics* that quantify desirable aspects of such maps. Two questions follow here: How can we define such quality metrics; and how do current decision map methods fare with respect to each other from the perspective of such metrics?

Coverage (RQ2): A decision map aims to depict the behavior of a trained model by mapping its high-dimensional decision zones to a partition of a 2D image into corresponding label-colored zones. Formally speaking, for such a map to capture the *full* behavior of the model, it should be able to map *all* the high-dimensional points that the model can work on to 2D points. However, it is clear that densely sampling a high-dimensional space creates a sample count having an exponential size with the spatial dimension, so not all such samples can be realistically mapped to 2D. Hence, existing decision map methods do internally make some (non-transparent) choices as to which areas of the data space they sample to depict. Key questions related to this coverage are: How do decision map methods differ in their sampling of the data space? Does this sampling depend on the data dimensionality or depicted classification model?

Control (RQ3): If, as stated above, current decision map methods do not densely cover their entire data space when creating decision map images, and arguably complete coverage is impossible in practice due to the aforementioned dimensionality problem, it follows that users would benefit from being able to specify how a decision map method samples this space. Such functionality would allow one to *e.g.* explore, in turn, different parts of the space, to compose a final picture (or insight) on how the model actually works – much like radiologists compose their understanding of a 3D CT or MRI volume by examining individual 2D slices thereof. The ensuing question is: How can we enable users to control the parts of the data space depicted by a decision map in simple, interactive, ways?

Interactivity (RQ4): One of the key added-value claims of decision maps resided on their ability to examine the behavior of a classifier and, upon seeing problems such as misclassifications, tune its hyperparameters and/or augment its training data to obtain an improved model (Rodrigues, 2020). For this to effectively work in a VA setting, the loop consisting of model training, decision map construction, and user-driven parameter changing should execute as fast as possible – ideally, at near-interactive rates. If this were possible, users could literally 'shape' the model's decision zones by interacting with data depicted in a decision map visualization. However, none of the current decision map methods offers this performance – the fastest such methods we are aware of still require several seconds to generate a decision map image of roughly 500^2 pixels (Oliveira et al., 2022). Can we significantly improve this computation time for creating decision maps for any classification model and for any type and dimensionality of datasets?

Summarizing the above, we can next state our key research question:

How can we improve decision map methods to understand and control which parts of the data space they sample, leading to desirable quality values, with high computational performance?

1.7 CONTRIBUTIONS

This thesis aims to answer the above-stated central research question by covering all sub-questions **RQ1** – **RQ4** that are subsumed therein. We next outline how we approach this goal and also thereby detail the structure of the thesis.

Chapter 2 presents related work in machine learning (ML) and visual analytics (VA) and, more specifically, the way in which ML can help VA and conversely. We introduce here basic notations used throughout the thesis and also explain the key methods and techniques that form the basis of ML4VIS and VIS4ML. While VIS4ML is the area that the use of decision maps fall into, ML4VIS provides several important techniques for building decision maps. We focus here chiefly on methods and quality metrics related to high-dimensional data and, more specifically, projection (dimensionality reduction) and inverse projection, since these are the key mechanisms underlying the computation of decision maps. Apart from presenting related work, this chapter outlines an important claim of this thesis, namely that ML4VIS and VIS4ML can be, and should be, approached *jointly* for most effective results in both fields. At a higher level, this chapter details the context in which we are going to improve decision maps to further answer **RQ1** – **RQ4**.

Chapter 3 presents an application of decision maps, as we encountered them at the onset of our work, for understanding classification models constructed for a geoscience application, namely the understanding of the genesis of mineral deposits. The main added value of this application is to expose both the advantages, but also the limitations, of current decision map methods in a *practical* setting, when used by scientists who are not ML specialists. Our work outlines that decision maps can indeed help to answer concrete questions in our chosen application domain – an insight which, to our knowledge, was not presented before in the DBM literature. This insight justifies the added value of decision maps and, thus, implicitly, our goal of further analyzing **RQ1** – **RQ4**. In the same time, our work exposed several limitations of existing DBM methods, thus provides starting points to further answer **RQ1** – **RQ4** in the following chapters.

Chapter 4 addresses **RQ1** by proposing a set of novel metrics for the comparison of decision maps – more precisely, the inverse projection techniques which are fundamental to any decision map construction. We use these metrics to evaluate all the decision map techniques (and inverse projections) we are aware of on a variety of datasets and classification models. Our findings show to-date unknown differences between the evaluated DBM techniques and, also, propose a practical workflow for choosing suitable techniques in practice based on the aspects quantified by our metrics. To our knowledge, this is the first – and, to writing date, only – study that quantitatively compares decision maps and inverse projections. This chapter also outlines a surprising observation we made during our study, namely that decision maps seem to cover only a small, manifold-like, surface embedded in the high-dimensional data space they aim to depict.

Chapter 5 further explores the above-mentioned observation on the coverage of decision maps. We study this issue of coverage further using a larger set of decision maps (and inverse projections), datasets, and classification models. We also propose both visual (qualitative) and computational (quantitative) ways to assess the dimensionality of this coverage – thereby addressing **RQ2**. Our findings confirm our earlier observation, namely that all decision map methods cover only a manifold of intrinsic dimensionality two embedded in the data space, regardless of any other parameters. Our findings – to our knowledge, observed for the first time by our work – significantly affect the way in which decision maps can be next used in practice.

Chapter 6 builds on the findings explored in Chapter 5 by proposing an interactive mechanism that allows users to control which parts of the data space an inverse projection – and thus a decision map – cover. We propose a simple, generic, and computationally efficient mechanism that allows users to 'pull' the aforementioned manifold surface closely towards selected data samples, thereby addressing **RQ3**. We illustrate the added value of this mechanism by presenting an application in style transfer in image datasets. To our knowledge, our work is the first technique that allows users to explicitly control inverse projections. In contrast to the previous chapters, which focus on VIS4ML application, chapter falls under the topic ML4VIS, as it uses deep learning based methods to construct the inverse projections.

Chapter 7 attacks our last research question, **RQ4**, namely the accelerated computation of decision maps. We propose Fast-DBM, a simple heuristic that reduces the number of times that an inverse projection needs to be called to compute a decision map by exploiting smoothness and continuity properties which should be present in the underlying mappings. Our method can be generically applied to any inverse projection and classification method, thus, any decision map method. Our results show that we can compute decision maps at about one order of magnitude faster than existing methods and with minimal quality loss. We also generalize this algorithm to accelerate the computation of other classifier maps such as gradient maps and distance-to-boundary maps (see Sec. 2.3.4.2). This enables one, for the first time, to practically compute maps that explain classifiers at high resolutions. Our work next opens the possibility for using decision maps in near-interactive visual analytics loops where users iteratively refine a classification model based on insights drawn from the visualized decision maps.

Finally, Chapter 8 concludes this thesis by revisiting our research questions, summarizing our results, and outlining potential directions of future work for the computation and application of decision map techniques.

2 RELATED WORK

2.1 INTRODUCTION

This chapter introduces basic notations and concepts used throughout the thesis and explain how both dimensionality reduction (DR) and machine learning (ML) can help each other in achieving their respective aims, *i.e.*, ML4VIS and VIS4ML. By presenting an overview of the state-of-the-art in both fields, we aim to provide readers with a better understanding of the existing synergies between ML and DR. More importantly, we highlight the role of decision maps in this context, *i.e.*, their importance in VIS4ML, and how they can benefit from ML4VIS[1].

ML has become one of the indispensable instruments in data-driven science and virtually any data-intensive application domain in our society. At a high level, and without loss of generality, ML models can be described as engines which process *high-dimensional data* – that is, collections of observations (samples) consisting of tens up to millions of individual measurements (dimensions) of a given phenomenon. Such data occurs throughout the ML pipeline – it is present in the input of the models (*e.g.*, images consisting of millions of pixels); in the internal working of such models (*e.g.*, the so-called activations of neural units in the many intermediate layers of a DL model), and also in the models' output (*e.g.*, the image created by generative AI techniques from given inputs). As such, it is not surprising that understanding high-dimensional data, and how it is transformed by ML models, is a key goal and challenge in ML.

In a separate field, exploring and understanding high-dimensional data is one of the top goals of information visualization (infovis) (Munzner, 2014; Telea, 2014; Liu et al., 2015). During the last decades, many techniques have been proposed to this end, including scatterplots and scatterplot matrices (Yates et al., 2014; Lehmann et al., 2012), parallel coordinate plots (Inselberg and Dimsdale, 1990), table lenses (Rao and Card, 1994; Telea, 2006), and glyphs (Borgo et al., 2013). However, most such techniques are fundamentally limited in the size of the datasets they

1 This chapter is based on the paper "Seeing is Learning in High Dimensions: The Synergy Between Dimensionality Reduction and Machine Learning" (Telea et al., 2024).

can depict: They can show either datasets having many samples with few dimensions, few samples having many dimensions, but not both.

Dimensionality reduction (DR) techniques, also called multidimensional projections (MPs), are a family of visualization techniques that aim to solve the aforementioned visualization scalability issue in both the sample and dimension count. Simply put, given a high-dimensional dataset consisting of several samples, DR techniques create a low-dimensional (typically 2D or 3D) scatterplot in which close points correspond to similar samples in the input dataset. This allows users of DR visualizations to identify salient *patterns* in the dataset in the form of clusters of closely-packed scatterplot points, clusters of points with different shapes or densities, or outlier points (Lespinats and Aupetit, 2011; Sorzano et al., 2014; Nonato and Aupetit, 2018). Tens, if not hundreds, of DR techniques have been designed to cater for the various functional and non-functional requirements inherent to the DR process, such as computational scalability, ability of treating data of various types and dimensionality, handling time-dependent data or data with missing values, ability of depicting new samples along existing ones (out-of-sample property), stability in the presence of noise, ability of capturing specific aspects present in the input dataset, and ease of use (Cunningham and Ghahramani, 2015; Espadoto et al., 2019a; Nonato and Aupetit, 2018).

At a first glance, ML and DR are separate fields with different goals. ML is chiefly concerned with *learning* a model to predict the behavior of some phenomenon from existing samples thereof. In a more general setting, this has been extended to additional tasks such as data representation (autoencoders) or generative AI. For the purpose of our discussion next, we will mainly focus on prediction tasks, either in a classification or regression setting. Separately, DR aims at depicting, or *seeing*, the samples of such a phenomenon. We argue that these two goals – learning and seeing – are, however, strongly related, and advances in one field directly support requirements of the other field in both directions. Simply put, we argue that *seeing is learning*, in both directions of the implication, as outlined below:

- **Learning needs seeing (VIS4ML):** The ML field generates complex models, whose 'black-box' behavior is increasingly hard-to-understand by both their developers and users. Understanding such models is increasingly important for fine-tuning their behavior but also gaining trust in their deployment. Such understanding can be massively aided by seeing (visualizing) their structure and operation. Since ML models revolve around high-dimensional data, and DR techniques are ideally suited for depicting such data, DR techniques are a candidate of choice for visualizing them;

- **Seeing needs learning (ML4VIS):** Existing DR techniques are increasingly challenged by the already-mentioned sum of requirements they have to cope with. Few, if any, of such existing techniques can cope with all these requirements. In contrast, many ML techniques are designed upfront to handle such requirements, especially computational scalability, accuracy, stability, and out-of-sample ability. Given these, it makes sense to use ML techniques to learn the high-to-low dimensional mapping and thereby assist the DR task.

In this chapter, we explore the commonalities of ML and DR techniques and bring evidence of existing, emerging, and potentially new interactions between these two fields. We proceed by introducing our two fields of interest – ML and DR – with an emphasis on their commonalities (Sec. 2.2). We next explore in Sec. 2.3 how learning (ML) is supported by seeing (DR), especially in the creation of visual analytics (VA) solutions for explainable artificial intelligence (XAI), in particular decision maps. Subsequently, we study in Sec. 2.4 the converse connection, that is, how seeing (DR) is supported by leaning (ML). We further outline in Sec. 2.5 new, emerging, connections between the two fields that point to promising future research directions in which the DR and ML fields can benefit from each other. Finally, Section 2.6 concludes the chapter.

2.2 BACKGROUND

In this section, we provide a general introduction to ML and DR concepts, notations, and principles, with a focus on highlighting the commonalities between the two fields, which will be further explored in the remainder of the chapter.

Notations: Let $D = \{\mathbf{x}_i\}$ be a dataset of n-dimensional samples, also called observations or data points $\mathbf{x}_i$, $1 \leq i \leq N$. A sample $\mathbf{x}_i = (x_i^1, \ldots, x_i^n)$ is a tuple of n components x_i^j, also called features, variables, attributes, or dimension values. For exposition simplicity, we next consider that $x_i^j \in \mathbb{R}$; other data domains are treated similarly for the purpose of our discussion. We denote by $Z \subset \mathbb{R}^n$ the spatial subset where samples of a given phenomenon are found. For instance, considering image data, only positive values (possibly bound by a maximum) can denote pixel intensities. Following this notation, D can be depicted as a table with N rows (one per sample) and n columns (one per dimension). Typically, these dimensions are considered to be *independent* variables, *i.e.*, whose values are measured from the behavior of a given phenomenon over Z. Atop of these, D can have one or more dimensions (columns) of so-called *dependent* variables, also called labels or annotations. We next consider a single such dependent variable $y_i \in C$, where C is the domain of definition of the labels, unless specified otherwise. We denote the annotated dataset D by $D_a = [D|\mathbf{y}]$.

Machine learning basics: Given a so-called test set $D_T \subset D_a$, machine learning (ML) techniques aim to create a function

$$f : Z \rightarrow C \tag{2.1}$$

which predicts data values for most (ideally, all) samples in Z. Models f are built by using a so-called training set $D_t \subset D_a$, $D_t \cap D_T = \emptyset$ so as to maximize the number of test set points $\mathbf{x}_i \in D_T$ for which $f(\mathbf{x}_i) = y_i$. ML models can be further split into *classifiers*, for which C is typically a categorical, or label, dataset; and *regressors*, for which C is typically a subset of $\mathbb{R}$. For regressors, f typically strives that $f(\mathbf{x}_i)$ is as close as possible to y_i, whereas for classifiers exact equality is aimed at.

Many methods exist to measure the performance of ML models. The most widespread such methods measure several so-called quality metrics on the training set (training performance) and, separately, on the unseen test set (testing performance). Common metrics include accuracy, precision, recall, F-score, and Cohen's kappa score. More advanced methods take into account hyperparameters that allow optimizing between precision and recall, *e.g.* the Receiver Operator Characteristic (ROC) curve and area underneath (Botchkarev, 2019; Jiang et al., 2020; Thiyagalingam et al., 2022).

Dimensionality reduction basics: A dimensionality reduction technique, or multidimensional projection P, is a function that maps every $\mathbf{x}_i \in D$ to a point $P(\mathbf{x}_i) \in \mathbb{R}^q$. For convenience, we next denote by

$$P(D) = \{P(\mathbf{x}_i) | \mathbf{x}_i \in D\} \tag{2.2}$$

the projection of an entire dataset D. For visualization purposes, $q \in \{2, 3\}$, *i.e.*, $P(D)$ is a 2D, respectively 3D, scatterplot. Since 2D projections are by far the most commonly used in VIS4ML, we will focus on them for the remainder of this thesis.

At a high level, all projection techniques P aim to preserve the so-called *structure* of the dataset D, so that users can infer this structure by visualizing $P(D)$, following a well-known inverse mapping principle in data visualization (Telea, 2014). Forms of such structure include, but are not limited to, clusters of similar samples; clusters having different sample densities; similarities between different samples; and outlier samples. Structure-preserving projections map (some) of these data properties to the corresponding properties of their generated scatterplots. Usually, projections do not use data annotations (even when these are available), but only independent variables – more on this aspect to be discussed further in Sec. 2.4.1.

Since data structure preservation entails several aspects, as outlined above, different so-called *quality metrics* have been devised to capture the abilities of a given P. A quality metric is a function

$$M(D, P(D)) \in \mathbb{R}^+ \tag{2.3}$$

that tells how well the scatterplot $P(D)$, or a part thereof, captures a given aspect present in the dataset D. At a high level, such metrics can be grouped into (1) measuring *distance* preservation between pairs of samples, respectively pairs of projection points, in $\mathbb{R}^n$ and $\mathbb{R}^q$ respectively, such as normalized stress and the Shepard diagram correlation (Joia et al., 2011); and (2) measuring if neighborhoods (groups of close points) in D are mapped to neighborhoods in $P(D)$, such as trustworthiness and continuity (Venna and Kaski, 2006), false and missing neighbors (Martins et al., 2014), and the Kullback-Leibler divergence (van der Maaten and Hinton, 2008). The latter class is extended for projections of labeled data $P(D_a)$ by metrics such as neighborhood hit and class consistency (Paulovich et al., 2008; Sips et al., 2009). Detailed surveys of projection quality metrics are given in (Aupetit, 2007; Lespinats and Aupetit, 2011; Nonato and Aupetit, 2018; Espadoto et al., 2019a).

Projection quality is implicitly linked to the measuring of the quality of inverse projections (discussed further in Sec. 2.5.2.1). Indeed, if a direct projection P has poor quality, then it will not preserve some aspects of its input data D. As such, and since inverse projections P^{-1} are constructed based on the projection $P(D)$ of the data D, it follows that also P^{-1} will inherently have low quality. In turn, poor direct and/or inverse projection methods will affect the quality of visualizations constructed using them such as the decision maps we discuss further in Sec. 2.3.4. Importantly, we note here that, in general, no projection technique P can preserve *all* structure in D perfectly in $P(D)$ – that is, at least some quality metrics will have relatively low values. This is inherent to the operation of mapping datasets having high *intrinsic* dimensionality to a (very) low-dimensional target space.

To disentangle such quality assessments, a simple but effective rule of thumb is to first measure the quality of $P(D)$ for some given dataset D. If this quality is deemed by the users to be too low for their application at hand, then $P(D)$ should be discarded and one should attempt to create better results by using *e.g.* different projection techniques or their hyperparameter settings (Martins et al., 2014). Conversely, if the quality of $P(D)$ is deemed good enough, then one can further measure the quality of P^{-1} to decide if this latter technique is good enough for the application context. We will use this approach throughout all our work in this thesis.

2.2.1 *Interaction between ML and DR*

As mentioned in Sec. 2.1, our central statement is that *learning* (accomplished using ML) and *seeing* (visualization accomplished using DR) are intimately related to each other. This assertion, illustrated by Fig. 2.1, is explored next in detail.

2.2.1.1 *How DR helps ML and conversely*

Machine learning pipeline: The central box (Fig. 2.1 blue) shows a technical view on the typical ML pipeline which maps an input real-valued dataset D into class labels or another real-valued signal by means of a classifier, respectively regressor. Such ML pipelines can be next deployed to assist a wide variety of tasks. In our work here, we do not further detail these, but rather focus on how DR techniques can be used to assist the technical aspects of a typical, task-generic, ML pipeline; and conversely, how ML techniques can generically assist constructing better DR methods. As explained earlier in Sec. 2.1, ML models operate on high-dimensional data. The green arrows atop this pipeline point to various visualization methods that use DR to depict such data. By using such visualizations, one can literally 'see' how the model learns. We further exemplify the use of such visualizations for ML tasks such as semi-automatic labeling (Sec. 2.3.2), assessing classification difficulty (Sec. 2.3.1), and assessing training of DL models (Sec. 2.3.3).

Dimensionality reduction pipeline: The bottom box (Fig. 2.1 yellow) shows how ML regressors can be used to create better DR projections of any high-dimensional data. Examples of this process include (self-)supervised projections and sensitivity analyses (Sec. 2.4.1), inverse projections (Sec. 2.4.2), and quality analysis for inverse projections (Sec. 2.5.2). Such better DR methods can be next used for assisting ML engineering tasks, as shown by the red arrow in Fig. 2.1.

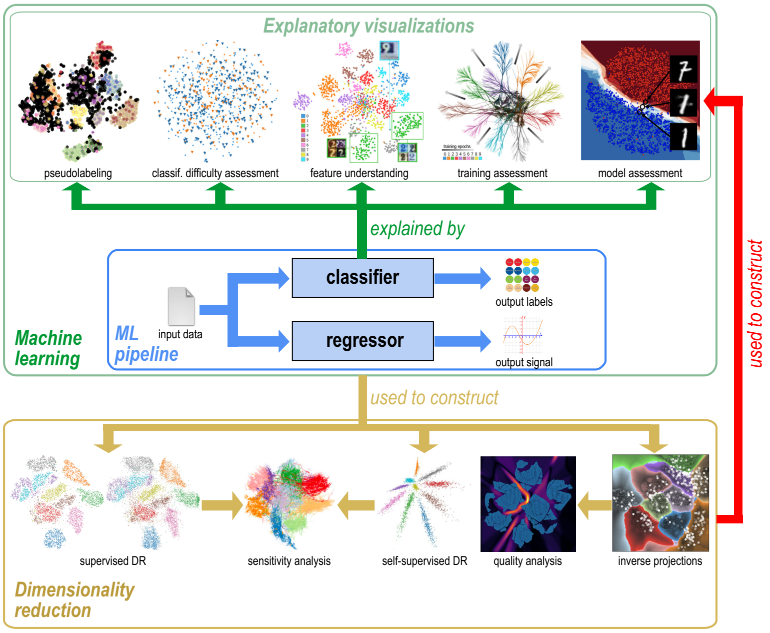


Figure 2.1: Two-way interaction between machine learning (ML) and dimensionality reduction (DR) workflows. ML algorithms can be used to construct DR techniques. In turn, these can be used to construct explanatory visualizations for ML. See Sec. 2.2.1.

2.2.2 *Common aspects of DR and ML*

Section 2.2.1.1 and Fig. 2.1 have outlined how DR can help ML and conversely. As such, it is not surprising that DL and ML share many common aspects. We detail next such commonalities, grouped in functional and non-functional ones, following a systems engineering perspective (Sommerville, 2015).

2.2.2.1 *Functional commonalities*

Functional aspects describe how a system should operate. As already outlined, both ML models f and DR projection methods P are specialized cases of *inference* involving high-dimensional data. More specifically, P can be seen as a particular type of regressor from $\mathbb{R}^n$ to $\mathbb{R}^2$. Given this, we next use the notation $\mathcal{M}$ to jointly denote an ML model or DR algorithm, when distinguishing between the two is not important.

2.2.2.2 *Non-functional commonalities*

Non-functional aspects describe how a system should behave in practice. Without claiming full coverage, we identify the following key aspects that both ML and DR techniques $\mathcal{M}$ strive to achieve in their operation. We also outline cases where these two classes of techniques achieve the respective requirements up to different degrees, thereby pointing to potential synergies where one technique family can be used to assist the other.

Genericity: In the ideal case, $\mathcal{M}$ should be readily applicable to any dataset D – that is, of any dimensionality, attribute types, and provenance application domain.

Accuracy: $\mathcal{M}$ should deliver highly accurate results (inferences for ML; projection scatterplots for DR) as gauged by specific quality metrics in the two fields.

Scalability: $\mathcal{M}$ should scale well computationally with the number of samples N and dimensions n – ideally, $\mathcal{M}$ should be linear in both N and n. In practice, $\mathcal{M}$ should be able to handle datasets with millions of samples and hundreds of dimensions on commodity hardware at interactive rates. This further enables the use of $\mathcal{M}$ in visual analytics (VA) scenarios where the iterative and interactive exploration of complex hypotheses via data visualization is essential. We discuss this aspect further in Secs. 2.3 and 2.4.

Understandability: For a technique to be useful and usable in practice, its operation should be easily understandable by its intended users. This requirement takes different forms for ML and DR techniques. In general, ML techniques have an easy-to-understand *output* – they are designed to infer features having a clear meaning, *e.g.*, the classes present in a dataset. However, due to their often black-box nature, the way in which they *operate* to do this is far less understandable, leading to challenges for their design, deployment, and acceptance (see next Sec. 2.3.3).

In contrast, most DR methods have a relatively clear way of operation – the projection aims to minimize a cost function that preserves certain aspects of the data D in the projection scatterplot $P(D)$ (Sorzano et al., 2014; Nonato and Aupetit, 2018). However, their output – a raw scatterplot, in the minimal case – is hard to interpret and requires additional explanatory mechanisms (Martins et al., 2014; da Silva et al., 2015; Coimbra et al., 2016; Chatzimparmpas et al., 2020b; Marcilio and Eler, 2021; Tian et al., 2021; Thijssen et al., 2023).

Understandability is subtly related, but not identical, to the concept of *interpretability*. As mentioned above, we refer to understandability as the 'low level' ability of the intended users of a technique or tool to grasp how the tool works, at a basic level, so they are able to deploy it in practice. Interpretability operates at a higher conceptual level and refers to the ability of the users to reason about how the tool operates *internally* when executing its work. For ML models, for instance, linear regression is arguably more interpretable than deep neural networks due to its inherent linear model. Similarly, PCA's operation based on a global and linear data transformation is easier to understand than local and/or non-linear DR techniques such as t-SNE. In our further discussion, we mainly focus on the lower level of understandability.

Out of sample (OOS): An operator $\mathcal{M}$ is said to be OOS if it can extrapolate its behavior beyond the data from which it was constructed. In ML, this usually means that the model f extrapolates from a training set D_t to an unseen test set D_T and beyond. By analogy, a projection P is OOS if, when extending some dataset D with additional samples $D' \nsubseteq D$, the projection $P(D \cup D')$ ideally keeps the samples of D at the locations they had in $P(D)$, *i.e.*, $P(D \cup D') = P(D) \cup P(D')$. If P is OOS, this helps users to maintain their 'mental map' obtained by studying $P(D)$ when they further study $P(D \cup D')$. As most ML methods are OOS by design, they can be potentially used to design OOS projections (see next Sec. 2.4).

Stability: Small changes in the input dataset D should only lead to small changes in the output dataset $\mathcal{M}(D)$. If not, spurious perturbations in D can massively affect the resulting inference $\mathcal{M}(D)$ thereby rendering such results potentially unusable and/or misleading. Similarly, large-scale changes in D should arguably lead to correspondingly large changes in $\mathcal{M}(D)$. Stability is related but not the same as OOS: An OOS algorithm is stable by definition but not all stable algorithms are OOS (Vernier et al., 2021; Espadoto et al., 2019a; Oliveira et al., 2023b). Most ML methods are OOS by design, a property which is not shared by many projection techniques – therefore opening up an interesting case for using ML for DR. We discuss stability and OOS in more detail in Secs. 2.4.1 and 2.5.2.

Ease of use: Visualization methods aim, by construction, to be easily usable by a wide range of users and with minimal or no programming effort. In contrast, building – and especially debugging and fine-tuning – an ML pipeline can be challenging for practitioners with limited training in ML. As such, this offers opportunities for using visualization (and DR in particular) to ease the task of ML practitioners.

Availability: $\mathcal{M}$ should be readily available to practitioners in terms of documented open-source code. While sometimes neglected, this is a key requirement for ML and DR algorithms to become adopted and impactful in practice.

Table 1 summarizes the above observations at a high level by comparing how ML and DL techniques satisfy in general the above requirements. Scores are given on a 5-point Likert scale (++: best; --: worst), according to our own experience. Besides genericity, where both ML and DR algorithms score equally well, all other requirements are met complementarity by the two algorithm families. This supports our earlier point that the two technique classes can support each other, if combined properly.

Table 1: Comparison of how ML and DR methods satisfy desirable requirements (**Gen**ericity, **Acc**uracy, **Scal**ability, **Out** (understandability of output), **Alg** (understandability of algorithm), **OOS**, **Stab**ility, **Ease** of use, **Avail**ability).

Methods	Gen	Acc	Scal	Out	Alg	OOS	Stab	Ease	Avail
ML	++	++	++	++	--	++	++	-	++
DR	++	-/+	-	--	+	--	-	++	++

We next explore these commonalities and contrasts by first discussing how DR is used to help ML (Sec. 2.3) and next how ML is used to create better DR algorithms (Sec. 2.4).

2.3 SEEING FOR LEARNING: DR ASSISTS ML

Many examples of visualization applications that assist ML workflows exist, most often coming in the form of complex multiple-view visual analytics systems (Garcia et al., 2018; Hohman et al., 2019; Yuan et al., 2021; Alicioglu and Sun, 2022; Chatzimparmpas et al., 2023). An exhaustive presentation thereof is out of the scope of this chapter. Rather, we focus in the following on selected use-cases where DR techniques have been used, with minimal additions, to assist ML workflows: assessing and improving classifiers (Sec. 2.3.1), pseudolabeling for enriching training sets (Sec. 2.3.2), exploring deep learning models (Sec. 2.3.3), and exploring classifier outputs via decision boundary maps (Sec. 2.3.4). We also highlight connection points between the discussed techniques and the focus of our research, namely decision maps.

2.3.1 *Assessing and improving classifiers*

One of the simplest, and still most frequently used, application of DR in ML is to project a labeled training or test set D_a with points $\mathbf{x}_i$ colored by their ground-truth labels y_i or labels $f(\mathbf{x}_i)$ inferred by some classifier f. The rationale for this use-case is straightforward: A projection places similar samples close to each other; a classifier labels similar samples similarly; hence, the visual structure of the projection helps several tasks:

- see how (and where) are *misclassified* samples distributed over the extent of a test set D_T (to next elicit what makes them hard to classify);
- see how well a training set D_t *covers* the data space (to *e.g.* determine where extra training samples are needed);
- see how well a training set D_t is *separated* into different same-label sample groups (to next predict the classification ease).

The two first tasks are quite straightforward. In contrast, the last task is particularly interesting. The intuition that a projection $P(D)$ which is well separated into compact same-label groups indicates that D is easy to classify is quite old. Yet, a formal study of this correlation was only relatively recently presented (Rauber et al., 2017b). In the respective work, the authors show that, given a range of classifiers, a dataset D whose projection $P(D)$ has well-separated classes (as measured by the neighborhood hit metric (Paulovich et al., 2008)) is far easier classifiable than a dataset whose projection shows intermixed points of different labels (low neighborhood hit). The projection $P(D)$ becomes a 'predictor' for the ease of classifying D, helping one to assess classification difficulty *before* actually embarking on the expensive cycle of classifier design-train-test.

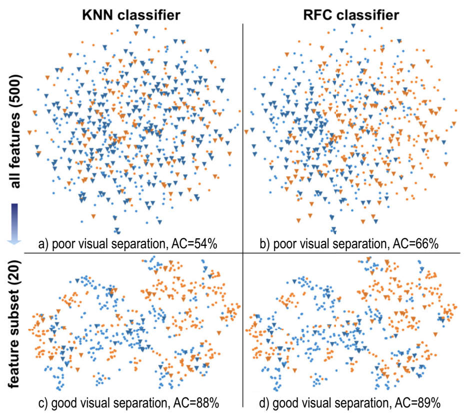


Figure 2.2: Classification difficulty assessment via projections (Rauber et al., 2017b).

Figure 2.2 illustrates the above usage of projections. Images (a) and (b) show the two-class Madelon dataset (Guyon et al., 2004) ($n = 500$ dimensions, $|C| = 2$ classes) classified by KNN and Random Forests (RFC) respectively, with samples projected by t-SNE (van der Maaten and Hinton, 2008) and colored by class labels. The two projections show a very poor separation of the two classes, in line with the obtained low accuracies $AC = 54\%$ and $AC = 66\%$ (also visible by the misclassified samples, marked as triangles). Images (c) and (d) show the same dataset where extremely randomized trees (Geurts et al., 2006) was used to select $n = 20$ dimensions. The projections show a much higher visual separation of the two classes, in line with the higher accuracies $AC = 88\%$ and $AC = 89\%$ obtained. Many other examples in (Rauber et al., 2017b) show that projections can predict classification accuracy quite well.

Connection to DBMs: Decision maps, discussed next in this chapter, keep the predictive power of projections for assessing classification models and also enhance this simple scatterplot-only view by filling in the gaps between projected data points.

2.3.2 *Pseudolabeling for ML training*

If projections are good predictors of classification accuracy, it means that their low-dimensional (2D) space captures well the similarity of the high-dimensional samples. This leads to the idea of using projections to create, rather than just explain, ML models. A first attempt was shown by Bernard et al. (2017) in the context of an user evaluation that compared classical active learning with a user-supported procedure they dubbed Visual Interactive Labeling (VIL). Next after that, Benato et al. (2018) proposed a very similar approach to VIL, called visual pseudolabeling, aimed to assist building a classifier from a training set having only very few labeled points: The entire training set, including unlabeled points, is projected and the user explores the projection to find unlabeled points tightly packed around labeled ones. Next, the user employs a tooltip to study the attributes of these points to confirm that they have the same class as the surrounded labeled one. If so, the user simply assigns that label to the unlabeled points. This workflow minimizes the user's labeling effort to quickly lead to sufficiently-large labeled sets for training the desired model. Interestingly, *automatic* label propagation in the embedded space using state-of-the-art methods (Belkin et al., 2006; Amorim et al., 2016) leads to *poorer* results than user-driven labeling, which confirms the added value of the human-in-the-loop, and thus of the projections.

However, optimal results are obtained when humans and machine *cooperate*, rather than aim to replace, each other. Benato et al. (2021) refined the above workflow to (a) use automatic label propagation (Belkin et al., 2006; Amorim et al., 2016) for the projection points where the propagation confidence is high; and (b) expose only the remaining unlabeled points to manual labeling (see Fig. 2.3). This way, many 'easy to label' points are handled automatically and the user's effort is channeled towards the hard cases, further reducing the manual labeling effort. This strategy also leads to increasing model accuracy and, again, surpassed confidence-based label propagation into the high-dimensional space.

Connection to DBMs: Decision boundaries are a crucial concept in the so-called *uncertainty sampling* strategy in active learning (Monarch, 2021). As such, decision maps can be used to visualize the decision boundaries of a classifier, and therefore provide a visual cue to the user to select the most informative samples for annotating. This strategy was recently explored showing an increase of the efficiency of users in constructing annotated training sets as compared to using only raw projections (Benato et al., 2024).

2.3.3 *Understanding DL models*

Deep learned (DL) models, with their millions of parameters, are among the hardest artifacts in ML to understand (Shwartz-

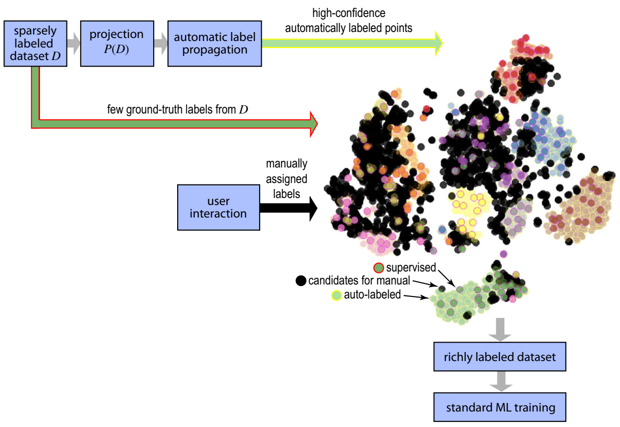


Figure 2.3: Semi-automatic label propagation for constructing training sets. An algorithm propagates ground-truth labels from a small set of supervised samples towards unlabeled neighbor samples in the projection. When this algorithm is uncertain, samples are left for manual labeling (Benato et al., 2021). See Sec. 2.3.2.

Ziv and Tishby, 2017; Azodi et al., 2020). Visualization has been listed early on as the technique of choice for explainable AI (XAI) for DL models (Tzeng and Ma, 2005). A recent survey (Garcia et al., 2018) outlines a wide spectrum of visual analytics techniques and tools used for DL engineering, grouped along how they support the tasks of training analysis (TA), architecture understanding (AU), and feature understanding (FU). Given the diversity of these tasks, the variety of the proposed visualization solutions – *e.g.* matrix plots (Pezzotti et al., 2017), icicle plots (Alsallakh et al., 2018), parallel coordinate plots (Strobelt et al., 2017), stacked barcharts, annotated networks (Liu et al., 2016), activation maps (Chattopadhay et al., 2018) – is not surprising.

Projections occupy a particular role among such visualizations due to their ability to compactly capture high-dimensional data – in the limit, a projection needs a single pixel to represent an n-dimensional point, for any value of n. As such, they are very suitable instruments to depict several aspects of a DL model. For example, in Fig. 2.4a, every point denotes a high-dimensional sample in D, in this case a digit image from the SVHN dataset (Rauber et al., 2017b). The points, colored by their ground-truth classes, have as dimensions all activations of the last hidden layer – also called *learned features* – of a DL model trained to classify this dataset. We notice a good separation of same-class images (the projection contains compact same-color groups), which tells that the model's training went well. We also see, for each color (class), two distinct such groups. This tells that the model has learned to split images of the *same* digit into two subclasses. Upon inspection, illustrated by the tooltips in the figure, we see that the model has learned *by itself* to separate dark-on-bright-background digits from bright-on-dark background ones. Such findings would be hard to get without the projection-based visual exploration tool. Moreover, such findings can help fine-tuning the model to increase performance – in this case, eliminate the learning of the background-*vs*-foreground artificial separation for same-class digits.

Figure 2.4b explores a different DL aspect, namely how the model learns. For every epoch, a projection of all training-set samples is made, using as dimensions the samples' last hidden layer activations, similar to image (a). To maintain temporal coherence, *i.e.*, have similar-value samples from the same or different epochs project to close locations, a *dynamic* projection algorithm, in this case dt-SNE (Rauber et al., 2016), was used (see further Sec. 2.5.2). Next, same-sample points from all epochs are connected by a trail. As the last step, trails are bundled in 2D (van der Zwan et al., 2016) to reduce visual clutter. The resulting image (b) shows how the projection literally 'fans out' from a dark clump (in the middle of the image), which represents the similar activations of all samples in the first epoch, to separate clusters of same-label points (in the final epoch). This effectively summarizes the training success – the model has increasingly learned to separate the classes throughout its training. We also see some challenges of this model: The purple bundle (digit 4) is less well separated from the others, which indicates difficulties in classifying this digit.

Figure 2.4c shows a similarly-constructed visualization but where the trails connect projections of test-set image activations through all network's hidden *layers*. Bundles start fanned out but apart from each other, indicating that the trained model successfully separates the classes even after its first layer. Same-color trails in a bundle progressively fan in and also stay separated from trails in other bundles, indicating that, as we go down the model towards its further layers, class separation only becomes better, *i.e.*, that the chosen network architecture is indeed good for the classification task at hand.

Connection to DBMs: If visual structures in projections help understanding the operation of a DL model, then the richer visual structures shown by decision maps – such as the shapes and

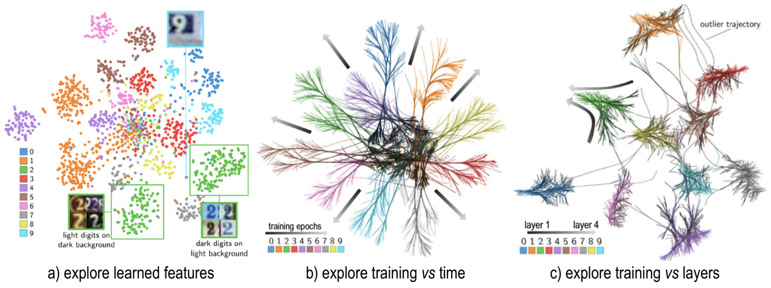


Figure 2.4: Projections for understanding DL models. Exploring (a) activations of similar instances, (b) evolution of activations over training epochs, and (c) evolution of activations over network layers (Rauber et al., 2017a). See Sec. 2.3.3.

sizes of decision zones and the smoothness (or lack thereof) of decision boundaries – will be able to give richer insights on such operation.

2.3.4 *Decision boundary maps*

A key aspect of ML classification models are points in their input data space $Z \subset \mathbb{R}^n$ where f changes output, *i.e.*., the inferred class. Given the continuity assumption behind most ML models, such points are located on hypersurfaces (manifolds) embedded in Z, also called *decision surfaces* (see the light blue surfaces in Fig. 2.5a). These partition the Z space into compact regions where the classifier has the same output, also called *decision zones*.

As described so far, projections $P(D)$ depict a *discrete* set of samples D, optionally color-coded to show the behavior of a ML model f. For the dataset D represented by the green points in Fig. 2.5a, this would yield the red-points scatterplot in Fig. 2.5b. Such images, however, do not explicitly show where decision boundaries are – we know that they occur somewhere between the red dots, but not where precisely. Depicting such boundaries, along the color-coded training and/or test sets of f, significantly improves the understanding of how f actually behaves. This can help ML engineers to find where in the input space more training samples are needed to improve a classifier or, conversely, assess in which such areas would samples be misclassified.

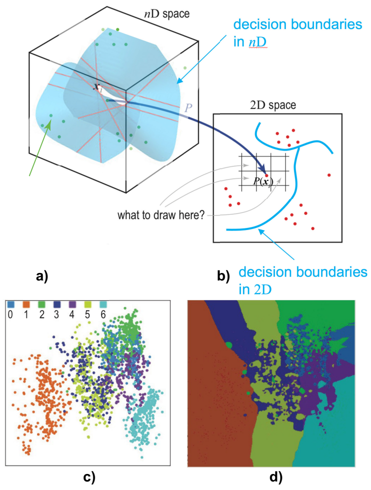


Figure 2.5: Decision boundary maps. (a) A high-dimensional dataset with its decision boundary hypersurfaces. (b) Projecting the samples (green) and decision boundaries (light blue) of this dataset yields the red 2D points, respectively light blue 2D curves. (c) Example of such a 2D projection with samples colored by the class inferred by a ML model. (d) The decision zones for this classifier are depicted in the 2D projection space as same-color areas. See Sec. 2.3.4.

2.3.4.1 *Basic idea of decision boundary maps*

Decision boundary maps (DBMs, sometimes called simply decision maps in various papers and also this thesis) are a visual representation for both decision zones and boundaries for any classifier (Schulz et al., 2015; Rodrigues et al., 2018). Intuitively put, DBMs aim to map the entire space Z (as classified by f) to 2D rather than the discrete sample set D, as follows. Given a training and/or test set D, a direct projection P is used to create a 2D embedding thereof. After that, given an image space $I \subset \mathbb{R}^2$, a mapping $P^{-1} : I \rightarrow \mathbb{R}^n$ is constructed to reverse the effects of P. We describe P^{-1} in detail further in Sec. 2.4.2. The mapping P^{-1} is then used to 'backproject' each pixel $\mathbf{p} \in I$ to a high-dimensional point $\mathbf{x} = P^{-1}(\mathbf{p})$, $\mathbf{x} \in Z$. Finally, each pixel $\mathbf{p}$ is colored by the label $f(\mathbf{x})$ assigned to it by the trained model to be explored. Same-color areas emerging in I indicate f's decision zones; pixels on the frontiers of these areas show f's decision boundaries. Figure 2.5d shows this for a KNN classifier trained to produce the test set depicted by the projection in Fig. 2.5 for a six-class problem.

The key to DBM construction is creating the mapping P^{-1} for a given direct projection P. In principle, any combination of P and P^{-1} can be used to construct a DBM for any given classifier by directly following the per-pixel procedure outlined above. However, earlier studies have shown that, for certain classification problems where one has ground-truth information about the expected outcomes – for example, in the sense of the shapes, sizes, and smoothness of the decision zones that a given classifier should create for that dataset – certain direct projections P and P^{-1} combinations work better (Rodrigues et al., 2019; Oliveira et al., 2022). We discuss these aspects separately in Sec. 2.4.2.

2.3.4.2 *Enhancements of basic DBMs*

DBMs can be further enhanced to encode, via brightness, the model's confidence at every 2D pixel (Figs. 2.6a,c) or the actual distance, in Z, to the closest decision boundary (Fig. 2.6b). The appearing brightness gradients tell which areas in the projection space are more prone to misclassifications. Importantly, this does not require actual training or test samples to exist in these areas – rather, such samples are synthesized by P^{-1}.

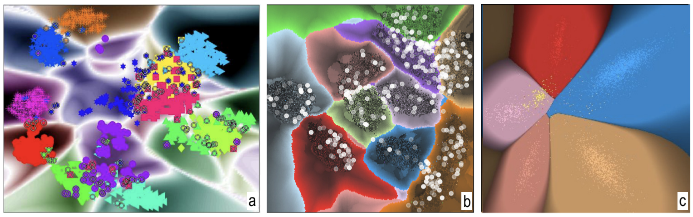


Figure 2.6: Decision boundary maps for classifier analysis with luminance encoding classifier confidence (a,c) (Schulz et al., 2015; Oliveira et al., 2022), respectively distance-to-decision-boundary (b) (Rodrigues et al., 2019). See Sec. 2.3.4.

Interpreting confidence or distance-enhanced DBMs is, however, not trivial, as illustrated next by the example in Fig. 2.7. Image (a) shows the MNIST digit dataset (LeCun et al., 2010) ($n = 782$ dimensions, $|C| = 10$ classes) projected to 2D by using t-SNE and classified by a neural network. Image (b) shows the DBMs for this problem. Image (c) enhances this by encoding the classifier confidence encoded into brightness (dark=lower confidence). For clarity, image (d) shows the confidence information separately (green=low confidence; yellow=high confidence). The images (c) and (d) convey the impression that the visualized classifier is highly, and equally, confident in all areas except very close to its decision boundaries.

Combining this information with the distance-to-closest-decision boundary reveals a different story. Image (e) shows this distance. In contrast to earlier techniques (Rodrigues et al., 2019) (Fig. 2.6b) which use expensive iterative-search in the high-dimensional space to locate, for each pixel $\mathbf{p}$, the distance from $\mathbf{x} = P^{-1}(\mathbf{p})$ to its closest decision boundary, Machado et al. (2024) proposed recently to create such images (e) by a simpler, and much faster approach. For each such point $\mathbf{x}$, they synthesize its closest adversarial example $\mathbf{a} \in Z$ and approximate the sought distance as $\|\mathbf{x} - \mathbf{a}\|$ using DeepFool (Moosavi-Dezfooli et al., 2016). This is orders of magnitude faster than iterative search and allows generating the desired distances in subsecond time on a commodity PC. Examining image (e), we see that the distance-to-boundary evolves very differently for the different decision zones and has complex patterns even in a single such zone, indicating that certain points are far closer to decision boundaries than others. For example, the red decision zone, although appearing very close to its neighbors in the raw projection (Fig. 2.7a, is quite bright, telling that it is farther away from its surrounding decision boundaries, than the other, darker, zones. Image (f) shows the same information, but with inverse brightness mapping than in (e). This highlights zones close to decision boundaries, *i.e.*, where the classifier may have trouble. We see, for example, a small yellow decision zone (marked by a white triangle). This zone, which is also disconnected from the other, larger, yellow decision zone (thus, for the same class), is very bright in image (e), indicating that it is very close to decision boundaries. This likely indicates potential model instabilities in this area.

To further explore this hypothesis, Machado et al. (2024) performed a simple experiment, as follows. They select ten pixels in the above-mentioned small yellow region, synthesize their corresponding data samples by P^{-1}, and add to them a wrong label – corresponding to the cyan color instead of the correct, yellow, label (see Fig. 2.7g, with the selected pixels marked in red). They next add these mislabeled points to the training set, re-train the model, and visualize its DBM. The result (Fig. 2.7h), shows how the small yellow region has become cyan, which is potentially not surprising given the newly added labels. More interestingly, however, we see large changes in decision zones of *different* classes adjacent to the yellow region: The dark-blue zone grows significantly to cut away a portion of the brown zone. This shows that few data changes in a small decision zone, potentially flagged

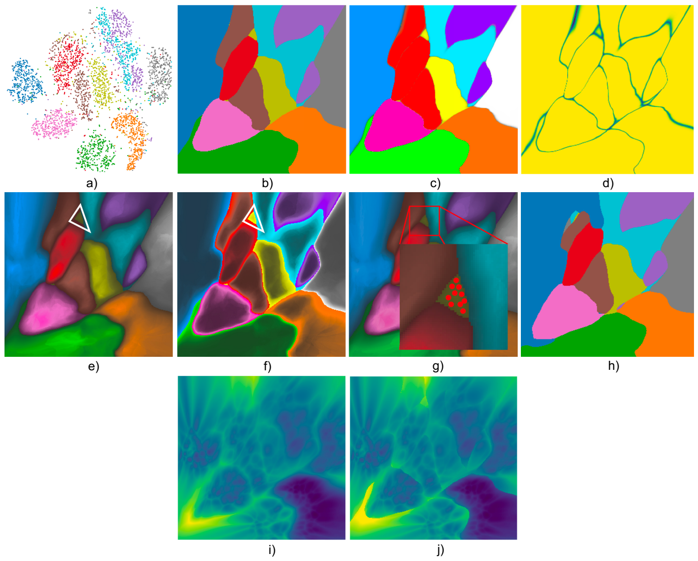


Figure 2.7: Explanatory visualizations for interpreting DBMs created by the technique of Machado et al. (2024). See Sec. 2.3.4.

by the DBM visualization as unstable, change indeed the overall behavior of the classifier even outside this zone.

Annotating a DBM with the classifier confidence and/or distance to closest decision boundary does not, however, reveal all information that characterizes different decision zones. Indeed, one additional such information involves how close the DBM points are to the actual training points that the classifier was constructed from. We illustrate the added-value of this information next. Images (c-f) in Fig. 2.7 show several large decision zones, *e.g.* the green and orange ones, which look quite similar from the perspective of confidence and distance to boundary – their inner pixels appear to be quite confident and far away from the surrounding decision boundaries. To gain more insight, Machado et al. (2024) visualize, for each pixel $\mathbf{p}$, the distance of its corresponding data sample to the closest training-set point, *i.e.*

$$d_D(\mathbf{p}) = \min_{\mathbf{x} \in D_t} \|P^{-1}(\mathbf{p}) - \mathbf{x}\|, \tag{2.4}$$

which we will also use in Chapters 4 and 7. Figure 2.7i shows the distance d_D for the MNIST classifier, with dark blue indicating small distances and yellow large ones, respectively. We immediately see that all pixels of the orange decision zone are very close to the training-set, whereas pixels in all other zones appear much farther away. This indicates non-linear behavior of the DBM construction algorithm – the visible sizes of the decision zones in the DBM do not indicate actual sizes in the data space. Differently put, the orange decision zone is much closer 'wrapped around' training-set points than the other zones. This indicates that, all other aspects being equal, one should have more trust in the classifier behavior in the orange zone, as its depicted points are much closer to the training set that the classifier was built from.

Additionally, we see in image (i) a bright yellow band at the bottom of the corresponding pink decision zone. This tells that points around this decision boundary (between the pink and green zone) are quite far away from *any* training-set point. As such, even if the confidence of the classifier appears quite high in this area, apart from points very close to the decision boundary (see image (d)), the classifier extrapolates much farther away from its training data here, so, it is more prone to errors. Note that one would expect the confidence to drop as the data points become further apart from the training set (intuitively, what the classifier learned from that training set is now 'stretched' to account for very different data), but this is not the case for this classifier. The visualizations show that purely relying on classifier confidence is not sufficient for users to gain enough understanding of what the classifier does in specific situations and, hence, whether they trust (or not) the classifier in those situations.

Besides the above, we see, within each decision zone, a varying color pattern consisting of dark 'cells' separated by slightly brighter bands. These indicate how the respective sub-areas in a decision zone have been created by samples in the training-set – much like the visualization of a Voronoi diagram whose sites are the training-set samples.

Figure 2.7j shows a final variation of the explanatory visualizations for DBMs. Here, instead of depicting the distance of a map pixel to the closest training-set point, Machado et al. (2024) show the distance to the closest training-set point of the same class as the pixel itself.

$$d_D^{sameclass}(\mathbf{p}) = \min_{\mathbf{x} \in D_t | f(\mathbf{x}) = f(P^{-1}(\mathbf{p}))} \|P^{-1}(\mathbf{p}) - \mathbf{x}\|. \tag{2.5}$$

The distance $d_D^{sameclass}$ shows how far away samples that map to a decision zone are from training-set samples that led to the creation of that zone in the model f. We see that image (j) is quite similar to image (i). This is a positive finding, as it tells that pixels in a decision zone correspond to data points which are close to the training samples for that zone, which is indeed what a good DBM should show. In the same time, we see that the contrast between the orange and green zones, visible as dark blue, respectively bright green in image (j), has increased. This tells that the decision boundary between the orange and green zones is far *closer* to the orange training samples than to the green ones – an insight which the basic confidence or distance-to-boundary maps do not reveal.

2.3.5 *Putting it all together: Visual analytics workflow*

At this point, we can further detail the general visual-analytics workflow of ML assistance by DR techniques introduced in Fig. 2.1 (green arrows). Figure 2.8 does this by refining the workflow for using DR to assist ML proposed earlier by Rauber *et al.* (Fig. 1 in Rauber et al. (2017b)) to include the DR-based techniques discussed above in this section – all which are novel in comparison with Rauber et al. (2017b)), apart from the classification ease analysis (see next). This gives a practical guideline on how to use the presented visualization techniques in practical ML engineering. In the following, numbers in the text indicate steps in the workflow Fig. 2.8.

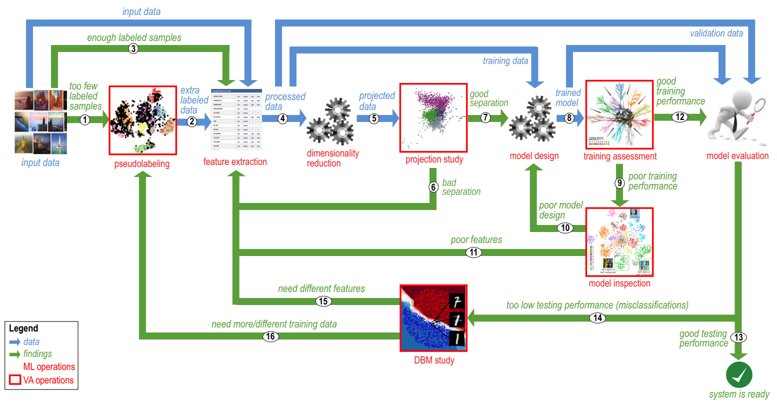


Figure 2.8: Workflow of using DR techniques to assist ML (see Fig. 2.1, green arrows). Visual analytics (VA) operations enabled by DR are marked by red-outlined boxes.

- The process starts by acquiring a dataset that one wants to further analyze, *e.g.* classify, using ML.
  - If not enough labeled samples are available (1), pseudolabeling (Sec. 2.3.2) can be used to create additional ones (2), else the process continues with the avail-

able data (3). Projections and decision maps can guide users during the pseudolabeling process.

- From these data, features are typically extracted by various processing operations (4).
- Next, DR is used to construct a projection (5) from the data. The projection is visually studied to assess whether the data form sufficiently separated classes to suggest a feasible classification problem (Sec. 2.3.1).
  - If so (7), a ML architecture is chosen to design and train a model. Else, the workflow reverts to extracting better features (6).
- DR-enabled techniques are next used to assess (8) whether the training performed well (Sec. 2.3.3).
  - If training is found unsatisfactory (9), the model is further inspected (Sec. 2.3.3) to find whether it has a poor design (10) or was fed by poor features (11).
    - In the former case, the model goes to redesign stage; in the latter, different features are extracted.
  - If the model's training was positively assessed (12), the flow continues with standard ML evaluation (testing).
    - Upon measuring satisfactory performance (13), the workflow ends with an operational model ready for use.
    - If testing performance is found too low (14), decision-map techniques (Sec. 2.3.4) can be used to find out whether different features (15) or if more (or different) training data is needed (16). In both such cases, the workflow continues from the respective earlier steps, as indicated in the figure.

2.4 LEARNING FOR SEEING: ML ASSISTS DR

Section 2.3 has shown several examples of how DR visualizations help in several use-cases of ML engineering. In this section, we outline the opposite path, *i.e.*, how ML techniques can be used to create DR visualizations so as to surpass limitations of existing DR algorithms. We discuss two classes of such methods for creating projections (Sec. 2.4.1), respectively inverse projections (Sec. 2.4.2).

2.4.1 *Deep learning projections*

Tens of DR techniques have been developed in the visualization community. However, choosing such a technique to apply in practice, for instance for the ML use-cases outlined in Sec. 2.3, is challenging, as few comparisons of such techniques exist following all desirable requirements listed in Sec. 2.2.2. A recent survey (Espadoto et al., 2019a) addressed this question at scale for the first time by comparing 44 projection techniques P over 19 datasets D from the perspective of 6 quality metrics M, using grid-search to explore the hyperparameter spaces of the projection techniques. Equally important, all its results – datasets, projection techniques, quality metric implementations, study protocol – are automated and freely available, much like similar endeavors in the ML arena. Following the survey's results, four projection methods consistently scored high on quality for all datasets, namely UMAP (McInnes et al., 2018), t-SNE (van der Maaten and Hinton, 2008), IDMAP (Minghim et al., 2006), and PBC (Paulovich and Minghim, 2006), with several others close to them. However, none of the top-ranked surveyed techniques also met the OOS, computational scalability, and stability criteria. As such, the survey concluded that better DR techniques are needed.

Following the analogy with ML regressors and given that such regressors meet the OOS, scalability, and stability criteria (Sec. 2.2.1), it becomes interesting to consider ML for building better projection algorithms. Autoencoders (Hinton and Salakhutdinov, 2006) do precisely that and meet all requirements in Sec. 2.2.1 except quality – the resulting projections have in general poorer trustworthiness and continuity than state-of-the-art methods like UMAP and t-SNE. Figure 2.9 illustrates this: The well-known MNIST dataset, which is well separable into its 10 classes by many ML techniques, appears, wrongly, poorly separated when projected by autoencoders. Following (Rauber et al., 2017b) (see also Sec. 2.3.1), we can conclude that autoencoders are a poor solution for DR.

2.4.1.1 *Basic idea of learning projections*

The idea of using deep learning to create an OOS projection is quite old. Pekalska et al. (1999) proposed to do this to approximate Sammon's mapping in a way that could be extended to approximate also other DR techniques. Autoencoders, mentioned earlier, are another early approach for the same task. More recently, Espadoto et al. (2020) proposed Neural Network Projections (NNP), a supervised approach to learning DR: Given any dataset D and its projection $P(D)$ computed by the user's technique of choice P, a simple three-layer fully-connected network is trained to learn to regress $P(D)$ when given D. Despite its simplicity, NNP can learn to imitate any projection technique P for any dataset D surprisingly well. While NNP's quality is typically slightly lower than state-of-the-art projections like t-SNE and UMAP, it is a *parametric* method, stable as proven by sensitivity analysis studies (Bredius et al., 2022), OOS, linear in the sample count N and dimensionality n (in practice, thousands of times faster than t-SNE), and very simple to implement.

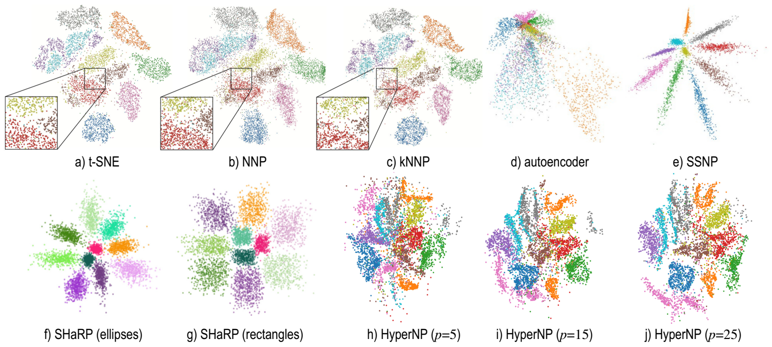


Figure 2.9: Projection of MNIST dataset with (a) t-SNE (van der Maaten and Hinton, 2008) and with deep learning methods: (b) NNP (Espadoto et al., 2020), (c) kNNP (Modrakowski et al., 2022), (d) autoencoders (Hinton and Salakhutdinov, 2006), (e) SSNP (Espadoto et al., 2021b), (f,g) SHaRP (Machado et al., 2023) with elliptic, respectively rectangular, cluster shapes, and (h-j) HyperNP (Appleby et al., 2021) imitating t-SNE for three different perplexity values p. See Sec. 2.4.

2.4.1.2 *OOS and sensitivity analysis*

As explained earlier, the OOS and sensitivity (stability to small changes of the input) are related, but not identical, desirable properties. We illustrate both properties for NNP next, noting also that all other similar deep-learned projection algorithms (kNNP, SNNP, SHaRP, autoencoders) share by construction the same properties, since they use very similar neural network architectures.

Figure 2.10 (top two rows) illustrates NNP's out-of-sample ability. The top row shows t-SNE projections of the MNIST dataset for an increasing number of samples (from 2K to 100K). As visible, the projection continuously changes, making it hard for users to maintain their mental map of the studied data. The row below shows NNP trained to mimic t-SNE. We see that the shape of the projection and location of its ten clusters (one per class) stays the same as more samples are added. A drawback of this is that the cluster separation is lower than for the t-SNE projection as more samples are added. This is expected since NNP was trained only on the initial set of 2K samples, *i.e.*, it did not have a chance to see the additional ones.

Figure 2.10 (bottom row) illustrates NNP's stability. An NNP model is trained to project the MNIST dataset, after which is asked to project MNIST images where an increasingly larger number of dimensions (pixel values) have been cancelled, *i.e.*, set to zero. Surprisingly, NNP can capture the cluster structure of the data (10 classes for the 10 digits) up to 40% cancelled dimensions. The aggregated image shows the 'movement' of the points in the NNP projection as increasingly more dimensions get dropped. Similar insights are obtained for other input-dataset perturbations such as sample jitter, translation, and scaling. At a higher level, we see sensitivity analysis as a very powerful, yet under-explored, technique – well known in the ML repertoire – to assess the quality of DR projections.

2.4.1.3 *Refinements of NNP*

Subsequent refinements of NNP aim to keep the attractive aspects of the method (speed, OOS, genericity, stability, simplicity) while increasing its *quality* and controlling the visual *appearance* of the resulting projections (see further Fig. 2.9). k-NNP (Modrakowski et al., 2022) enhances the projection quality, measured following the metrics introduced in Sec. 2.2, by learning to project sets of neighbor samples rather than individual samples. SSNP (Espadoto et al., 2021b) works in a self-supervised way, similar to autoencoders, thus dispenses of the need of a training projection. The self-supervised information comes either in the form of ground-truth labels (when available) or pseudolabels computed by clustering the input data. Since based on an autoencoder structure, SSNP can also create inverse projections (see next Sec. 2.4.2). SDR-NNP (Kim et al., 2022) increases NNP's ability to separate clusters of different observations by pre-sharpening the input training set D_t via mean shift (Comaniciu and Meer, 2002). SHaRP (Machado et al., 2023) refines SSNP to allow one to control the shapes of clusters of similar observations to match desired templates such as ellipses, rectangles, or triangles. Finally, HyperNP (Appleby et al., 2021) extends NNP by learning the behavior of a projection technique P for all its hyperparameter values, thereby allowing users to explore this parameter space at interactive rates. All the above results prove that DL is a serious contender for generating projections that comply with all requirements set forth by practice.

We will use deep-learned projections at several points in our work such as when quantitatively comparing decision maps created using such techniques (Chapter 4).

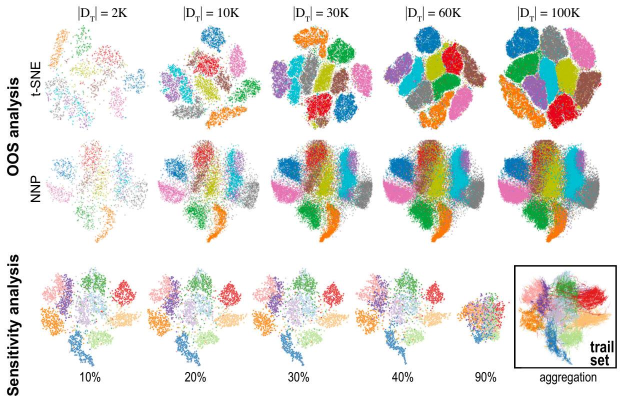


Figure 2.10: NNP out-of-sample (OOS) analysis. Image taken from (Bredius et al., 2022). The top row shows projections of the MNIST dataset using t-SNE for increasing numbers of samples in the test set D_T. The projection does not maintain stability. The second row shows how NNP maintains stability as new samples are added to the test set. Bottom row: NNP sensitivity analysis when removing between 10% and 90% of the dataset's dimensions. NNP can robustly depict the data structure even when a large part of the input information is missing. See Sec. 2.4.1.

2.4.2 *Inverse projections*

Following the success of DL for constructing projections P outlined above, it becomes immediately interesting to consider their use for the complementary problem of computing *inverse* projections P^{-1}. Introduced in Sec. 2.3.4 for constructing DBMs, inverse projections have additional uses, *e.g.*, generating synthetic samples for data augmentation scenarios (Rodrigues et al., 2018) and hypothesizing the unexplored regions of a sampled data space for *e.g.* shape or image morphing applications (dos Santos Amorim et al., 2012).

Definition: Formally put, given a direct projection function $P : \mathbb{R}^n \rightarrow \mathbb{R}^q$, with $q \ll n$ typically, the inverse projection is just the inverse of that function, *i.e.*, a function

$$P^{-1} : \mathbb{R}^q \rightarrow \mathbb{R}^n \tag{2.6}$$

so that $P^{-1}(P(\mathbf{x})) = \mathbf{x}$ for any $\mathbf{x} \in \mathbb{R}^n$. Unfortunately, computing such an exact inverse function is not possible for most of the existing projection algorithms P for one or several of the following reasons:

- *non-injectivity*: Typical projection functions P are not injective – that is, they can map different points in $\mathbb{R}^n$ to the same location in $\mathbb{R}^q$. This is a direct, and likely unavoidable in general, consequence of the fact that $q \ll n$;
- *non-parametric nature:* While we are talking about P as being a *function* between two spaces ($\mathbb{R}^n$ and $\mathbb{R}^q$), many projection algorithms do not work in this fashion. Rather, they map a given *sampling*, or dataset, $D \subset \mathbb{R}^n$ to another dataset $P(D) \subset \mathbb{R}^q$ (see Eqn. 2.2). When the dataset D changes, the mapping can change as well – that is, the same point $\mathbf{x} \in \mathbb{R}^n$ can be mapped to different locations in $\mathbb{R}^q$, depending on which other points it comes along with in the new D. This is precisely the lack of OOS ability discussed in Sec. 2.4.1 and illustrated, for t-SNE, in Fig. 2.10. Only a (small) subset of projection techniques are *parametric*, *i.e.*, comply with the functional definition given above (van der Maaten, 2009; Jolliffe, 2002);
- *inverse problem:* Even for cases when an algorithmic functional definition of P is available and P is injective, computing its inverse can be quite challenging since inverse problems do not always have guaranteed unique and/or stable solutions.

To address the above three problems, the practical approach to computing P^{-1} is by various forms of approximation, which share one or several of the following characteristics:

- *non-injectivity:* This aspect is usually neglected by practical algorithms that compute P^{-1}. That is, if two (or more) points $\mathbf{x}_i \in \mathbb{R}^n$ map to the same location $\mathbf{p} \in \mathbb{R}^q$, $P^{-1}(\mathbf{p})$ will yield a single value in $\mathbb{R}^n$;

- *non-parametric nature:* Since most projections P are of this nature, inverse projections are usually defined in terms of a given dataset $D \subset \mathbb{R}^n$. For such a dataset, an inverse projection is a function that aims to yield $P^{-1}(P(\mathbf{x})) \approx \mathbf{x}$ for all $\mathbf{x} \in D$. However, inverse projections *need* to be parametric themselves, *i.e.*, applicable to any other values $\mathbf{p} \in \mathbb{R}^q$ apart from $P(D)$, otherwise they would not be useful for the tasks mentioned earlier in this chapter – in particular, for the construction of decision maps. For such 'unseen' points $\mathbf{p}$, P^{-1} is usually defined to work in a *smooth* manner – that is, the closest $\mathbf{p}$ is to a known $P(\mathbf{x})$ value, the closest should $P^{-1}(\mathbf{p})$ be to $\mathbf{x}$.

- *inverse problem:* Computing P^{-1} is usually done by minimizing suitably-chosen (convex) error functions that model the goal $P^{-1}(P(\mathbf{x})) \approx \mathbf{x}$ for all $\mathbf{x} \in D$ mentioned above. This simplifies and accelerates the computation problem by using existing numerical optimization methods.

2.4.2.1 *Early methods for computing inverse projections*

Likely one of the earliest methods for computing inverse projections is by training an autoencoder to jointly perform P and P^{-1} (Hinton and Salakhutdinov, 2006). While this method is simple to implement and fast to compute, it does not allow inverting any user-chosen direct projection P. Moreover, projections created by autoencoders are of lower quality than other state-of-the-art techniques (see Fig. 2.9 and related text and also the evaluation in (Espadoto et al., 2019a)).

Mamani et al. (2013) presented a method that uses inverse projections to transform the high-dimensional data space based on manipulations of the 2D projection space. This allows users to execute several potentially complex operations that affect the invisible high-dimensional data based on a simple interface that allows direct manipulation of 2D projection points. However, the proposed method does not directly allow inversely projecting new points – that is, 2D points which are not the direct projection of existing data points.

iLAMP (dos Santos Amorim et al., 2012) uses local information in $P(D)$ to build affine transformations that map $\mathbb{R}^2$ to $\mathbb{R}^n$. Although iLAMP was proposed to reverse the LAMP projection technique (Joia et al., 2011), it can be used to reverse other projections P with reasonable results (Rodrigues et al., 2018; Espadoto et al., 2019c). The same authors next proposed an inverse projection method using Radial Basis Functions (RBFs) to gain continuity and global behavior.

Schulz et al. (2015, 2020) proposed DeepView to compute inverse projections using UMAP (McInnes et al., 2018) as direct projection P. In contrast to UMAP, however, similarities of points are computed by combining their high-dimensional attributes with the outcome of a classifier f (similar to SSNP). The inverse projection is then computed also by UMAP from the projection of the given dataset $P(D)$ and then extrapolated to the entire 2D space by minimizing the Kullback-Leibler divergence (similar to t-SNE). DeepView yields quite smooth results (see its application for DBM construction in Fig. 2.6a) but is over an order of magnitude slower than all other P^{-1} techniques described here.

More recently, Blumberg et al. (2024) proposed iMDS to invert MDS projections using multilateration with randomly selected samples to estimate the inverse projection. While this method is simple to implement, it only gives good results for datasets having a quite low intrinsic dimensionality (under 10) and can only invert MDS. Although this method is quite recent, we include it in this section which discusses early inverse projections since it is, fundamentally, sharing the same characteristics as all other methods which do not use the more recent deep-learning approach which are discussed in the next section.

2.4.2.2 *Deep learning inverse projections*

Following the success of NNP for direct projections (Sec. 2.4.1), Espadoto et al. (2019c) computed inverse projections by simply 'switching' the input and output of NNP, *i.e.*, given a dataset D that projects to a 2D scatterplot by some technique P, train a regressor to output D when given $P(D)$. This technique, called NNInv inherits all the desirable properties of NNP (Sec. 2.4.1) and also produces higher-quality inverse projections than autoencoders and iLAMP. The usage of NNInv is illustrated in the DBM construction in Fig. 2.6b. Further variations of this design include SSNP (Espadoto et al., 2021b) (used to construct the DBM in Fig. 2.6c and SHaRP (Machado et al., 2023)). Both techniques use an autoencoder basis so produce both a direct and inverse projection (see also Sec. 2.4.1). However, their quality is higher than plain autoencoders and also than plain NNInv given their (self-)supervised operation based on class (pseudo)labels. Figure 2.11 illustrates this by comparing NNinv (first row) with SHaRP (second and third rows) for the construction of a decision map for a simple k-nearest neighbors (KNN) classifier ($k = 21$) for four datasets of varying dimensionalities n. This is a novel insight as SHaRP has, so far, not been gauged on its performance for computing decision maps. We see that, similarly to SSNP (shown in Fig. 2.6c), SHaRP produces decision zones with smoother boundaries than NNInv, which are closer to the known ground-truth smooth boundaries (hyperplanes) that a KNN classifier should have.

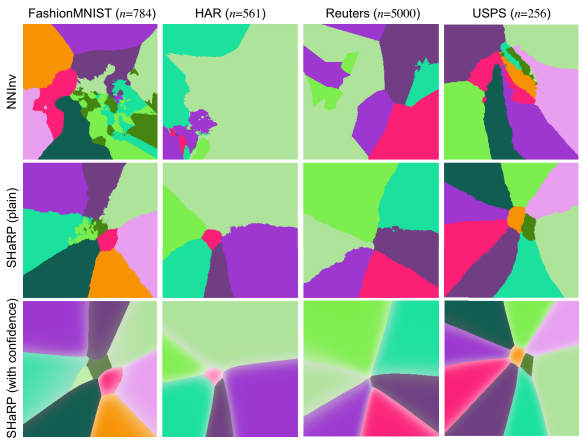


Figure 2.11: Comparison of decision maps constructed by NNinv (Espadoto et al., 2019c) (top row) and SHaRP (Machado et al., 2023) (middle row: plain map; bottom row: map with classifier confidence encoded into color saturation) inverse-projection techniques, for a KNN classifier and four datasets (n indicates the dataset dimensionalities). See Sec. 2.4.2.

2.4.2.3 *Applications of inverse projections*

NNInv was further explored in detail for visual analytics scenarios involving dynamic imputation and exploring ensemble classifiers (Espadoto et al., 2021a). Figure 2.12 shows the latter use-case: In the image, each pixel is backprojected and ran through a set of nine classifiers, trained to separate classes 1 and 7 from the MNIST dataset. The pixel is then colored to indicate the classifiers' agreement. Deep blue, respectively red, zones show areas where all classifiers agree with class 1, respectively 7. Brighter areas indicate regions of high classifier disagreement – which are thus highly difficult to decide upon and are prime candidates for ML engineering, regardless of the used classifier.

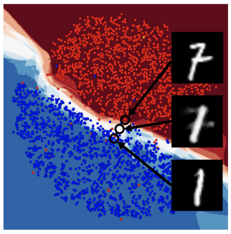


Figure 2.12: Classifier agreement map for 9 classifiers, two-class problem (MNIST datasets digits 1 and 7). Dark colors indicate more of the 9 classifiers agreeing, at a pixel in the map, with their decisions (red=1, blue=7). Brighter, desaturated, colors indicate fewer classifiers in agreement (white=four classifiers output 1, the other five output 7, or conversely). Image taken from (Espadoto et al., 2021a). See Sec. 2.3.4.

2.5 FUTURE EXPLOITATIONS OF THE ML-DR CONNECTION

Reflecting upon the current achievements of using ML for DR and conversely, we see a bright future ahead for research where the two directions assist each other. We illustrate this with a few selected, non-exhaustive, examples of such potential ML-DR synergies. Moreover, we connect these examples to concrete cases where we leverage this synergy in our own work on decision maps in the next chapters.

2.5.1 *Prospects of DR assisting ML: Seeing to learn better*

2.5.1.1 *DBMs in use*

For researchers in various fields, DBMs can provide insights into the behavior of classifiers in their respective applications, thereby helping them to understand and interpret the results of their ML models. For machine learning engineers, they offer a visual and informative way to refine models.

Apart from the scenarios depicted in (Espadoto et al., 2021a; dos Santos Amorim et al., 2012; Amorim et al., 2015), DBMs could be readily used in a visual analytics explorative scenario to drive a classifier's training. If computable in real-time, users could visualize the DBMs, find problematic areas with respect to how the decision boundaries wrap around samples, and next modify the training set by *e.g.* adding or deleting labels, adding new augmented samples, or even moving samples. We envisage a tool in which users could effectively 'sculpt' the shape of decision boundaries by sample manipulation much as one edits 2D shapes by manipulating spline control points. This would offer unprecedented freedom and a wholly new way of fine-tuning classifiers to extend the approaches pioneered in Benato et al. (2018, 2021).

This thesis contributes to this topic from two aspects: First, Chapter 3 presents an application of decision maps to understand classification models constructed for geoscience research, which demonstrates how decision maps can be used to help research in other fields in practice. Secondly, Chapter 7 presents a novel method to accelerate the construction of decision maps (and other maps related to inverse projections). Our method can compute such maps interactive rates, which is a prerequisite for the interactive exploration mentioned above. An early version of our acceleration technique has been used in a practical application, namely for pseudo-labeling high dimensional datasets to improve classification performance (Benato et al., 2024).

2.5.1.2 *Visualizing regressors*

All visualizations examples shown in this chapter have covered only the depiction of classifiers that output a single categorical value. However, as Sec. 2.2 mentions, ML also studies multi-valued classifiers and, further, single-valued and multi-valued regressors. Concerning decision maps, we are not aware of their extension to multi-valued classifiers. This could be achieved by using multiple-view maps, one per classifier output, or categorical color-coding of all multi-valued class combinations in a single decision map. Concerning regressors, recent results have shown how to extend the decision map metaphor to visualize single-valued regressors (Espadoto et al., 2021c, 2023). However, this research only used a relatively low-quality projection (PCA), so it could be readily explored how better direct and inverse projections, like the ones described in Secs. 2.4.1 and 2.4.2 could improve its results. Visualizing multi-valued regressors is a harder problem as several continuous values would need to be displayed at each pixel. To assist this, techniques developed earlier in scientific visualization, such as tensor visualization (Weickert and Hagen, 2005), could offer an outcome.

We leave this challenge of visualizing general regressors open in our work in this thesis.

2.5.2 *Prospects of ML assisting DR: Learning to see better*

2.5.2.1 *Inverse projection quality*

While many metrics exist to gauge the quality of direct projections (Sec. 2.2), there are no established ways to measure the quality of an inverse projection, apart from the simple mean-square-error (MSE) $\sum_{\mathbf{x}\in D} \|\mathbf{x} - P(P^{-1}(\mathbf{x}))\|$ (Espadoto et al., 2019c). This is not surprising since, as explained in Sec. 2.4.2, inverse projections are mainly used to infer, or hypothesize, what the data would be in locations where no ground-truth is present. As such, defining what a good inverse projection should return in such areas is conceptually hard. Yet, possibilities exist. One can *e.g.* use a ML approach where an unseen test set is kept apart from the construction of the inverse projection and is used to assess the quality of such a trained model using the aforementioned MSE. An equally interesting question is how to design a scale, or hierarchy, of errors. It is likely that differently inversely-projected points $\mathbf{x}' = P^{-1}(P(\mathbf{x}))$ that deviate from its ground-truth location $\mathbf{x}$ by the same distance $\|\mathbf{x}' - \mathbf{x}\|$ are not equally good, or equally bad, depending on the application perspective. As such, inverse projection quality metrics may need to be designed in an application-specific way.

Similarly to direct projections (Espadoto et al., 2019a), the quality of inverse projections can be measured not only globally (by a single aggregate metric) but also locally, at every pixel. The explanatory visualizations in Fig. 2.7 can be thought as being such per-pixel quality maps (for classifiers). For inverse projections, we are aware of a single such per-pixel quality visualization – gradient maps (Espadoto et al., 2021a). Figure 2.13a shows this gradient map, which depicts the gradient magnitude of the P^{-1} function (in this case constructed with NNInv) at every pixel. Hot, respectively dark, regions in the map indicate nearby 2D points which backproject far away from, respectively close to, each other. Points in the hot regions thus indicate areas where the inverse projection may be unstable, and as such, potentially create misleading data. However, one cannot directly say that this is an error of the inverse projection P^{-1}. Such regions may correspond to areas where the *direct* projection P squeezed faraway data points to fit them in the 2D space – thus areas of low continuity (Venna and Kaski, 2006). Hence, analyzing inverse projection errors should go hand-in-hand with analyzing the errors of the direct projection it was computed for. For the latter, many per-pixel techniques are readily usable (Aupetit, 2007; Lespinats and

Aupetit, 2011; Martins et al., 2014). We will further use gradient maps in evaluating our results in Chapters 4 and 6.

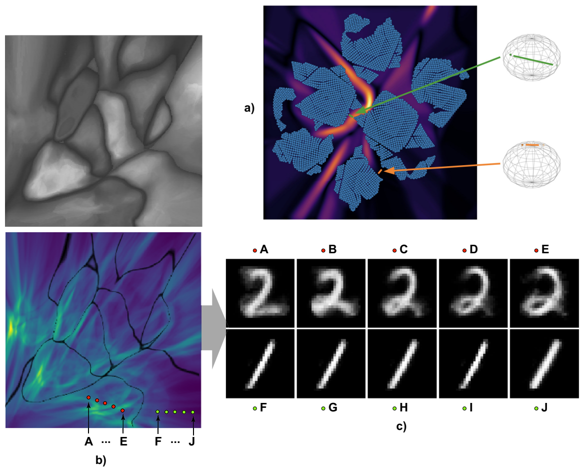


Figure 2.13: (a) Gradient map of NNInv inverse projection constructed from a t-SNE projection of an uniformly sampled sphere. Hot, respectively dark, regions indicate nearby 2D points that inversely project to far-apart, respectively close, nD points (green line, top sphere; orange line, bottom sphere, respectively). (b) Gradient map of NNInv inverse projection used to construct the decision maps for the MNIST classification in Fig. 2.7, with distance-to-closest-boundary map at the top (grayscale). (c) Two regions of large, respectively low, gradients are sampled by the red, respectively green, points. The corresponding images generated by NNInv are shown and confirm the large, respectively low, variations of the inverse projection in these areas. See Sec. 2.5.2.

Figure 2.13b shows an additional use-case for gradient maps. The image depicts the gradient map of the NNInv inverse-projection method used to construct the decision map visualizations for the MNIST classifier explored in Fig. 2.7. Atop of this gradient map, we overlaid the classifier confidence (Fig. 2.7d), so the dark bands in the image correspond to the classifier's decision boundaries. For clarity of exposition, we show atop image (b) the distance-to-closest-boundary, *i.e.*., the same information as encoded in the luminance in Fig. 2.7e. Image (b) gives us several insights. We see that large inverse-projection gradients occur both along decision boundaries but also deep inside the decision zones. Also, these large gradients are not correlated with areas of low, or high, distance-to-closest boundary. Hence, the gradient map tells additional information not present in earlier visualizations. This information helps seeing where a classifier will be exposed to high data variability, thus, meet more challenges. We show this by taking five points ($A \dots E$) in a low-gradient, and five others ($F \dots J$) in a high-gradient area, respectively. Fig. 2.13c shows the MNIST images corresponding to these points. Indeed, we see how the respective digits vary significantly more in high-gradient areas than in low-gradient ones.

Another aspect related to inverse projection quality is the *coverage* of an inverse projection function. Simply put: Consider an image space $I = \{\mathbf{p}\}$, $I \subset \mathbb{R}^2$ as a set of pixels $\mathbf{p}$, and an inverse projection function $P^{-1} : \mathbb{R}^2 \rightarrow \mathbb{R}^n$. Let $I^{-1} = \{P^{-1}(\mathbf{p}) | \mathbf{p} \in I\}$ be the mapping of the entire pixel image I to the data space via P^{-1}. Current decision maps only 'sample' the data space by examining I^{-1}. The question is: How well does I^{-1} cover, or represent, the entire data space Z on which some machine-learning model is supposed to work?

We will describe a comprehensive quantitative and qualitative evaluation of the decision maps in Chapter 4 which addresses the inverse projection quality aspect discussed in this section. Separately, we will further investigate the coverage of inverse projections in Chapter 5.

2.5.2.2 *Increasing user control*

Clearly, both direct and inverse projection operations can be defined, and applied, by only having an input present in the form of a high-dimensional dataset, respectively a 2D scatterplot. However, in many cases, this lack of controlling how the direct and/or inverse projection actually work can lead to suboptimal results. Typical ways to address this are of offer various hyperparameters to control the direct and/or inverse projection (Appleby et al., 2021). However, controlling such hyperparameters is not always easy or intuitive for actual users (Wattenberg et al., 2016). Several other mechanisms for control exist and could be further exploited, as follows.

All projection methods aim to encode the relative distance between data points in their resulting scatterplot. Atop of this, parametric projections aim to encode the actual data values. SHaRP extends this to force data clusters to specific shapes (Sec. 2.4.1). Such strategies could be extended to map other data attributes, such as sample density or specific value ranges, to the size, shape, and/or position of point clusters in a projection. For DL methods, this could be done by refining their loss function. Additionally, SHaRP could be extended to create a hierarchy-aware projection algorithm that would combine the advantages of treemaps and classical projections, extending earlier ideas in this class (Duarte et al., 2014). All in all, projection techniques can be extended to take more properties of the input data into account when computing the output 2D scatterplot than pairwise point distances. We will not explore this direction in our work.

A second extension would be to design *local* cost functions that attempt to construct the projection by combining different criteria for different subsets of the input data – for example, to achieve a globally-better projection that locally behaves like t-SNE in some areas and like UMAP in others. ML techniques can help here by *e.g.* extending the HyperNP idea (Appleby et al., 2021) to train from a set of projection techniques run on the same input dataset. Further inspiration can be gotten from recent ways in which DL is used for image synthesis and style transfer, *e.g.*, Luan et al. (2017). We will also not explore this direction further.

Finally, one can control the way that inverse projections work. Indeed, as mentioned in Sec. 2.5.2.1, current inverse projections simply map a given 2D image, discretized as a set of pixels, to the high-dimensional data space, without any user control. It is very likely that this simple construction will not be able to cover all the high-dimensional data space equally well. As such, it is interesting to consider ways to control how the inverse projection samples this data space so that regions of higher interest, for any given application, are offered a higher chance to show up in the 2D representation – for example, in a decision map. We will present such a mechanism that allows users to control the inverse projection intuitively and interactively in Chapter 6, with the help of deep learning.

2.5.2.3 *Dynamic projections*

Section 2.3.3 has briefly introduced dynamic projections. These are extensions of the standard, static, projection techniques which aim to handle a dataset consisting of high-dimensional points which maintain their identity while changing their attribute values through time. Dynamic projections have a wealth of applications – simply put, anywhere one wants to study high-dimensional data which changes over time. However, only a handful of dynamic projection techniques exist (Rauber et al., 2016; Vernier et al., 2021, 2020; Neves et al., 2022), and their quality – as gauged by established quality metrics – is good in data structure preservation *or* data dynamics preservation but not both aspects. Designing a dynamic projection technique that accurately maps both data structure and dynamics is a grand challenge for the infovis community. Following the good results of using ML for DR (Sec. 2.4), it looks highly interesting to explore ML (and in particular DL) to create dynamic projections. An issue here is that, since good ground-truth dynamic projections are relatively hard to construct, the supervised way (NNP-class methods) may be less preferable than the self-supervised (SSNP-like) direction.

We leave this challenge open in our work in this thesis.

2.6 CONCLUSIONS

In this chapter, we have presented an overview of recent connections between the two fields, with a focus on techniques and methods in one field which assist tasks and use-cases in the other, and also satisfy overal desirable criteria as genericity, computational scalability, stability, and ease of use. We have made the case that the two fields are complementary, with key features being offered by methods in one field being required by methods in the other, therefore the potential for cross-fertilization. The first part of our overview (Sec. 2.3) showed how DR can assist ML tasks by examples in assessing the behavior of general-purpose classifiers, pseudolabeling for creating large training-sets, exploring the training and inference of deep learning models, and depicting the high-dimensional decision zones and boundaries of classifiers. The second part (Sec. 2.4) showed how ML can assist DR by examples covering the deep learning of projections and inverse projections.

In the third part (Sec. 2.5), we have outlined several high-potential research directions at the crossroads of ML and DR based on the techniques discussed in this chapter, and we have shown which of these directions we will further explore in the context of this thesis.

Decision maps are an important tool of the use-case where DR can assist ML as they help exploring the behavior of a trained classifier. Conversely, decision maps rely on direct and inverse projections which can be improved in several ways by using ML. In the remainder of this thesis, we will explore both these directions and propose improvements for both use-cases.

3 APPLICATIONS OF DECISION MAPS IN GEOSCIENCE

3.1 INTRODUCTION

To date, decision maps have not been adapted or used to actually assist in solving scientific problems. As Oliveira et al. (2022) remark, studies are needed to show how users actually interpret such maps to extract information on the visualized classification problems. In this chapter, we fill this gap by presenting a case study where we apply decision maps to a geological problem – the classification of mineral deposit genetic environments. We show how decision maps can be used to interpret the machine learning classification results and how they can be used to guide the exploration of the data in a *practical* setting. This chapter contributes indirectly to our research questions **RQ1** – **RQ4** listed in Chapter 1 by (1) making the case that decision maps are indeed useful instruments in classifier engineering in a real-world setting; and (2) outlining several limitations of decision maps which we discovered in this setting, which further justify our choices for exploring **RQ1** – **RQ4** in the next chapters[1].

The accelerating pace of data generation and computational power, coupled with the burgeoning interest of geoscientists in machine learning, is leading to significant breakthroughs in the applications and discoveries in geosciences (Petrelli and Perugini, 2016; Bergen et al., 2019; Karpatne et al., 2019; Petrelli, 2021). The data-driven study in geosciences essentially aims at digging deep information from complex/huge data sets, rather than merely and simply producing classification or prediction models. The 'black box' nature of machine models, however, hinders our understanding of decision-making processes during machine learning (Lipton, 2018; Carvalho et al., 2019; Molnar, 2020). Although pioneering explorations on the transparency of the working pathway of machine learning have emphasized the significance of the interpretability machine learning model (Lipton, 2018; Carvalho et al., 2019; Molnar, 2020; Yuan et al., 2021), such work is lacking in the classification of mineral deposit genetic environments.

1 This chapter is based on the paper: "Interpreting mineral deposit genesis classification with decision maps: a case study using pyrite trace elements" (Wang et al., 2024).

Understanding the mineral deposit genetic environments is important to explore the physio-chemical conditions that are responsible for the ore formation (Rusk, 2012). To improve the precision of the ore deposit classification environment, with a transparent and interpretable machine learning approach, we apply decision maps.

We underscore the potential of decision maps within a fundamental geological domain: the genesis of mineral deposits. The dwindling supply of near-surface ore deposits necessitates deeper exploration (Gregory et al., 2019). The ability to recognize the type of mineralization present in a given context can offer critical insights, thus streamlining exploration efforts and minimizing associated costs (Gregory et al., 2019). Trace elements measured in specific minerals, such as quartz, pyrite, apatite, and zircon, can serve as unique identifiers for understanding their genesis, revealing types of minerals deposits and host rock genetic environments (Belousova et al., 2002b; Rusk, 2012; O'Sullivan et al., 2020; Wang et al., 2021; Zhong et al., 2021a; Zhu et al., 2022; Zhou et al., 2023).

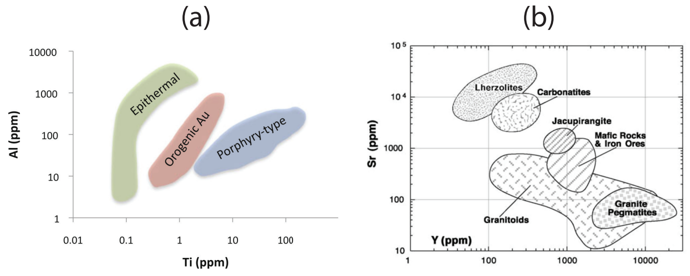


Figure 3.1: Discriminant diagrams for minerals using their trace elements concentrations (a) Ti versus Al diagrams for quartz (Rusk, 2012). (b) Sr versus Y diagram for apatite (Belousova et al., 2002a).

Classification of mineral deposits environments has traditionally been studied using visual tools, including discriminant diagrams (Fig 3.1) (Pearce and Cann, 1973; Bralia et al., 1979; Belousova et al., 2002a; Rusk, 2012; Li et al., 2019; Breiter et al., 2020; Zhou et al., 2022), and, more recently, machine learning-assisted approaches (Belousova et al., 2002a,b; Petrelli and Perugini, 2016; Gregory et al., 2019; Wang et al., 2021; Zhong et al., 2021a; Liu et al., 2023). However, striking a balance between visual interpretability and accuracy is still a challenge. There have also been some attempts of using machine learning to optimize geochemistry discriminant diagrams (O'Sullivan et al., 2020; Wang et al., 2022b). Such applications improve the quality of the patterns depicted by the diagrams but still do not take full advantage of high-dimensional information. Here, decision maps come to the fore, combining the high-accuracy of machine learning models with visual accessibility to decision boundaries, greatly promoting transparency and interpretability. This study represents the first application of visualization to elucidate machine learning classification in mineral deposit genetic types, highlighting the paramount role of visualization techniques in modern data interpretation and decision-making.

Here, our contributions straddle both information visualization and mineralogy domains: (1) We offer a unique pyrite trace elements dataset comprising six genetic populations. (2) We illuminate the added value of the decision map technique in deciphering the machine learning classification results, opening up new avenues for using decision maps. (3) We introduce a method that seamlessly blends the merits of traditional 2D discriminant diagrams (visual interpretability) and machine learning methods (high accuracy), providing a robust framework for mineral genesis classification problems. This blend of visualization and machine learning underlines the evolving landscape of data science, championing transparency and interpretability.

3.2 RELATED WORK

We divide our discussion of related work into two main topics – traditional mineral genetic discriminant diagrams (3.2.1) and machine learning for mineral genetic type classification (3.2.2). For each discussed topic, we outline the main advantages and disadvantages, in support of our overall claim of creating a new technique with decision maps that optimizes across existing approaches.

3.2.1 *Traditional trace element discriminant diagrams*

Trace element *discriminant diagrams* were introduced in the 1970s and are still widely used as a tool for identifying the types of deposit/host rock/tectonic setting with which a sample is associated (Belousova et al., 2002a,b; Bralia et al., 1979; Breiter et al., 2020; Pearce and Cann, 1973; Rusk, 2012). Discriminant diagrams are basically classical scatterplots that use only a few elements (data dimensions) plotted as binary or ternary diagrams. The axes of the diagrams, *i.e.*, data dimensions to explore, are usually selected based on the geologists' experience. For example, diagrams of Co *vs* Ni and Au *vs* As are widely used to discriminate the type of pyrite (Bajwah et al., 1987; Bralia et al., 1979; Deditius et al., 2014); Ti, Al, Ge are common axes used in quartz forming environment classification (Fig. 3.1a) (Breiter et al., 2020; Rusk, 2012); finally, Sr, Y, Mn and rare earth elements (REE) of apatite are useful axes for recognizing the apatite's host rock type (Fig. 3.1b) (Belousova et al., 2002a).

Advantages: The most important property of discriminant diagrams is that they offer a direct, visual, insight into how the depicted data dimensions relate to each other. The information that geologists can get from such diagrams includes labels of the depicted data samples and, at a higher level, trends, correlations, and outliers of the depicted elements and labels, which next help interpreting the phenomenon captured by the plotted data. In some cases, information – such as mineralization stages within a deposit, which often involves sequential patterns or transitions – cannot be easily conveyed to machine learning. Visualizing how such data points scatter in the diagram helps scientists understand data evolution. Based on domain knowledge, scientists can explore and get valuable understanding from such diagrams.

Disadvantages: The key shortcoming of discriminant diagrams is their inability to fully use all high-dimensional information available in the studied dataset. Only two elements (or ratios of a few more elements) can be depicted. These dimensions have to be hand-picked by geologists based on their experience, which means that potentially interesting, but unknown, data patterns present in other dimensions will not be found (Petrelli and Perugini, 2016).In addition, when classifying more complicated cases, the diagrams are usually heavily overlapped (Rottier and Casanova, 2020).

There are also some examples of using machine learning to optimize X-Y geochemistry diagrams (Hu et al., 2022; O'Sullivan et al., 2020; Wang et al., 2022b). Such applications improve the quality of the patterns depicted by the diagrams but still do not take full advantage of high-dimensional information.

Such limitations of low-dimensional diagrams are well-known in information visualization. Several partial solutions have been proposed to address them. Arguably the simplest and most used solution is to plot several diagrams side-by-side, a technique known as *small multiples* (Tufte, 1983). Users can then extract insights involving multiple attributes by visually comparing such diagrams. However, this solution scales poorly with the number of involved dimensions and requires careful ordering of the diagrams in the plotted sequence for efficient visual inspection. Another class of methods generically known as *scagnostics* (Wilkinson et al., 2005) computes all possible diagrams from a given set of data dimensions and next selects "interesting" diagrams to be shown to the user based on predefined data patterns. In contrast to small multiples, this selection scales visually well with the number of dimensions. However, computing and analyzing all possible diagrams is computationally very demanding. More importantly, defining and detecting what "interesting" patterns are is a challenging problem.

Discriminant diagrams have an additional important limitation: As they only plot a *sparse* set of observations (samples), users have to mentally group these visually into clusters and decide by themselves where actual "boundaries" exist that separate different phenomena captured by the data. Doing this can be challenging especially in cases when the depicted scatterplots do not show clear-cut visual separation between point clusters. As we shall see, DBMs compute and depict such boundaries explicitly, thereby significantly easing the analyst's task.

3.2.2 *Machine learning classifiers for mineral genetic type classification*

Machine learning is the emerging approach to solving geochemistry data classification problems (Gregory et al., 2019; O'Sullivan et al., 2020; Petrelli and Perugini, 2016; Wang et al., 2021).

We start by recapping a few notations introduced earlier in Sec. 2.2. Let $D = \mathbf{x}_i \subset \mathbb{R}^n$, $1 \leq i \leq N$, be a dataset of n-dimensional data points $x_i = (x_i^1, \ldots, x_i^n)$ with corresponding labels $y_i \in \mathrm{C}$, where C is the set of classes. Given a dataset D, a machine learning classifier constructs a function $f : \mathbb{R}^n \to C$ so that $f(\mathbf{x}_) = \mathbf{y}_i$ for ideally all $\mathbf{x}_i \in D_t$, where $D_t \subseteq D$ is so the called training set. After training, one uses the model f to infer labels of unseen points $\mathbf{x}_i$.

In the present work, we considered several machine learning classification algorithms, including Logistic Regression (Cox, 1958), Support Vector Machines (SVM) (Cortes and Vapnik, 1995), Random Forests (Breiman, 2001), and Neural Networks. These represent distinct families of algorithms: Logistic Regression is a linear classification model; SVM stands as a maximum margin classifier; Random Forest embodies an ensemble method; and Neural Network signifies deep learning. Crucially, these classifiers are frequently examined in mineral classification studies (Gregory et al., 2019; Zhong et al., 2021a; Wang et al., 2023a).

This frequent examination not only enables a thorough comparative analysis but also underscores the relevance and robustness of our conclusions within the machine learning applications in geosciences.

Advantages: Machine learning methods can efficiently handle high-dimensional data. More specifically for the mineral genetic type classification problem, such methods can make, by default, full use of all available attributes (*i.e.*, elements in the geochemistry context) of the minerals. Also, ML methods can distinguish significantly more classes, and with higher accuracy (and limited or no user effort) than discriminant diagrams.

Disadvantages: Most ML models are considered black-box which lack *interpretation* (Molnar, 2020). In particular, ML classifiers only output a label (with optional confidence values) for each sample. This information is usually insufficient to interpret scientific phenomena as it tells which type (class) a sample has but not why. Additionally, when ML methods fail to correctly predict classes, tuning them to do so is often a complex trial-and-error process.

Our work aims to reveal the black box by extending machine learning classifiers with decision maps. In other words, we keep the advantages of machine learning for mineral genetic type classification while compensating for their disadvantages by providing extra visual insights into the classification process. Specifically, we use Supervised Decision Boundary Map (SDBM) (Oliveira et al., 2022), as this is the most updated version of decision maps method that increases both the speed and quality of the original DBM method. Thus, SDBM was employed to construct decision maps for all the following experiments in this chapter.

3.3 METHODS

3.3.1 *Dataset collection*

The dataset used in this study is a compilation of published pyrite trace elements datasets. Pyrite is a ubiquitous mineral in the crust. Appearing in various mineral deposit types, its trace elements can fingerprint its forming environments (Belousov et al., 2016; Zhong et al., 2021a). In this study, we compiled a dataset with 3571 pyrite LA-ICP-MS analyses from different origins, including Ni-Cu/platinum group element deposits (Ni-Cu-PGE, igneous deposits), porphyry deposits, orogenic deposits, Carlin-type Au, volcanic-hosted massive sulfide (VHMS) deposits, and barren sedimentary pyrite. Eleven trace elements (Co, Ni, Cu, Zn, Se, As, Ag, Sb, Au, Bi, Pb) are selected as features, or dimensions, for our study. Each trace element was measured in parts per million (ppm) and these measurements were used to train machine learning classifiers which are next explored using the decision map. Detailed information on the compiled dataset is shown in Table 2, including the used data sources.

Table 2: Published pyrite trace element datasets used in this study.

Class	No. of samples	References
Ni-Cu-PGE	263	Mansur et al. (2021)
Porphyry	658	Hong et al. (2018); Keith et al. (2022); Liu et al. (2020); Mavrogonatos et al. (2020); Sheng (2022); Tang et al. (2021); Zhang et al. (2016)
Orogenic	615	Zhong et al. (2021a); Large et al. (2007)
Carlin	487	He et al. (2021); Large et al. (2009); Liang et al. (2021); Lin et al. (2021); Xie et al. (2018)
VHMS	150	Revan et al. (2014); Zhong et al. (2021a)
Sedimentary	1421	Zhong et al. (2021a)

3.3.2 *Workflow*

After assembling the dataset to be used for classification, the following workflow was conducted: data preprocessing, SDBM training, search for best classifiers, map building, and evaluation.

3.3.2.1 *Metrics*

To select the best (classifier decision-map) pair, we use three metrics, introduced next. We fully elaborate these metrics further in Sec. 4.3.1 which is our main contribution to quality assessment of decision maps.

Classifier accuracy ACC_C, likely the best-known and most frequently used of the three metrics, is the fraction of correct predictions in a high-dimensional dataset and its respective labels. It is defined as

$$ACC_C = \frac{|\{\mathbf{x}_i \in D \mid C(\mathbf{x}_i) = f(\mathbf{x}_i)\}|}{|D|}, \tag{3.1}$$

where $|\cdot|$ denotes the size of a set and D is the sample set, with labels in C, used for evaluation.

Map accuracy ACC_M is the proportion of correctly positioned data points in the decision zones for a given dataset. It is defined as

$$ACC_M = \frac{|\{\mathbf{x}_i \in D \mid C(\mathbf{x}_i) = f(P^{-1}(P(\mathbf{x}_i)))\}|}{|D|}. \tag{3.2}$$

Data consistency $Cons_d$ measures the proportion of samples that retain their predicted labels, as determined by the classifier f, after a direct-inverse projection cycle. It is defined as

$$Cons_d = \frac{|\{\mathbf{x}_i \in D \mid f(P^{-1}(P(\mathbf{x}_i))) = f(\mathbf{x}_i)\}|}{|D|}. \tag{3.3}$$

3.3.2.2 *Data preprocessing*

The data were processed by the following steps:

Data missing value imputation: Unless not measured, missing values in the input dataset indicate analyses below detection limits. Missing values were set to half the detection limit to keep the data distribution.

Data transformation: Normality of the features is desired for downstream machine learning model training. Trace elements in minerals are lognormal distributed. A power transformation (Yeo and Johnson, 2000), given by

$$T(x_i^j) = log_{10}(x_i^j + 1) \tag{3.4}$$

was applied to each sample i in each dimension j to obtain this desired normality.

Data splitting: The whole dataset was randomly split into a training set D_t (80%) and a test set D_T (20%) by stratified sampling while keeping each class's proportions. D_t was used to train the classifier and SDBM, while the D_T was used to evaluate the performance of the classifier, the quality of the computed SDBM, and finally the classifier-SDBM combination.

Oversampling: Decision functions would favor the class with the larger number of samples as our dataset is unbalanced. To correct this, the Synthetic Minority Oversampling Technique (SMOTE) (Chawla et al., 2002) was applied to D_t. Note that this does not affect our final results since we split D_T before oversampling.

3.3.2.3 *Optimal decision boundary map construction*

In the following, we describe the pipeline we use to construct the optimal decision map. The workflow is summarized in Figure 3.2.

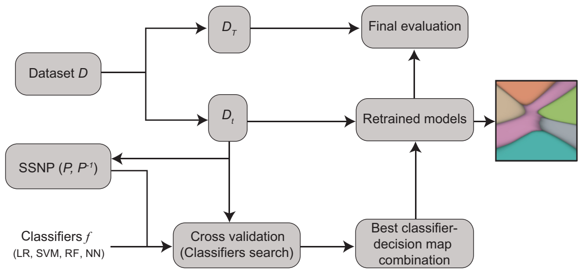


Figure 3.2: Workflow of the optimal decision map construction and evaluation. Abbreviation: LR, Logistic Regression; SVM, Support Vector Machine; RF, Random Forest; NN, Neural Network.

SDBM training: Building decision maps followed the SDBM pipeline (Oliveira et al., 2022), except that we trained SSNP, the technique used for constructing P and P^{-1} before training the classifier. This was needed because our aim next was to search for the best classifier among candidates evaluated using the same SSNP instance.

Classifier search: Four classifiers were evaluated by stratified K-fold cross-validation on the training set using the metrics described by Equations 1-3. These classifiers included Logistic Regression, SVM (with an RBF kernel), Random Forests (200 estimators), and a Neural Network (3 hidden layers of 100 units each). All these models were constructed using scikit-learn (Pedregosa et al., 2011). The classifier with the highest cross-validation scores (Equations 1-3) was selected and retrained to build the final decision map.

Map building: We created the final decision map following the procedure detailed in Oliveira et al. (2022). The decision map resolution was set to 300^2 pixels. Pixels $\mathbf{p}$ were colored by the class value $f(P^{-1}(\mathbf{p}))$. To represent confidence levels (prediction probability of f) on the decision map, we adjusted the brightness of each pixel. Pixels $\mathbf{p}$ in areas with lower confidence, typically near the boundaries where decisions change, are shown in darker shades. In contrast, $\mathbf{p}$ in high confidence areas, well inside a clear decision region, are shown in brighter shades. The visual approach allows users to quickly see where the model's predictions are more or less certain.

Evaluation: The retrained classifier and SDBM were finally evaluated on D_T with the metrics in Equations 1-3.

3.4 RESULTS

The results of the classifier search are shown in Table 3. Random forests got the highest ACC_C but the lowest ACC_M, which can be considered a poor generalization; SVM ranked third in ACC_C and first in both ACC_M and Cons; Neural Network had slightly lower results than Random forests for all three considered metrics; Logistic regression did not obtain competitive results in classifier accuracy compared to the other three models, its ACC_C being 0.09 lower than the penultimate one (SVM). Based on all three metrics, we selected SVM as the best classifier for building the decision map. The resulting map of pyrite classification built for SVM is shown in Fig. 3.3 with samples of both the training and test set plotted. Test set samples are dots with black outlines; training set samples are dots without outlines. We see that most samples fall within their respective decision zones, which already indicates a good classification performance.

Table 3: Search results of the classifiers for building the Decision Boundary Map. The highest value per metric type is indicated in bold.

Model	ACC_C	ACC_M	$Cons_d$
Logistic Regression	0.855	0.918	0.871
SVM	0.942	**0.926**	**0.922**
Random Forest	**0.984**	0.886	0.886
Neural Network	0.978	0.874	0.875

For the evaluation on the test set D_T, SVM got an overall accuracy ACC_C = 0.91 (Equation 3.1), while the SDBM got an overall accuracy ACC_M = 0.88 (Equation 3.2) and a consistency $Cons_d$ = 0.90 (Equation 3.3). The confusion matrices of both the SVM and the SDBM are shown in Fig. 3.4. ACC_M is 0.03 lower than ACC_C. This minor discrepancy, which is nearly uniform across all classes, suggests that the SDBM's (inverse) projection process (P and P^{-1}) introduces a minimal classification error for the SVM. This negligible drop of accuracy indicates that the SDBM faithfully represents the actual classifier's decision boundaries.

3.5 APPLICATIONS

We next present two applications of the decision maps to show their added-value in classifier construction and analysis. First, we demonstrate how decision maps work on samples from unseen locations and show their added-value in conjunction with regu-

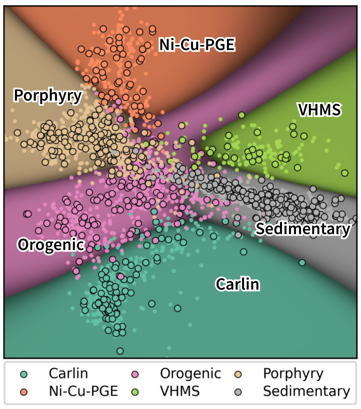


Figure 3.3: Decision Map built from the training set and the trained SVM. Training set samples are plotted as colored dots without outlines. Test set samples are plotted as colored dots with black outlines. Darker pixels in the map (mainly pixels close to the decision boundaries) show lower classification confidence.

lar machine learning methods. Second, we demonstrate how decision maps can help data exploration and model explanation.

3.5.1 *Unseen location application example*

3.5.1.1 *Case Study: Analysis of the Zaozigou Gold Deposit*

The trained classifier and its decision map were applied to data of pyrite trace elements from a new location – the Zaozigou gold deposit, which is unseen by the models. Zaozigou is the largest gold deposit (118 tons Au) that is under operation in the Gannan

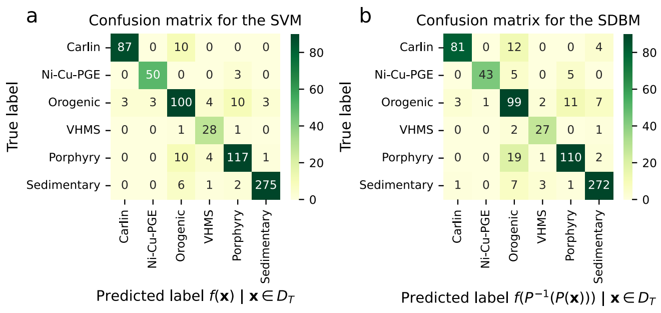


Figure 3.4: (a) Confusion matrix for the actual SVM classifier. (b) Confusion matrix for the trained decision map for this classifier.

area in the Triassic West Qinling orogenic belt in China (Qiu et al., 2020). Pyrite is the main gold-bearing mineral in this deposit, and its trace elements can be used to identify the physicochemical conditions of gold mineralization (Rusk, 2012). The genetic classification however is still in debate, which hinders our understanding for ore formation and future explanation strategy (Qiu et al., 2020). Sui et al. (2020) considered that the Zaozigou deposit is a reduced intrusion-related gold system (magmatic); Qiu et al. (2020) argued that this deposit is best classified as an epizonal orogenic Au-Sb deposit (metamorphic hydrothermal) based on in situ monazite geochronology.

The fine-labeled pyrite trace element data from Sui et al. (2020) was analyzed using the trained classifier and decision map. The pyrites are categorized into three types: (1) Py1a: pyrites in sedimentary rocks, (2) Py1b: pyrites in dike-hosted ores, and (3) Py2: pyrite grains in quartz-sulfide-ankerite veinlets. We believe that this example demonstrates our new approach's utility in solving real scientific problems.

3.5.1.2 *Classifying Pyrite from Zaozigou*

The data from the Zaozigou deposits yield results in two parts: the regular machine learning classifier (SVM) results (Table 4) and the decision map (SDBM) results (Table 5, Fig. 3.5). (1) For the samples labeled Py1a (pyrite sedimentary rocks), the SVM classified 56% of them as orogenic pyrite and 44% as pyrite in Carlin-type deposits; on the decision map, 46% of these samples were plotted in the sedimentary zone, 39% in the orogenic zone, and 15% in the Carlin zone. (2) For samples labeled Py1b (dike-hosted ores), most are classified as orogenic (94% and 84% for SVM and SDBM, respectively). (3) Most samples labeled Py2 (grains in quartz-sulfide-ankerite veinlets) are also classified as orogenic (70% and 78% for SVM and SDBM, respectively). In summary, SVM and SDBM yield similar results: Py1b and Py2 samples are classified as orogenic class; Py1a samples, however, exhibit ambiguity between Carlin, orogenic, and sedimentary types. The decision map tends to classify Py1a samples as sedimentary more than the SVM.

Focusing on the decision map (Fig. 3.5), Py1b and Py2 samples are mainly plotted in the orogenic zone, as expected. However, intriguingly, Py1a samples are divided into two clusters. One cluster is within the orogenic domain, while the other is located around the boundaries of the orogenic, Carlin, and sedimentary zones. From the geological perspective, this bifurcation suggests

Table 4: Zaozigou pyrite trace element data classification result from the SVM

	Ni-Cu-PGE	Porphyry	Orogenic	Carlin	VMS	Sedimentary
Py1a	0 (0.00%)	0 (0.00%)	18 (43.90%)	23 (56.10%)	0 (0.00%)	0 (0.00%)
Py1b	0 (0.00%)	0 (0.00%)	30 (93.75%)	2 (6.25%)	0 (0.00%)	0 (0.00%)
Py2	0 (0.00%)	2 (5.41%)	26 (70.27%)	7 (18.92%)	2 (5.41%)	0 (0.00%)
Total	0 (0.00%)	2 (1.82%)	74 (67.27%)	32 (29.09%)	2 (1.82%)	0 (0.00%)

Table 5: Zaozigou pyrite trace element data classification result from the DBM

	Ni-Cu-PGE	Porphyry	Orogenic	Carlin	VMS	Sedimentary
Py1a	0 (0.00%)	0 (0.00%)	16 (39.02%)	6 (14.63%)	0 (0.00%)	19 (46.34%)
Py1b	0 (0.00%)	2 (6.25%)	27 (84.38%)	2 (6.25%)	0 (0.00%)	1 (3.12%)
Py2	0 (0.00%)	0 (0.00%)	29 (78.38%)	5 (13.51%)	2 (5.41%)	1 (2.70%)
Total	0 (0.00%)	2 (1.82%)	72 (65.45%)	13 (11.82%)	2 (1.82%)	21 (19.09%)

that the first cluster may have interacted with ore fluids, resulting in a distinct geochemical signature. Consequently, their intricate geochemical features make these data to be a challenge to be classified. As a result, they landed near the decision boundaries of several related decision zones, which are areas of low confidence from the perspective of machine learning classification.

In summary, decision maps offer two significant pieces of additional information beyond mere agreement with the classifier: First, they reveal data clusters, which are crucial for interpreting the data, as demonstrated above; second, the decision maps demonstrate information for each individual sample, not as an aggregate score. This includes the level of classification confidence, for example, whether a sample is close to a decision boundary. Such detailed information offers a more granular understanding than an overall and general aggregate score.

3.5.2 *Exploratory data analysis and model explanation using decision maps*

3.5.2.1 *Feature Inverse Projection*

The decision maps shown so far are useful to show how all samples spread over the decision zones inferred by the trained model and also allow interpretation of the classification confidence in terms of the distance from a sample to its closest decision boundary in the map. However, they do not show which features are most responsible for the emergence of the respective decision

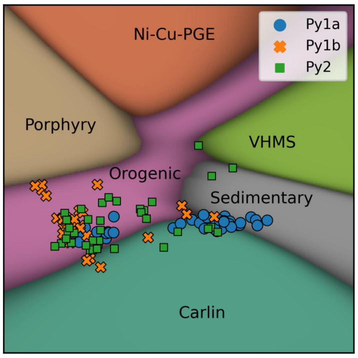


Figure 3.5: Zaozigou pyrite trace element data plotted on the trained decision map. The pyrite trace element data is from Sui et al. (2020).

zones. Understanding this is essential to further explain the studied phenomenon. To address this goal, we propose a new visualization called feature inverse projection.

To see the relationship between each feature and the decision zones/boundaries, we created a corresponding map to each feature (pyrite trace elements). For the map of each feature j, $j \in$ Co, Ni, Cu, Zn, Se, As, Ag, Sb, Au, Bi, Pb, each pixel $\mathbf{p}$ was colored by $T^{-1}(P^{-1}(\mathbf{p})^j)$, which is the value of the respective feature j, where $T^{-1}(t) = 10^t - 1$ is the inverse function of the power transformation given by Equation 3.4.

3.5.2.2 *Ranking the features*

To better guide a better reading of the maps, we propose to rank the features quantitatively. While there are multiple ways to rank the features based on their importance, here we suggest two options: (1) permutation feature importance (Breiman, 2001) for global ranking (all classes), and (2) mutual information (Ross 2014) for local ranking (user selected class).

The permutation feature importance of the classifier gives an intuition of the importance ranking of these trace elements in pyrite genetic type classification globally. The rank of the permutation feature importance of the SVM classifier on the test set is Ni > Au > Sb > Pb > As > Se > Co > Bi > Cu > Ag > Zn (Fig. 3.6a). The importance value of each feature is the decrease in SVM accuracy on D_T when a single feature value is randomly shuffled. All the importance scores are above zero, which means that all these trace elements are helpful in the classification.

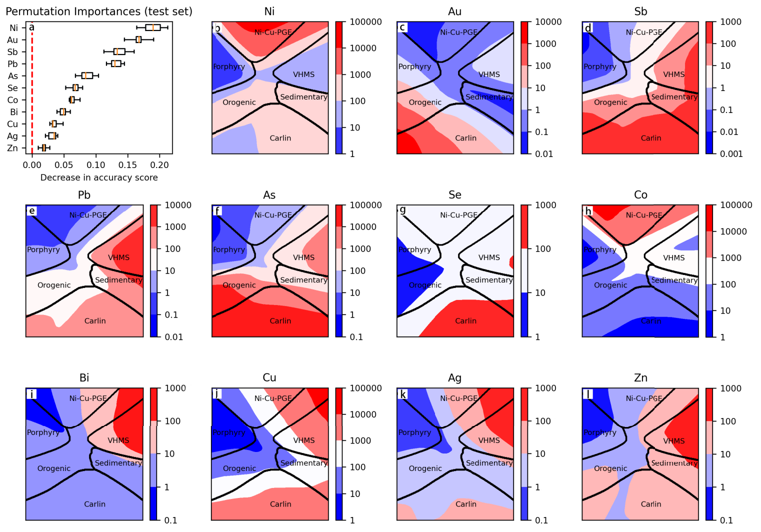


Figure 3.6: (a) Permutation feature importance of the SVM classifier. (b-l) Feature inverse projections for all the 11 features of the considered dataset that explain which features and feature-values are most responsible for the appearance of the learned decision zones.

The permutation importance provides a glimpse of the overall feature ranking. However, when the users are interested in how much a feature helps with distinguishing a certain class from all the others, a better option is to design an algorithm to rank for this specific class. Therefore, we tailor mutual information to our decision map case. For each feature j, we calculate the mutual information $I(S_c(f(P^{-1}(\mathbf{p}))), P^{-1}(\mathbf{p})^j)$ for all pixel $\mathbf{p}$, where S_c is a function that masks off all labels which are not c (the class label selected by users). Mutual information is a non-negative value. Simply put, it measures the dependence of the feature j and the user-selected class c. It equals 0 if feature j and label c are independent. The higher the value, the stronger the dependency, and thus the visual pattern of the feature aligns better with selected decision zone c and its decision boundaries (discussed below). This quantitative measurement is particularly useful when multiple features show similar patterns.

Note that the ranking methods are to provide the users (geologists) with clues for exploring the data. Geologists' knowledge is still crucial in interpreting the data in this human-centered application case.

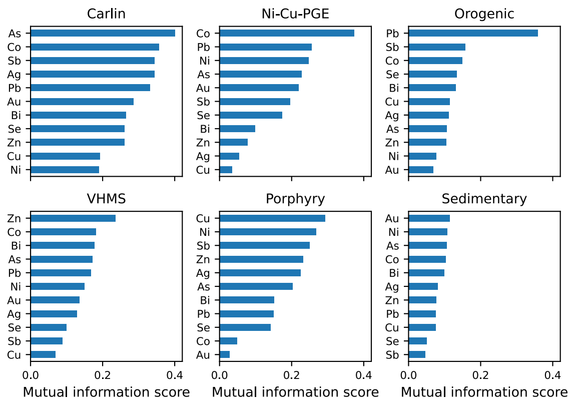


Figure 3.7: Mutual information scores ranking feature importance for each class respectively.

3.5.2.3 *Visualizing feature patterns*

The resulting inverse projected features are displayed in the permutation importance order (Fig. 3.6b-l). Black lines in these images show the decision boundaries. Actual feature values of the high-dimensional samples corresponding to every pixel are color-coded on an ordinal colormap (blue=low, red=high feature values). We used a banded colormap having a small number of discrete levels. This way, color changes in the images indicate the actual isolines (equal-feature-value contours) of the respective features in the data. Simply put, if a color band created by the above colormap for value v of feature j has a shape that matches well the shape of a decision zone for class c it is plotted over, it means that the value v of j is a strong predictor of class c. Conversely, if all color bands of feature j have shapes that do not match well any of the decision zones, it means that j is not a strongly useful feature for the classification. This can be exemplified by either permutation importance or mutual information ranking: (1) Ni, which is ranked as the most important feature for prediction, shows three color bands (dark blue, light blue, red) which match quite well the Porphyry, VHMS, and Ni-Cu-PGE zones, respectively (Fig. 3.6b). In contrast, Zn, the least important feature for prediction, shows color bands that match far less well than any of the six decision zones (Fig. 3.6l). While permutation importance gives us an initial understanding of feature relevance, mutual information can provide a more nuanced view, especially in terms of how specific features align with a certain class. (2) For instance, in mutual information ranking, the features with the highest and lowest scores for the orogenic class are Pb and Au, respectively (Fig. 3.7). The isolines of Pb align well with the shape of the orogenic decision zone (Fig. 3.6e), highlighting Pb is a strong indicator for predicting orogenic class. Conversely, the isolines of Au, being roughly perpendicular to the orogenic decision zone (Fig. 3.6c), indicate that Au is less useful for discriminating this class.

Let us explore in detail how the feature inverse maps show the relationships between pyrite trace elements and their forming environment types learned from the model. We consider both visual patterns (relations between color bands and decision zones) and feature ranking. We do this in order of permutation importance:

1. The color bands show that Ni > 1000 ppm can distinguish Ni-Cu-PGE from other classes, and Ni < 1 ppm can distinguish porphyry from other classes (Fig. 3.6b); Mutual information feature ranking confirms the importance of Ni for both Ni-Cu-PGE and porphyry classes (Fig. 3.7)

2. Au > 100 ppm characterizes pyrites from orogenic and Carlin-type deposits. Au < 0.1 ppm is the character of pyrites from barren sedimentary and Ni-Cu-PGE (magmatic) deposits (Fig. 3.6c); However, Au is not a strong predictor for any single class, as indicated by its lower mutual information scores (Fig. 3.7).

3. Pyrite with Sb < 0.1 ppm is more likely from Ni-Cu-PGE or porphyry deposits, while pyrite with Sb > 10 ppm is more likely from the other four classes (Fig. 3.6d); The mutual information score robustly supports the visual pattern indicating the importance of Sb for the porphyry class (Fig. 3.7).

4. Pyrites from VHMS deposits, sedimentary and Carlin-type deposits tend to have Pb values > 100 ppm (Fig. 3.6e); Moreover, as mentioned above, the color band of Pb concentration ranging 10 - 100 ppm aligns well with the shape of the orogenic decision zone, the significance of which is also confirmed by mutual information score for orogenic class (Fig. 3.7).

5. Pyrite from Carlin, orogenic and VHMS deposits have high As values. Most Carlin pyrite and some orogenic pyrite could have As > 10000 ppm (Fig. 3.6f); When focusing on a single class, As appears to be an efficient predictor for only Carlin class (Fig. 3.7).

6. Pyrites with Se < 10 ppm are more likely to be from porphyry or orogenic deposits (Fig. 3.6g); Se, however, does not have a significant mutual information score for distinguishing any single class, as shown in Figure 3.7.

7. Co > 1000 ppm characterizes Ni-Cu-PGE pyrite (Fig. 3.6h); Figure 3.7 highlights Co as the most significant element for the Ni-Cu-PGE class.

8. VHMS and part of the Ni-Cu-PGE zone have Bi > 10 ppm (Fig. 3.6i); However, Bi's insignificance for single-class discrimination is evident in Figure 3.7.

9. Cu < 10 ppm is the character of porphyry pyrite. Pyrites in the other four classes have Cu varying from 10 to 10000 ppm (Fig. 3.6j); The significance of Cu for identifying porphyry class is also strongly confirmed by the mutual information score (Fig. 3.7).

10. The color band of Ag < 1 ppm align with the porphyry zone(Fig. 3.6k); However, similar to Au, Ag is overall also not efficient for identifying any class, as it never ranks in the top 3 for any class in mutual information score (Fig. 3.7).

11. The Zn value color bands do not match the decision zones well, except for the band of Zn > 100 ppm, which matches the VHMS zone well (Fig. 3.6l); And, indeed, Zn is the most efficient element for distinguishing VHMS from other classes according to mutual information score for VHMS class (Fig. 3.7).

3.6 DISCUSSION

3.6.1 *Interpretability and limitations of decision maps*

Based on the evaluation metrics (Equations 1-3), we established the optimal decision map for the pyrite genetic type classification task. As shown in the results, the SVM has an ACC_C of 0.91, while the decision map for the aforementioned SVM has an ACC_M of 0.88 and a $Cons_d$ of 0.90, on the test set D_T. This means that decision maps can be used to accurately predict how a classifier works. From a visual perspective, there is only a slight overlap of data points in the center of the map (Fig. 3.3), a property with which no existing 2D discriminant diagram can compete.

Decision maps provide a novel way to get insight into how machine learning classifiers work and where each data point lands in the context of decision boundaries. They should not be seen as a replacement, but rather an enhancement, of traditional classifier metrics (*e.g.*, accuracy): Classifier metrics give a highly aggregated quality score (for the entire problem or per class), but do not tell how specific instances (train, test, or new) get classified. This is exactly the addition that decision maps provide. More specifically, the actual shapes of the decision zones and the spread of instances over them tell how easy is for a given classifier to handle a given data distribution, *e.g.*, which classes are easily separable from the others and/or which parts of the data distribution are easily classifiable.

On top of the samples being categorized into a major class label, decision maps show how samples are similar to certain other classes via their distances to the closest decision boundaries. Samples near decision boundaries are more uncertain about the predicted label and thus more likely to be misclassified. Feature inverse maps (discussed below) provide additional insights into why these samples may have such problems. However, all these tools need to be complemented by an expert's knowledge to lead to effective interpretations and understanding of the studied phenomenon. Decision maps provide thus a way to combine human knowledge with machine learning predictions when interpreting classification results in a way that cannot be obtained from regular machine learning classification routines. The application of the decision map on the Zaozigou pyrite data discussed next further illustrates the added-value of our visualizations (detailed in the next subsection).

Besides observation-centric interpretation (seeing how samples spread with respect to each other and the inferred decision zones), our new addition to decision maps – the feature inverse projections – provides a class-centric interpretation, *i.e.*, allows analysts to understand which features and feature values are key responsible for the appearance of specific decision zones or even separate sample groups. These feature inverse projections can be seen as a summary of real-world complex data that show what the model learned and how it decides when values vary. We further discuss this in the next subsection.

Our visual analysis techniques scale well with both the number of samples and the number of dimensions. As shown in Figure 7, the computation time of SDBM is minimally affected by changes in either the number of dimensions or the number of samples. It consistently remains at approximately 3-10 seconds on a standard desktop computer with a consumer-level graphics card. This contrast is particularly noticeable when compared to machine learn-

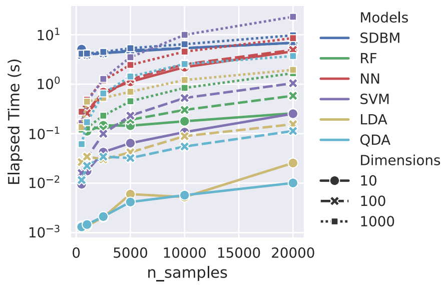


Figure 3.8: Plot showing the time taken by SDBM and 5 common classifiers, utilizing synthetic datasets of varying dimensionality and number of samples (n_samples). The time recorded for SDBM includes the duration for fitting training samples and inverse projecting grids (grid size: 300^2); whereas, for classifiers, it records the time for fitting and predicting labels for the training samples. Abbreviation: RF, Random Forest; SVM, Support Vector Machine NN, Neural Network; LDA, Linear Discriminant Analysis; QDA, Quadratic Discriminant Analysis.

ing classifiers such as SVM, LDA, and QDA, which can be highly sensitive to changes in data dimensionality (Fig. 3.8). In general, using SDBM to obtain a decision map for a given classifier requires only a few additional seconds after the classifier is trained. Every sample point is reduced to a single 2D scatterplot point. While overplotting does occur, this does not affect, we argue, the usability of our proposal. Indeed, in most applications, one is interested in reasoning about groups of similar samples and not every single individual. Such groups become actually better visible when large amounts of samples are plotted. Decision maps also inherit by construction the scalability of the underlying projection techniques to tens or even hundreds of dimensions. Feature inverse projections are less scalable in this sense since we need to plot (and study) one map per feature. However, as discussed above, such maps can be ordered by feature ranking methods (*e.g.*, permutation feature importance, mutual information), so that analysts can focus on a small set of most relevant features. Similar techniques have been used for explaining projections of high-dimensional data for 3D projections (Coimbra et al., 2016).

Decision maps and their proposed extensions are also generically deployable and easy to use: They can be generated automatically using any trained classifier and any relevant data set, whether it be training, test, or new data. Exploring the created visualizations also does not require any complex interaction from the user except the optional brushing of points to show details in a tooltip. While the standard implementation of SDBM (Oliveira et al., 2022) does not provide this feature, adding it is very simple. Note that this functionality can target both existing points from the projected dataset D used to construct the SDBM, and more interestingly, new points that correspond to pixels in the decision map to which no actual data point projects. These effectively generate new, unseen, data points in the high-dimensional space (via the inverse projection P^{-1}) which allow the analyst to reason about how the classifier, or more generally phenomenon under study, would behave for data outside the actually measured dataset one has.

However, decision maps also have some limitations. As explained earlier, both direct and inverse projections have inevitable errors which cannot be fully eliminated in the generic case. We address this issue by quantifying the magnitude of errors and demonstrating that, for classifier analysis, these errors are minimal and do not significantly impact the interpretability of the decision maps. If desired, one can easily extend our proposal by visualizing errors locally in the decision maps following Espadoto et al. (2021a). Studying how such more refined error views can help interpret classifiers is an important future work topic. A separate limitation of decision maps is that they do not explicitly depict individual dimensions along the two axes of the map, unlike classical discrimination diagrams. Combined with the nonlinear nature of the projections used to create the maps, this asks analysts to deploy more effort to understand how dimensions vary across the map. For example, the pyrite decision zones in Fig. 3.3 show a trend from the high-temperature forming environment to the low-temperature forming environment in sequence: Ni-Cu-PGE – Porphyry – Orogenic – Carlin. The feature inverse projections help this analysis by mapping the feature variations, one by one, to the respective maps. An interesting future work direction is to summarize several such feature inverse projection images in a single map, thereby reducing the number of different visualizations one needs to study to interpret a decision map.

3.6.2 *Implications for mineral deposit genesis classification studies*

The dataset used in our work includes Carlin-type pyrite and Ni-Cu-PGE pyrite trace elements, which fills the gap of previous pyrite machine learning related work (Gregory et al., 2019; Zhong et al., 2021b). This dataset provides a more comprehensive view of pyrite from magmatic to hydrothermal origins. More importantly, we present a solution to the current problem of the lack of visual interpretability of machine learning in geochemical data classification work. Visual interpretability is a valuable property of traditional geochemistry discriminant diagrams, and it is also a desire for geochemistry data exploration and analysis. Our decision maps solution provides a unique perspective to reveal the structure and properties of data hidden from regular machine learning routines, offering new opportunities for analyzing and explaining geological problems. More specifically, the decision map application on Zaozigou pyrite trace elements shows how seeing the data clusters and their locations on the decision map can help interpretation compared to regular machine learning routines; the feature inverse projection application shows how the decision map can uncover what the model learned from the mapping of pyrite trace elements to pyrite forming environments, and displays how the model decides the type of pyrite when the trace element values vary.

We now discuss the specific findings we obtained using decision maps for our specific use-case of studying mineral deposit genesis. Pyrite trace element data of the Zaozigou deposit from Sui et al. (2020) are plotted mainly in the orogenic zone on the decision map. Within this zone, the plotted data clusters are closer to the Carlin and sedimentary zones than the porphyry and Ni-Cu-PGE zones. From the view of pyrite trace elements, Zaozigou shows little similarity to magmatic-related (Ni-Cu-PGE, Porphyry) deposits. Instead, it shows some more similarity to low-temperature Carlin-type deposits. Therefore, it is reasonable that Py1a are plotted around the boundaries of the orogenic zone, Carlin zone, and sedimentary zone (Fig. 3.5): Since Py1a samples are pyrites in sedimentary within the gold deposit district, Py1a shares similarities to pyrite in barren sedimentary geologically; Carlin-type deposits, which were first found in Nevada, USA, are sediment-hosted, disseminated Au deposits. So, they also share some similarities to pyrite in barren sedimentary. Most of the Carlin-type samples in the dataset are from gold deposits near the edge of the Yangtze craton, in southeast China. These deposits are also argued to be epizonal orogenic gold deposits (Bodnar et al., 2014). If we regard the Carlin class as an epizonal orogenic class, Py1a pyrites are more similar to pyrite from epizonal orogenic deposits than from classic orogenic deposits; Pyrites from Py1b and Py2 are more similar to pyrites from classic orogenic deposits. The conclusion from pyrite trace elements and the decision map method closely agrees with the monazite geochronology conclusion from Qiu et al. (2020).

According to the feature inverse projections (Fig. 3.6, Fig. 3.7), some trace elements can be considered indicator elements in discriminating the mineral-forming environments. For example, the model learned that Co, Ni, and Pb are efficient features when classifying Ni-Cu-PGE from others. This model learned knowledge is consistent with geologists' experience that Co, Ni, and their ratio in pyrite are considered reliable indicators and geochemical tools in ore deposit genesis (Bajwah et al., 1987; Bralia et al., 1979). Knowing what the model learned for classifying the pyrite genetic types makes it easy to find other elements as indicators. For example, Pb, which is less discussed in the literature, could be an indicator for discriminating Ni-Cu-PGE, porphyry, and orogenic pyrites from the other classes. In Figure 5, we can observe that the model considers pyrites of Ni-Cu-PGE and porphyry classes to have the feature that Pb < 10 ppm, while orogenic pyrite has Pb roughly between 10 to 100 ppm; Cu could be another indicator for discriminating porphyry pyrite from the other classes, *i.e.*, the model considers porphyry pyrite has the feature that Cu < 10 ppm (Fig. 3.6j). The effectiveness of Pb and Cu as indicators remains to be further proven in practice.

3.7 IMPLICATIONS

3.7.1 *Implications for the geoscience community*

The union of information visualization and mineralogy, as presented in this study, heralds a transformative era in geoscience research. By harnessing the capabilities of enhanced decision maps, we have illuminated a novel approach to interpret classification models, deepening our comprehension of multifaceted geochemistry data dimensions.

The introduction of inverse projections is particularly groundbreaking for geology. This feature unravels the depth of understanding models extracting from complex geochemical data, enabling researchers to directly correlate predictions with specific mineralogical features or value-ranges. In the realm of mineral geochemical discrimination, this research signifies a monumental shift. Transitioning from traditional machine learning classification to the advanced visual analytics of machine learning, we're effectively merging the precision and scalability of modern computational methods with the rich, interpretative legacy of discriminant diagrams.

As it continues to lean into data-driven methodologies, our work offers a robust toolset for enhanced mineral genesis classification and exploration. Beyond the immediate applications, this study promises to influence a range of geochemistry subdisciplines, driving more informed, nuanced, and efficient research and exploration endeavors in the future.

3.7.2 *Implications for visualization community*

This chapter shows that decision maps are practically useful in geosciences research, a discipline which is far away from computer science, and whose users are non-specialists in data visualization. While applications of decision maps in geosciences are also reported in Zhou et al. (2023), our work is, to our knowledge, the first in this area. On the one hand, these results are a good indicator that decision maps are an effective tools for solving problems in *practice* when one needs to explore and/or improve a trained classification model. On the other hand, we discovered that current decision map methods have several challenges that need to be addressed to make them even more effective in such practical use cases. We summarize these challenges next.

First, our work investigated only the SDBM decision map method – and, while doing so, used only three relatively simple metrics to gauge quality. As mentioned in Chapter 2, several other decision map methods exist. As such, a comprehensive comparison of such methods with additional quality metrics is needed to understand the pros and cons of each method. This is addressed in Chapter 4.

Secondly, we found that a single view – or single decision map image – cannot capture everything we aim to understand in the behavior of a trained model. To alleviate this, we customized the decision map with so-called feature inverse projection (Sec. 3.5.2). While this is a good start, more refined user customizations are needed to make decision maps more flexible and more informative. We propose such user-driven customizations in Chapter 6.

Finally, we only created decision maps with a fixed resolution of 300^2 pixels. The key reason for this was the limited computational efficiency of SDBM. As more user control is eventually incorporated, such as zooming in and out, the resolution of the decision map should be increased, which will require faster ways to compute it. We address this topic in Chapter 7.

4 QUALITATIVE AND QUANTITATIVE EVALUATION OF DECISION MAPS

4.1 INTRODUCTION

As introduced in Chapter 2, several techniques exist to construct decision maps for arbitrary classifiers (Hamel, 2006; Migut et al., 2015; Schulz et al., 2015, 2020; Rodrigues et al., 2019; Oliveira et al., 2022), These techniques have found applications in areas like model steering (Rodrigues, 2020), detecting backdoor attacks (Schulz et al., 2020), and our own work in interpreting geoscience models (Chapter 3). Overall, such techniques take away some of the complexity of interpreting the working of a machine learning model while, in the same time, depicting the functioning of the model in more detailed ways than classical aggregate performance metrics. As such, decision map techniques are particularly attractive for users of machine learning who are not domain experts in the operation of the underlying ML algorithms (Zhou et al., 2023)[1].

Despite the above, no framework to comprehensively evaluate and compare such techniques exists (Oliveira et al., 2022). This makes it hard for actual users to know which technique to pick and use in a given application context.

Lacking an in-depth assessment for the quality of decision maps lead to some issues. From a technical perspective, it is unclear how effectively different decision map techniques capture information present in high-dimensional decision zones and boundaries. For instance, it is unclear whether the smoothness or fragmentation visible in decision boundaries truly depicts the same properties of the actual, high-dimensional, boundaries. Similarly, it is unclear whether a sample located close to a decision boundary in the map genuinely reflects its proximity to the high-dimensional point where a classifier changes output. From a practical perspective, this lack of evaluation makes it hard to select the most suitable decision map technique for a given dataset and classifier.

1 This chapter is based on the paper “Quantitative and Qualitative Comparison of Decision Map Techniques for Explaining Classification Models” (Wang et al., 2023b).

We aim to fill this gap – and therefore answer our research question **RQ1** introduced in Chapter 1 – by proposing a framework that evaluates decision map techniques. We proceed by identifying the desirable aspects that a decision map technique should have – accuracy, reliability, interpretability, and computational efficiency. Next, we design several metrics to quantify these aspects. Taken together, these metrics aim to capture the concept of 'quality' of a decision map technique, much like classical metrics in ML such as accuracy, area under the ROC curve, F1-scores, and similar aim to capture the concept of quality of a ML model. We then use these metrics to conduct a multi-faceted comparison of existing decision map techniques over several classifiers and datasets. This helps us identify several strengths and limitations of the evaluated decision map techniques. Finally, we propose a workflow for choosing the best decision map technique for a given dataset in light of a set of desirable requirements.

Summarizing the above, our key contributions are as follows:

- We propose a suite of metrics to quantitatively evaluate decision map techniques;
- We conduct a comprehensive comparison of existing decision map techniques, both quantitatively (by comparing the aforementioned metrics) and qualitatively (by comparing visually the obtained decision map images);
- We propose a workflow to guide the selection of the most suitable decision map technique for a given dataset, based on a set of desirable requirements.
- At a higher level, this chapter answers our question RQ1 introduced in Sec. 1.

4.2 RELATED WORK

4.2.1 *Overall workflow of decision map*

Decision maps – introduced in Sec. 2.3.4 – are techniques that visualize the decision boundaries and decision zones of a classifier: Given a classifier f, a *decision boundary* is a surface in $\mathbb{R}^n$ that separates high-dimensional data points $\mathbf{x} \in \mathbb{R}^n$ into regions, also called *decision zones*. All points in a given zone are assigned the same label $y \in C$ by the model f. Decision zones are separated by *decision boundaries*, which are hypersurfaces embedded in $\mathbb{R}^n$ where the classifier f changes output. Understanding how the high-dimensional space is partitioned into such decision zones, and how data samples in a training or test set are distributed across the zones, effectively helps understand how a classifier behaves (Rodrigues et al., 2019; Schulz et al., 2020). For example, seeing how labeled samples distribute close to decision boundaries can help categorize misclassification problems; seeing how unlabeled samples (whose class is to be predicted by f) spread across decision zones helps understand how well f can handle a given data distribution.

The general workflow of constructing decision maps is as follows (see also Figure 4.1):

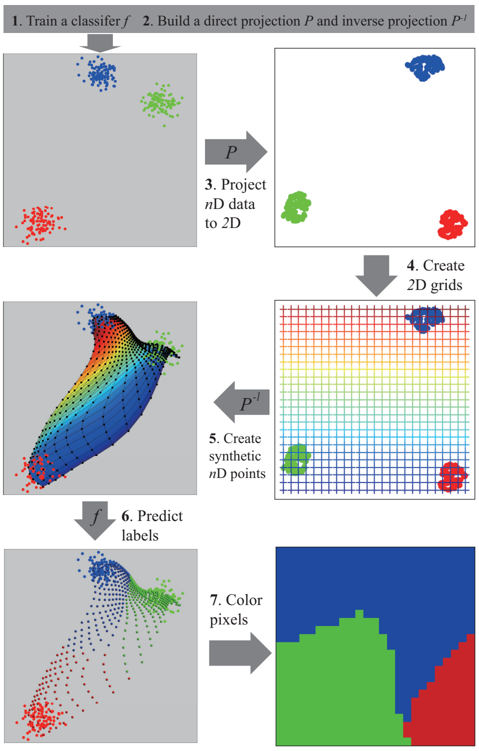


Figure 4.1: General workflow of decision map techniques (see Sec. 4.2.1).

1. Train a classifier f on a dataset D_t. This is the classifier whose decision map we next want to visualize;

2. Construct a direct projection P and inverse projection P^{-1} using a dataset D';

3. Project D' to create a 2D scatterplot $P(D')$;
4. Sample the extent of $P(D')$ on a uniform pixel grid I;
5. Backproject all pixels $\mathbf{p} \in I$ to the data space using P^{-1};
6. Use f to predict the labels of the backprojected points $P^{-1}(\mathbf{p})$;
7. Color I according to the predicted labels $f(P^{-1}(\mathbf{p}))$.

In the above, D' can be, at a minimum, a subset of the training set D_t. However, if more samples of the target domain are available, such as in the form of test data D_T or even unlabeled data, these can be added to D'. By doing this, the constructed decision map will better sample the actual data distribution of the investigated phenomenon and, thus, the classifier's behavior. In this work, we set $D' = D_t$ following earlier examples of this workflow (Oliveira et al., 2022).

We next describe three specific techniques that implement the above workflow and introduce some of their key features. For completeness, these are the techniques we further selected for quantitative and qualitative evaluation in this chapter. The selection was based on the fact that these techniques pioneering in the DBM area (Rodrigues et al., 2019); are computationally effective and simple to use (Oliveira et al., 2022); and achieve a high quality representation (Schulz et al., 2020), respectively. For each technique, we also indicate the exact origin of its source code, for replicability purposes of our comparison.

4.2.2 *Decision Boundary Maps (DBM)*

Decision Boundary Maps (DBM) (Rodrigues et al., 2019) closely follow the above workflow. While the original DBM used t-SNE (van der Maaten and Hinton, 2008) and, alternatively, UMAP (McInnes et al., 2018) for the projection P, any user-chosen projection method can be used, such as PCA (Jolliffe and Cadima, 2016), LAMP (Joia et al., 2011), Least Square Projection (LSP) (Paulovich et al., 2008), or Piecewise Laplacian Projection (PLP) (Paulovich et al., 2011). For the inverse projection P^{-1}, DBM evaluated two techniques, namely iLAMP (dos Santos Amorim et al., 2012) (the inverse of the LAMP projection technique mentioned above) and NNinv (Espadoto et al., 2019c). Compared to iLAMP, which constructs the backprojection by interpolating between the samples of D and $P(D)$ using linear or radial kernels, NNinv deep learns a regressor to output D using $P(D)$ as input. NNinv achieves inverse projections having a lower mean absolute error than P and is also simpler to implement and significantly faster (Espadoto et al., 2019c; Rodrigues et al., 2019). Note that the same deep learning idea has been used to construct direct projections with similar speed, quality, and implementation simplicity advantages (Espadoto et al., 2020).

DBM is simple to implement and allows one to use any direct and inverse projection techniques. Also, DBM works without requiring label information, which means that constructing decision maps only depends on the distribution of points in D' in the feature space. In our subsequent evaluation, we use DBM with UMAP as the direct projection P and NNinv for P^{-1}, as this combination led to the best results in earlier DBM evaluations (Rodrigues et al., 2019). More information about DBM is available at `https://mespadoto.github.io/dbm/`.

4.2.3 *Supervised Decision Boundary Map (SDBM)*

While flexible, the independent choice of P and P^{-1} in DBM means that these operations have to be constructed separately (including fine-tuning their hyperparameters), which incurs additional effort. Self-Supervised Neural Network Projection (SSNP) (Espadoto et al., 2021b) alleviates this by constructing P and P^{-1} jointly. Briefly put, SSNP follows a classical autoencoder architecture but adds a classification loss atop the standard reconstruction loss. To minimize this classification loss, either true labels (supervised mode) or pseudo-labels (semi-supervised mode; labels are obtained by running a clustering algorithm on the feature space) can be used. The encoder part then delivers P, whereas the decoder delivers P^{-1}.

Supervised Decision Boundary Map (SDBM) (Oliveira et al., 2022) directly applies SSNP to construct decision maps following the workflow in Fig. 4.1. Like DBM, SDBM is also simple to implement and has a similar speed. More interestingly, SDBM seems to produce smoother decision boundaries than DBM and these boundaries appear to better agree with ground-truth information on the visualized classifiers. The price to pay for this is the inability to choose a specific direct projection P. This can be suboptimal in cases where one has such a technique that is known to be best for depicting the structure of a given dataset. The code of SDBM is available at `https://github.com/mespadoto/sdbm`.

4.2.4 *DeepView (DV)*

DeepView (DV) (Schulz et al., 2020) constructs the direct projection P by using UMAP. However, in contrast to classical UMAP, the points' similarities are computed using a Fischer distance which combines their high-dimensional features with the classification function f. This approach, also called discriminative dimensionality reduction (Schulz et al., 2015), favors grouping points in the projection that are similar both feature-wise and in terms of classification by f. The inverse projection P^{-1} of DV also uses UMAP, with the roles of input and output swapped. The inverse projection is next extrapolated to all 2D points by minimizing a Kullback-Leibler (KL) divergence that captures probabilities of closeness in both 2D and the data space. Once P^{-1} is available, DV colors the 2D pixels following the same approach as DBM and SDBM (Fig. 4.1, step 7).

While DV produces decision maps with smooth boundaries, the process is quite expensive, given that the computation of the Fisher distance is squared in the number of samples in D'. Also, the process involves optimizing several hyperparameters to construct an accurate P^{-1}. Just as DBM, the direct and inverse projections P and P^{-1} can be in principle freely chosen. However, given the aforementioned optimization complexity, we next use the exact original proposal in Schulz et al. (2020) to construct both P and P^{-1}. The code of DV is available at `https://github.com/LucaHermes/DeepView`.

4.2.5 *Limitations*

As indicated in Chapter 2, the by far most evident issue of current decision map techniques is the very limited *evaluation* they come with. DBM was evaluated qualitatively by using 28 projection techniques P (and iLAMP for P^{-1}) to conclude that UMAP and t-SNE are among the best options for P for creating smooth decision boundaries (Rodrigues et al., 2019). However, this evaluation was purely *qualitative*, *i.e.*, based on visually examining the respective decision maps for smoothness. SDBM was evaluated on four classifiers and four real-world datasets and compared against DBM. However, as in the previous case, the comparison was purely qualitative, based on a visual assessment of the map's smoothness. Finally, DV was evaluated on two real-world datasets. Its quality was measured by computing two metrics related to our map accuracy and data consistency (discussed further in Sec. 4.3.1). However, it was not compared to any other

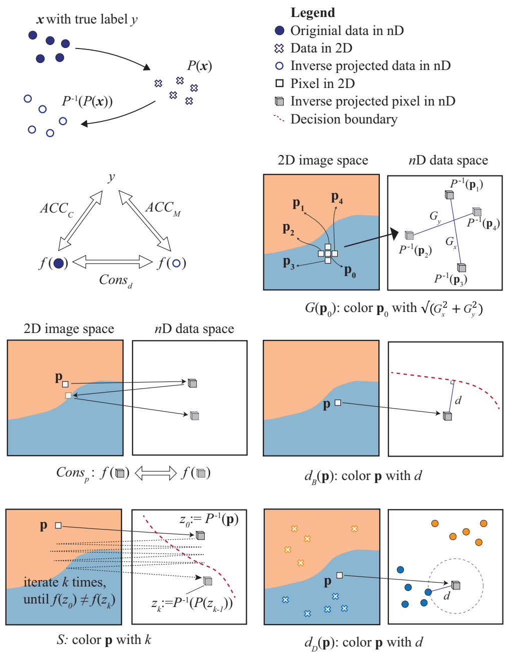


Figure 4.2: Illustration of the metrics used to evaluate decision maps. (see Sec. 4.3)

decision map technique. We expand on all these aspects by our proposed evaluation method described next.

4.3 EVALUATION METHOD

As mentioned in Sec. 4.2.5, current evaluations of decision map techniques are limited. Due to inevitable errors in the decision map creation process (detailed next in Sec. 4.3.1), using the classifier's accuracy to gauge a decision map's quality is neither appropriate nor comprehensive. Simple visual inspection of a decision map is equally limited in gauging its ability to correctly capture a classifier's behavior since we do not usually know this ground-truth behavior upfront. We aim to improve upon this by proposing an extensive set of both quantitative and qualitative evaluations, each characterizing a different desirable property of decision maps, as follows.

4.3.1 *Global metrics*

Global metrics aim to characterize the quality of a decision map by a single (scalar) value, much like metrics for direct projections such as trustworthiness, continuity, or normalized stress (Venna and Kaski, 2006) (see also Eqn. 2.3 and related text).

We propose five such metrics, as follows (see also Figure 4.2). Note that we used the first three metrics in Sec. 3.3.2.1 for selecting the best (classifier, SDBM) combinations for our geoscience application. In this chapter, we discuss these three metrics in further detail and also introduce additional metrics for decision map evaluation. We then use these metrics to evaluate three decision map techniques (DBM, SDBM, DV) with a broader set of classifiers and datasets than in our work in Chapter 3.

Classifier accuracy ACC_C: Accuracy is one of the most common metrics for evaluating the performance of a classifier and is defined as the ratio of correct predictions made by the model f to the total number of predictions, *i.e.*

$$ACC_C = \frac{|\{\mathbf{x}_i \in D \mid C(\mathbf{x}_i) = f(\mathbf{x}_i)\}|}{|D|}, \tag{4.1}$$

where $|\cdot|$ indicates set size, D is the sample set used for evaluation, and $C(\mathbf{x}_i)$ is $\mathbf{x}_i$'s ground-label y_i assigned by the trained model f. We will use these notations throughout this section when defining global metrics. In the following, we will set D to either D_t (training set) or D_T (test set) and thereby compute the corresponding classifier accuracies which we denote as ACC_C^t and ACC_C^T, respectively. ACC_C ranges in $[0,1]$, where $ACC_C = 1$ indicates perfect classification. While accuracy does not, as we already noted, gauge the quality of a decision map, it helps calibrate the understanding of subsequent metrics. For example, if we know a classifier is accurate, we expect its decision map to reflect this accordingly.

Map accuracy ACC_M: We define map accuracy as the fraction of data points (of a given dataset D) that are drawn in the correct decision zones. We define map accuracy as

$$ACC_M = \frac{|\{\mathbf{x}_i \in D \mid C(\mathbf{x}_i) = f(P^{-1}(P(\mathbf{x}_i)))\}|}{|D|}, \tag{4.2}$$

Intuitively, this says that data points $\mathbf{x}_i$ for which we have ground-truth labels y_i are indeed colored correctly (*i.e.*, by y_i) in the computed decision map. As for class accuracy, we compute ACC_M^t and ACC_M^T for the training, respectively test, sets. Note that map accuracy only evaluates the map pixels onto which the data in D projects since, for all other pixels, we do not have ground truth labels. The range of ACC_M is $[0,1]$, where $ACC_M = 1$ tells that all data points are drawn on the pixels with the same color as their ground-truth labels. Another way to evaluate map accuracy is to employ an additional 2D classifier, as used in the DV evaluation (see Q_{kNN} in (Schulz et al., 2020)). We do not use this option since we believe it introduces an additional degree of complexity in the selection and training of this additional classifier.

Data consistency $Cons_d$: In general, both direct and inverse projections P and P^{-1} unavoidably introduce errors (Espadoto et al., 2019a; Nonato and Aupetit, 2018; dos Santos Amorim et al., 2012; Espadoto et al., 2019c). That is, in general, P^{-1} is not an exact inverse of P, *i.e.*, $P^{-1}(P(\mathbf{x})) \neq \mathbf{x}$ for several data points $\mathbf{x}$. Such errors can be evaluated by the mean square error $MSE = \sum_{\mathbf{x} \in D} \|\mathbf{x} - P^{-1}(P(\mathbf{x}))\|^2 / |D|$ computed over the set D (Espadoto et al., 2021a). However, for decision maps, MSE is not the most relevant metric to consider: A pixel $P(\mathbf{x})$ may be backprojected away from $\mathbf{x}$, *i.e.*, the MSE may be nonzero; still, if the backprojection has the same label as the original point $\mathbf{x}$, then there is no visible error in the map. To account for this, we gauge whether the error introduced by the 'round-trip' direct-and-inverse projection creates inconsistencies in the decision maps. For this, we define the projection consistency metric

$$Cons_d = \frac{|\{\mathbf{x}_i \in D \mid f(P^{-1}(P(\mathbf{x}_i))) = f(\mathbf{x}_i)\}|}{|D|}. \tag{4.3}$$

Simply put, $Cons_d$ measures the fraction of so-called consistent projections, *i.e.*, samples $\mathbf{x}_i$ from a given set D that keep the same classification label after one round-trip given by the projection P and inverse-projection P^{-1}. This metric is also used in DV's evaluation under the name Q_{data} (Schulz et al., 2020). As for class and map accuracy, we compute $Cons_d^t$ and $Cons_d^T$ for the training, respectively test, sets. Note that, in contrast to ACC_C and ACC_M, $Cons_p$ does not use ground truth labels y_i. That is, consistency assesses only how much the decision map can represent a given classifier, and not whether the DBM is correct with respect to ground-truth labels. The range of $Cons_d$ is $[0,1]$, where $Cons_d = 1$ indicates perfect consistency.

Map consistency $Cons_p$: All above metrics evaluate a decision map only at locations where an actual data point in D would project. However, as already noted, most pixels in such a map are not covered by data points. We extend $Cons_d$ to cover all pixels in a map by

$$Cons_p = \frac{|\{\mathbf{p} \in I \mid f(P^{-1}(P(P^{-1}(\mathbf{p})))) = f(P^{-1}(\mathbf{p}))\}|}{|I|}, \tag{4.4}$$

where $\mathbf{p}$ is a pixel of the decision map image I. $Cons_p$ calculates the fraction of consistent pixels in the decision map, that is, pixels whose corresponding data points (obtained by inverse projection P^{-1}) have the same classification label after a round-trip of projection P and inverse-projection P^{-1}. Note that $Cons_p$ extends the idea of using the round-trip to visually evaluate an inverse projection (Espadoto et al., 2021a) to a quantitative metric used to evaluate a decision map. $Cons_p$ ranges in $[0, 1]$, where $Cons_p = 1$ implies perfect pixel-level consistency in a decision map.

Class stability $\bar{S}$: One application of decision maps concerns improving a classifier by, for instance, adding extra labeled training samples by backprojecting selected decision map pixels. In this case, *multiple* round-trips between the 2D map space and data space occur. Since P^{-1} is not an exact inverse of P, several such round-trips can increasingly accumulate errors. We measure this as follows. Given a pixel in the decision map, we apply P^{-1} to obtain a data point and then apply the classifier to it. Next, we project this point to 2D and repeat the backprojection-and-labeling process until the class label changes or a maximum number of iterations (set to $k_{max} = 10$ in our experiments) is reached. Let S be an image recording this maximum iteration count (normalized by k_{max}) that keeps the class label constant at every map pixel. We then define the class stability $\bar{S}$ as the average of S over all pixels, with values in $[0, 1]$. A value $\bar{S}$ close to 1 indicates a stable decision map with consistent class assignments through multiple P and P^{-1} round-trips. Conversely, a value $\bar{S}$ close to 0 suggests an unstable decision map, sensitive to distortions introduced by (inverse) projection, potentially misrepresenting the classifier's decisions. As explained next in Sec. 4.3.2, we also directly visualize S to get local insights in the class stability over the map.

Average gradient $\bar{G}$: Since P^{-1} is a function of two variables (the x and y coordinates of a pixel), one can measure its gradient magnitude G at every pixel $\mathbf{p}$ by central differences (Espadoto et al., 2021a) as

$$G_x(\mathbf{p}) = \frac{P^{-1}(\mathbf{p} + (w, 0)) - P^{-1}(\mathbf{p} - (w, 0))}{2},$$

$$G_y(\mathbf{p}) = \frac{P^{-1}(\mathbf{p} + (0, w)) - P^{-1}(\mathbf{p} - (0, w))}{2},$$

$$G(\mathbf{p}) = \sqrt{\|G_x(\mathbf{p})\|^2 + \|G_y(\mathbf{p})\|^2}, \tag{4.5}$$

where w is a small step size, set to the size of one pixel for all our experiments. Large gradient values indicate pixels where the decision map has a high likelihood of being incorrect since neighboring pixels correspond to faraway data points, thus, data points can be classified differently. Separately, discontinuous changes in this map indicate areas where P^{-1} is not smooth, where we likely expect to see errors in the decision map.

We can reduce G to a scalar value by computing the average $\bar{G}$ of the normalized values $G(\mathbf{p})/G_{max}$ over all the decision map pixels, where G_{max} is the maximal value of G over the set of decision maps being compared. The value $1 - \bar{G}$, ranging in $[0, 1]$ thus signals how smooth a decision map is. Values close to 0 indicate a smooth map (with low average gradients). Values close to 1 indicate a map having many discontinuities – thus, more prone to errors, as explained above.

4.3.2 *Local metrics*

The metrics presented so far aggregate the quality of a decision map to a single scalar number. While simple to interpret, such metrics only give a *global* assessment of the decision map. Since typical direct projections and, in any case, inverse projections are nonlinear functions, large errors can occur *locally* in such maps. Such local errors – if not too numerous – will not show up in global metrics. Moreover, the *position* of such local errors is very important for interpreting a decision map. For example, errors appearing close to a decision boundary will influence the shape of this boundary and, subsequently, how one uses the map to interpret and/or improve a given ML model.

To get more insight into such local phenomena, one can use so-called *local metrics*. Introduced for studying direct projections (Aupetit, 2007), these metrics evaluate the quality of a map at every 2D spatial position and typically display the result as a color-coded visualization. We propose three local metrics to assess the quality of decision maps, as follows (see also Figure 4.2):

Gradient map: We display the gradient map G, computed as explained in Sec. 4.3.1, to help understand the smoothness of the inverse projection and, as outlined earlier, check for the presence of high-gradient regions which are prone to creating errors in the map. Figure 2.13 shows a typical gradient map and how to interpret it. Simply put, dark regions indicate areas of low gradients G, where the decision map is reliable; bright regions indicate areas of high gradients G, where the decision map is very likely unreliable.

Distance to boundary: Due to the nonlinearity of the mappings P and P^{-1}, if a pixel is visually close to a decision boundary *in the map*, this does not necessarily mean that its corresponding data point is close to the classifier's decision boundary *in the data space*. Yet, as already mentioned in Chapter 2, a key aim of decision maps is precisely to indicate points which are close to decision boundaries, since these are prone to misclassifications upon slight changes in the data or model parameters.

We alleviate this by computing the distance to the boundary using DeepFool (Moosavi-Dezfooli et al., 2016), as described by Machado et al. (2024). An adversarial example for a given classifier is a synthetic data point $\tilde{\mathbf{x}} = \mathbf{x} + \Delta\mathbf{x}$ such that $f(\tilde{\mathbf{x}}) \neq f(\mathbf{x})$ with $\|\Delta\mathbf{x}\|$ as small as possible. Here, $\Delta\mathbf{x}$ indicates a small perturbation, or change, of the sample $\mathbf{x}$. In practice, it is complicated to generate the smallest perturbation possible that generates an adversarial example, so DeepFool approximates it instead for every map pixel $\mathbf{p}$ as

$$d_B(\mathbf{p}) = \min_{\Delta\mathbf{x}\in\mathbb{R}^n} \left\{ \|\Delta\mathbf{x}\|_2 \mid f(P^{-1}(\mathbf{p}) + \Delta\mathbf{x}) \neq f(P^{-1}(\mathbf{p})) \right\}, \quad (4.6)$$

In other words, $d_B(\mathbf{p})$ tells how close $P^{-1}(\mathbf{p})$ is to a change in the class predicted by f, *i.e.*, which is the distance between the backprojection of $\mathbf{p}$ and the closest decision boundary in data space. Note that, to compute the above, a differentiable classifier is needed. For this, we use a Logistic Regression classifier, implemented in PyTorch.

Distance to data: Besides knowing how close a decision-map pixel $\mathbf{p}$ is to its closest decision boundary, it is also useful to know how close such a pixel is (via backprojection) to the nearest data point in the training set D_t used to construct f. Following Machado et al. (2024), as introduced in Chapter 2 (Eqn. 2.4), we compute this as

$$d_D(\mathbf{p}) = \min_{\mathbf{x}\in D_t} \|P^{-1}(\mathbf{p}) - \mathbf{x}\|, \quad (4.7)$$

that is, the smallest distance between the data point $P^{-1}(\mathbf{x})$ corresponding to the pixel $\mathbf{p}$ and samples in the training set D_t. Interpreting d_D works as follows: Decision map pixels $\mathbf{p}$ which are close (in 2D) to projections $P(\mathbf{x})$ of training-set points $\mathbf{x} \in D_t$ should represent data points which are close to such $\mathbf{x}$. If this is not the case, *i.e.*, if we see high $d_D(\mathbf{p})$ values for pixels close to the training-set projection, it means that the decision map has issues there with extrapolating from the training-set – that is, it takes classifier values from points *far* from the training-set and depicts them *close* to the projection of the training-set.

Class stability map: As outlined in Section 4.3.1, the class stability map S can act as a local metric. As explained there, pixels with high S values will have the same class label even after multiple round-trip P and P^{-1} iterations. When a pixel is close to a decision boundary, S will likely be lower since the chance that a point there 'jumps' on the other side of a decision boundary due to such round-trips increases. However, if the pixel is far from a decision boundary, its S value should be higher. If this is not the case, then the decision map may be unreliable in such areas. As such, visualizing how S matches the distances to the depicted boundaries in a decision map tells us about confident we are about the quality of that map.

4.3.3 *Datasets*

We evaluate the three decision-map techniques (DBM, SDBM, and DV) using both a synthetic dataset and real-world datasets. The real-world datasets are chosen following the same criteria as in Oliveira et al. (2022). They are chosen to be openly accessible, representative of different types of data (e.g., time series, image, text), and to have different numbers of classes and dimensions. The datasets are listed as follows and summarized in Table 6.

Synthetic Blobs: This is a synthetic dataset with 5 classes, 100 dimensions, 1500 samples. All data points in a blob (following a Gaussian distribution) have the same label. Therefore, it is an easily classifiable dataset, for which we expect all three decision-map techniques to produce good-quality metric values.

Human Activity Recognition (HAR) (Anguita et al., 2012): This a dataset with time-series data from smartphone sensors. The goal is to classify the type of physical activity (e.g., walking, climbing stairs) performed by the user. This dataset has 10299 samples, 561 dimensions, and 6 classes.

MNIST (LeCun et al., 2010): This dataset is a collection of handwritten digits that is commonly used for training various image classification systems. The dataset contains 60,000 training images and 10,000 testing images. Each image is a 28x28 grayscale

Table 6: Datasets used in our decision map evaluations and their properties.

Dataset	Type	$\lvert D_t \rvert$	$\lvert D_T \rvert$	Dimensionality	No. of Classes
Synthetic Blobs	Synthetic	1000	500	100	5
HAR	Time Series	5000	2352	561	6
MNIST	Image	5000	5000	784	10
Fashion MNIST	Image	5000	5000	784	10
Reuters Newswire	Text	5000	2432	5000	6

image, associated with a label from 0 to 9. The dataset is downsized to 10k for all our experiments.

FashionMNIST (Xiao et al., 2017): A dataset of Zalando's article pictures, with images of 10 fashion categories. It has the same size as MNIST, and is downsized to 10k samples for our experiments.

Reuters Newswire Dataset (Thoma, 2017): This dataset contains 8432 samples of news report documents, from which 5000 features were extracted using the standard TF-IDF (Salton and McGill, 1986) text processing method. From the full dataset, we use only the 6 most frequent classes to favor the creation of easily-interpretable decision maps.

4.3.4 *Classifiers*

We evaluate decision maps for four classifiers – Logistic Regression (Cox, 1958), Random Forest (200 estimators) (Breiman, 2001), Neural Network (having 3 hidden layers with 200 units each), and Support Vector Machines (SVM, with an RBF kernel) (Cortes and Vapnik, 1995) – which are all extensively used in machine learning. They represent different families of algorithms: Logistic Regression is a linear classification model; Random Forest is an ensemble method; Neural Network represents deep learning; and SVM is a maximum margin classifier. Importantly, these classifiers are frequently studied in existing decision map research, thus allowing for meaningful comparisons. The four machine learning classifiers are implemented using *scikit-learn* (Pedregosa et al., 2011).

We train all four classifiers on 5000 samples from the four real-world datasets and use the remaining samples for testing. We construct the corresponding three decision maps for each combination. As a result, we get $4 \times 4 \times 3 = 48$ combinations of datasets, classifiers, and decision maps to study.

4.4 COMPARISON RESULTS

We next discuss the results of our evaluation metrics for the constructed decision maps.

4.4.1 *Global metrics of real-world datasets*

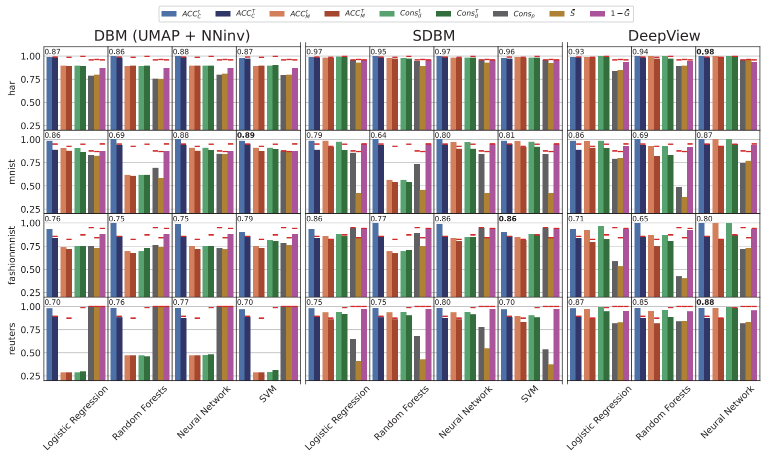


Figure 4.3: Aggregated global metrics for each combination of decision map, classifier, and dataset. Herein, ACC_C, ACC_M, $Cons_d$, $Cons_p$, $\bar{S}$, $1 - \bar{G}$ are the global metrics defined in Sec. 4.3.1.Red dashes '-' show the highest values across each dataset (row). Top-left numbers in each plot give the average value of $ACC_C{}^T$, $ACC_M{}^T$, $Cons_d{}^T$, $Cons_p$, and $\bar{S}$. Bold values indicate the highest value along the dataset (row).

Figure 4.3 shows all global metrics for all combinations of decision map technique, classifier, and dataset. Note that DV failed to run with SVM on the real-world datasets, and thus is not included in the figure. From a dataset perspective, the results differ heavily based on the specific dataset being considered. On HAR, SDBM and DV show comparable (high metric) results while DBM got slightly lower scores; on MNIST, DBM shows the best results in all aspects, even though DBM's training does not use label information. on FashionMNIST, SDBM gets the highest scores, while DV shows comparable results for data-level metrics (ACC_C, ACC_M, $Cons_d$) but much lower results on pixel-level metrics ($Cons_d$, $\bar{S}$); finally, on Reuters, the overall winner is DV. From a classifier perspective, the results are more stable. The accuracy of classifiers themselves varies little. The most noticeable difference is that random forests always have lower scores (on all global metrics except ACC_C and $1 - \bar{G}$) than the other classifiers for all considered metrics, particularly on MNIST dataset. For $1 - \bar{G}$, almost all results are close to 1. This is due to a large maximum value of DV which influences the normalization (see the definition of $\bar{G}$ in Sec. 4.3.1). We explore this in more detail next in Section 4.4.3.

The datasets in Figure 4.3 are sorted top-to-bottom on increasing dimensionality (see also Table 6). The higher the dimensions, the more difficult for the models to learn. So, it is not surprising that DBM, which is a fully unsupervised method, has trouble when aiming to depict classifiers for higher-dimensional datasets like Reuters (5000 dimensions). In contrast, for this dataset, SDBM reaches higher quality metrics, while DV reaches the highest values. This suggests that DV is the decision map method of choice – from the perspective of global quality metrics – for high-dimensional datasets. However, as we shall see next, other aspects weigh the three decision-map methods differently.

4.4.2 *Interpreting local metrics on synthetic data*

We start explaining the local metrics proposed in Sec. 4.3.1 using the simple synthetic blobs dataset which is, as explained earlier, straightforward to classify. As such, we only study its decision maps for the simple Logistic Regression classifier (we obtained very similar results for the other three considered classifiers). The blobs dataset is split into 1000 training and 500 testing samples. All three map methods – namely, DBM, SDBM, and DV – achieve 100% on the test set for all data-wise global metrics. We next use the blobs dataset to establish and validate our *expectations* for a 'good' decision map based on the local metrics introduced in Section 4.3.2 as follows:

1. Distance to the nearest data d_D: We expect this distance to be small for pixels close to actual projections of data points and larger for pixels far away from these points. In other words, we expect that the 2D distance (to projections of data points) to mimic the nD distance (to actual data points).

2. Distance to the decision boundary d_B: Similar to the above, we expect that points close to decision boundaries (in 2D) are also close to decision boundaries (in nD) and conversely.

3. Class stability map S: Ideally, we would like to have most pixels with high stability values, especially close to decision boundaries (where we are most interested in studying a decision map).

4. Gradient map G: We expect (1) a *smooth* gradient map without any discontinuities or peaks. Ideally (2), we would also like to have low gradients close to the decision boundaries (for the same reason mentioned above for class stability).

With these expectations in mind, we analyze the results shown in Figure 4.4 for the three decision-map methods and the above-mentioned four local metrics. The first row in Figure 4.4 shows the decision maps and projected data points for the three methods. At a glance, all maps appear similar. However, a deeper analysis of the local metrics reveals more. The results for d_D (second row) and d_B (third row) meet our expectations across all methods. Some discontinuities in DV, however, are noticeable – the d_D and S images appear to contain some 'cuts' that perturb their overall smoothness. S (fourth row) primarily manifests as expected, with all low-value pixels situated around the decision boundaries. A notable difference arises with SDBM showing larger regions of unstable pixels, indicating its vulnerability to projection errors. For G, DBM and SDBM are smooth, whereas DV exhibits peaks, thus not satisfying our first expectation of G. Even more interestingly, the local maxima for G for DV takes the shape of *lines* roughly connecting the clusters of projected points. As for our second expectation, we find a surprising result: DBM and SDBM show relatively higher values close to decision boundaries, while DV shows lower values in these areas. Consequently, no method satisfies both expectations of G simultaneously.

In conclusion, on the synthetic blob dataset, all three methods generally align with our expectations of local metrics. However, despite the decision maps' similarities for this simple example, there are subtle but significant differences between them. This is the first indication that, while superficially similar, the three studied decision map techniques behave quite differently. We examine this aspect further on real-world data.

4.4.3 *Analyzing local metrics on real-world data*

To refine our preliminary insights concerning the differences between the three studied decision-map techniques, we now use our four proposed local metrics to study their behavior on the

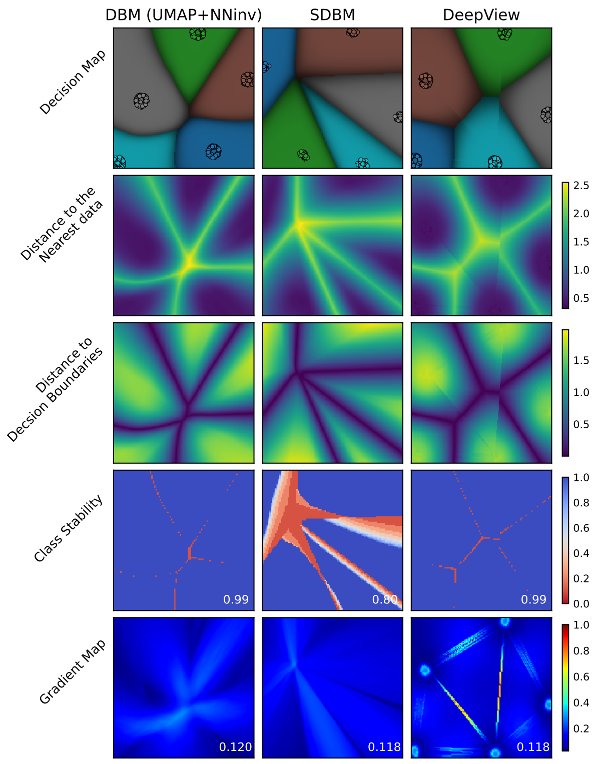


Figure 4.4: Decision maps and local metrics of the synthetic blob dataset (see Sec. 4.4.2). Row 1: Decision map with data points projected onto it (marked black). The brightness of pixels encodes the confidence of f's prediction (dark=low, bright=high confidence). Row 2: Distance of each map pixel to the nearest data point d_D depicted using a blue (low distance) to yellow (high distance) colormap. Row 3: Distance of each map pixel to the closest decision boundary d_B, using the same colormap as in row 2. Row 4: Class stability map S depicted using a red-white-blue colormap. Blue values indicate stable pixels, while red ones pixels for which fewer direct-and-inverse projection round-trips change the classifier's decision depicted at that pixel. Row 5: Gradient map G showed using a rainbow colormap. Blue and red pixels indicate low, respectively high, values of the derivatives of the inverse projection function.

four real-world datasets in Tab. 6. We start by presenting the actual decision maps (Sec. 4.4.3.1). The next sections (Sec 4.4.3.2-4.4.3.5) interpret these maps using our four local metrics.

4.4.3.1 *Decision Maps*

Figure 4.5 shows the decision maps computed using training, respectively testing, data. The decision zones are color-coded categorically, while the brightness of pixels encodes the confidence of f's prediction (as used earlier in all decision-map techniques). The projected samples are also color-coded according to their class, but made slightly brighter to distinguish them from their surrounding zones. Misclassified samples, *i.e.*, samples for which $f(\mathbf{x}^i) \neq y^i$, are marked with a white outline. Theoretically, points near decision boundaries represent samples about which the classifiers are most uncertain. This can be observed at the boundaries between the pink and the yellow zone in the HAR dataset, where some misclassified points are highlighted with white outlines. Notably, there are even some misclassified light blue points at the boundaries of the pink and yellow zones. Knowing the nature of this dataset – classification of human activities – this observation aligns well with the understanding that static activities (corresponding to the class labels yellow and pink). are more challenging to distinguish than dynamic activities, such as various walking activities.

We see that each decision-map technique has its own 'signature': DBM is more random; SDBM shows radial patterns; and DV presents blob-like patterns. Also, we see that prediction confidences are always lower near decision boundaries and higher within decision zones, which is expected. In more detail, DBM shows some 'island' decision zones that have no data points. Without more information, it is hard to tell whether these islands actually do exist in the high-dimensional data or are artifacts of the decision map. In contrast, DV shows some discontinuities, indicated by 'breaks' or 'jumps' in the decision boundaries. Regarding the smoothness of decision boundaries, SDBM displays the smoothest ones; DBM has more complex but still smooth boundaries; and DV exhibits the least smoothness. The shapes of the decision zones seem to be strongly influenced by the underlying projection method P: SDBM consistently shows radial, elongated, star-like structures (a property known to autoencoders used for dimensionality reduction); DV presents well-separated, blob-like, structures (highly likely due to its use of discriminative dimensionality reduction); and DBM has more variable structures (due

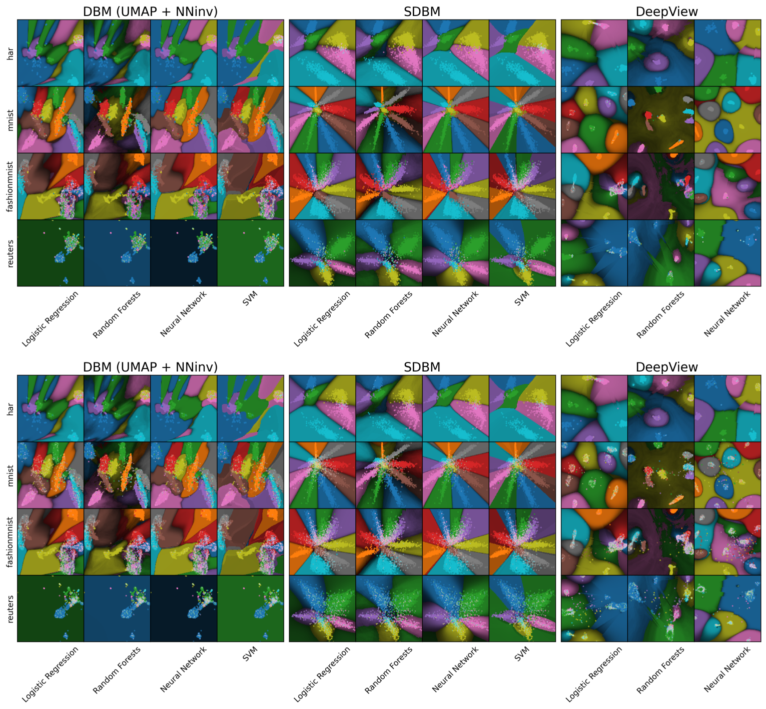


Figure 4.5: Decision maps with training data (top) and test data (bottom).

to its UMAP projection). It is worth noting that DBM failed on the Reuters dataset, as also reflected by the low global metrics score in Figure 4.3.

Separately, we examine the issue of overfitting. By displaying both training and test data, we can check whether decision maps can visually indicate overfitting. This overfitting is reflected by the noisier scatters plotted on incorrect background colors in the test sets (Figure 4.5 bottom). For the test data, we observed overfitting in all three methods across all four datasets, except for the simplest one, the HAR dataset. Among the three methods, DV appears more prone to overfitting, a fact also evident from the noticeable difference between ACC_M^t - ACC_M^T and $Cons_d^t$ - $Cons_d^T$ in Figure 4.3.

4.4.3.2 *Smoothness*

The gradient map G, as defined in Equation (4.5), captures the smoothness of decision maps. Figure 4.6 shows this map for the three studied decision-map techniques. We see that each tech-

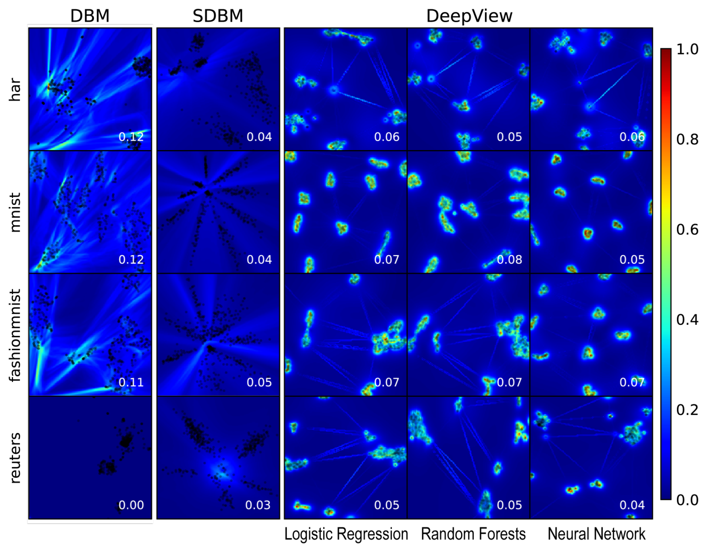


Figure 4.6: Gradient maps G of the studied decision map methods. Gradient values are scaled to [0,1] within each dataset. The number in the bottom right corner of each plot is the average $\bar{G}$ of each map.

nique creates its unique type of gradient, as follows. The DV technique shows high gradient values close to the projected points, which indicates a high degree of data compression during the projection P in these areas. Additionally, the DV technique shows some high gradient 'lines' connecting the clusters of projected points, indicating discontinuities. The remaining areas have uniformly low gradient values. In contrast, SDBM shows low gradient values almost everywhere, with slightly higher values in the gap areas between point clusters. Both the above patterns for DV and SDBM are similar to what we observed on the blob dataset in Figure 4.4. Finally, DBM's gradient map presents a more complex pattern, entirely different from what we observed on the easily classifiable blob dataset (Figure 4.4). As stated in Section 4.4.2, we expect gradient maps to not show peaks and to have low values in areas close to decision boundaries. However, *no* method simultaneously exhibits these two properties. Notably, SDBM's relatively high gradient values in the gap areas between point clusters are still low in absolute terms, making SDBM the method that best satisfies the two mentioned gradient-map properties. SDBM's high smoothness can be attributed to the joint training of P and P^{-1}. DV is clearly the least smooth method, as indicated by the number of discontinuities reflected by the 'peaks' in its gradient map. DBM falls in between, with its complex pattern showing less smoothness but at least continuity.

4.4.3.3 *Class stability map*

As explained earlier in Sec. 4.3.1, the global average value for class stability $\bar{S}$ indicates whether a decision map is robust to (inverse) projection errors. The class stability map S, which we study next, reveals *where* the decision map is susceptible to such errors. Figure 4.7 shows the S maps for the three studied decision-map techniques. As explained in Section 4.4.2, pixels with low S values are expected to appear primarily near decision boundaries. This assumption holds true, especially for simpler datasets such as the HAR dataset.

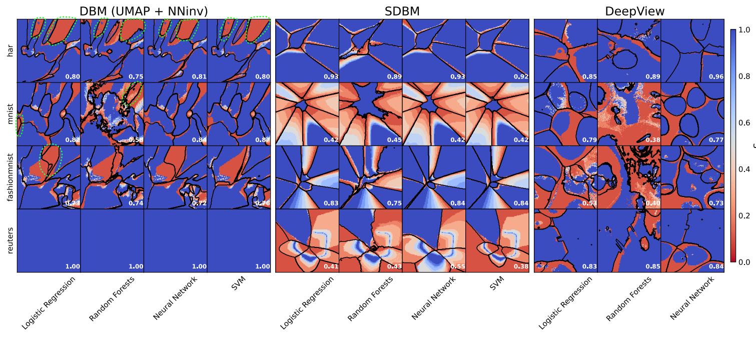


Figure 4.7: Class stability map S. The number in the bottom right corner of each plot is $\bar{S}$ of each map.

However, some anomalies can be seen for each decision-map technique: DBM displays large zones with low S values. Referring to the decision map (Figure 4.5), most of these zones are *no data zones* (NDZs), in which no data points are projected – see the zones circled by dotted green lines in Figure 4.7 (leftmost image). Another interesting observation is that the most unstable pixels tend to change their labels *immediately* after the first round-trip in DBM. This observation also applies to DV. Unlike these two methods, SDBM continues to be affected by round-trip errors. This can be observed by the gradual changes in S values for SDBM in Figure 4.7. After several rounds, most of the pixels in SDBM change their labels, except for those with the most robust labels, which almost never change. DV presents another salient pattern: Given the blob-like patterns in this decision map, a particular class often forms the 'base' of the blob-like shapes in some cases, *e.g.*, the MNIST dataset with Random Forest and Neural

Networks. Pixels in these areas are more prone to round-trip errors. In summary, SDBM performs the worst on S, as also indicated by the result global metrics (Figure 4.3) and the synthetic blob dataset (Figure 4.4). DBM and DV show similar patterns S, with DBM slightly outperforming DV on the MNIST and FashionMNIST datasets. However, *all* three methods are quite prone to instabilities and many such instabilities exist along their decision boundaries – which, as explained earlier in Sec. 4.4.2, is an undesirable aspect for trustworthy decision maps.

4.4.3.4 *Distance to decision boundary*

A key application of decision maps is to show how close a point on the map is to the actual decision boundary of the studied classifier. For this, the 2D distances in the map should depict as closely as possible the corresponding nD distances. Figure 4.8 (left) shows, for each map pixel, its distance to its closest decision boundary in the high-dimensional space, computed via Eqn. 4.6. From this figure, we see that our expectations align with the observations – pixels close to the decision boundaries in the map are dark (meaning, they map points close to the actual decision boundaries in nD), while pixels deep in the decision zones are bright (meaning, they map points far away from the nD decision boundaries). Yet, we also see variation in these distances across the decision-map methods. SDBM and DV display patterns closer to the desired outcome than DBM. DV shows a wider area with low distance values, while SDBM shows a more concentrated area. This follows the patterns observed in gradient maps (Figure 4.6), where DV expands the gaps between point clusters, whereas SDBM compresses them. DBM exhibits a complex pattern, where certain zones display low distance values entirely, even for pixels located in the center of the decision zones. These zones are particularly noticeable in the upper areas of the HAR and FashionMNIST datasets, as well as the leftmost area of the MNIST dataset. Interestingly, these zones (circled by red dotted lines in Figure 4.8 left) are also NDZs, which coincide with the zones of low S (see Figure 4.7). This correlation indicates reduced confidence in the inverse projection in those zones. The importance of confidence in extrapolation is underscored by these findings and will be discussed further in Sec. 4.4.3.5.

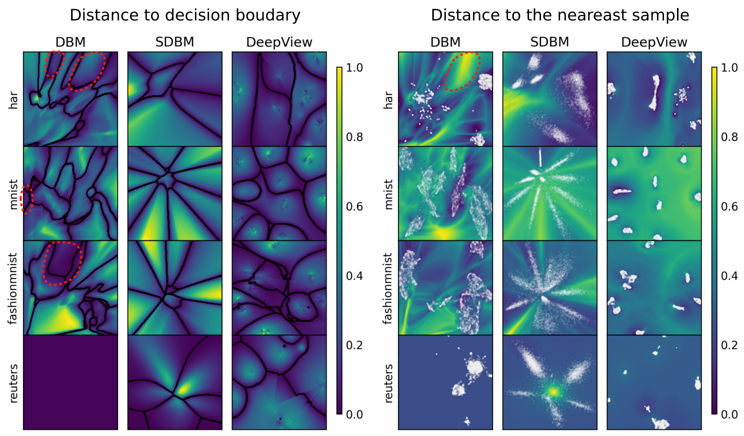


Figure 4.8: Left: Distance to decision boundary d_B; Right: Distance to the nearest training data d_D. Projections of training data are shown as white dots.

4.4.3.5 *Distance to the nearest training data*

The distance to the nearest data point d_D (Eqn. 4.7) indicates how far the inverse projection for a given pixel is from the actual data distribution. If an inverse projection is significantly far from this data distribution, t likely corresponds to a point where the classifier will have generalization difficulties. Ideally, a decision map should (1) not contain points which are far away from the actual training or testing points and (2) pixels close to these points in the map should actually also be close to the respective data points.

Figure 4.8 (right) shows the distance d_D for the three decision-map methods. Data points are marked as white dots. For DV, pixels near the data points exhibit low distance values, as expected. However, this pattern is less evident for DBM and SDBM. In the SDBM representation of the FashionMNIST dataset, a zone on the right side displays high distance values, even though a data point cluster is projected there. For DBM, an entire region in the HAR dataset map also shows very high distance values. Different from SDBM's case, this is an NDZ again, which also has low S values and low d_B in the entire zone (see Figure 4.7 and Figure 4.8 (left) respectively). This high-value region, which is significantly distant from the data distribution, indicates that the inverse projections in this zone are unlikely to align with user expectations based on the class labels. This might be due to the square shape of the 2D map. Depending on the distribution of the data, the inverse projection is likely not to populate the square region uniformly. In this case, the inverse projection has to extrapolate certain pixels which correspond to locations further away from the training data. This scenario underscores the value of the distance d_D in offering insights into potential issues.

Another interesting observation is that the pattern of the d_D metric is prone to outliers. For instance, we see some ripple-like patterns in HAR with DBM and DV. Although these patterns appear slightly different in each case, they are consistently caused by the presence of outliers. Other NDZs not showing high d_D values might be caused by the same reason.

Furthermore, a closer examination of the data point locations (white dots) reveals that not all close decision-map pixels correspond to data *samples* which are close to each other. That is, the 2D distance we see on the map is not the same as the high-dimensional distance we have in the data space. In other words, decision-map pixels equally (and very) close to data points actually depict points at various distances from such data. This can lead to interpretation problems of the decision maps created by *all* three methods – more so for DBM and SDBM, where this pattern is more visible, and less for DV.

4.4.4 *Computational efficiency*

We measured the training and inverse projection time of the three studied decision map techniques on an Intel Core i7-12700 CPU machine with 32GB of RAM and an NVIDIA GeForce RTX 3070 GPU with 8GB of RAM. For this, we used the synthetic blob dataset with varying dimensions (10-500) and numbers of samples (250-5000). Particularly, we evaluated DV, which requires a pre-trained classifier, with Logistic Regression, Random Forest, Neural Network, and SVM. It is important to note that DV's training time varies based on the classifier used (as detailed further below).

Figure 4.9 shows the training times for the compared methods (that is, the time needed to construct the functions P and P^{-1}). When the number of samples d_D and dimensions n are both small ($N < 1000$, $n < 50$), the training time of DV (with Logistic Regression) is comparable to SDBM and DBM. For larger n and/or d_D, comparable to the real-world datasets (Table 6), DBM and SDBM showed comparable training times. DV, however, had substantially higher training times. SDBM remained the fastest method, with average training times of less than 10 seconds across all tested scenarios. DBM's training time is affected by the number of samples d_D but far less by the number of dimensions n. DV's training time, when using SVM, increased drastically for larger n and/or d_D – training DV with SVM for $n = 500$, $N = 5000$ took over 2.7 hours. This steep increase made SVM-based DV experiments unfeasible for our real-world datasets.

Figure 4.10 shows the time required to create the decision maps (given a trained direct and inverse projection P and P^{-1}) for various numbers of dimensions n, samples d_D, and grid resolution I. As for training, DV took the longest and was heavily influenced by all these parameters, even failing to handle resolutions over $I = 150$ pixels squared. Conversely, both DBM and SDBM had a relatively high and consistent performance, roughly linear in the number of pixels in the map and the number of samples d_D. Summarizing all the above, we conclude that DBM and SDBM are practical methods for creating decision maps for real-world datasets, while DV is not.

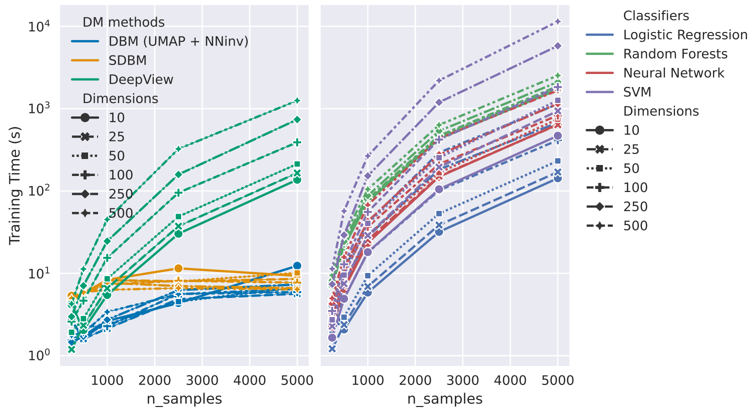


Figure 4.9: Time to train P and P^{-1} pairs. Left: Training time of DBM, SDBM, and DV with Logistic Regression. Right: Training time of DV with 4 different classifiers.

4.5 DISCUSSION

4.5.1 *Decision Maps for Deep Learning Variations*

The previous evaluations have covered the way the three studied decision map methods – DBM, SDBM, and DeepView – perform over a wide range of classification models. However, in the respective study, we used a single such model based on a deep learning architecture (see Sec. 4.3.4). It can be argued that, with the increasing prominence of deep learning, it is especially impor-

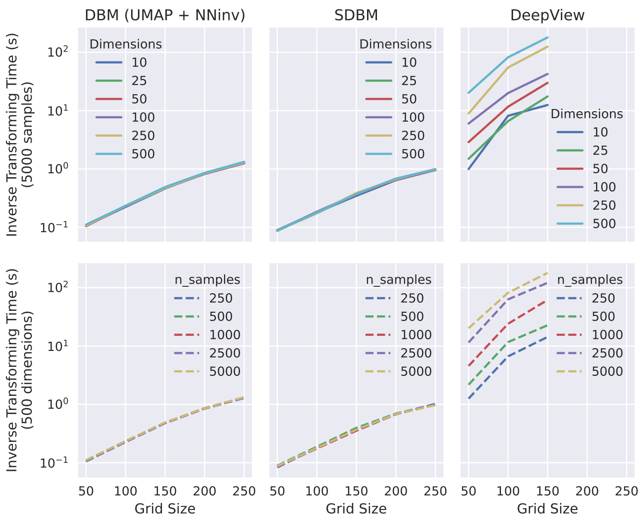


Figure 4.10: Time to create decision maps for DBM, SDBM, and DV.

tant to gather insights on how decision map methods perform on this particular family of algorithms.

We cover this desiderate next by studying, separately, the decision maps created for four such deep learning architectures, as follows. First, we consider the original, and relatively simple, neural network architecture already used in the previous evaluations, for comparison purposes. This architecture has three fully-connected layers of 200 units each. Note that, albeit small, it is precisely the type of architecture used earlier in all deep-learning applications for computing direct projections (Espadoto et al., 2020), inverse projections (Espadoto et al., 2021a), and decision maps (Oliveira et al., 2022). We then used two larger variants of this architecture having four layers of 1024 units, respectively four layers of 2048 units. Finally, we used TabNet (Arik and Pfister, 2021), a very recent architecture designed for tabular data, which combines the principles of decision trees and neural networks and uses attention mechanisms (Vaswani et al., 2017) to prioritize important features during decision-making. We did not include in this comparison more specialized architectures like Convolutional Neural Networks (CNN) (LeCun et al., 1989), Recurrent Neural Networks (RNN) (Elman, 1990), Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997), or Deep Belief Networks (DBMs) (Salakhutdinov and Murray, 1998). While these architectures are in themselves quite powerful, they are most often designed for being used with one particular type of data, *e.g.*, images or time-dependent signals.

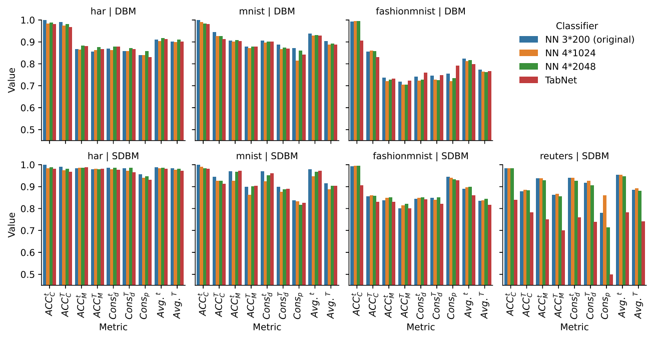


Figure 4.11: Label related metrics for different neural networks. NN $m * n$ denotes a neural network with m hidden layers of n units each.

Figure 4.11 shows the performance metrics for three decision map techniques trained to classify four datasets using the above-mentioned four deep learning architectures. We see that, across the four deep learning techniques – that is, within one set of four-colored bars in the respective plots – is quite similar. The large differences which occur are, rather, dependent on the dataset or metric, much as we have seen for the earlier-evaluated architectures (see Fig. 4.3). The only outliner in this respect is TabNet when run on the Reuters dataset to compute the SDBM map and, up to a much lesser extent, when run on the FashionMNIST dataset to compute the DBM map. Let us examine these two situations separately next.

In the latter case, TabNet has low ACC_C but relatively higher $Cons_d$ and $Cons_p$ values. This demonstrates the situation when the classifier f performs poorly and the decision map has good quality. This is a very good example of the use of DBMs – the maps can serve as a reliable indicator of the classifier's underperformance.

In the former case, TabNet shows clearly lower scores in all metrics. If we look at the actual decision maps (Figure 4.12), we see that these are, for all four architectures, overall quite similar for the same dataset. Moreover, these maps look quite different than those of the other classifiers depicted in Figs. 4.4 to 4.6 – apart from neural network classifier to the two sets of images, of course. In other words, the variability of decision maps is much smaller over one type of architecture (neural networks) than across different architectures, which is expectable. However, Figure 4.12 also shows some subtle differences between the maps, in the middle of the depicted data clusters, that is, close to the locations where the decision boundaries appear. This indicates that the four studied deep learning classifiers are, indeed, behaving slightly differently in the most uncertain areas. Also, we see that, the more complex a model is, the more complex its decision boundaries are. For example, in the case of SDBM-HAR, the decision boundary along the green zone is relatively simple for the NN $3 * 200$ model. For the NN $4 * 1024$ model, the dark blue intersects between the green zone and the other zones (pink and yellow). Further, for the most complex model NN $4 * 2048$, the pink zone thrusts into the middle of the green zone and the light blue zone. Finally, with TabNet, the decision boundary of the green zone becomes rugged, and the whole decision boundaries become more complicated. This can be explained by the fact that the more complicated the classifier is, the more chance it will overfit the given training data. While this aspect is known in machine learning, the fact that we can show it directly using decision maps has, to our knowledge, not been done earlier.

Summarizing the above, our findings highlight the consistent performance of decision map methods across different datasets and classifiers. The absence of a clear best method for computing decision maps underscores the importance of a comprehensive selection workflow for decision maps. We propose such a workflow for choosing the decision map method in the next section (see Sec. 4.5.2).

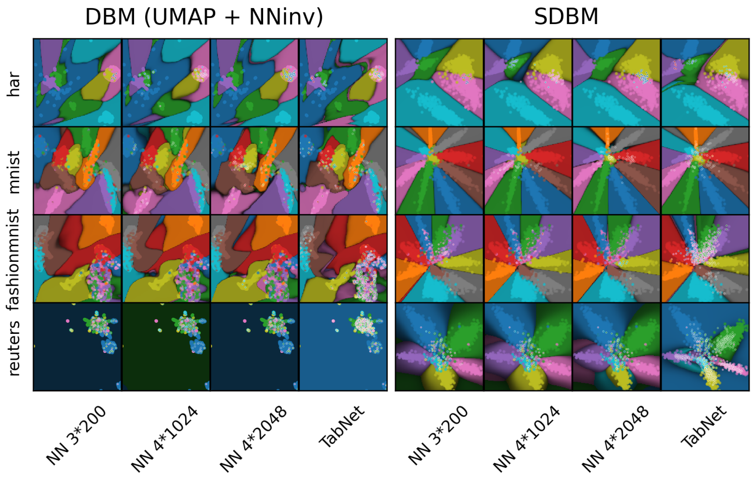


Figure 4.12: Decision maps for different neural networks. See Sec. 4.5.1.

4.5.2 *Workflow to guide the selection of decision map techniques*

Putting together all the results of our evaluation, we see that there is no clear winner among the studied decision map techniques. When we consider the global quality metrics (Sec. 4.3.1), we found that each method reached its maximal quality on different datasets. Surprisingly, the unsupervised method (DBM) sometimes scored higher than the supervised ones (SDBM and DV) on the MNIST dataset. However, DBM's failure on the Reuters dataset indicates that more challenging datasets may still require some level of supervision.

The results of the local metrics bring more insights that lead us to the no-winner conclusion. For the synthetic dataset, all methods showed quite similar patterns, except for DV's poor smoothness and SDBM's slightly worse class stability. For real-world datasets, the patterns are less distinct. Yet, some trends can be discerned. The smoothness of DV is consistently low, irrespective of the simplicity or complexity of the datasets. Additionally, DV and SDBM consistently show more representative distances (d_B and d_D) than DBM.

Concerning speed, SDBM and SBM are clear winners – they definitely surpass DV in both training and inference (map construction) time by one up to three orders of magnitude, which makes the latter not suitable for creating decision maps in real-world application scenarios, especially when using more time-consuming classifiers such as SVM.

Compared to SDBM and DV, DBM has a separate advantage. As outlined in Section 4.2, DBM allows one to use any chosen direct projection function P to create decision maps. This can be important in cases where one knows that a given P is optimal, either for quantitative reasons or because it creates projections which are easier to interpret by users.

In conclusion, since there is no clear winner, the choice of the decision map method should be guided by the specific requirements and constraints of individual use-cases. We capture this by proposing a workflow for choosing a method to create decision maps as illustrated in Figure 4.13. Given a specific dataset and a set of potential classifier candidates, the workflow assists users in making the choices following the steps below:

1. If the user has already chosen a projection function P, they should select DBM, as this is the only method that can accommodate a predefined P, and proceed to step 4.

2. If the user does not have a specific P, the next key aspect to consider is computational efficiency. If speed is important and the data to be visualized is large, DV should be excluded from consideration, and the workflow proceeds to step 4.

3. If the user does not have a specific P, and computational efficiency is not a concern, one should consider if smoothness is important. If yes, DV should be excluded, and one can proceed to step 4.

4. If more than a single classifier-decision map combination remains to choose from, global quality metrics can be used to select the optimal one. The key metrics to use here are ACC_C, ACC_M, and $Cons_d$ (defined in Sec. 4.3.1), which can be computed for any combination of direct projection, inverse projection, and classifier. Note that $Cons_p$ and $\bar{S}$ are not available when the selected projection P cannot infer new data, such as for the case of non-parametric, non-out-of-sample, projections like t-SNE.

5. Finally, visualizations of local metrics can be used to gain more trust and/or understanding of the behavior of the chosen decision map, such as described in detail in the scenarios in Sec. 4.4.

4.5.3 *What decision maps really are*

A key observation running through all our experiments so far is that decision maps are imperfect instruments that map a *part* of the high-dimensional space of a classification model to a 2D surface or map. While our various metrics have shown differences between the three evaluated decision-map methods, it is still unclear how these 2D maps are created. To gain more insight in this, we consider a simple experiment: We create a three-dimensional dataset having 6 concentrated blobs (data point clusters), each of a separate class. Next, we use the three studied techniques to construct the respective decision maps for a given classifier (note that the classifier choice is not important next). Finally, we backproject the map pixels to the data space, which is three-dimensional in our case, and directly visualize the obtained set of 3D points.

Figure 4.14a-c show these 'backprojected' decision maps for DBM, SDBM, and DV, with colors mapping the six classes. Surprisingly, in all cases, these zones appear as residing on a *surface*,

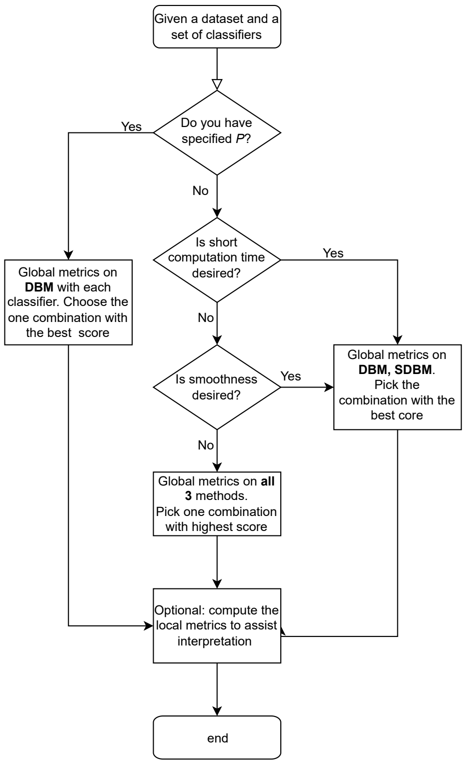

Given a dataset and a set of classifiers
Do you have specified *P*?
Yes
No
Global metrics on **DBM** with each classifier. Choose the one combination with the best score
Is short computation time desired?
Yes
No
Global metrics on **DBM, SDBM**. Pick the combination with the best core
Is smoothness desired?
Yes
No
Global metrics on **all 3** methods. Pick one combination with highest score
Optional: compute the local metrics to assist interpretation
end

Figure 4.13: Workflow for choosing the most suitable decision map in a given user application context.

whereas, knowing the structure of the underlying dataset, they should actually be *volumetric* zones that partition the 3D space into six regions corresponding to the six class labels. In other words, the evaluated DBM techniques only choose a very specific two-dimensional surface-like subspace Z' of the entire data space to visualize. For clarity, note that this surface Z' is not the same as the actual decision *boundaries* of the studied classifier. These boundaries cannot be directly shown by the studied DBM visualizations. Rather, only their *intersections* with the artificial surface Z' are shown as the curves that separate different-color patches in Z'.

Image (d) illustrates this for the DBM map shown in image (a). Here, we sketch how the decision zones of three classes (yellow, blue, and purple) would arguably look like in 3D. As said, these are volumetric objects that enclose the training samples of their three respective classes. The border between the yellow and blue zones is the decision boundary B_{yb} between these classes, which is a *surface*. However, only the curve-like intersection $B_{yb} \cap Z'$ of this surface, indicated by the black curve in the figure, is shown by the DBM. Similarly, the border between the yellow and purple zones is the decision boundary B_{yp} between these two classes. However, in this case, the DBM does not show anything, since the surface Z' it constructs does not reach to that area of the 3D space, *i.e.*, since $B_{yb} \cap Z' = \emptyset$ (dotted black curve in the figure). This is due to the finite size of the 2D image that these methods construct.

Figure 4.14e summarizes the above by a simpler, lower-dimensional, 2D sketch (all quantities in Fig. 4.14d thus become one dimension lower). We see here the decision zones (2D yellow and pink surfaces), actual decision boundary B_{yb} (1D curve), surface Z' constructed by the DBM method (1D curve), and the part of the decision boundary that a DBM method can depict ($B_{yb} \cap Z'$, 0D point). As stated earlier, DBM methods only visualize a *subset* (0D point in this sketch) of the actual decision boundaries (1D curves in this sketch).

Summarizing our findings: (1) the way that a DBM method constructs the surface Z' will strongly influence which parts of the actual decision zones of a classifier will be offered for visualization; (2) only a part of the actual decision boundaries of a classifier are visualized by DBM methods; and (3) different DBM methods will produce different decision map visualizations for the same dataset and classifier – therefore, potentially leading to different interpretations. To our knowledge, none of these three findings have been outlined by earlier work on decision maps.

A final observation from Fig. 4.14 is that the above-mentioned surfaces Z' appear to *smoothly* connect the samples D used by the direct projection P that go into generating the inverse projection P^{-1}. Intuitively put, they look like tension surfaces (Colding and Minicozzi, 2006) that pass close to samples in D. This further suggests that, if the data to classify Z lives in high dimensions on a surface, and if D closely samples this surface, DBM methods will work predictably well and, also, deliver similar results – so, the choice of the DBM method to use is less relevant. Conversely, if D contains points that cannot be fit along such a surface – in other words, the sampled phenomenon has intrinsic dimensionality higher than two – DBM methods may generate very different results depending on the actual dataset and DBM algorithm. This matches our earlier observation concerning the challenge of DBM methods to 'squeeze' a high-dimensional space into a 2D image. Designing more refined DBM algorithms that offer users a way to control which part of the high-dimensional space they sample to construct their classifier explanations is, thus, an important direction for future work (see next Sec. 2.5.1).

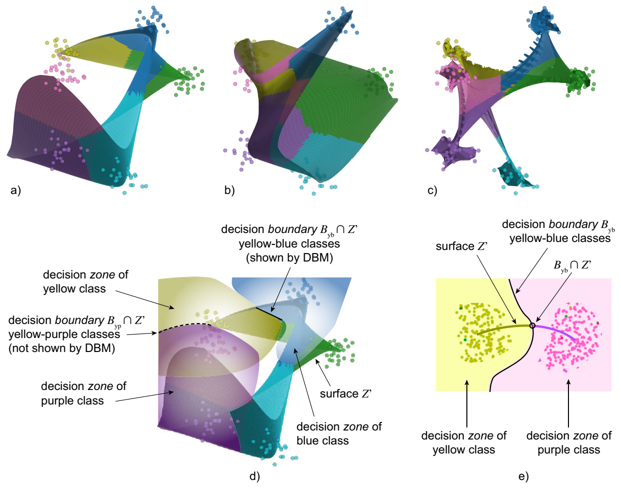


Figure 4.14: Decision map methods construct and visualize the classifier (Logistic Regression) only over an implicitly-constructed *surface* embedded in the high-dimensional space (see Sec. 4.5.3). (a) DBM (Rodrigues et al., 2019). (b) SDBM (Oliveira et al., 2022); (c) DeepView (Schulz et al., 2020). (d) Limitations of decision map visualizations. (e) 2D sketch of (d). We discuss the implications of this important observation in more detail in Chapter 5.

4.5.4 *Limitations*

We single out two limitations of our current work:

Decision maps: While they significantly simplify understanding how a trained ML model works, decision maps require quite some effort to become *actionable* – that is, lead to concrete insights that explain and/or improve the working of a given ML model. In other words, it is still challenging for users to make decisions using decision-maps. Our work helps partially in this direction by helping users to make decision about (a) how and where they trust a given decision map and (b) which decision map technique to next use in practice in a given context. Improvements can, and should, be done to the understandability of decision map visualizations, *e.g.*, explain which parts of a given training set, training process, or model are responsible for visible patterns in such a map. Such extensions are not the scope of this work but are very promising directions for future work.

Evaluation: Our evaluation, covering only five datasets and seven classifiers, is necessarily limited in its generalizability. More models and architectures exist in machine learning. How the studied three decision map techniques (DBM, SDBM, and DeepView) cope with these models is not necessarily the same as for the models we studied so far, so our evaluation can be extended by considering additional models. However, we argue our choice for our *initial* evaluation consisting of the aforementioned datasets and classifiers as follows: (1) As no similar evaluation of decision maps existed before our work, we had to start with relatively simple cases – that is, datasets and classifiers with known behavior. This way, we could use these datasets and classifiers as 'ground truth' to actually assess the produced decision maps. (2) Our limited evaluation already pointed out several key insights such as the very limited computational scalability of DeepView and the fact that *all* decision map methods only visualize a surface subset that passes close to the training samples in the data space. These limitations will exist for any more complex model visualized by the current methods. As such, future work can already focus on removing these limitations before applying decision map techniques to more complex models.

4.6 conclusion

In this chapter, we have presented a framework for exploring and comparing methods for constructing decision maps for visualizing the behavior of general-purpose classifiers of high-dimensional data. To this end, our framework proposes six global metrics and four local metrics to gauge the overall quality, respectively the local quality, of a decision map. We validated our framework by applying it to a simple synthetic dataset for which the expected behavior of the decision map constructed by three state-of-the-art decision map techniques (DBM, SDBM, DV) was known. Furthermore, we compared these three techniques for a combination of four real-world datasets and four classifiers.

Our results showed that there is no decision map method from the evaluated ones that consistently scores better than its competitors in *all* aspects deemed relevant for quality. Furthermore, we outlined that all the studied decision map methods have inherent limitations in various quality aspects and that these limitations can fluctuate quite significantly as a function of the studied dataset and/or classifier being explored. To aid users in choosing a suitable decision map method in a practical setting, we proposed a workflow that considers all studied quality aspects and proceeds by elimination and next optimization of these aspects.

Separately, we showed that all studied decision maps have an inherent, previously unknown, limitation – they only visualize a surface from the entire high-dimensional space. The way this surface is constructed depends on the actual decision map technique. As a consequence, the decision map visualization reflects both the actual decision boundaries in the data and the way these intersect with the surface constructed implicitly by the decision map technique. However, this surface-like behavior is only observed on that 3D datasets with a logistic regression as the classifier. We further investigate this issue in Chapter 5.

Several directions of future work exist, as follows. First, our evaluation can be extended by considering more datasets and classifiers and, when these will appear, more decision map techniques. Second, our finding that decision maps actually visualize a single surface from the high-dimensional space can turn this inherent limitation of decision map techniques to a strength. We can imagine ways to parameterize this implicit surface under user control so as to let it 'slice' through the actual high-dimensional decision boundaries in an interactive way, thereby offering the user the possibility to examine these decision boundaries in a more controlled, and global, way. Chapter 6 will explore this parameterization direction.

5 FUNDAMENTAL LIMITATIONS OF DECISION MAPS

5.1 INTRODUCTION

Chapter 4 (Sec. 4.5.3) explored the behavior of decision maps and their related inverse projection techniques. In particular, an experiment we performed during this evaluation showed that, for a linear regressor model trained on a synthetic 3D dataset, all three techniques we examined, *i.e.*, DBM (Rodrigues et al., 2019), SDBM (Oliveira et al., 2022), and DeepView (Schulz et al., 2020), created essentially smooth *surfaces* that interpolate between the 3D samples (see Fig. 4.14). Practically, this means that decision maps created by these methods only show the model's behavior on a *small* 2D subset of the entire 3D data space the model works on, namely the aforementioned surfaces. How the model behaves on samples away from these surfaces is not shown. In this chapter, we conduct an in-depth investigation about the coverage question of decision maps – thereby answering our research question **RQ2** introduced in Chapter 1). Our results show that this 2D behavior is intrinsic to all decision maps that we area ware of and is independent on the studied classifier and dataset[1].

To further explore the surface-like behavior of inverse projections observed in the simple experiment in Chapter 4, we aim to answer several questions:

- Q1 How do decision maps look like for different ML models than the one used in Figure 4.14?
- Q2 How do the boundaries we see in Figure 4.14 relate to the actual decision boundaries of a classifier?
- Q3 Which parts of the data space do decision maps cover for data spaces having more than three dimensions?
- Q4 How do different inverse projection techniques influence the answers obtained for Q1-Q3?

1 This chapter is based on the papers "Fundamental Limitations of Inverse Projections and Decision Maps" (Wang and Telea, 2024) and "Investigating Desirable Properties of Inverse Projections and Decision Maps" (Wang and Telea, 2025)

Q5 How is the smoothness of the backprojection influenced by the direct-and-inverse projection technique choice?

To answer Q1-Q4, we study the behavior of DBM (which uses the NNInv inverse projection (Espadoto et al., 2019c)), SDBM, DeepView, and a few additional direct projection and inverse projection pairs, on a combination of several datasets (of varying dimensionality) and classification models. Next, we propose a way to measure how far an inverse projection can produce structures away from a single surface using intrinsic dimension estimation (Bennett, 1969; Camastra, 2003; Campadelli et al., 2015; Bac et al., 2021). Jointly put, our findings show that, for all datasets, all studied techniques essentially create surface like structures – with varying local smoothness – when mapping all points of the 2D space, except for points very close to the ones created by the direct projection (Q1-Q3). Answering Q4 is important for practitioners aiming to choose which (inverse) projections to use for any of the aforementioned applications (imputation, morphing, decision maps). In this chapter, we address this by studying not only NNInv, SDBM and DeepView, but also several additional combinations of direct projections (UMAP (McInnes et al., 2018), MDS (Borg and Groenen, 2005)) and inverse projections (iLAMP (dos Santos Amorim et al., 2012), RBF (Amorim et al., 2015), and iMDS (Blumberg et al., 2024)). Answering Q5 is important as a smooth backprojection is essential for interactive applications such as morphing where users want that small changes of a selected 2D point yield only small changes of the inferred data sample (dos Santos Amorim et al., 2012). Conversely, a backprojection that aims to cover as much as possible from the data space will be more effective in *e.g.* creating decision maps that capture more of a ML model's behavior, but will be likely less smooth. Understanding the inverse projection's smoothness is thus important for users to make informed choices for such applications.

The structure of this chapter is as follows. Section 5.2 introduces related work. Section 5.3 presents our results for 3D datasets, for which direct visual evaluation can be used to answer our questions. Section 5.4 extends our evaluation with new methods that address high-dimensional data. Section 5.5 discusses our findings. Finally, Section 5.6 concludes this chapter.

5.2 BACKGROUND

Decision maps, introduced in Sec. 2.3.4, aim to construct dense visualizations of a trained ML model f. We refer to Sec. 2.3.4

and Sec. 4.2.1 for a detailed description of how such maps are computed, including notations used in the process. We further denote the set

$$I^{-1} = \{P^{-1}(\mathbf{p}) | \mathbf{p} \in I\} \tag{5.1}$$

of all pixels $\mathbf{p}$ of an decision map image I which get mapped via P^{-1} to the data space as the *backprojection* of the decision map. The surfaces in Fig. 4.14 are examples hereof. As such, Q3 and Q5 (see Sec. 5.1) relate to how much of the data space does I^{-1} cover, respectively how smooth is I^{-1}. Similarly, we denote the set

$$P^{-1}(D) = \{P^{-1}(P(\mathbf{x})) | \mathbf{x} \in D\} \tag{5.2}$$

as the backprojection of the dataset D.

As explained in Sec. 2.3.4, decision maps can be constructed using any inverse projection P^{-1}, which is in turn suitably constructed from any direct projection P. However, only a limited number of (P, P^{-1}) combinations have been used to this end, as follows. Espadoto et al. (2019b) tested 28 projection techniques P with iLAMP as the inverse projection to construct decision maps and concluded that t-SNE and UMAP were the best choices for P. Following this, DBM (Rodrigues et al., 2018) used UMAP (McInnes et al., 2018) or t-SNE (van der Maaten and Hinton, 2008) for P and NNinv (Espadoto et al., 2019c) for P^{-1}. Espadoto et al. (2019c) used UMAP and t-SNE for direct projection while using iLAMP or RBF for the inverse one. SDBM (Oliveira et al., 2022) uses SSNP which, as already mentioned in Chapter 2 and Chapter 4, provides both P and P^{-1}. DeepView (Schulz et al., 2020) leverages discriminative dimensionality reduction (Schulz et al., 2015) to enhance the direct projection UMAP (McInnes et al., 2018), which also provides an inverse projection. Finally, the recent inverse projection iMDS (Blumberg et al., 2024) can also potentially be used to construct decision maps if P is set to multidimensional scaling (MDS, Torgerson (1952)).

In this chapter, we investigate the coverage and smoothness of decision maps constructed by the all the above mentioned methods/combinations.

5.3 VISUAL EVALUATION ON 3D DATA

To answer questions Q1-Q4 introduced in Sec. 5.1, we first extend the visual evaluation for 3D datasets in Fig. 4.14 to use more inverse projection techniques and more classifiers. We next evaluate these techniques on high-dimensional data (Sec. 5.4).

5.3.1 *Method*

Dataset: We conduct this evaluation using the well-known three-class Iris flower dataset (Fisher, 1988). As explained in Chapter 2, the key idea of visual evaluation is to directly *draw* the backprojected decision maps I^{-1} and see how these actually cover the data space of the trained ML model they are supposed to depict. To be able to create such visualizations, we need our dataset to have maximally $n = 3$ dimensions. We achieve this by restricting the Iris dataset to its last three features.

Decision map methods: Besides the three decision map methods used in Fig. 4.14, *i.e.*, DBM (Rodrigues et al., 2018), SDBM (Oliveira et al., 2022), and DeepView (Schulz et al., 2020), we also consider three additional inverse projection techniques: iLAMP (dos Santos Amorim et al., 2012), RBF (Espadoto et al., 2019c), and iMDS (Blumberg et al., 2024). For all methods except SDBM and DeepView (which use their own direct projection techniques, see Sec. 5.2), we now use MDS as direct projection. This is because (a) one of the considered inverse projections, iMDS, only works with MDS as direct projection; and (b) this setting allows us to minimize the number of direct projection techniques we use and thus provide a more intuitive comparison.

Classifiers: We study the behavior of decision maps using six classifiers: Logistic Regression (Cox, 1958), Support Vector Machines (Cortes and Vapnik, 1995, SVM), Random Forests (Breiman, 2001), Neural Networks, Decision Trees, and K-Nearest Neighbors (KNN). All are implemented using Scikit-Learn (Pedregosa et al., 2011) with default parameters, except Neural Networks, which uses three hidden layers each with 256 units. For each classifier, we not only construct the backprojected decision maps I^{-1} (see Sec. 5.2) for the six studied decision map techniques, but also visualize the actual decision boundaries in the 3D data space.

5.3.2 *Results*

5.3.2.1 *Preliminary comparison*

Figure 5.1 (top two rows) shows the backprojected decision maps, each from two different viewpoints (for better interpretation), for the six studied direct-and-inverse projection combinations (columns). For ease of interpretation of the results, we use here only the simple Logistic Regression classifier. The corresponding

2D decision maps are shown in Fig. 5.1 (bottom row). This preliminary investigation already reveals several interesting facts.

Firstly, we see that the backprojected decision maps for the first three methods (DBM, SDBM, DeepView) have very similar smooth-surface-like shapes as the ones shown in Fig. 4.14 for the synthetic blobs dataset. The backprojected surfaces of DBM and SDBM are quite smooth and, as such, cannot get very close to (all) the actual data points; In contrast, DeepView creates a much more noisy surface which 'connects' the data points better. This is also observed in the actual 2D decision maps (bottom row in Fig. 5.1): The maps for DBM and SDBM have far smoother decision boundaries than the DeepView map. This tells us that DBM and SDBM can depict the classifier's behavior *further* from the training set (extrapolation), whereas DeepView shows this behavior *close to and inside* this set (interpolation). Further on, we see that the backprojected decision maps for MDS+iLAMP, MDS+RBF, and MDS+iMDS behave very differently from the first three techniques. The latter two generate decision maps and backprojections which are very similar to each other and also quite close to a planar surface. Slight differences exist though: MDS+RBF creates a quite smooth backprojection that strictly passes through every sample $\mathbf{x}$, *i.e.*, $P^{-1}(P(\mathbf{x})) = \mathbf{x}$ for all $\mathbf{x} \in D$. In contrast, MDS+iMDS creates a noisier backprojection that does not strictly pass through the data samples. The most noticeable outlier is the result of MDS+iLAMP. It shows the appearance of a 'triangle soup' that exhibits practically no smoothness. In contrast, its backprojection covers far more of the 3D data space than all other compared methods. It is worth mentioning here that such discontinuities, originating from the iLAMP inverse projection, is precisely why the iLAMP authors next proposed the RBF inverse projection which is continuous and smooth (Amorim et al., 2015).

5.3.2.2 *Detailed comparison*

We now extend the findings obtained so far using the Linear Regressor classifier to all six classifiers mentioned in Sec. 5.3.1. At the same time, we extend the visual exploration used in Fig. 5.1 to show not only the backprojections I^{-1} but also the actual decision zones and decision surfaces. As this creates quite complex imagery, we now restrict the Iris dataset to its last two classes. This will decrease the amount of colors we need to use in our visualizations to two. Note that these two classes are not fully linearly separable, which makes our classification task more challenging than the synthetic blob dataset used in Fig. 4.14.

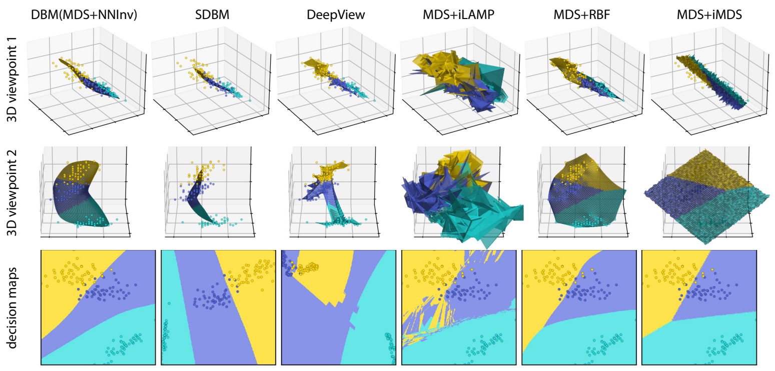


Figure 5.1: Top two rows: Backprojections of the decision maps constructed by 6 inverse projection techniques (columns) with Logistic Regression on IRIS dataset, viewed from two viewpoints. Bottom row: Corresponding (2D) decision maps.

Figure 5.2a shows the actual decision zones and decision boundaries and the backprojected decision maps for the six classifiers mentioned in Sec. 5.3.1 and the same six decision map methods already explored in Fig. 5.1. The actual 2D decision maps are shown in Fig. 5.2b. We further study the differences between the backprojected decision map and the *actual* decision zones and surfaces as follows. We sample the 3D data space on a voxel grid of size 100^3 (to limit computational effort); compute, for each voxel $\mathbf{v}$, the predicted class $f(\mathbf{v})$, and color code it; and draw this color-coded volume half-transparently (Fig. 5.2a, bottom 6 rows). The yellow, respectively purple, volumes are thus the *actual* decision zones of f. For clarity, we show these volumes, without the backprojection I^{-1}, in the leftmost column in Fig. 5.2a. Also, we draw the actual decision boundary $\mathcal{S}$ that separates the two decision zones in beige – see Fig. 5.2a, leftmost column, top cell for an example.

Figure 5.2 leads us to several insights. First, we see that the backprojected decision maps I^{-1} (shaded surfaces in Fig. 5.2a, top row), *i.e.*, the part of the data space that a decision map visualizes, are roughly orthogonal to, and intersecting, the actual decision surfaces (pale brown in Fig. 5.2a). That is, the boundaries which we *see* in a decision map (curves where yellow meets purple in Fig. 5.2b) are the intersection $\mathcal{S} \cap I^{-1}$. Separately, if we scan a column in Fig. 5.2a, we see that the backprojections I^{-1} are the same – or almost the same in the case of MDS+iMDS and DeepView (the reason for this is discussed separately below). How-

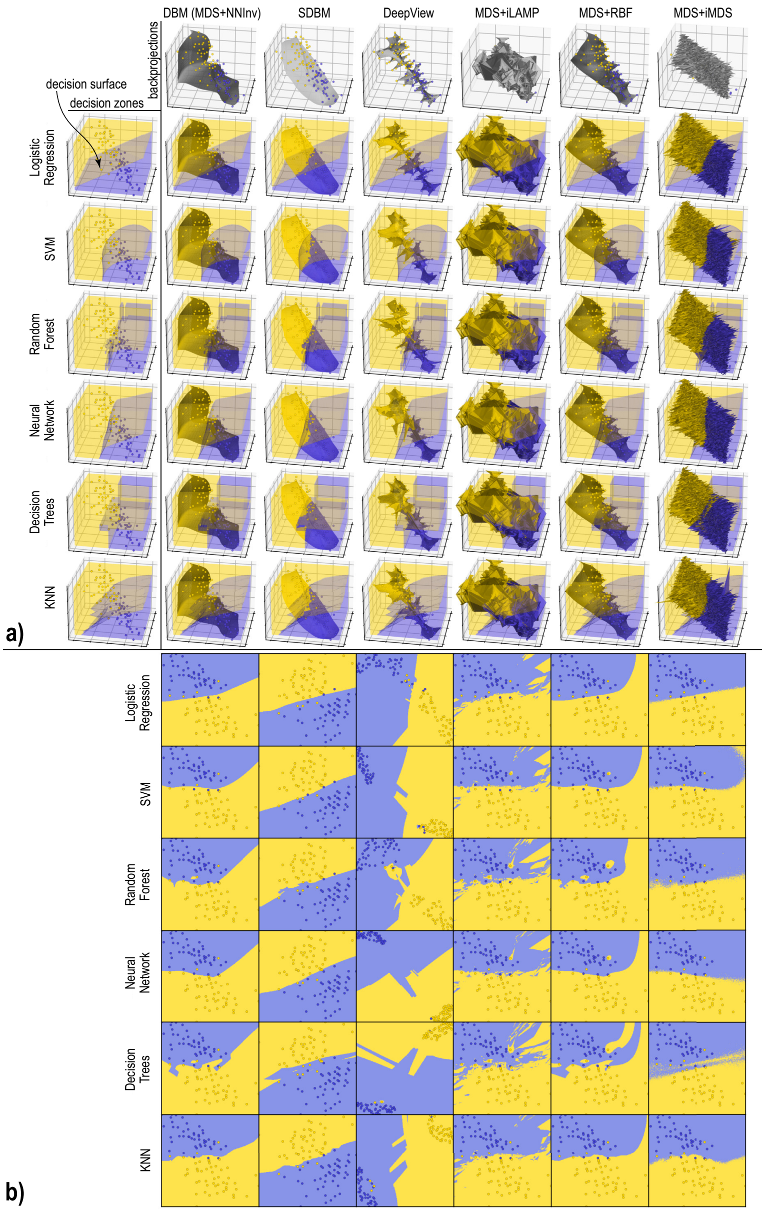


Figure 5.2: Decision maps (a) backprojected in 3D; (b) original in 2D of six classifiers, modified Iris dataset, computed by six techniques. The decision zones are yellow, respectively purple; the decision surface separating them is beige. The shaded surfaces are the backprojected decision maps.

ever, as the classifiers change (rows), the decision boundaries change – see Fig. 5.2a, leftmost column. Hence, the *intersection* of I^{-1} with the actual decision boundary will change. As mentioned, this intersection is precisely what we see as color boundaries in a 2D decision map. So, if the backprojection I^{-1} is not smooth, this intersection can change *significantly*, even when the visualized classifier model changes only slightly. Simply put, this means that decision maps whose backprojections look 'crumpled' (non-smooth) can be very unstable and thus unsuited for practical use. Even stronger: the non-smoothness of the backprojection can create the false impression that a classifier has complex decision maps. Take for instance Logistic Regression, SVM, or Neural Networks; these classifiers show very smooth decision boundaries (Fig. 5.2b, leftmost column). However, their 2D decision maps created with DeepView or MDS+iLAMP show complex, non-smooth boundaries, which is clearly misleading. All in all, the above insights argue in favor of *e.g.* (S)DBM and MDS+RBF as techniques for creating decision maps and definitely against DeepView and MDS+iLAMP.

Secondly, we see that no decision map technique can actually depict the *full* decision boundaries of any classifier. For example, the linear decision boundary of Logistic Regression is not well captured, except by MDS+iMDS. The other decision maps show non-linear boundaries or even disconnected decision zones, see *e.g.* DeepView and MDS+iLAMP. Another example is for Decision Trees. We see that the actual decision zone (purple) is split into two disconnected components (top and bottom purple cubes (Fig. 5.2a, leftmost column)). However, none of the tested decision map techniques shows two such separated purple decision zones (Fig. 5.2b).

Finally, let us revisit the issue of the backprojection shapes generated by a given technique. DBM, SDBM, iLAMP, and RBF produce exactly the same shapes regardless of the classifier they depict – indeed, their P and P^{-1} do not depend on the classifier. In contrast, MDS+iMDS and DeepView can generate (slightly) different shapes for different classifiers, for different reasons, as follows. By design, DeepView uses discriminative dimensionality reduction (Schulz et al., 2015), so its P depends on f. As for the reason why MDS+iMDS has different shapes for rows, this is because iMDS uses *random* selections of samples to compute its P^{-1}. While one can argue that DeepView's design shows more information on f, *controlling* how DeepView's decision maps actually sample the data space as a function of f is unclear. As such, we believe that the approaches of (S)DBM, iLAMP and RBF where this sampling only depends on the training set, are more intuitive and stable.

5.4 EVALUATION ON HIGH DIMENSIONAL DATA

We next conduct a quantitative evaluation of inverse projections and decision maps using high-dimensional datasets for all the six inverse projection techniques listed in Sec. 5.3.1. Additionally, we substitute UMAP for MDS when using it in combination with the inverse projection techniques NNInv, RBF, and iLAMP, as UMAP is far better suited to handle high-dimensional data than MDS. We also present additional quantitative measurements that gauge the quality of the studied inverse projections and corresponding decision maps, as well as a visual exploration of the smoothness of the studied inverse projection techniques.

5.4.1 *Method*

Since decision maps fundamentally depend on inverse projections, it makes sense to first and foremost quantify the quality of P^{-1}. Further on, for $n > 3$ dimensional data, we cannot directly *draw* the backprojected images I^{-1}, as already mentioned in Sec. 5.3.1. Recall now our question Q3 (Sec. 5.1). To answer it, we measure how far I^{-1} is, locally, from a two-dimensional manifold embedded in $\mathbb{R}^n$. For this, we use intrinsic dimensionality (ID) estimation (Bac et al., 2021) with a linear method, *i.e.*, Principal Component Analysis (PCA), due to its intuitiveness, computational efficiency, ease of use, and popularity (Espadoto et al., 2019a; Bac et al., 2021; Tian et al., 2021). Finally, we use the gradient map technique (Espadoto et al., 2021a) to get insights into the decision maps' smoothness. All these steps are detailed further below.

5.4.1.1 *Datasets*

We use five synthetic and real-world datasets, all having $N = 5000$ samples (Tab. 7). The synthetic datasets, with dimensionality n of 10, 30, and 100, consist of each of $C = 10$ isotropic Gaussian blobs. Using isotropic blobs ensures that the ID is the same as the dimension count n for these datasets. As real-world datasets, we use HAR (Anguita et al., 2012) and MNIST (LeCun et al., 2010). The intrinsic dimensionality of these datasets has been estimated by prior work (El Moudden et al., 2016; Facco et al., 2017; Aumüller and Ceccarello, 2019; Bahadur and Paffenroth, 2019). We use Logistic Regression as an example classifier f. Note that this does not affect the ID estimation, as f is not involved in the construction of P^{-1}.

Table 7: Datasets used for ID estimation. For each dataset, we list the provenance, dimensionality n, sample count N, and class count $|C|$.

Dataset	n	N	$\|C\|$
Blobs 10D (synthetic)	10	5000	10
Blobs 30D (synthetic)	30	5000	10
Blobs 100D (synthetic)	100	5000	10
HAR (Anguita et al., 2012)	561	5000	6
MNIST (LeCun et al., 2010)	784	5000	10

5.4.1.2 *Error of the inverse projection*

We measure the quality of an inverse projection P^{-1} for a given dataset D and its projection $P(D)$ by the mean squared error (MSE) of the data backprojection $D' = P^{-1}(P(D))$ which is defined as

$$MSE = \frac{1}{|D|} \sum_{\mathbf{x} \in D} \|\mathbf{x} - P^{-1}(P(\mathbf{x}))\|^2. \tag{5.3}$$

An ideal inverse projection P^{-1} should yield $P^{-1}(P(\mathbf{x})) = \mathbf{x}$ for all $\mathbf{x} \in D$, *i.e.*, have zero MSE, which is the same as saying that $D' = D$ (Espadoto et al., 2019c). Conversely, if this error is large, then the inverse projection is likely poor and will lead to misleading or even meaningless decision maps.

5.4.1.3 *Intrinsic dimensionality estimation*

Let X be a dataset in $\mathbb{R}^n$ with $S(\mathbf{x})$ being its k nearest neighbors in X. Let $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ be the n eigenvalues of $S(\mathbf{x})$'s covariance matrix, sorted decreasingly. A common way to define the ID of $S(\mathbf{x})$ as the smallest d value so that the sum of the first d eigenvalues is larger than a given threshold θ, where θ was set to a value close to 1, typically 0.95 (Jolliffe, 2002; Fan et al., 2010; Tian et al., 2021). This method is also known under the name *total variance* (Tian et al., 2021). When using the total variance method for computing d_i, we found that, in the case of iLAMP (an inverse projection method they didn't study but we do), sometimes the first two eigenvalues capture a significant portion of the variance (*e.g.*, 85%); however, to arrive at 95%, one would need a large number of additional eigenvalues (*e.g.*, over 500 on MNIST dataset), each contributing less than 1% to the total variance. Obviously, this is not desirable, as it would highly overestimate the intrinsic dimensionality. To cope with this, we adopted the alternative definition of intrinsic dimensionality known as *minimal variance* which solves precisely this problem (Tian et al., 2021) – that is, we define ID as the number of eigenvalues each accounting for at least θ percent of the data variance, where θ is set typically to a small value.

Algorithm 1 shows our computation of ID values. We set $\theta = 0.01$, thereby identifying the principal components that capture more than 1% of the total variance as significant for the intrinsic dimensionality. The size k of the local neighborhood $S(\mathbf{x})$ needs careful setting. A too large k leads to overestimating the local ID. Conversely, too small k values lead to noisy estimations. Note that $d + 1$ independent vectors are required to span d dimensions, so k should be at least equal to the actual ID of $S(\mathbf{x})$ (Verveer and Duin, 1995). We have ID estimations ranging from 13 to 33 for MNIST (Facco et al., 2017; Aumüller and Ceccarello, 2019; Bahadur and Paffenroth, 2019); and from 15 to 61 for HAR, depending on the estimation method (El Moudden et al., 2016); our synthetic datasets have known ID values of 10, 30, and 100 (see Tab. 7). To cover all the above cases, we globally set $k = 120$.

Algorithm 1: Intrinsic Dimensionality Estimation

```
Data: $X$, set of data points in $\mathbb{R}^n$ (can be $D$, $D'$, or $I^{-1}$);
      neighborhood size $k = 120$; threshold $\theta = 0.01$
Result: $\bar{d}$, the estimated ID of $X$ (average among all local
        neighborhoods)
1 begin
2     for $\mathbf{x} \in X$ do
3         Find the $k$ nearest neighbors $S(\mathbf{x})$ of $\mathbf{x}_i$ in $X$;
4         Compute the covariance matrix $\mathbf{Cov}$ of $S(\mathbf{x})$;
5         Compute the eigenvalues $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ of $\mathbf{Cov}$;
6         Sort $\boldsymbol{\lambda}$ in descending order;
7         Calculate ID $d(\mathbf{x})$ of $S(\mathbf{x})$ as
              $d(\mathbf{x}) = \left| \left\{ \frac{\lambda_i}{\sum_{i=1}^{n} \lambda_i} \geq \theta, 1 \leq i \leq n \right\} \right|;$
8     Calculate average ID $\bar{d} = \sum_{\mathbf{x} \in X} d(\mathbf{x}) / |X|$;
```

We perform two different ID estimations, as follows. First, for a given dataset D and its 2D projection $P(D)$, we compute the average ID of the data backprojection $D' = P^{-1}(P(D))$ over all neighborhoods $S(\mathbf{x})$, denoted $ID_{D'}$. We then compare $ID_{D'}$ with the ground-truth average ID of D, denoted ID_D. Both ID_D and $ID_{D'}$ are computed using Alg. 1 with D and D' as inputs, respectively. For an ideal inverse projection P^{-1} that perfectly reverses the effects of a direct projection P on D, we would obtain $ID_{D'} = ID_D$. Secondly, to study how well a decision map covers the data space it aims to depict (see Q3, Sec. 5.1), we create a pixel grid I of size 500^2 and backproject it by P^{-1} to obtain a sample set I^{-1}. We next measure the ID at each sample $\mathbf{p} \in I^{-1}$ using Alg. 1 with I^{-1} as input. Let the resulting value at $\mathbf{p}$ be called ID_p. Finally, we color the image I by the values ID_p and also compute the average $\overline{ID_p}$ over all pixels in I.

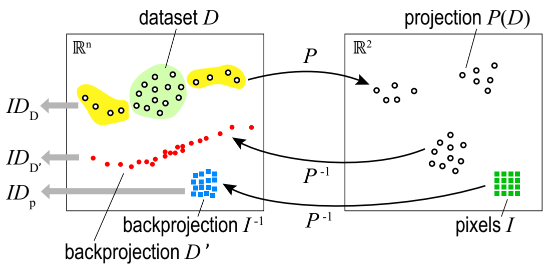


Figure 5.3: Computing the intrinsic dimensionality of data, backprojection of data, and backprojection of pixels.

Figure 5.3 depicts all the above processes: Given a dataset D, we compute its 2D projection $P(D)$. We inversely project these points via P^{-1} to get the backprojection D'. Separately, we inversely project all pixels in the image I to get the sample set I^{-1}. In this example, the intrinsic dimensionality ID_D is the same to $ID_{D'}$ for the yellow areas in D; and higher than $ID_{D'}$ for the green areas in D, respectively.

5.4.1.4 *Gradient maps*

To study the smoothness of the computed decision maps, we use the gradient map technique (Espadoto et al., 2021a), following Equation (4.5). Recall from Sec. 4.3.2 that areas in a decision map where G is large mean that neighboring pixels are backprojected by P^{-1} far away from each other in the data space, hence the map is unreliable at those locations. Conversely, areas in a decision map with low G mean that neighboring pixels sample the $\mathbb{R}^n$

space at close locations. Assuming a (relatively) smoothly evolving classifier f over $\mathbb{R}^n$, such areas will accurately capture the local behavior of f.

5.4.2 *Results*

5.4.2.1 *Error assessment*

Table 8 shows the MSE results for all our datasets and decision map computation methods. Values for the iLAMP and RBF inverse projections are exactly zero since these methods enforce that $P^{-1}(P(\mathbf{x}_i)) = x_i$ for all $\mathbf{x}_i \in D$ by construction. We see that the MSEs of DBM, SDBM, and DeepView are quite low and comparable across all datasets, indicating that these inverse projections are similar and reliable. In contrast, the MSE of MDS+iMDS is significantly higher. On the synthetic datasets (Blobs), the errors of MDS+iMDS are 2 to 3 orders of magnitude higher than for the other tested methods, which is already quite high. However, on the real-world datasets (HAR and MNIST), the errors of MDS+iMDS become even higher. This indicates that the backprojections of MDS+iMDS, and thus the corresponding decision maps, are likely meaningless. Figure 5.4 confirms this by running a simple test on the MNIST dataset. For 14 images $\mathbf{x}_i$ in this dataset (top row), we show the corresponding inverse projections $P^{-1}(\mathbf{x}_i)$ computed by DBM, SDBM, DeepView, and MDS+iMDS. The first three methods yield very similar images to the original ones, as expected due to the low MDS thereof. In contrast, MDS+iMDS yields basically noise. Note that this agrees with the preliminary discussion of Blumberg et al. (2024) which mentioned that iMDS may be unsuitable for inversely projecting scatterplots created from data of a too high dimensionality.

Table 8: MSE of the backprojection for the studied datasets.

	Blobs 10D	**Blobs 30D**	**Blobs 100D**	**HAR**	**MNIST**
DBM	1.83×10^{-3}	2.13×10^{-3}	2.16×10^{-3}	5.30×10^{-3}	3.67×10^{-2}
SDBM	2.22×10^{-3}	2.06×10^{-3}	2.16×10^{-3}	8.69×10^{-3}	5.28×10^{-2}
DeepView	1.42×10^{-3}	1.67×10^{-3}	1.90×10^{-3}	4.40×10^{-3}	2.92×10^{-2}
UMAP+iLAMP	0	0	0	0	0
UMAP+RBF	0	0	0	0	0
MDS+iMDS	1.07×10^{-1}	6.11×10^{-1}	5.71×10^{0}	5.15×10^{33}	6.19×10^{5}

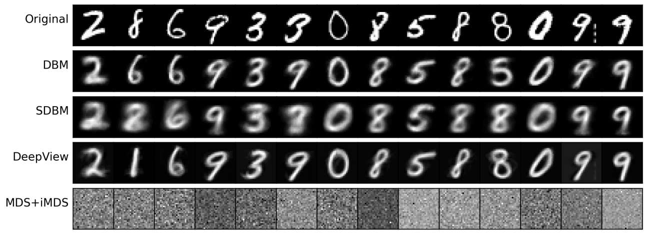


Figure 5.4: Selected samples from MNIST and their corresponding back-projections using different P^{-1} methods. Compare with the MSE values in Tab. 8.

5.4.2.2 *Intrinsic dimensionality estimation*

To answer Q3, we first test how well an inverse projection P^{-1} covers the *data space* D it aims to depict. For this, we compare the estimated ID of the actual data (ID_D) with that of the round-trip consisting of the direct and inverse projections ($ID_{D'}$). As explained in Sec. 5.4.1, an ideal inverse projection would yield $ID_{D'} = ID_D$. Table 9 shows the results for our five studied datasets. A first consistency check is to see how good the estimated ID_D is. We see that these values align well with the expected (ground-truth) ID values for most datasets. The largest difference occurs for Blobs 100D, which is due to the fact that this dataset isotropically spreads in 100 dimensions at every blob by construction (see Sec. 5.4.1); the ID estimation by PCA (Alg. 1) is heavily affected by the well-known curse of dimensionality.

Given the above, we can next compare the estimated ID_D with the round-trip estimation $ID_{D'}$ to gauge the inverse projection quality (see Tab. 9). Just as for the 3D data discussed in Sec. 5.3.2, we see that (S)DBM creates basically a *two-dimensional*, surface-like, structure in the data space. DeepView is slightly better in capturing the ID_D of the data – which matches the fact observed for 3D datasets that its backprojected surfaces have more complex shapes that aim to connect the data samples (Fig. 5.2). Still, DeepView's $ID_{D'}$ values are much lower than the estimated ID_D. Note that iLAMP and RBF are not included in the comparison as they have $P^{-1}(P(\mathbf{x}_i)) = \mathbf{x}_i$ for all $\mathbf{x}_i \in D$ by construction, which means $ID_{D'} = ID_D$. Finally, for MDS+iMDS, we see that $ID_{D'}$ is much closer to the estimated ID_D than for all other methods. This may suggest that MDS+iMDS is better at capturing the data space. Yet, as observed earlier, this method has a very high MSE (Tab. 8) and also generates meaningless inverse projections (Fig. 5.4). As such, the high $ID_{D'}$ for this method is rather an indication of its random sampling pertaining to its implementation (see (Blumberg et al., 2024) for details) than its intrinsic higher quality. The authors of iMDS also noted that this inverse projection may not be effective for datasets of high intrinsic dimensionality. Our experiment here confirmed this observation.

Table 9: Estimated intrinsic dimensionalities ID_D and $ID_{D'}$ for our studied datasets. The expected ID values for HAR and MNIST are taken from prior studies (El Moudden et al., 2016; Facco et al., 2017; Aumüller and Ceccarello, 2019; Bahadur and Paffenroth, 2019).

	Blobs 10D	**Blobs 30D**	**Blobs 100D**	**HAR**	**MNIST**
Expected ID	10	30	100	15-61	13-33
ID_D	10.00	29.03	39.63	24.62	20.04
$ID_{D'}$ DBM	2.04	2.10	2.04	3.56	4.71
$ID_{D'}$ SDBM	2.23	2.14	2.11	2.09	2.47
$ID_{D'}$ DeepView	4.98	4.71	4.63	8.25	7.60
$ID_{D'}$ UMAP+iLAMP	-	-	-	-	-
$ID_{D'}$ UMAP+RBF	-	-	-	-	-
$ID_{D'}$ MDS+iMDS	10.00	22.95	37.69	11.77	28.68

To further answer Q3, we want to know how well a decision map image covers the entire data space of the classifier it aims to visualize. We measure this by comparing the ID of the backprojected decision map image ID_p at each pixel (see Sec. 5.4.1) with the ID_D of the dataset D the classifier is trained (or tested) on. Areas where ID_p is close to ID_D indicate that the decision map covers well the local distribution of D; areas where $ID_p \ll ID_D$ indicate that the decision map can only capture a part of this local distribution.

Figures 5.5—5.7 show this comparison. In each figure, the top row shows the actual decision maps computed by our six decision map techniques for the studied Logistic Regression classifier. Colors in these images indicate the inferred class by the trained model f at each pixel; brightness encodes the confidence of f at those locations (dark values indicate low confidence); for details of this computation, see (Rodrigues et al., 2018, 2019). These decision map images are only provided for illustration purposes *e.g.*, showing the location of decision boundaries and the data clusters;

the ID analysis presented next does not depend on the classifier choice.

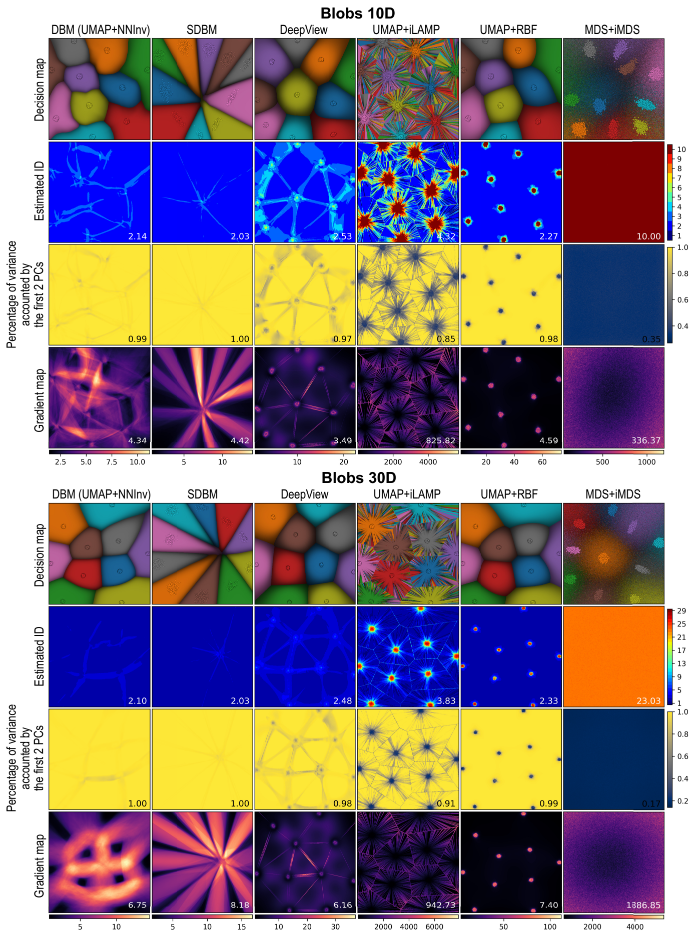


Figure 5.5: Decision maps and ID estimation, Blobs 10D and 30D.

The second rows in Figs. 5.5-5.7 show the estimated ID_p at each decision map pixel, with the average value $\overline{ID_p}$ over the entire map shown bottom-right in the images. The results are very interesting to examine.

For DBM and SDBM, the estimated ID_p are *exactly* 2 almost everywhere, which means that these decision maps precisely correspond to *surfaces* in the data space. This extends our earlier findings (Sec. 5.3.2) to $n > 3$ dimensions. DeepView yields higher ID_p values (but still much lower than ID_D, peaking at 10 for the HAR dataset) close to the actual data points; and values roughly equal to 2 further away from these points, with an $\overline{ID_p}$ over all datasets of 2.49 ± 0.03.

For UMAP+RBF, in areas close to the data points, the estimated ID_p is high (about the same as ID_D), see the red-colored spots in Figs. 5.5-5.7 (second rows). This is expected by the design of the RBF method, as mentioned earlier. Between the data points, the estimated ID_p for this method is exactly 2. UMAP+iLAMP shows more complicated patterns: This method also yields high ID_p values (basically equal to ID_D) close to the data points areas, as expected by construction for this method, as explained earlier. Between the data points, this method yields an estimated ID_p roughly equal to 2. However, in contrast to all other methods, UMAP+iLAMP shows strong radial-like patterns of high estimated ID_p that 'fan out' from the data points. These radial patterns in the ID_p images match similar ones in the decision maps (Figs. 5.5-5.7, first rows). We see here again an instance of iLAMP's behavior discussed in Fig. 5.2 for the 3-dimensional dataset case: iLAMP covers the data space better than other methods (thus answers Q3 better) but does this at the expense of continuity – that is, it produces decision maps which can be hard to interpret.

To further understand the high ID_p values for iLAMP in image areas far away from data samples, we select an area having such high values for the MNIST dataset (Fig. 5.7, second row, white square). We next oversample this area at a resolution of 500^2 pixels to compute ID_p. The result (Fig. 5.7 bottom row) shows that ID_p is actually almost 2 in such areas as well, apart from very close to the data points. Hence, the observed higher intrinsic dimensionality ID_p for iLAMP (and, actually, all other tested methods) is only an effect of the image resolution; all methods have the low ID_p values they exhibit virtually everywhere except infinitesimal neighborhoods around the data points. Further, an interesting observation is that the aforementioned radial patterns seem to be less noisy as the data dimensionality increases – compare the Blobs 10D, 30D, and 100D images in Figs. 5.5-5.6. Indeed, as the data is increasingly higher dimensional, iLAMP has more difficulties to 'cover' the entire data space with a two-dimensional map, even close to the data points.

Finally, MDS+iMDS shows a quite different result: The ID_p values it produces are nearly identical over the entire image and also roughly equal to ID_D. The fact that ID_p is nearly constant matches the linear behavior of this inverse projection method that we discussed for the 3D dataset case (Sec. 5.3). Separately, the fact that $ID_p \simeq ID_D$ is due to the random sampling process used

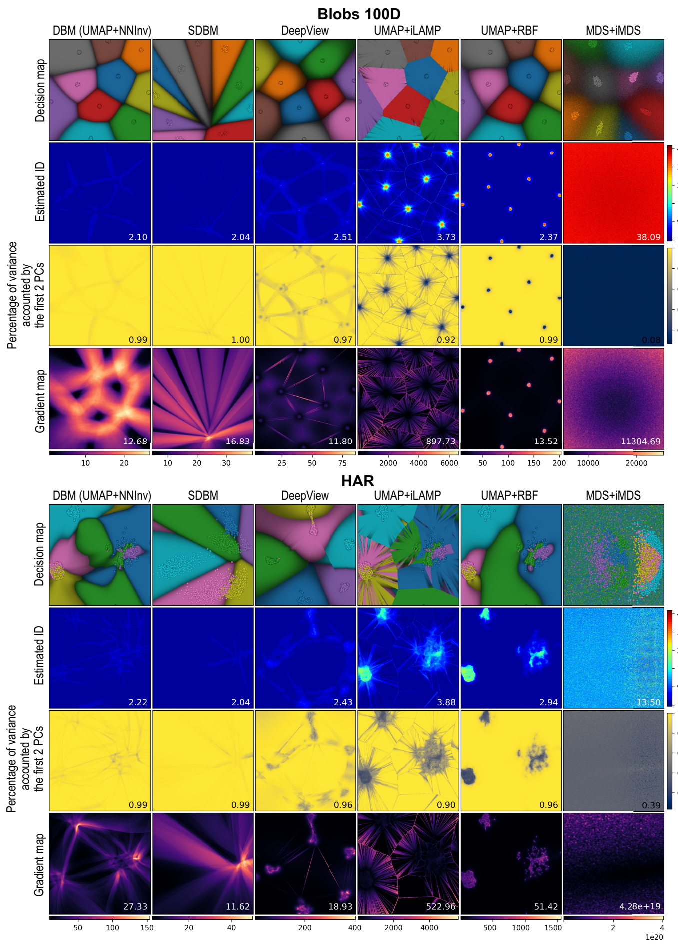


Figure 5.6: Decision maps and ID estimation, Blob 100D and HAR.

by iMDS, see Sec. 5.2. We also see that the decision maps for this method are quite dark in all areas, even in those near the actual samples. This correlates with the relatively high MSE of MDS+iMDS (Tab. 8). Intuitively put, these findings indicate that this inverse projection quickly 'goes away' from the data samples $\mathbf{x}_i$ for pixels which are not very close to the locations $P(\mathbf{x}_i)$. Again, this is due to the linear nature of iMDS – the backprojected surface I^{-1} cannot, by construction, follow the likely curved manifolds on which the samples $\mathbf{x}_i$ are spread. Separately, we see strong noise in the decision maps for this method, which is due to the aforementioned random sampling process. Again, we see here the earlier-mentioned trade-off between coverage and continuity.

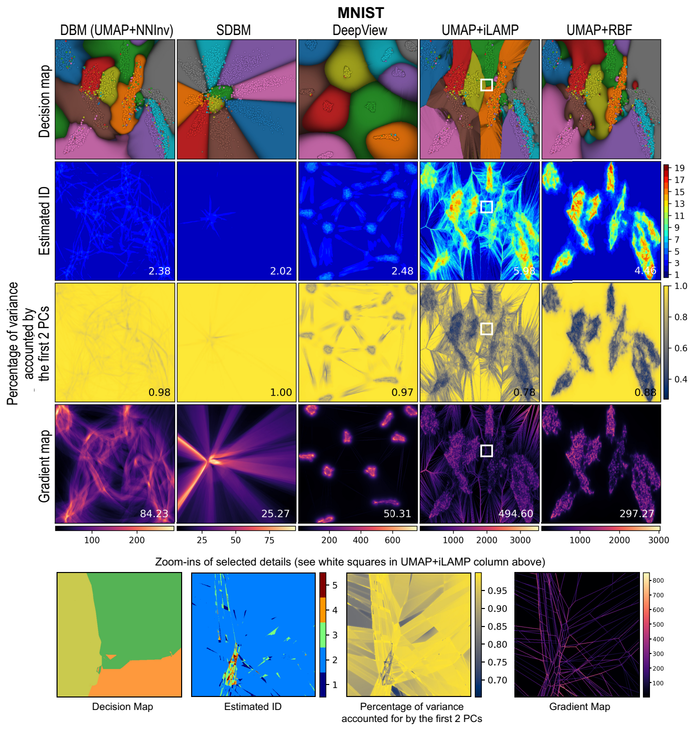


Figure 5.7: Decision maps and ID estimation, MNIST. Bottom images show selected detail zones sampled at a high resolution of 500^2 pixels.

As iMDS has a global linear behavior, we can study it in further detail as follows. We compute the covariance matrix for the whole set of backprojected points I^{-1} and then analyze its eigenvalues $\lambda_1, \dots, \lambda_{100}$ (Fig.5.8). We observe that there is always a clear drop from the second to the third eigenvalue, indicating that the back-projected points are also dominated by a 2D planar-like structure. This matches the visual observation for the 3D dataset shown in Fig. 5.2. Hence, the earlier discussed fact that ID_p is overall high (Figs. 5.5-5.6, second rows) is purely due to the random sampling of iMDS. Separately, we see that as the dimensionality of the data increases, this drop becomes less significant. This suggests that the structure becomes more dominated by noise as the dimensionality increases, which correlates with the fact that the MSE of MDS+iMDS is significantly higher for higher-dimensional data (Tab.8).

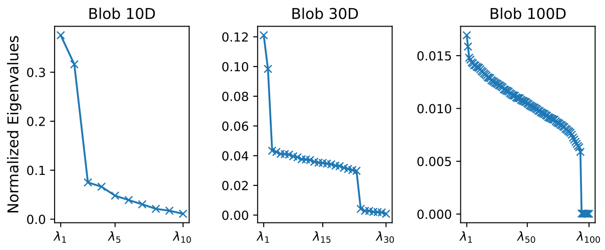


Figure 5.8: Eigenvalues of the covariance matrix of the whole set of back-projected points I^{-1} for MDS+iMDS.

The third and fourth rows in Figs. 5.5—5.7 refine the above insights. The third rows show the percentage of data variance in a neighborhood S captured by the eigenvectors corresponding to the two largest eigenvalues λ_1 and λ_2. Yellow values indicate that almost all the data variance is captured by these two eigenvalues, so the inverse projection creates locally planar structures there. Dark blue values indicate that the opposite, *i.e.*, the inverse projection creates high-dimensional structures in the respective areas. The fourth rows in the figures show the gradient maps of the inverse projections. Dark values in these maps indicate low values of G (Eqn. 4.5), *i.e.*, areas where the backprojection changes slowly and smoothly. Bright areas indicate the converse phenomenon – rapid and potentially non-smooth changes in the backprojection. The computation details are described in Sec. 5.4.1.

These visualizations lead us to additional interesting observations. First, we see that, in nearly all cases, all inverse projection methods except MDS+iMDS create large yellow areas far away from the data points (third rows in Fig. 5.5—5.7) – that is, they essentially create two-dimensional surface-like backprojections I^{-1}. In contrast, MDS+iMDS shows dark blue values nearly everywhere in these images, *i.e.*, it creates nearly everywhere a high-dimensional sampling of the data space. As explained earlier, this is due to the random sampling process inherent to this method.

A second observation pertains to the presence of 1D dark filament-like linear structures that connect the projected data points which we notice for DeepView and UMAP+iLAMP. These filaments seem to connect the projected points much like a Delaunay triangulation. These structures match quite well high values in the corresponding gradient maps. Taken together, these findings indicate that the backprojections of DeepView and UMAP+iLAMP consist of a set of planar-like facets, separated by sharp creases or gaps. This generalizes our earlier findings on the 'crumpled' aspect of these backprojections, observed for 3D datasets (Fig. 5.2a) to higher dimensions.

The gradient maps allow us to draw some other insights on the behavior of the inverse projections. For (S)DBM, these maps have high values that align quite well with the corresponding decision boundaries shown in Figs. 5.5–5.7, first rows. In contrast, UMAP+RBF has high gradients systematically close to the projected data samples only. UMAP+iLAMP shows an almost complementary behavior to UMAP+RBF, that is, high gradient values on the aforementioned filaments connecting the projected data points and relatively low gradient values close to the data points. Overall, these insights tell that the studied inverse projection methods have very different smoothness behaviors: (S)DBM is relatively smooth overall except close to the decision boundaries; UMAP+RBF is also quite smooth except close to the data samples; and UMAP+iLAMP is overall smooth except close to lines that connect neighboring data samples. All these findings match our earlier observations in the visual study of these backprojections for the 3D dataset case (Fig. 5.2a).

5.5 DISCUSSION

We next discuss our findings on the interpretation, added value, and found limitations of decision maps and their accompanying inverse projections, and summarize our answers to the questions Q1-Q5 listed in Sec. 5.1.

5.5.1 *Surface behavior of inverse projections and decision maps*

All six inverse projection pipelines we studied essentially generate surface-like structures embedded in the high-dimensional data space, with some local differences. (S)DBM tends to create relatively smooth and compact surfaces that closely interpolate the data samples. UMAP+RBF does the same but passes exactly through the data samples while being slightly less smooth. DeepView and UMAP+iLAMP create highly twisted surfaces with a similar type of trade-off, *i.e.*, DeepView interpolates the data points less accurately but yields smoother surfaces, while UMAP+iLAMP interpolates the data points exactly but yields very non-smooth results. Finally, MDS+iMDS yields a structure which formally speaking has higher intrinsic dimensionality than a surface. Upon closer examination, we see that this structure is essentially a plane jittered by high amounts of noise (Q1, Q4). We also saw that this surface-like property does not depend on the intrinsic or total dimensionality of the studied datasets (Q1), the studied classifiers (Q1), or resolutions of the decision map images (see Figs. 5.4).

5.5.2 *Coverage of decision maps*

Given the aforementioned surface property, we conclude that current decision maps only depict a *small* part of the behavior of a given classifier (Q3). The only (relative) exception here is MDS+iMDS which succeeds in covering a higher proportion of the data space. However, this is done by using a random sampling mechanism which leads to high inverse projection errors (Tab. 8) and noisy results in both the inverse projections (Fig. 5.4) and decision maps (Figs. 5.5, 5.6). We conclude that this method is not suitable for creating general-purpose inverse projections and decision maps for high-dimensional data.

The boundaries shown by the studied decision maps (1D curves separating same-color regions in *e.g.* Fig. 5.2) are actually the intersections of the aforementioned surfaces with the actual decision boundaries in high-dimensional space (Q2). Intuitively put, a decision map thus shows a 'slice' through the high dimensional data space – its pixels are located on the aforementioned 2D surface; and its decision boundaries are 1D curve subsets of the actual decision surfaces. It is tempting to argue that, since inverse projections take a 2D space as input, they will always produce also a 2D surface as output and not a higher-dimensional object. Yet, this does not need to be so. Space-filling curves (Peano, 1890) and space-filling surfaces (Paulsen, 2023) can map low-dimensional sets to higher-dimensional ones in a continuous fashion. By combining such primitives, we could in principle create continuous mappings of intervals between any two dimensions q and n, $q < n$. Our study – in particular, the ID and gradient map estimations – showed that all evaluated inverse projections (DBM, SDBM, DeepView, iLAMP, RBF, iMDS) do not even get close to such behavior – which can be explained by the fact that they are constructed by differentiable mappings which cannot in principle exhibit fractal behavior. iMDS has the highest coverage but, as we saw, this is achieved by random sampling, which completely loses continuity.

5.5.3 *Comparing decision map methods*

Different decision map techniques sample the high-dimensional space quite differently (Q4). As such, they produce different maps for the *same* classifier (which, obviously, has a unique set of actual decision surfaces). Each such map provides its own insights for the same classifier (see *e.g.* Figs. 5.2, 5.5, 5.6, 5.7), each with its own advantages and limitations. At a global level, we see a clear trade-off between *smoothness* and *precision* (Q5). Methods that generate the smoothest surfaces (DBM, SDBM) cannot approximate very well the data samples. Conversely, methods that pass very close or exactly through the data samples (DeepView, UMAP+iLAMP) generate non-smooth surfaces. UMAP+RBF falls somewhere in the middle of these two types. These aspects affect in turn the *interpretability* and ultimately *usability* of the corresponding decision maps. Smooth-surface methods yield maps which are easier to interpret and show better how a classifier *extrapolates* from its training set but are harder to control in terms of *where* they are actually constructed; tighter-surface methods approximate data samples better and, for the case of DeepView, are also easier to control in terms of where they sample the data space. However, they only *interpolate* the classifier behavior close to and between the training points, and can create decision maps which are hard to interpret (UMAP+iLAMP). Summarizing the above, we believe that smooth-surface methods are overall preferred to tight-surface ones – they ultimately yield decision maps which are easier to interpret at the small cost of not perfectly approximating the data samples.

As a separate point, we note that none of the studied techniques aims to explicitly sample a classifier close to its *actual* decision boundaries – which, arguably, are the most interesting areas to understand (Q2). For this task, new inverse projections and/or decision map methods need to be devised.

5.5.4 *Limitations caused by the low dimensionality of decision maps*

Inverse projection tasks are structurally similar to data reconstruction or data generation tasks – all of these aim to output high-dimensional data from low-dimensional representations. From the perspective of data reconstruction, the projection and inverse projection pipeline (P, P^{-1}) can be seen as a special case of an encoder-decoder structure, where the bottleneck, or latent space, is two-dimensional. Existing works show that the dimensionality of the latent space, which is analogous to the input of

P^{-1}, is a critical factor for the reconstruction or generation quality (Wang et al., 2016; Marin et al., 2021; Padala et al., 2021).

Inspired by these findings, we wonder how the dimensionality q of the latent representation affects the quality (MSE and ID) of an inverse projection. To explore this, we ran DBM (UMAP+NNInv) and SDBM (SSNP) with q values in the range 2 to 25. We chose these methods since they are easily modifiable to use a different latent dimensionality than $q = 2$ and also since, following our earlier results, they seem to offer a good balance in terms of desirable properties of inverse projections. For each q value, we recorded the $ID_{D'}$ and MSE of the inverse projection again on D. The results, shown in Fig. 5.9, reveal that, although DBM and SDBM exhibit similar surface behavior, their outcomes differ. The MSE of both methods decreases with q increasing, which is expected – a higher q is closer to the data dimensionality n, so both P and P^{-1} have an easier task. This drop in MSE is however more pronounced for DBM. The $ID_{D'}$ of both methods increases until reaching a plateau around 10-20 (for DBM) and 5 (for SSNP). This tells that the inverse projection task is *fundamentally* harder than the direct projection – indeed, even when having a much higher number of dimensions q than two as input for P^{-1}, it is not always possible to fully recover the full dimensionality n of the data.

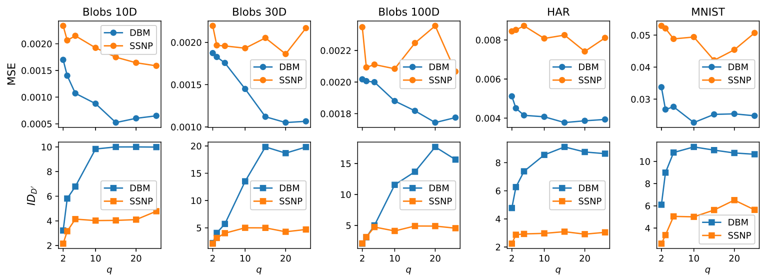


Figure 5.9: Changes of MSE and $ID_{D'}$ as a function of the latent dimensionality q for DBM (UMAP+NNinv) and SDBM (SSNP). See Sec. 5.5.4.

From this experiment, we infer that the 2D bottleneck of a (P, P^{-1}) transformation strongly affects the quality (error and ID) of the inverse projections. Increasing the number of dimensions q available as input for inverse projections can increase the quality of their output, but only up to a given limit. Moreover, from a practical viewpoint, increasing q is not possible – after all, we need to have $q \in \{2, 3\}$ if we want to directly visualize the decision maps. Exploring how this barrier can be overcome, *e.g.*, by more sophisticated inverse projection methods, are an important direction for future work.

5.5.5 *Limitations*

Our results are limited in their power by several factors. We used only two real-world datasets. Datasets having different characteristics, *e.g.*, local intrinsic dimensionality, data distribution, sparsity, or dimensionality, could potentially lead to new insights on how the tested decision maps and inverse projections work. A challenge here is to find datasets having ground-truth estimations of their intrinsic dimensionality. Separately, apart from the technical estimation of MSE, ID estimation, and gradient maps, we gauged the *suitability* of decision maps to practical applications only by qualitatively interpreting the visual smoothness of the resulting decision zones and boundaries. It is expected that, for the tested datasets and classifiers, such zones and boundaries should be smooth (Oliveira et al., 2023b). A more powerful ranking of decision maps would need to consider their actual use in ML engineering scenarios such as data augmentation or adversarial attacks, see *e.g.* (Machado et al., 2024).

5.6 CONCLUSIONS

We have analyzed the limitations of current inverse projection and decision map techniques used to visualize the behavior of machine learning classification models. Specifically, we compared the decision zones and boundaries depicted by six inverse projection techniques (and corresponding decision maps), with the actual zones and boundaries created by six classifiers on a three-dimensional real-world dataset. We found out that, in all cases, all the studied maps essentially capture a 2D structure embedded in the data space. We further extended our analysis to high-dimensional data by comparing the intrinsic dimensionality of the data with that of the inverse projection and backprojection of the map to the data space. We found that, as for the 3D data case, all studied map techniques still only cover essentially two-dimensional structures in the data space (modulo a certain amount of noise). We found that this surface-like limitation is particularly visible in areas located between the projected data points. Apart from this common aspect, we found several differences between the studied methods in terms of smoothness of the generated surface and accuracy by which it approximates the data points.

Our conclusion is that, when selecting inverse projection methods for constructing decision maps, methods which create smoother surfaces, *i.e.*, DBM, SDBM, and UMAP+RBF, are preferred in terms of predictability and ease of interpretation of the resulting maps, to methods that create tighter-fitting, less smooth, surfaces, and thus harder to interpret decision maps, *i.e.*, UMAP+iLAMP and DeepView. Finally, we showed that the recent MDS+iMDS inverse projection method is not suitable for constructing meaningful decision maps. Our work highlights fundamental limitations of all studied decision map techniques in terms of how much of a classifier's behavior they capture, but also where and how they choose to capture this behavior. These limitations are essential to understand when choosing which such technique to use in practice to construct decision maps but also when actually interpreting the resulting maps.

Future work can advance in a number of directions. The key one, we believe, is overcoming the 'surface limitation' of current decision map techniques. Likely, capturing a full high-dimensional space in a 2D map is not possible in general. Rather, one can focus on capturing specific areas in this space which are important to ML engineering, such as low-dimensional (curved) subspaces which contain most of a given dataset; areas close to the actual decision zones, where a classifier is most interesting to study; or areas where a classifier exhibits poor testing performance. An alternative way is to involve interacting allowing the user to move the current backprojected surfaces so as to sweep interesting zones of the data space, by *e.g.* generalizing the approach of Sohns et al. (2023), which interactively explores decision boundaries by the simple but limited PCA projection. In Chapter 6, we present a general purpose approach – that can accommodate any direct projection technique – that allows users to achieve the above by directly controlling the backprojected surface in an interactive way.

6 LOSS-CONTROLLED INVERSE PROJECTIONS

6.1 INTRODUCTION

As outlined several times from Chapter 1 to Chapter 4, multidimensional projections and their inverse projection counterparts methods enable a wealth of applications such as data imputation, classification model evaluation, and shape morphing (Rodrigues, 2020; Rodrigues et al., 2019; Oliveira et al., 2022; Schulz et al., 2020; Benato et al., 2024; Amorim et al., 2015; Machado et al., 2024). However, it is well known that direct projections suffer from information loss when the intrinsic dimensionality of their input data significantly exceeds that of the projection space (Sec. 2.2). Additionally, our work in Chapter 5 showed that all inverse projection techniques we know can only generate fixed, surface-like, structures in the data space. In other words, both direct and inverse projections are subject to significant *information loss* (Zheng et al., 2023). The key problem with this is that users expect to use inverse projections – or techniques using these such as decision maps – to examine large parts of a data space starting from the (2D) visual space. If both direct and inverse projections have limitations in this sense, such visualizations may be misleading in several ways.

In this chapter, we address the above-mentioned limitations by proposing a novel controllable inverse projection method, thereby answering research question **RQ3** introduced in Chapter 1. In contrast to our contributions in previous chapters, which fall under the topic *VIS4ML* (see Chapters 1 and 2), the work in this chapter falls under the topic *ML4VIS* as it employs deep learning techniques to create better visualizations[1].

The key novelty of our work is that we enable inverse projections to represent surface-like structures that can "sweep" through the high-dimensional data space in a user-controlled manner, achieving significantly higher coverage than all existing fixed-surface inverse projection techniques. In more detail, for a sample $\mathbf{x}$ projecting to $\mathbf{y} = P(\mathbf{x})$ by some user-chosen technique P, we find the information $\mathbf{z}$ that is lost during the projection P. Next, we enable users to (interactively) control how to combine $\mathbf{y}$

1 This chapter is based on the paper "LCIP: Loss-Controlled Inverse Projection of High-Dimensional Data" (Wang et al., 2025a).

(the information preserved by P) and $\mathbf{z}$ (the information lost by P) to compute our controllable inverse projection.

To realize this, we must answer three questions:

1. How to ensure $\mathbf{z}$ is *independent* of $\mathbf{y}$ (as we want to let P compute $\mathbf{y}$ and the user control $\mathbf{z}$)?

2. How to compute $\mathbf{z}$ for locations in the 2D space where *no ground-truth* sample $\mathbf{x}$ projects?

3. How to *control* the structure created by the inverse projection, to enable users to explore large parts of the data space?

Our proposal, called Loss-Controlled Inverse Projection (LCIP) answers the above questions as follows:

1. We minimize mutual information to disentangle the $\mathbf{y}$ and $\mathbf{z}$ spaces;

2. We use interpolation to fill in the 'empty' areas using learned $\mathbf{z}$ values for known data samples;

3. We allow users to interactively adapt $\mathbf{z}$ to control the shape of the inverse projection.

We introduce LCIP in Sec. 6.3. To show our method's abilities, we first study its disentanglement effect (Sec. 6.4.1) and next compare LCIP both visually and via quality metrics with 3 state-of-the-art inverse projections for 4 datasets and 2 direct projections (Sec. 6.4.2). Further on, we show how LCIP's main feature – its ability to control the shape of the inversely-projected structure in data space – creates higher-dimensional structures than surfaces, something existing methods cannot achieve (Sec. 6.4.3). Finally, we illustrate the added-value of being able to control the inverse projection by using LCIP to create a style transfer application (Sec. 6.5).

6.2 BACKGROUND AND RELATED WORK

Preliminaries: The key concepts and definitions we will use in this chapter have been introduced and used multiple times in this thesis. Projections map high-dimensional datasets in $\mathbb{R}^n$ to two-dimensional scatterplots (see Eqn. 2.2 and related text). Inverse projections map the entire 2D projection space to the $\mathbb{R}^n$ data space aiming to reverse the effects of a direct projection (see Eqn. 2.4.2 and related text). Most importantly, Chapter 5 showed that all existing inverse projections we are aware of map the 2D space to a *surface-like* structure embedded in data space – or, more formally, a structure with intrinsic dimensionality close to two (Ansuini et al., 2019). While this is not entirely unexpected given that an inverse projection aims to be a smooth mapping of $\mathbb{R}^2$ to $\mathbb{R}^n$, this means that inverse projections can only cover a very *limited subspace* of a given data space. Moreover, which exact such subspace an inverse projection generates in a given context is completely non-transparent to, and not controllable by, its users.

The above imply, in turn, serious limitations for inverse projection applications. Using inverse projections for data augmentation (Benato et al., 2024) will lack diversity. Separately, one cannot be sure that decision maps, which need to use inverse projections in their construction (Chapter 4), truly depict a classifier's behavior, as they show only a small part of the data space – more specifically, the intersections of the true decision boundaries with the surface-like structure created by P^{-1}. Finally, inverse projections generate only surface-like structures in the data space (Chapter 5). Hence, how to explore the data space *outside* such structures, and how to *control* this exploration, are challenges not answered by current inverse projection methods.

Information loss: Direct and inverse projections share some limitations concerning information loss. For direct projections, if the data X do not land on or close to a manifold, it is hard to construct a mapping $P(X)$ that fully preserves the data structure (Zheng et al., 2023). For inverse projections, the limitations are stronger – these always create a surface-like structure when mapping the 2D space to data space (Wang and Telea, 2024). Combining the above, if we consider the project-unproject cycle, *information loss* will always occur.

Inverse projections are structurally similar to data reconstruction or data generation tasks which aim to output high-dimensional data from low-dimensional representations. From this perspective, the (P, P^{-1}) cycle can be seen as an encoder-decoder structure, where the bottleneck is the 2D latent space. The dimensionality of this space is a critical factor for the quality of reconstruction and generation (Wang et al., 2016; Marin et al., 2021; Padala et al., 2021). Inspired by these observations, we aim to break the limitations imposed by our bottleneck – the visual space – by retrieving information lost during P and using it to drive the construction of P^{-1} under user control. This will enable our P^{-1} to span structures with higher intrinsic dimensionality than two, and also control where these are placed in data space.

6.2.1 *Learning disentangled representations and adversarial training*

As Sec. 6.1 stated, our first goal is to find the information $\mathbf{z}$ lost during projection independent of the projection $\mathbf{y}$. Independence is often relaxed to minimizing mutual information or separating complementary factors (Moyer et al., 2018). When achieved, this improves interpretability, reduces potential bias, and enhances generalization. For example, in handwriting recognition, separating text content $\mathbf{y}$ (what an actual letter *is*) from its style $\mathbf{z}$ (how the letter is *written*) helps model generalization. In speech processing, one aims to separate the speech content $\mathbf{y}$ from the speaker's identity $\mathbf{z}$ (Hadad et al., 2018).

Minimizing mutual information between two latent representations, also called learning disentangled representations, can be done via adversarial training (Mathieu et al., 2016; Hadad et al., 2018; Xie et al., 2017; Jaiswal et al., 2019; Zheng and Sun, 2019). Adversarial training, first used by generative adversarial networks (GANs) for image generation (Goodfellow et al., 2014), can generate high-dimensional realistic samples from low-dimensional latent codes. GANs jointly train a generator G and a discriminator Dis; G aims to create samples that are indistinguishable from real samples; Dis tries to distinguish real from generated samples. Adversarial training has been extended to other areas like robust machine learning, domain adaptation, and disentangled representation learning. While diffusion models are now more popular for image generation, GANs are significantly more efficient (Pan et al., 2023). For example, DragGAN (Pan et al., 2023), an interactive image manipulation method, uses the StyleGAN2 architecture (Karras et al., 2020).

Directly projecting high-dimensional datasets such as high-resolution images is challenging. In such cases, one typically describes the data using the latent space of a pre-trained classifier such as InceptionV3 or VGG16 (Espadoto et al., 2020; Benato et al., 2024). Since we focus on inverse projection, we prefer to retrieve images from the latent space of a generative model designed for this purpose such as StyleGAN2. In more detail: Let $\mathbf{w} \in \mathcal{W}$ be latent code that StyleGAN2 uses to generate images. Codes $\mathbf{w}$ can be obtained by inverting StyleGAN2 pre-trained on the same dataset (Karras et al., 2020). For a given $\mathbf{w}$, the corresponding image is then given by $G(\mathbf{w})$.

Adversarial training connects to our work in two ways: (1) We use it to enforce disentanglement between the information $\mathbf{z}$ and the projection $\mathbf{y}$ during the training of our inverse projection. (2)

We use the $\mathcal{W}$ space of an image dataset to ease the projection process.

6.3 DESIGN OF LOSS-CONTROLLED INVERSE PROJECTION

6.3.1 *Inverse Projection Deep Learning Network Architecture*

We implement our inverse projection using neural networks. Consider a dataset X and its projection $Y = P(X)$ computed by any user-chosen projection technique P. Our network has two key parts: an encoder *Enc* that computes the information in X lost by P; and a decoder *Dec* which is also our inverse projection P^{-1}. *Enc* reads the data X and outputs a latent code $Z = Enc(X) = \{\mathbf{z}_i\}$. *Dec* reads the concatenation $Y \oplus Z$ and outputs the inversely projected data $X' = Dec(Y, Z)$ (Fig. 6.1a). We also need to ensure that Z is not related to Y (see Sec. 6.2.1). We do this by a third network *Dis* which reads Z and outputs $Y' = Dis(Z)$ and is adversarially trained to minimize the error between Y and Y'. *Enc* and *Dec* are jointly trained to (i) minimize the reconstruction error between X and X', and (ii) maximize the difference between Y and Y'.

Let θ_{Enc}, θ_{Dec}, θ_{Dis} be the weight and bias parameters of *Enc*, *Dec*, and *Dis*, respectively. Let $L_{adv}(Y, Y')$ be the reconstruction loss between Y and Y', and $L_{rec}(X, X')$ the reconstruction loss between X and X'. When optimizing θ_{Dis}, L_{adv} is minimized, so *Dis* learns to predict Y from Z. The cost function J for optimizing *Enc* and *Dec* is defined as

$$J = L_{rec}(X, X') - \lambda L_{adv}(Y, Y'), \tag{6.1}$$

where $\lambda > 0$ is a hyperparameter that balances reconstruction *vs* adversarial loss, with the target of the optimization being

$$\min_{\theta_{Enc}, \theta_{Dec}} J, \tag{6.2}$$

that is, we minimize L_{rec} and maximize L_{adv} while keeping θ_{Dis} fixed. Once trained, *Enc* infers Z from X and *Dec* next inversely projects $Y \oplus Z$ to X'.

Implementation: We use fully-connected networks for *Enc*, *Dec*, and *Dis*. *Enc* has 3 hidden layers (sizes 512, 256, and 128). *Dec* has 4 hidden layers (sizes 128, 256, 512, and 1024). Each hidden layer of *Enc* and *Dec* is followed by a ReLU activation function. The final layer of *Dec* is followed by a sigmoid activation function. *Dis* has 2 hidden layers, each with a size of 128. Each hidden layer is

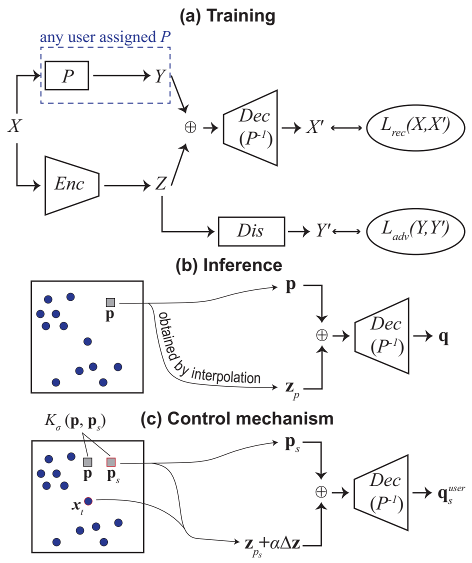


Figure 6.1: Illustration of our LCIP method. $\oplus$ denotes concatenation. (a) Training the networks. P is a user-selected DR method that projects X to Y. Enc encodes X into Z. Dis uses Z to predict Y. Dec (P^{-1}) uses Y and Z to reconstruct X. Disentanglement is enforced by the adversarial network Dis (see Sec. 6.3.1). (b) Inversely projecting a 2D point $\mathbf{q}$ to the data sample $\mathbf{p}$ (see Sec. 6.3.2). (c) The control mechanism refines the inverse projection by maneuvering $\mathbf{z}$ (see Sec. 6.3.3).

followed by batch normalization and a ReLU activation function. The dimension of Z is set to 16 – a value we empirically found to be sufficient to capture the information loss of all studied P techniques. We use all these settings consistently for all tested datasets.

We use mean squared error (MSE) for both L_{rec} and L_{adv} (Eqn. 6.1). For λ (Eqn. 6.1), we ran a grid search over the range $[0.005, 4]$, and found that $\lambda \in [0.01, 0.1]$ gives good results for all tested datasets. We train all networks using the Adam optimizer with a learning rate of 0.001. While training Dis, we have noticed that the adversarial training requires more steps to stabilize, since it learns from a changing input. Hence, at each iteration, we update Dis 5 times, then update Enc and Dec once. Our work is implemented using PyTorch (Paszke et al., 2019) with PySide6 (Qt) for the GUI, and is publicly available (Wang et al., 2025b).

6.3.2 *Computing* $\mathbf{z}$ *for the entire projection space*

To apply our inverse projection presented in Sec. 6.3.1 at a 2D point $\mathbf{p}_i$, we need a latent code $\mathbf{z}_i = Enc(\mathbf{x}_i)$, which in turn requires that we know the data sample $\mathbf{x}_i \in X$ that projects to $\mathbf{p}_i$, *i.e.*, $\mathbf{p}_i = P(\mathbf{x}_i)$. To apply our method to *any* 2D point $\mathbf{p}$, we need to estimate $\mathbf{z}_\mathbf{p}$ at that location. We do this by interpolating the $\mathbf{z}_i$ values of the training points X (see Fig. 6.1b). We tested two methods: weighted k-NN ($k = 10$ neighbors) and smoothed RBF with a parameter-free thin plate spline kernel. RBF gives a smooth surface, while weighted K-NN is slightly faster. We discuss the results of both interpolation methods in Sec. 6.4.2.

Having now the latent code $\mathbf{z}_\mathbf{p}$ for any $\mathbf{p} \in \mathbb{R}^2$, we can inversely project $\mathbf{p}$ to the data space (Fig. 6.1b) as

$$\mathbf{q} = P^{-1}(\mathbf{p}) = Dec(\mathbf{p}, \mathbf{z}_\mathbf{p}). \tag{6.3}$$

6.3.3 *Controlling the inverse projection*

To allow users to effectively control the shape of the inverse projection in data space, two questions arise: (1) How to do this *easily*, *i.e.*, by changing a small number of intuitive parameters in a direct, visual, way; and (2) How to make our P^{-1} cover zones in data space where *plausible* samples exist, so that P^{-1} is useful for real-world applications.

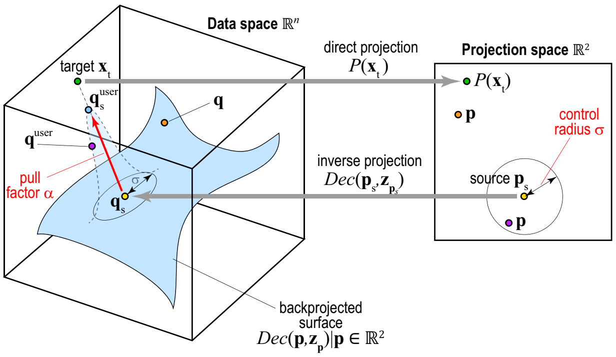


Figure 6.2: Controlling the inverse projection. User parameters are marked in red.

We solve both problems by the pipeline shown in Fig. 6.1c, which we explain next (see also Fig. 6.2). Let $\mathbf{x}_t \in X$, called a *target* sample, be a data point that we want to make our inverse projection go close to. Users can discover such points by brushing the projection with a tooltip to see their values $P(\mathbf{x}_t)$ (Fig. 6.2, green point). The user next selects a 2D *source* point $\mathbf{p}_s$ and manipulates it to control P^{-1} (Fig. 6.2, yellow point). We then 'pull' the inverse projection $P^{-1}(\mathbf{p}_s)$ of $\mathbf{p}_s$ towards $\mathbf{x}_t$ by adjusting $\mathbf{z}_{\mathbf{p}_s}$ (initially computed as described in Sec. 6.3.2) – see the red arrow in Fig. 6.2. To achieve this pull, we first compute the difference

$$\Delta\mathbf{z} = \mathbf{z}_t - \mathbf{z}_{\mathbf{p}_s} = Enc(\mathbf{x}_t) - \mathbf{z}_{\mathbf{p}_s} \tag{6.4}$$

between $\mathbf{z}_t = Enc(\mathbf{x}_t)$ and $\mathbf{z}_{\mathbf{p}_s}$. Intuitively put, Δz tells how much the latent codes of the source and target differ – that is, what we need to change in the source's inverse projection to make it become the target. Having Δz, we now compute the user-controlled inverse projection of $\mathbf{p}_s$ as

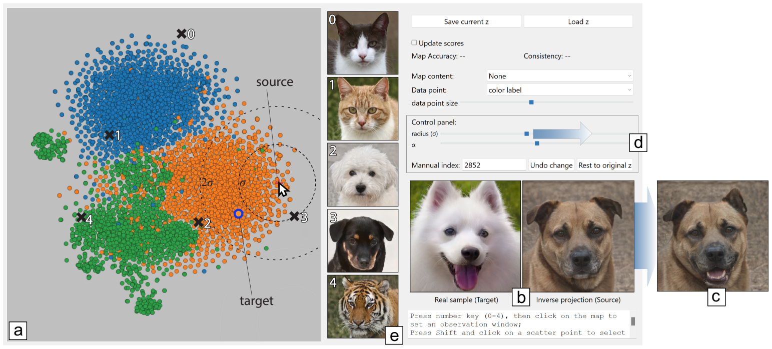
$$\mathbf{q}_s^{user} = Dec(\mathbf{p}_s, \mathbf{z}_{\mathbf{p}_s} + \alpha\Delta\mathbf{z}), \tag{6.5}$$

where $\alpha \in \mathbb{R}$ is a factor giving the pull magnitude (Fig. 6.2, light blue point). Yet, this adjustment only changes P^{-1} at the *single* location $\mathbf{q}_s^{user}$. 2D points close to $\mathbf{p}_s$ will not be affected by this pull, as their inverse projections still follow Eqn. 6.3. The inverse projection will exhibit a discontinuity or lack of smoothness around $\mathbf{q}_s^{user}$, which is undesired (Sec. 2.4.2). We could get smoothness by applying $\Delta\mathbf{z}$ to all such 2D points. However, this changes the inverse projection *globally* – the source $\mathbf{p}_s$ will equally influence all inversely-projected points, no matter how far these are from $\mathbf{p}_s$. We jointly achieve smoothness and local control by weighing the adjustment based on distances to the source $\mathbf{p}_s$. That is, after adjusting the inverse projection at $\mathbf{p}_s$ (Eqn. 6.5), we replace Eqn. 6.3 by

$$\mathbf{q}^{user} = Dec(\mathbf{p}, \mathbf{z}_p + \alpha K_\sigma(\mathbf{p}, \mathbf{p}_s)\Delta\mathbf{z}), \tag{6.6}$$

where $K_\sigma(\mathbf{p}, \mathbf{p}_s) = e^{-\frac{\|\mathbf{p}-\mathbf{p}_s\|^2}{2\sigma^2}}$ is a Gaussian centered at $\mathbf{p}_s$ and σ controls the source's influence. Larger σ values make the control more global and yield smoother inverse projections; smaller values have the opposite effect. When $\mathbf{p}$ is close to the source $\mathbf{p}_s$ (Fig. 6.2 purple point), its inverse projection gets pulled towards the target $\mathbf{x}_t$ – see light-blue bump on the surface in Fig. 6.2. When $\mathbf{p}$ is far from $\mathbf{p}_s$ (Fig. 6.2 orange point), its inverse projection stays on the surface given by Eqn. 6.3.

Figure 6.3 shows this control mechanism in action for a simple style transfer application. Here, the dataset X contains images of various animal faces. The user selects the source $\mathbf{p}$ by picking a location in the projection – not necessarily an actual projected sample. The image **q** for this point is computed by the inverse projection – see the sad-looking dog in Fig. 6.3a, right. Next, the user selects a target sample $\mathbf{x}_t$ – see the happy-looking dog in Fig. 6.3a, left. Pulling a slider changes α and morphs the source image towards the target. The user can see how far/strong the effect of changing the source propagates over the projection by selecting other images (Fig. 6.3e) and assessing their changes during source manipulation.



Map Accuracy: --
Consistency: --
Map content: None
Data point: color label
Mannual index: 2852

```
Press number key (0-4), then click on the map to
set an observation window;
Press Shift and click on a scatter point to select
```

Figure 6.3: LCIP illustrated by a style transfer application. If we want to make a dog smile, we proceed as follows: **1)** Brush the projection (a) to find a frowning dog which we want to change. The actual image of this dog is shown in (b, right). **2)** Use the same brushing to find a target image with a smiling face (b, left). **3)** Pulling a slider (d) changes the source *towards* the target – that is, makes the frowning dog (b) smile like the target, without changing its identity, see result in (c). Optionally, we can adjust the influence range σ to affect how other images close to the source also get changed towards the target – see the circles centered at mouse location in (a). We can also select a few additional points in the projection – see locations marked 0-4 in (a) – and inspect how the user-controlled inverse projection behaves there. Point 3 is close to the source, so it will be strongly influenced by the source's change; points 0, 1, and 4 are far away from the source, so they will not be affected.

6.4 EVALUATION

We evaluate our proposed inverse projection method on several datasets and projection techniques. We first study the effect of disentanglement both qualitatively and quantitatively and show that our latent codes **z** are indeed independent on the projected information $\mathbf{y}$ (Sec. 6.4.1). Then, we compare our inverse projection (without interactive control) to existing inverse projection methods and show we reach similar quality (Sec. 6.4.2). Finally, we show that our interactive control can break the surface-like limitation discussed in Sec. 6.4.3.

For P, we consider t-SNE and UMAP as these are the two highest-quality techniques in DR landscape (Espadoto et al., 2019a) and are also used in other inverse-projection studies (Espadoto et al., 2019c, 2021a; Schulz et al., 2020; Wang et al., 2023b). We use the following datasets:

MNIST: 70K samples of handwritten digits (0-9), each a 28^2 grayscale image flattened to a 784-size vector (LeCun et al., 2010).

Fashion-MNIST: 70K samples of 10 fashion categories (e.g., T-shirts, trousers, dresses), each a 28^2 grayscale image flattened to a 784-size vector (Xiao et al., 2017).

HAR: 10K samples of smartphone accelerometer and gyroscope data capturing six human activities (walking, walking upstairs, walking downstairs, sitting, standing, laying) (Anguita et al., 2012).

$\mathcal{W}$ of AFHQv2: AFHQv2 contains 15K color images of animal faces in 3 classes: dogs, cats, and wild animals (Choi et al., 2020). Its $\mathcal{W}$ is a $\mathbb{R}^{512}$ latent space used by StyleGAN2 models (Karras et al., 2020) to generate images (see Sec. 6.2.1). In the following, for ease of exposition, we will show the generated images $G(\mathbf{w})$ instead of the raw codes $\mathbf{w}$.

For a dataset $X = \{\mathbf{x}_i\}$, we first compute the projection $Y = \{\mathbf{y}_i\}$, $\mathbf{y}_i = P(\mathbf{x}_i)$. We next split D in a training $D_T = \{(Y_T, X_T)\}$ and testing $D_v = \{(Y_v, X_v)\}$ set (see Sec. 6.4.2). We set λ (Eqn. 6.1) to 0.01 (AFHQv2), 0.05 (HAR) and 0.1 (MNIST, Fashion-MNIST) respectively. Table 10 summarizes the above.

Table 10: Datasets used in our evaluation with dimensionality n, sample count $|D|$, training and testing set sizes $|D_T|$ and $|D_v|$, and values of hyperparameter λ (Eqn. 6.1).

Dataset	n	$\|D_T\|$	$\|D_v\|$	$\|D\|$	λ
MNIST	784	5000	5000	60000	0.1
Fashion-MNIST	784	5000	5000	60000	0.1
HAR	561	5000	5000	10299	0.05
$\mathcal{W}$ of AFHQv2	512	5000	5000	14336	0.01

6.4.1 *Added value of disentanglement*

To show the added value of the disentanglement, we compare our results using the loss in Eqn. 6.1 (called next *WithDis*) with the same network trained without L_{adv} (called next *NoDis*).

Quantitative evaluation: We measure disentanglement by how well can **z** predict **y**. The *worse* **z** can predict **y**, the better the disentanglement, *i.e.*, **z** and **y** are more independent (Jaiswal et al., 2018, 2019; Moyer et al., 2018). We measure this by training a *post-hoc* regression model to predict Y_T from $Z_T = Enc(X_T)$. This model is a neural network with one hidden layer (100 units) followed by ReLU activation, trained for 200 epochs using the Adam optimizer. We measure R^2 and MSE on a hold-out test set. We expect that *WithDis* should yield low R^2 and high MSE – that is, **z** and **y** are independent and/or different. Conversely, we expect that *NoDis* should yield high R^2 and low MSE – that is, **z** and **y** are correlated and/or similar. Table 11 shows that MSE and R^2 for *WithDis* and *NoDis* indeed match the expectations, thus confirm our claimed disentanglement.

Table 11: Measuring disentanglement: How well can **z** predict **y**?

Dataset	*P*	***WithDis***		***NoDis***	
		MSE	R^2	*MSE*	R^2
MNIST	UMAP	17.1	0.032	1.7	0.900
MNIST	tSNE	1497.5	0.011	161.9	0.892
FashionMNIST	UMAP	19.9	0.201	0.7	0.968
FashionMNIST	tSNE	1284.4	0.096	79.4	0.944
HAR	UMAP	48.8	0.098	6.9	0.862
HAR	tSNE	1337.8	0.170	152.7	0.892
$\mathcal{W}$ of AFHQv2	UMAP	5.8	0.014	3.6	0.402
$\mathcal{W}$ of AFHQv2	tSNE	664.8	0.024	336.2	0.500

Qualitative evaluation: We visually evaluate disentanglement by selecting two samples $\mathbf{x}_1$ and $\mathbf{x}_2$ of MNIST dataset and their t-SNE projections $\mathbf{y}_1$ and $\mathbf{y}_2$. These are two images of digits 0 and 7 – see Fig. 6.4a. Let $\mathbf{z}_1 = Enc(\mathbf{x}_1)$ and $\mathbf{z}_2 = Enc(\mathbf{x}_2)$ be the codes of these two images. These are shown in a UMAP projection of the **z** values for the dataset in Fig. 6.4b. We linearly interpolate between $\mathbf{y}_1$ and $\mathbf{y}_2$, and $\mathbf{z}_1$ and $\mathbf{z}_2$, respectively, with 10 steps in

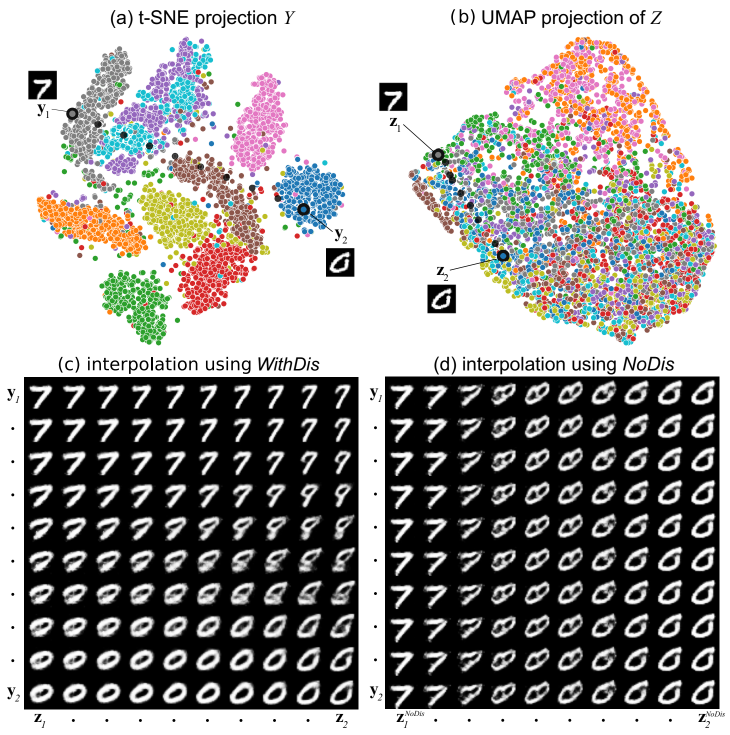


Figure 6.4: Showing disentanglement on the MNIST dataset. (a) 2D t-SNE projection Y. (b) UMAP projection of Z using *WithDis*. (c) Inverse projections of the linear interpolation between two data points in the Z and Y spaces, using *WithDis* (c) and *NoDis* (d).

each of these two dimensions. This yields a total of 10^2 interpolated values $(\mathbf{y}, \mathbf{z})$ which we next inversely project to the data space using both *WithDis* (Fig. 6.4c) and *NoDis* (Fig. 6.4d).

Several observations follow. First, we see from the t-SNE projection that the label information is well-preserved in the projection space (Fig. 6.4a) – that is, digits of the same *type*, *e.g.*, zeroes or sevens, are well grouped. In contrast, the UMAP projection of $\mathbf{z}$ strongly mixes labels (Fig. 6.4b). This is desired, as $\mathbf{z}$ should capture a digit's writing style and not its class. We next see that, as we change $\mathbf{z}$ using *WithDis*, the *style* of the digit changes – see columns in Fig. 6.4c; while, as we change $\mathbf{y}$, the digit *itself* changes – see rows in Fig. 6.4c. Hence, *WithDis* disentangles $\mathbf{y}$ and $\mathbf{z}$ well. In contrast, for *NoDis*, the inverse projection changes only when $\mathbf{z}$ changes – see columns in Fig. 6.4d; when $\mathbf{y}$ changes, the digit stays the same – see rows in Fig. 6.4d. Hence, *NoDis* keeps the latent codes **z** and **y** entangled.

6.4.2 *Comparison to other inverse projection methods*

We now compare our inverse projection LCIP to iLAMP (dos Santos Amorim et al., 2012), RBF (Amorim et al., 2015), and NNInv (Espadoto et al., 2019c). We show that LCIP yields similar if not higher quality even without using its control mechanism (which we discuss separately in Sec. 6.4.3).

Inverse projection error: We first measure the Mean Squared Error (MSE) of the inverse projection on the test set D_v

$$MSE = \frac{1}{|D_v|} \sum_{(\mathbf{y},\mathbf{x}) \in D_v} \|\mathbf{x} - P^{-1}(\mathbf{y}, \mathbf{z})\|^2, \tag{6.7}$$

where **z** is ignored for iLAMP, RBF, and NNInv. For LCIP, we compute **z** at $\mathbf{y} \in \mathbb{R}^2 \setminus Y$ by smoothed RBF and weighted k-NN interpolation of the $\mathbf{z}_i$ values (see Sec. 6.3.2). We call these two interpolation variants LCIP (rbf) and LCIP (knn), respectively. We also evaluate $\mathbf{z}_i = Enc(\mathbf{x}_i), \mathbf{x}_i \in X_v$ and denote this computation as LCIP*. While this is not the intended way to get **z** (as we cannot do this for *any* 2D point, see Sec. 6.3.2), this gives the performance of our method if we could compute **z** exactly.

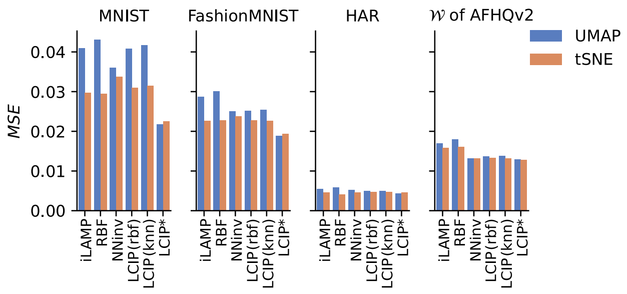


Figure 6.5: MSE of the studied inverse projections.

Figure 6.5 shows that iLAMP, RBF, NNInv, and LCIP have similar MSE on D_v. Yet, LCIP* has a lower MSE than the others on MNIST and Fashion-MNIST. Hence, our method can produce inverse projections with *lower error* when **z** is properly provided. This quality gap between LCIP* and LCIP (rbf) or LCIP (knn) can be filled by interaction (see next Sec. 6.5).

Visual quality in gap areas: The above MSE testing can only be done for points in D_v, *i.e.*, where we have ground truth on P^{-1}

in terms of data samples. As explained several times so far, a 2D projection space mainly consists of gap areas where no sample projects, *i.e.*, we have no ground truth – and it is precisely there that one wants to use inverse projections. One way to assess quality there is to study how inversely projected samples from gap areas *look* like. Ideally, we want to obtain *plausible* samples which follow the overall nature of the data in a given dataset.

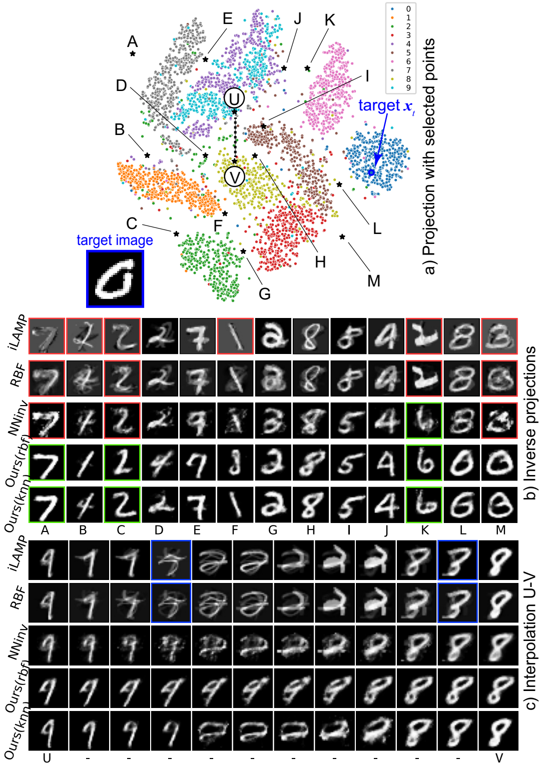


Figure 6.6: Visual comparison of the inverse projection on MNIST dataset. 13 points are selected in the projection space at various distance from projected samples (A-M, top image). The middle set of images shows the inverse projections at these locations using the tested inverse projection techniques. The bottom set of images shows results obtained by inversely projecting points located along a line between the locations U and V in the projection.

Figure 6.6 shows the four studied P^{-1} methods on MNIST. Images (b) show that LCIP produces more plausible digits than the other methods even at locations far away from data samples (*e.g.*, A, C, M). We also see that iLAMP and RBF are sensitive to outliers. For example, there is a single green point (near K, class 2) located in between pink (class 4) and purple (class 6) clusters, while the main green cluster is far down at the bottom of the projection. It makes sense that this green point is an outlier and this gap area is supposed to be something between 6 and 4. Yet, the inverse projection of 'K' (close to the green point) is a '2' in iLAMP and RBF, while it is a '6' in NNinv and LCIP. Also, iLAMP creates gray backgrounds (*e.g.*, A,B,C,F,M) which are not only far away from the actual data distribution but also not what one would expect for MNIST. Images (c) show the inverse projections from points along a line between images U (digit 9) and V (digit 8) in the projection. iLAMP, RBF, and NNInv generate spurious shapes that do not resemble any digit during this interpolation process. In contrast, LCIP morphs the 9 to an 8 following, we argue, a more natural set of intermediate images.

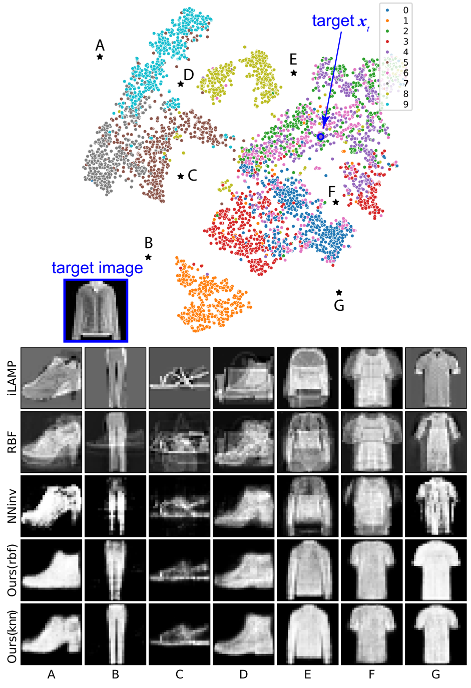


Figure 6.7: Visual comparison of the inverse projection on Fashion-MNIST dataset.

On Fashion-MNIST, iLAMP and RBF mix several images in the reconstruction (Fig. 6.7). For example, at point A, iLAMP and RBF mix high heels and boots; at point B, RBF mixes shoes and pants; at point F, iLAMP and RBF mix a wider and a narrower

T-shirt. NNinv doesn't have this problem, but it produces jagged (*e.g.*, at points A, B, C, G) or ambiguous shapes (*e.g.*, at points E, F). LCIP keeps the reconstructed shape clear and recognizable in all cases.

Finally, on AFHQv2, all methods work well when the inversely-projected 2D points are close to training samples. Once a 2D point is further away, iLAMP immediately becomes problematic (see A, B, C, E, J, P). In extreme cases, all methods but ours have some issues. For example, at point C (see Fig. 6.8), iLAMP produces a mass of colors with indiscernible shapes; the image by RBF has color distortions and the dog in the image only has one ear; NNinv produces a dog but in a strange appearance, also with color distortions. In contrast, LCIP produces realistic images in all cases.

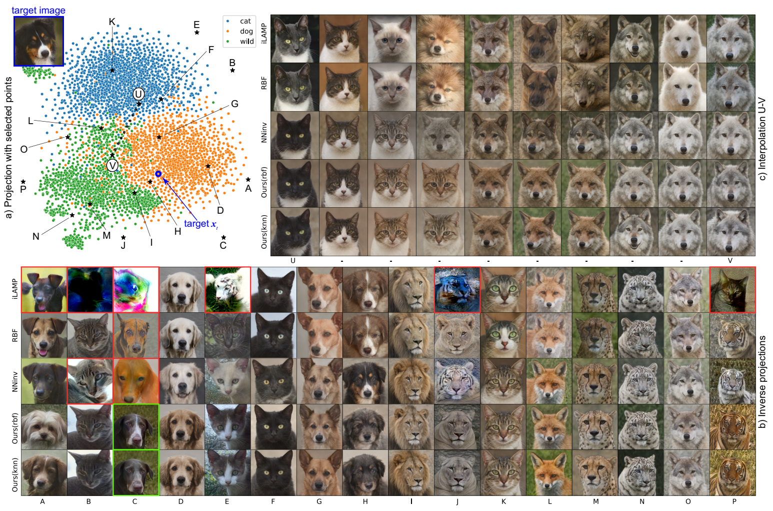


Figure 6.8: Visual comparison of the inverse projection, AFHQv2 dataset. (a) Dataset projection with 16 selected locations both close and far away from the data samples (A-P). (b) Inversely projected resulting images at these locations created by all tested methods. (c) Results obtained by inversely projecting points along a line between the locations U and V in the projection.

Smoothness comparison: We next evaluate smoothness using *gradient maps*, introduced in Sec. 2.5.2.1 and used extensively in Chapter 4. Figure 6.9 shows these maps for the tested inverse projections on the MNIST dataset with the UMAP projection; Bright colors indicate high gradients, that is, points where the inverse projection 'jumps' in data space when its input moves between neighbor pixels. Such jumps are undesirable (see Sec. 2.5.2.1). We

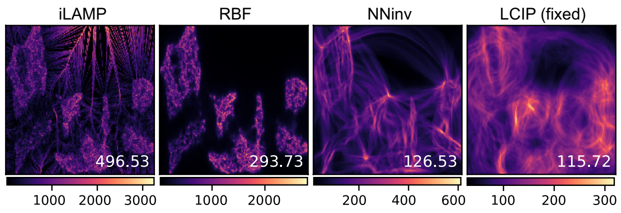


Figure 6.9: Gradient maps for the tested inverse projections for UMAP on MNIST.

see that LCIP achieves the smallest gradient norms (see color legends) for all datasets and direct projections.

Computational speed Figure 6.10 shows timings for the four studied inverse projection methods on a desktop with an Intel Core i5-12400 CPU and NVIDIA GeForce GTX 3090 GPU. The y-axis shows total time, *i.e.* the (constant) training time and inference time (linear in the projected points count). All methods show similar speed except iLAMP which is significantly slower. Although LCIP requires slightly more training time due to its adversarial training, its slope is nearly identical to NNinv and RBF, showing the same high scalability. Separately, our evaluations show that the results of LCIP (rbf) and LCIP (knn) are very similar. Since LCIP (rbf) is theoretically smoother, we will use it in our following experiments.

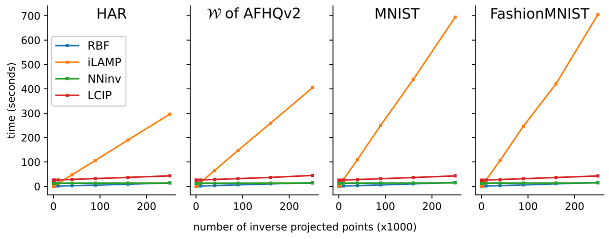


Figure 6.10: Inverse projection speed. The training time equals the y-intercept of the graphs. The slopes of the graphs depict how inference time depends on the number of inversely projected samples.

6.4.3 *Controllability: Going beyond a fixed surface*

We have shown so far that LCIP can construct a 'fixed' inverse projection from a given dataset X and its projection $P(X)$ with results which are comparable – and often better – than other existing inverse projection techniques in terms of generating plausible results in gap areas, inverse projection MSE, inverse projection smoothness, and speed. Yet, the key feature of our method is its ability to **dynamically** control the inverse projection, which we describe next.

Effects of control: Figures 6.11-6.12 show how the inverse projection changes when their $\mathbf{z}$ values are adjusted towards selected targets in MNIST, Fashion-MNIST, and AFHQv2. Targets are marked by $\mathbf{x}_t$ (blue) in Fig. 6.6-6.8 with their images shown as insets. In Figs. 6.11-6.12, topmost rows show the selected source images ($\alpha = 0$); rows below show how the inverse projection 'sweeps' the data space between source and target as the user increases α. For example, when selecting an italic-like '0' digit in MNIST as target, as we increase α, the inverse projection gradually become more italic, no matter which is the selected source image (Fig. 6.11). We see similar changes in Fashion-MNIST, such as subtle changes in the style of shoes and shirts (Fig. 6.11 bottom). For AFHQv2, the selected target is a dog tilting its head – see inset image in Fig. 6.8. As we increase α, all selected source animals in the inverse projection rotate their heads with similar angles (Fig. 6.12).

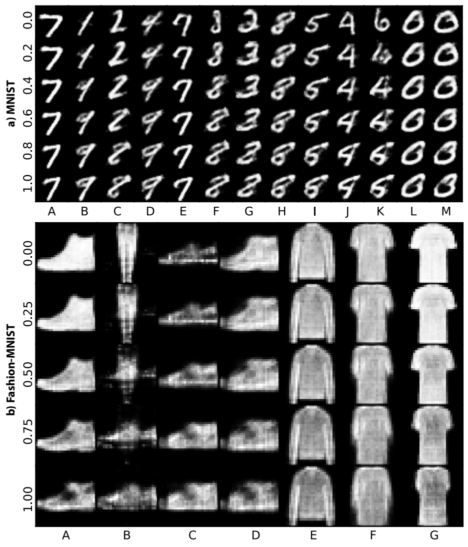


Figure 6.11: Controlling the inverse projection for MNIST (a) and Fashion-MNIST (b). Targets are the blue-outlined inset images in Fig. 6.6 for (a) and Fig. 6.7 for (b). Rows in the two images show the effect of increasing user control α.

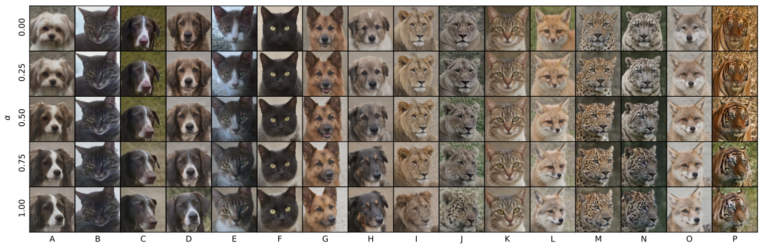


Figure 6.12: Controlling the inverse projection for AFHQv2. The target is the blue-outlined image in Fig. 6.8.

Increased reach: Recall, from Chapter 5, that the main limitation we identified for existing inverse projections was their fixed surface-like behavior. While LCIP can generate inverse projections that go beyond a fixed surface, a key question is *how much* beyond such a surface can we reach. To measure this, we evaluate the Intrinsic Dimensionality (ID) of the inverse projection by the Minimal Variance method (Tian et al., 2021), for all pixels in an image (300×300), without interaction. We call this ID value the *baseline*. Next, we adjust the inverse projection by globally adding $\Delta\mathbf{z}$ values, for 50 uniformly sampled values of $\alpha \in [0, 0.2]$, and measure the ID of all resulting inversely-projected points taken together. Figure 6.13 shows that the baseline has $ID \simeq 2$ at roughly all pixels (with small higher-ID areas close to the sample points), so we generate roughly a surface embedded in $\mathbb{R}^n$, much like other inverse projection techniques (Chapter 5). When using control, we get an ID roughly equal to 3, *i.e.*, $\Delta\mathbf{z}$ 'shifts' our inverse-projection surface to span a 3D space in $\mathbb{R}^n$. Some ID values of 2 are likely due to the $\mathbf{z}$ value of those areas being insensitive to the selected target, *i.e.*, $\mathbf{z}_t$ and $\mathbf{z}_{p_s}$ are close in the first place. Note that we only use a *single* target point here. If we used k target samples $\mathbf{x}_t$ which span a k-dimensional space in $\mathbb{R}^n$, we would obtain an inverse projection of ID $\simeq k$.

Flexibility of control: We further study the impact of using more control targets $\mathbf{x}_t$. For this, we consider a training set X_T with 5000 points. For each $\mathbf{x}_i \in X_T$, we compute its latent code $\mathbf{z}_i = Enc(\mathbf{x}_i)$, consider it as a target for our control mechanism, and consider each pixel $\mathbf{p}$ of the image as source point. For each such $\mathbf{p}$, we measure the variance

$$V(\mathbf{p}) = \frac{\sum_{j=1}^{n} Var(\{P^{-1}(\mathbf{p}, \mathbf{z}_i)\}^j)}{\sum_{j=1}^{n} Var({X_T}^j)} \tag{6.8}$$

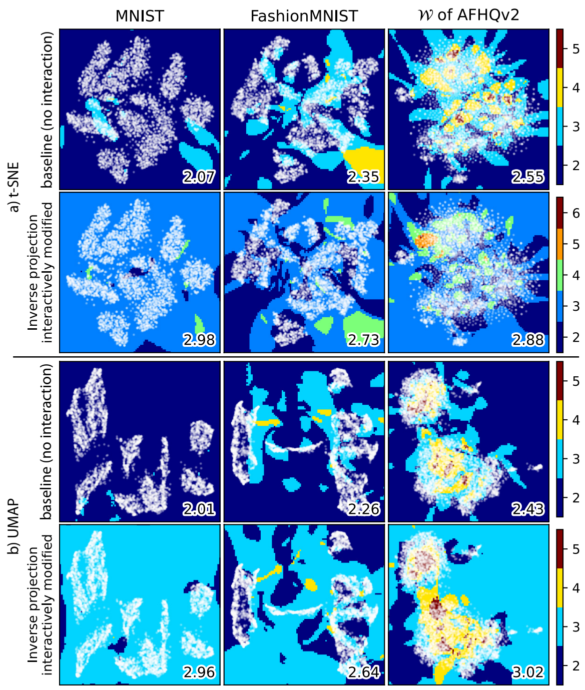


Figure 6.13: Intrinsic dimensionality of LCIP without and with interaction for different datasets and direct projections.

over the set $\{Dec(\mathbf{p}, \mathbf{z}_i)\}$, across all n data dimensions. Here, superscript j denotes the j^{th} dimension of a data sample. Higher values of $V(\mathbf{p})$ indicate that the inverse projection of $\mathbf{p}$ is more sensitive to control, and vice versa. The denominator normalization factor in Eqn. 6.8 accounts for the spread of the training set X_T in data space. If this spread is low, then we should not expect that our inverse projection reaches far further in the data space, and vice versa.

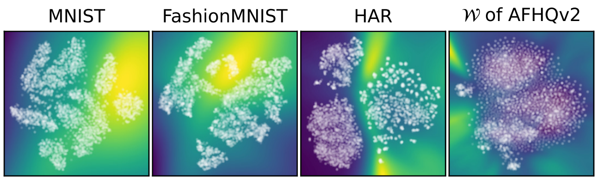


Figure 6.14: Normalized variance of inverse projections produced by 5000 different $\mathbf{z}$ values, t-SNE direct projection. See Eqn. 6.8 and related text.

Figure 6.14 shows the results. On $\mathcal{W}$ of AFHQv2 and HAR, variance is lower near data samples and higher in gap areas, telling that inverse projections are more 'nailed' when there is ground-truth data around, and more flexible when no ground truth nearby. On MNIST and Fashion-MNIST, the pattern seems not to be related to the distance to data samples. We believe that this is an effect of t-SNE's known tendency of t-SNE to compress (or stretch) point neighborhoods from data space to projection space (Chatzimparmpas et al., 2020b). That is, areas showing a low normalized variance in Fig. 6.14 may actually map points which are close in data space, where our inverse projection does not have the freedom to move much; conversely, areas of high variance may map points far away in data space, where our inverse projection has more freedom to move. These maps provide users with insights on where interaction is likely to be most *effective*: Interacting with source points in high-variance areas will produce inverse projections which 'sweep' the data space more freely, which is the core goal of interaction; interacting with source points in low-variance areas is likely not going to discover new areas in data space.

6.5 USER CONTROL OF THE INVERSE PROJECTION

We now show the added value of the controllability of our inverse projection method. All experiments use the interactive tool shown in Fig. 6.3a.

Based on how far the selected source $\mathbf{p}_s$ is from the selected target $P(\mathbf{x}_t)$ in projection space, we distinguish two types of control: (1) target is close to source and (2) target is far away from source. In (1), we expect the controlled P^{-1} to gradually but *fully* change towards the target; In (2), we only expect a *partial* change, *e.g.*, style-wise. We detail both scenarios next.

6.5.1 *Local control: Target is close to source*

Consider the limit case where $P(\mathbf{x}_t) = \mathbf{p}_s$. The difference between the target $\mathbf{x}_t$ and the controlled inverse projection $Dec(\mathbf{p}_s, \mathbf{z}_{p_s})$ is fully controlled by $\mathbf{z}_{p_s}$. When $\mathbf{z}_{p_s} = Enc(\mathbf{x}_t)$, the inverse projection of $\mathbf{p}_s$ becomes $Dec(P(\mathbf{x}_t), Enc(\mathbf{x}_t)) \simeq \mathbf{x}_t$. In other words, the inverse projection *fully* changes towards the target $\mathbf{x}_t$. We next show how our control helps with two challenges that frequently occur in inverse projection usage.

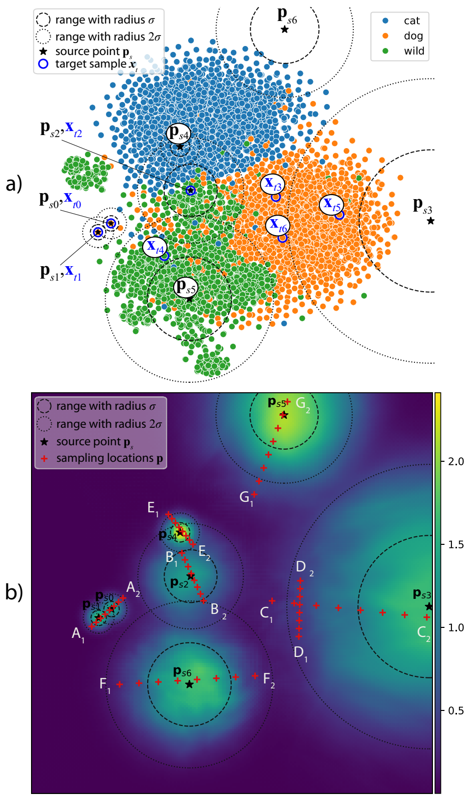


Figure 6.15: Close and far-away control of the inverse projection, AFHQv2 dataset. (a) 2D locations where we performed interaction. See also Figs. 6.16,6.17. (b) Assessing smoothness of controlled inverse projection at various sampling locations (+). Color encodes the distance $\|\mathbf{q} - \mathbf{q}^{user}\|$ at each pixel. Figure 6.18 shows the inverse projections at these sampling locations.

Inverse projection correction: While good inverse projections should have a low MSE (Eqn. 6.7), they do not yield $P^{-1}(P(\mathbf{x})) = \mathbf{x}$ at *all* projected samples $P(\mathbf{x})$. For instance, in Fig. 6.15a, $\mathbf{p}_{s0}$ and $\mathbf{p}_{s1}$ are two points at the margin of clusters. As such, their inverse projections $\mathbf{q}_{s0}$ and $\mathbf{q}_{s1}$ should barely be influenced by other points and, ideally, equal the data samples $\mathbf{x}_{t0}$ and $\mathbf{x}_{t1}$ that project there. Yet, this is not the case – compare images $\mathbf{q}_{s0}$, $\mathbf{q}_{s1}$ to $\mathbf{x}_{t0}$, $\mathbf{x}_{t1}$ in Fig. 6.16, respectively. We can adjust our controllable inverse projection to make it *closer to the data*. As we add $\Delta\mathbf{z}$ to $\mathbf{z}_{p_s}$, the inverse projection $\mathbf{q}^{user}$ gradually changes towards the

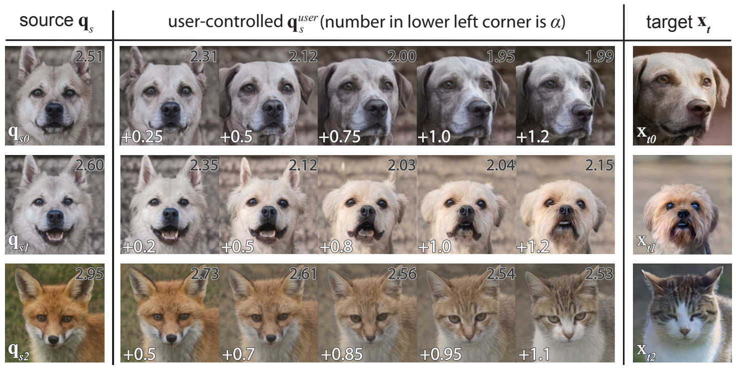


Figure 6.16: Local control – adjusting the inverse projection when $P(\mathbf{x}_t) \simeq \mathbf{p}_s$. Locations of $\mathbf{q}_s$ and $P(\mathbf{x}_t)$ are shown in Fig. 6.15a. Numbers in upper right corners show $\|\mathbf{x}_t - \mathbf{q}_s^{user}\|$. Numbers in lower left corners show the control parameter α.

target $\mathbf{x}_t$ – see Fig. 6.16 top two rows. The distances $\|\mathbf{q}^{user} - \mathbf{x}_t\|$ are also reduced, reaching minima for $\alpha = 1$ (Fig. 6.16).

Overlap separation: Overlapping points in a projection are common. In such areas, P is not injective, so formally it is not invertible. For example, in Fig. 6.15a, the area around point $\mathbf{p}_{s2}$ – which corresponds to a cat (denoted $\mathbf{x}_{t2}$) – shows several overlapping samples of cats (blue) and wild animals (green). This overlap is due to the projection – the samples are actually separable in data space. Existing inverse projection methods will map $\mathbf{p}_{s2}$ to a cat, a wild animal, or a mix of them in a *fixed* and *not controllable* way. Our method allows flexibly controlling the inverse projection of $\mathbf{p}_{s2}$ to be more like a cat or a wild animal: We see that $\mathbf{q}_{s2}$ is initially a fox (Fig. 6.16, bottom row, left). Setting the cat image $\mathbf{x}_{t2}$ as the target and $\mathbf{p}_{s2}$ as source, we see how the inverse projection gradually transforms from a fox into a cat as we increase α (Fig. 6.16, bottom row, columns $\mathbf{q}_s^{user}$).

6.5.2 *Far-away control: Target is far from source*

In this scenario, the controlled $P^{-1}(\mathbf{p}_s, \mathbf{z}_{p_s})$ will not go completely toward $\mathbf{x}_t$, since $P(\mathbf{x}_t) \not\simeq \mathbf{p}_s$. Rather, only what is controlled by $\mathbf{z}$ will change. Yet, it is hard to tell what $\mathbf{z}$ *exactly* controls without prior knowledge or exploration since this depends on the information that is not captured by the projection $\mathbf{y}$, which in turns depends on the actual projection technique P used. In our studies, we found that, for the MNIST–t-SNE combination, the digit is controlled by $\mathbf{y}$, while $\mathbf{z}$ controls the digit's style; for the AFHQv2–t-SNE combination, the type of animal faces is controlled by $\mathbf{y}$, while the animal poses are controlled by $\mathbf{z}$. Controlling $\mathbf{z}$ shapes the inverse-projected surface as desired, enabling applications like user-controlled data generation, as illustrated next.

Consider a source point $\mathbf{p}_s$ far from any target, *i.e.*, within a so-called gap area in the projection. All existing inverse-projection techniques produce surface-like structures in such areas (see Chapter 5). Our control breaks this limitation: Take point $\mathbf{p}_{s3}$ which is in a gap area (Fig. 6.15a). Its inverse projection is a sad-looking puppy ($\mathbf{q}_{s3}$, Fig. 6.17 top-left). We now pick as the target a happy dog ($\mathbf{x}_{t3}$ in Fig. 6.15a, Fig. 6.17 top-right). As we increase $\Delta\mathbf{z}$, the sad puppy gradually smiles and eventually laughs with open mouth (Fig. 6.17, $\mathbf{q}^{user}$, top row). Note that the puppy's appearance did not change to the target one (*e.g.*, hair color and ear shapes); only its 'style' changed. This control also works with source and target from different *classes*. The inverse projection $\mathbf{q}_{s4}$ of $\mathbf{p}_{s4}$ is a cat looking ahead (Fig. 6.17, left column, second-top image). We choose a tiger looking up as the target ($\mathbf{x}_{t4}$ in Fig. 6.17, right column, second-top image). By adding $\Delta\mathbf{z}$, the cat gradually looks up without becoming a tiger (Fig. 6.17, $\mathbf{q}^{user}$, second row).

Decreasing $\Delta\mathbf{z}$ gets an opposite effect. We first choose as source point $\mathbf{p}_{s5}$ a front-facing cheetah (Fig. 6.17, left column, third image from top). Our target is a left-facing dog $\mathbf{x}_{t5}$ (Fig. 6.17, right column, third image from top). By adding $\Delta\mathbf{z}$, the cheetah gradually turns its head to left (Fig. 6.17, $\mathbf{q}^{user}$, row 3). Decreasing $\Delta\mathbf{z}$, the cheetah turns its head to right (Fig. 6.17, $\mathbf{q}^{user}$, row 4). At around $\alpha = -0.55$, the cheetah starts changing into a tiger. This is not surprising since cheetahs and tigers overlap in the projection. This decrease operation triggers the overlap separation (see Sec. 6.5.1). A final example considers a point $\mathbf{p}_{s6}$ in a gap area whose inverse projection is a black-and-white, front-facing, cat (Fig. 6.17, left column, bottom image). We set a left-facing dog as the target ($\mathbf{x}_{t6}$, Fig. 6.17, right column, bottom image). Increasing $\Delta\mathbf{z}$, the cat gradually turns its head left (Fig. 6.17, $\mathbf{q}^{user}$, row 5). Decreasing $\Delta\mathbf{z}$, the cat tilts its head to the opposite direction and changes fur color to brown (Fig. 6.17, $\mathbf{q}^{user}$, row 6).

These examples show the flexibility and potential of LCIP. By adjusting $\Delta\mathbf{z}$, users can control subtle *style* features, such as the expression or direction a subject is facing, without altering a sample's fundamental *identity*. This ability extends across different classes and also in areas far from any projected data sample. This shows LCIP's support for user-controlled data generation, where

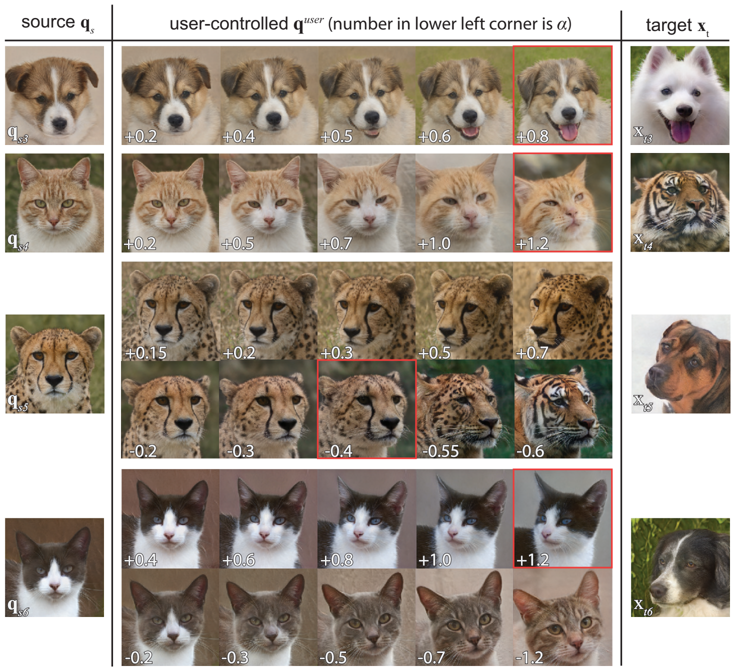


Figure 6.17: Far-away control – adjusting the inverse projection when $P(\mathbf{x}_t) \not\approx \mathbf{p}_s$. Locations of $\mathbf{q}_s$ and $P(\mathbf{x}_t)$ are shown in Fig. 6.15a.

maintaining the original data's integrity, while introducing desired variations, is crucial (Wang et al., 2022a).

6.5.3 *Smoothness of controlled projections*

Section 6.4.2 showed the smoothness of our inverse projection in absence of user control. User control should create a smooth inverse projection since the input of *Dec* smoothly varies with $\mathbf{p}$, α, and σ (Eqn. 6.6) and since *Dec* itself is smooth (being the neural network described in Sec. 6.3). Yet, it is worth to practically test this smoothness. For this, we choose 7 source points $\mathbf{p}_{s1} \ldots \mathbf{p}_{s6}$ and sample several lines $A_1 \ldots A_2$ – $G_1 \ldots G_2$ in the projection space, with 8 samples per line. Figure 6.15b shows the sampling locations (+) and source points (★). We next set different α and σ values for the source points and compute their inverse projections. Figure 6.18 shows these inverse projections without and with control. Images in each row – thus, over a set of linearly-spread sampling points – smoothly change in both cases. To further confirm this, Fig. 6.15b color-codes the difference between the inverse projection without and with control $\|\mathbf{q} - \mathbf{q}^{user}\|$ at each pixel. The result is a *smooth* signal, with large values close to the source points and small values further away, exactly as aimed by the local control aimed in Eqn. 6.6.

6.6 DISCUSSION

We next discuss several aspects of our controllable inverse projection method.

Controllability: To our knowledge, our method shows for the *first time* that an inverse projection can be controlled by users – rather than being fully prescribed by a dataset X and its projection $P(X)$. This also breaks the limitation that inverse projections land on a surface embedded in the data space (Sec. 6.4.3).

Genericity: While we illustrated LCIP with style transfer for images, our method is not limited to manipulating image data and, also, should not be seen as *competing* with other dedicated style-transfer methods. We chose the image style-transfer application as it is a very simple and illustrative one for our technique. Yet, our method can be applied to control inverse projections generated by any user-selected projection $P(X)$ of any high-dimensional dataset X. To our knowledge, this is the first inverse projection method that allows such control for any projection technique P and any dataset.

Quality: Our evaluations show that our inverse projection has better quality in gap areas as compared to existing inverse projection techniques (Sec. 6.4.2). Our inverse projection creates smoother-varying data samples (when the input 2D points smoothly change). This is an important added value point for any user-driven applications of inverse projections since a lack of smoothness in the inverse projection would cause at least difficulty, and most likely confusion, for users who aim to control how 2D points are backprojected to the data space.

Scalability: Our method scales in the size and dimensionality of its input data as well as state-of-the-art inverse projection methods, *e.g.*, NNinv (Sec. 6.4.2). Its speed is linear in the number of inversely projected points.

Ease of use: To inversely project a so-called source point, our method requires at minimum only selecting that point in a 2D projection space. If one desires to modify the inverse projection, the user needs additionally to select one target point (from the projected ones) and a 'pull' factor α that tells how much the source point will change towards the target. All such operations

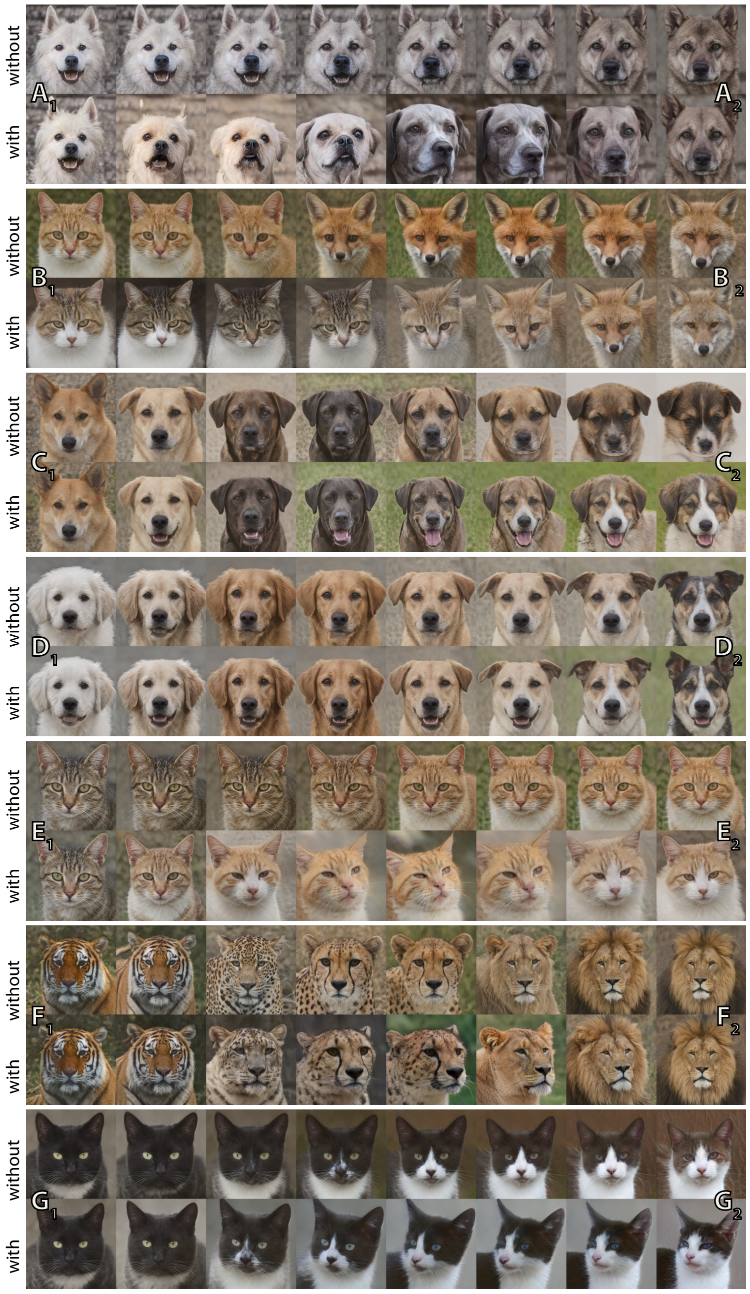


Figure 6.18: Control smoothness: Inverse projections at 8 sampling points (columns) around 7 different source points (rows), with and without user control. Each image corresponds to a '+' in Fig. 6.15b.

are simple to perform in a general-purpose user interface by clicking (to select points) and pulling a slider (to change α). Additionally, users can change how smooth the inverse projection is around a source point by changing σ – again, by pulling a slider.

Limitations: First and foremost, our method relies implicitly on disentangling the information captured by a projection operation P from what P cannot capture, and subsequently manipulating this information to control the inverse projection. From a technical perspective, we argued that this makes good sense. However, from a practical perspective, it is by far not clear to the common user what a given P would capture (and thus not subject to user control) and what would be, consequently, left to the said control. Our experiments showed that, for selected datasets, the projection captures the core similarity of items, while the lost information – under user control – mainly affects a generic attribute we called 'style'. Yet, what exactly style is; how it differs from what a given projection captures; and how much is this style controllable in practice, are all questions that we cannot formally answer for all datasets and all projection methods. In connection to this, it is not yet obvious how the control of inverse projections which we proposed can assist a *broad* set of applications. Our examples of style transfer on images are, we hope, convincing. Yet, more use-cases, and ideally user studies, would be needed to strengthen our claims of added value concerning our inverse projection method.

6.7 CONCLUSION

In this chapter, we have presented an inverse projection method that allows users to explicitly break the barrier of creating two-dimensional, fixed, surfaces embedded in the data space – a property exhibited by all inverse projection methods we are aware of. To do this, we split the information present in a high-dimensional dataset into the information which is captured by a projection technique and, separately, the information that such a technique cannot capture – the latter which we call a latent code. Next, we allow users to control this latent code in a simple but effective manner and thereby generate a dynamic inverse projection which can effectively 'sweep' the data space between the samples used by the direct projection it aims to invert.

Several experiments show that our proposal meets a number of key requirements which are important for inverse projections: Our method is – in absence of user control – at least as accurate, and practically as computationally scalable, as state-of-the-art inverse projection techniques. When user control is added, our method can generate a family of inverse projections that span the data space in a controlled manner – that is, by letting users tell where the inverse projection should adapt to so-called target data points, and where it should be purely driven by the underlying direct projection. This control is simple to perform as it involves changing two simple-to-understand linear parameters – an action radius and an action amount. Moreover, our inverse projection method has smoothness properties which are required from methods in this class for application deployment, up to a higher degree than other inverse projection methods we are aware of. Also, we showed that out method can cover a larger area of the data space – measured in terms of intrinsic dimensionality – than other inverse projection methods. Last but not least, we showed that our inverse projection method creates data samples (images in our studies) which look more natural than those created by existing methods in the same class.

The key following point to this work, we believe, is its direct application in practice, to validate its effectiveness. We envisage, in the short term, applications such as data augmentation, pseudolabeling, data interpolation, and morphing. Using our method in such contexts is, we believe, of significant added value – though, we acknowledge, such concrete studies are a subject of future work.

7 FAST COMPUTATION FOR DECISION MAPS AND CLASSIFIER MAPS

7.1 INTRODUCTION

As introduced in Chapters 2 and 4, decision maps and their associated extensions such as gradient maps (Espadoto et al., 2021a) or differential decision maps (Machado et al., 2024) are powerful tools for understanding the behavior of ML classification models and exploring high dimensional data. However, computing a decision map, even at quite small resolutions of hundreds of pixels squared, can take tens of seconds up to tens of minutes, depending on the decision map technique (Chapter 4). This precludes using such DBMs in scenarios where users aim to *interactively and iteratively* improve a classification model by *e.g.* changing its hyperparameters or performing data pseudo-labeling in active learning settings (Benato et al., 2018; Schulz et al., 2020; Benato et al., 2021). In this chapter, we propose a method that accelerates the computation of decision maps and their enhancements, thereby addressing our last research question **RQ4** introduced in Chapter 1[1].

To achieve the above, we propose FastDBM, a set of techniques that speeds up the computation of DBM images. FastDBM has a very low error rate – only a few tens of pixels, located on decision boundaries, are different from the ground-truth, slow, DBM computation. Our method can speed up any DBM that encodes classifier label and confidence without inner knowledge of how the model operates. Also, the method is simple to implement and has no hidden parameters.

Additionally, we show that the basic FastDBM can be easily extended to compute any real-value classifier map that depends only on sample positions, such as the gradient maps and differential decision maps mentioned above, by a simple modification. We show that this modification still keeps the attractive speed-up and low error rates proposed by the basic FastDBM technique. We also explore additional combination of techniques and qual-

1 This chapter is based on the papers "Computing fast and accurate decision boundary maps" (Grosu et al., 2024) and "Computing Fast and Accurate Maps for Explaining Classification Models" (Wang et al., 2025d).

ity metrics introduced in Chapter 4 to gauge the added value of our proposal.

The structure of this chapter is as follows. Section 7.2 introduces related work on decision maps and classifier maps and techniques used for computing these such as direct and inverse projections. Section 7.3 presents the core of our FastDBM technique which can be used to compute decision maps. ection 7.4 evaluates the three acceleration heuristics we proposed for FastDBM and outlines the winning heuristic: binary split. Section 7.5 presents additional evaluations focusing on the binary split heuristic. Section 7.6 presents our extension of FastDBM to handle general real-valued maps as well as examples of accelerating three such map types. Section 7.7 discusses the features of our method. Finally, Sec. 7.8 concludes the chapter.

7.2 RELATED WORK

Following our by now established notations, which we repeat here for easing the reading, let $D = \{\mathbf{x}_i\} \subset \mathbb{R}^n$ be a high-dimensional dataset with samples $\mathbf{x}_i$. A classification model $f : \mathbb{R}^n \rightarrow C$, trained and/or tested on D, maps samples from the data space to a categorical (label) domain C. Let $c : \mathbb{R}^n \rightarrow [0,1]$ denote the confidence of this classification. A *decision map* is a two-dimensional image I that aims to capture f's behavior by extrapolating it from D. Three elements are key to the construction of I, as follows:

Direct projection: Let $P : D \rightarrow \mathbb{R}^2$ be a projection, operation; and $P(D)$ be the mapping of D to a 2D scatterplot computed by P, that is, $P(D) = \{P(\mathbf{x}) | \mathbf{x} \in D\}$. P maps the input space of the model f to $\mathbb{R}^2$, next allowing the construction of the decision map image I.

Inverse projection: An inverse projection $P^{-1} : \mathbb{R}^2 \rightarrow \mathbb{R}^n$ inversely maps, or backprojects, any pixel $\mathbf{p} \in I$ to the data space location $P^{-1}(\mathbf{p})$, aiming to revert the effects of a given direct projection P.

Decision maps: These are images

$$F(\mathbf{p}) = f(P^{-1}(\mathbf{p})) \tag{7.1}$$

that depict, at every pixel $\mathbf{p}$, any property of interest f that is measured in the data space $\mathbb{R}^n$ at location $P^{-1}(\mathbf{p})$. In their simplest form, which we extensively discussed in the previous chapters, decision maps depict a classification model.

Besides depicting a classifier, maps can be used to visualize additional properties of f or P^{-1}. For instance, the gradient map G (Secs. 2.5.2.1 and 4.3.1) depicts the smoothness of the inverse projection P^{-1} computed at every pixel via Eqn. 4.5. Visualizing G over I shows areas where P^{-1} has high gradients, *i.e.*, where the extrapolation of the model f from the samples in D can be risky due to so-called compression of the high-dimensional space to the 2D projection space created by the direct projection P (Aupetit, 2007; Nonato and Aupetit, 2018). The distance to closest decision boundary d_B (Sec. 4.3.1, Eqn. 4.6) depicts at each pixel the distance to the closest decision boundary which allows one to find different areas in the data space where the trained model may be brittle (Rodrigues et al., 2018; Machado et al., 2024). Computing d_B is however quite expensive as it requires bisection-like search for the closest decision boundary (Rodrigues et al., 2018) or running adversarial example generation (Moosavi-Dezfooli et al., 2016). The distance to the closest training sample d_D, also called distance to data (Sec. 4.3.1, Eqn. 4.7) helps locating areas where we extrapolate the behavior of f far away from the training data D, *i.e.*, where the model's behavior can be less reliable, despite high confidence values (Machado et al., 2024). Any other characteristics of interest of f can be visualized via such maps. We now call these *classifier maps* as the mapped functions are in general real-valued ones (see Eqns. 4.5-4.7) as opposed to *decision maps* which map class values only (see Eqn. 7.1).

Scalability: All current decision map techniques are quite slow – on a typical commodity PC, computing a DBM for resolutions of 250^2 pixels reach about 10 seconds for all tested methods except DeepView; for the latter, the computation time can extend to several hours, depending on the choice of classifier (Chapter 4). As we shall see in Sec. 7.4, these costs increase quadratically with the DBM resolution – and higher resolutions are needed to create maps in which users can see subtle details such as the exact shape of decision boundaries close to groups of data samples or the variations due to compression in gradient maps (Aupetit, 2007; Nonato and Aupetit, 2018) or distance to boundary/training samples (Machado et al., 2024). This makes current DBM methods poorly suitable for visual analytics scenarios that require fast recomputation of decision and/or classifier maps upon re-training of the studied model.

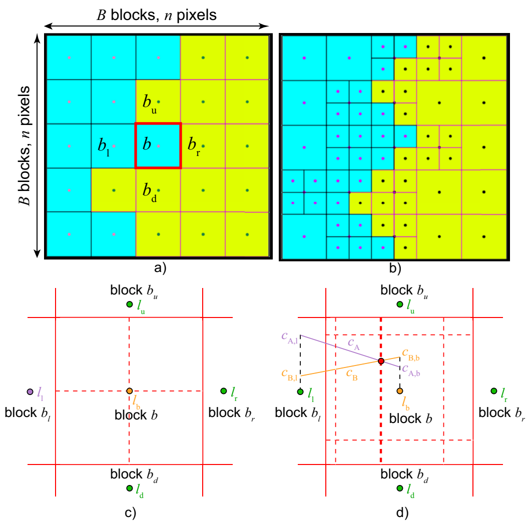


Figure 7.1: Illustration for binary and confidence-based splitting heuristics. Given the block-set in (a), binary split creates the refined block-set in (b). For the block in (c), binary split would create four equal-sized blocks along the thin dashed lines. In contrast, confidence-based splitting (d) examines the confidence values and splits the block in up to 9 smaller blocks along the thin dashed lines.

7.3 FAST DBM COMPUTATION

For an image I of $n \times n$ pixels, the complexity of current DBM methods is $O(n^2K)$, where K is the cost of a single $f(P^{-1}(\cdot))$ operation. Decreasing K is hard if we allow any generic inverse projections P^{-1} and classifier model f. Hence, to improve speed, we next aim to reduce the n^2 term.

A classification model f, in general, must fit its decision boundaries so that they (a) surround same-class training points, but (b) the boundaries are sufficiently *smooth* to allow for generalization without overfitting. Given (b), f, and thus a decision map that aims to accurately capture f, has in general relatively *few* compact decision zones (not necessarily one zone per class). We use this property to devise our acceleration as follows.

7.3.1 *Binary split*

We start by dividing the image I into B^2 blocks – each such block is a square of $\frac{n}{B} \times \frac{n}{B}$ pixels from I. For each block b, we evaluate the label $l_b = f(P^{-1}(\mathbf{p}))$ at its central pixel $\mathbf{p}$. Figure 7.1a shows this for a binary classifier (cyan and yellow are the two classes). Let l_u, l_d, l_l, l_r be the labels computed similarly for the up, down, left, and right neighbor blocks of b. Let N be the number of neighbors with labels different from l_b. If $N = 0$, then b is surrounded by same-label blocks, so, if we assume that a decision zone in the DBM is locally *thicker* than $\frac{n}{B}$ pixels, no decision boundary crosses it. Hence, we can assign l_b to all pixels in b. If $N > 0$, we split b into four equal smaller blocks (Fig. 7.1b shows the results of this splitting). We repeat the process, in a quadtree-like fashion, until we arrive at pixel-sized blocks or blocks do not need splitting anymore. During this, we note that (1) splitting larger blocks first helps to ensure a uniform refinement all over the image; and (2) splitting blocks having several neighbors with different labels is better than splitting blocks having a single such neighbor since the former cover more decision boundary fragments. We model this by keeping blocks to split in a priority queue sorted decreasingly on $d \cdot d \cdot \frac{N}{C}$ where d is the size of a block, N is its number of different-label neighbors, and C is its neighbor count (4 for blocks inside the DBM, 3 for blocks on the DBM boundary, and 2 for blocks on the DBM corners).

As Sec. 7.2 outlines, a DBM also often shows the *confidence* of the visualized model f at each map pixel. Per block, however, we have a single data sample $P^{-1}(\mathbf{p})$, computed at the block's center pixel $\mathbf{p}$. This is fine for class labels since these are *constant* over decision zones, thus also per block as per our splitting heuristic. In contrast, confidence varies *continuously* within a decision zone, hence can also vary within a block. We avoid computing additional confidence values apart from $c(\mathbf{p})$ by interpolating these values, computed at the blocks' centers $\mathbf{p}$, using nearest-neighbor, bilinear, and bicubic schemes.

7.3.2 *Confidence split*

The binary split is a simple bisection procedure to find the places in $\mathbb{R}^2$ where decision boundaries are, up to the pixel precision of I. We can potentially use the confidence values $c(\mathbf{p})$ to refine this process as follows. Take the block b shown in Fig. 7.1c. Binary split would divide b along the dashed lines in the image. Consider the confidence values c_A and c_B for the inferred classes purple, respectively orange, sampled at the centers of cells b_l and b, denoted next as $c_{A,l}$, $c_{B,l}$, $c_{A,b}$ and $c_{B,b}$ respectively (Fig. 7.1d). We next linearly interpolate these values to find the point where $c_A = c_B$ (red point, Fig. 7.1d). This is likely a good point to split cell b (along the thick dashed line, Fig. 7.1d) since, left to this point, class A has a higher confidence than class B (so the decision zone there should tell A) and, right to this point, class B has a higher confidence than class A (so the decision zone there should tell B). Note how this confidence maximization is precisely similar to how classification models internally decide on the class to output, albeit using more complex interpolation schemes than our linear one. We proceed in the same way for all class values with respect to all four boundaries of cell b. This yields a possible set of 2, 3, 4, 6, or 9 cells that split b as opposed to the fixed 4 cells done by binary split (see thin dashed red lines in Fig. 7.1d). Confidence is next interpolated as for the binary split method.

7.3.3 *Confidence sampling*

Our final acceleration heuristic uses the underlying idea that, if we can capture the confidence c at a *coarse* sampling resolution, then we can find decision zones (and boundaries) at pixel resolution by maximizing c over all inferred labels. This will reduce the costs implied by the block-splitting process. For this, given our initial B^2 blocks, we compute confidences $c(\mathbf{p})$ at block centers $\mathbf{p}$, for *all* inferred $|C|$ classes, and next interpolate these over I using nearest-neighbor, bilinear, or bicubic techniques – as described above, but now only over the initial blocks, which we do not further split. Next, for each pixel $\mathbf{p} \in I$, we compute which class yields the highest interpolated confidence and assign that class to $\mathbf{p}$.

7.4 EVALUATION OF ACCELERATION HEURISTICS

7.4.1 *Comparison of acceleration heuristics*

We now compare our three acceleration heuristics (binary split, confidence split, confidence sampling) against each other and with the ground truth. For this, we use two metrics:

Label errors: We ideally want to get the same labels for a Fast-DBM image I_{fast} and the ground-truth DBM image I. We evaluate this by the error

$$\epsilon_{label} = \frac{100}{n^2} \sum_{1 \leq x \leq n, 1 \leq y \leq n} \delta(I(x,y), I_{fast}(x,y)), \tag{7.2}$$

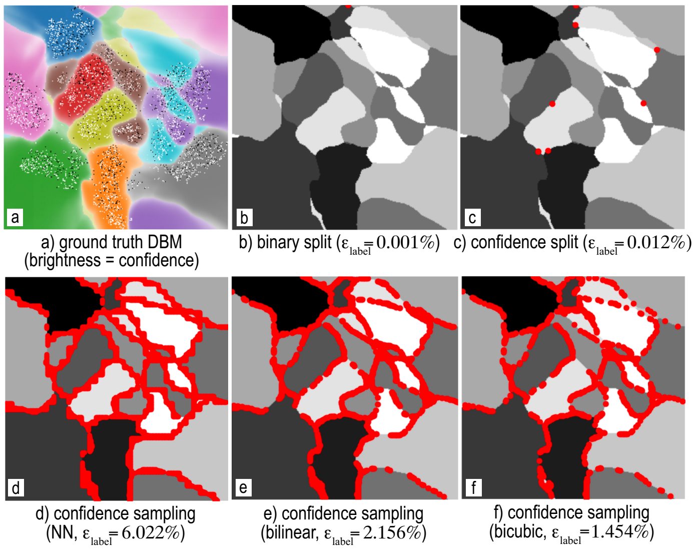


Figure 7.2: a) Ground-truth DBM with labels and confidence encoded into colors, respectively saturation, MNIST dataset. b-f) Class assignment errors for FastDBM method variants.

where $\delta(a,b)$ is 0 if $a = b$ and 1 otherwise. That is, ϵ_{label} measures the percent of the $n \times n$ FastDBM map image which is different from the ground truth.

Confidence errors: Our interpolated confidence c_{fast} should be as close as possible to the ground-truth one c. We evaluate this by the normalized MSE error

$$\epsilon_{conf} = \frac{\sum_{1 \le x \le n, 1 \le y \le n} (c(x,y) - c_{fast}(x,y))^2}{\sum_{1 \le x \le n, 1 \le y \le n} c(x,y)^2}. \qquad (7.3)$$

Figure 7.2 shows our results for the MNIST dataset (LeCun et al., 2010), classified with a simple deep learning network f (flatten layer, dense 10-unit layer and softmax activation, 20 training epochs, 3.5K training samples, 1.5K test samples); t-SNE and NNInv used for P and P^{-1}; DBM image size $n = 256$ pixels, $B = 8$ blocks. Image (a) shows the ground-truth DBM with labels and confidence color- respectively saturation-coded. Images (b-d) show the results of our binary split, confidence split, and confidence sampling heuristics, the latter using nearest neighbors, bilinear, and bicubic interpolation. Red points show pixels where ground-truth labels differ from our results. Our heuristics yield practically the same decision maps, with only a few different pixels. The binary split method is best – only 8 pixels of the 256^2

are different; the confidence sampling method is the worst; for the latter, errors appear *strictly* on the decision boundaries. This is likely since confidence varies slowly inside decision zones but rapidly close to boundaries (see Fig. 7.2a), so our interpolation has difficulties in the latter areas.

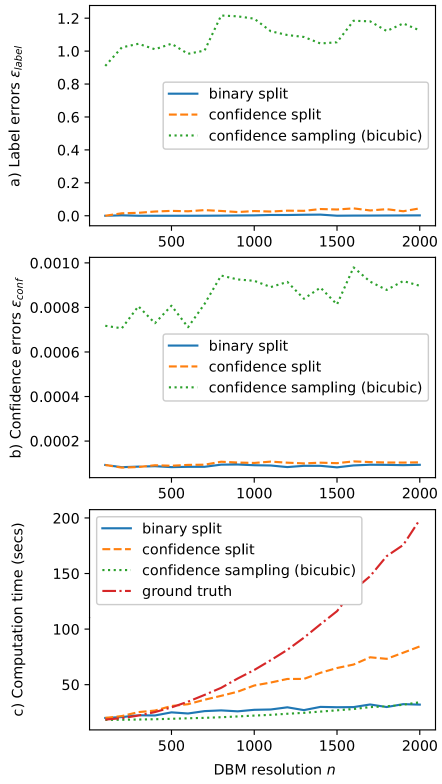


Figure 7.3: Label errors ϵ_{label} (a), confidence errors ϵ_{conf} (b), and computation time (c) for our three acceleration heuristics, MNIST dataset.

Figure 7.3 shows the errors ϵ_{label} and ϵ_{conf} and computing time for the above experiment for different image resolutions n (100 to 2000 pixels squared). For confidence sampling, we only use bicubic interpolation as this yields lower errors than nearest neighbor and bilinear (see Fig. 7.2). Error-wise, the binary split and confidence split methods are very similar and consistently lower than confidence sampling since the latter method uses a single *fixed* block resolution which, if too low, is unable to capture complex signal variations over the map image. Also, the binary split and confidence split errors are virtually constant with n, while confidence sampling errors show a slight increase with n. Speed-wise, the binary and confidence-sampling methods show near-linear behavior in n (with a very small slope) as opposed to the quadratic behavior of ground-truth DBM, with the confidence-split method in between the two. The binary and confidence-sampling methods are over one order of magnitude faster than ground-truth DBMs. The confidence split method's relative low speed can be explained by the fact that it can create up to 9 cells when splitting a single block as opposed to exactly four for the binary split (see Fig. 7.1 and related text). Note also that our maximal resolution $n = 2000$ exceeds *by far* all reported DBM results in the literature. From the above, we conclude that the binary split method is the clear winner when considering computational speed and accuracy factors. As such, we focus only on this method in our further evaluations.

7.4.2 *Parameter setting for binary split heuristic*

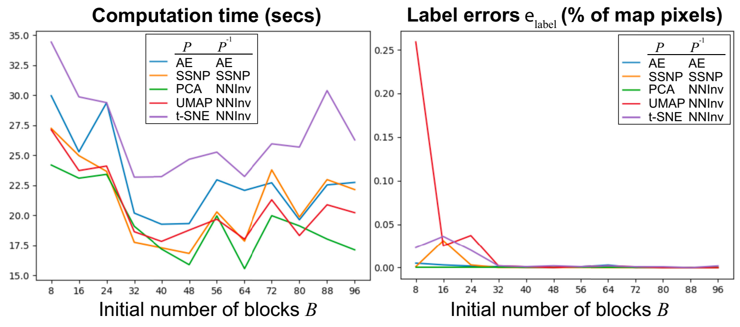


Figure 7.4: Speed (left) and label errors (right) of binary split method as function of initial block count B for decision maps constructed by various (P, P^{-1}) methods.

Binary split has one parameter – the initial block count B – so how to set its value? A high B will limit errors due to dense sampling of the image, but will be slow, since $f(P^{-1})$ must be evaluated on many blocks. A low B will be fast, but as Sec. 7.3 notes, decision map details under $\frac{n}{B}$ may be lost. To find a good initial value for B, we measure both speed and label errors ϵ_{label} for various B settings ranging from 8 to 96. To generalize our findings, we test several combinations of P and P^{-1} to compute our ground-truth decision maps, specifically autoencoders (AE, used for both P and P^{-1}); SSNP (used for both P and P^{-1}); and DBM (PCA, UMAP, and t-SNE used for P, NNInv used for P^{-1}). Figure 7.4 shows the speed and label errors as function of B for our maximally considered resolution $n = 2000$. We see that, label-error-wise, all B values above roughly 32 yield (very) low errors. Speed-wise, B values in the interval 32-64 offer best results, which confirms our earlier observations that too low or too high B will be slow. Also, we see that the overall speed trend as function of B does not strongly depend on the choice of (P, P^{-1}), up to a constant bias factor. Hence, we conclude that a block size $B = 32$ is a good preset for FastDBM.

7.4.3 *Implementation details*

Our FastDBM method is implemented in Python and runs fully on the CPU. The full source code, including datasets and experiments presented here, is publicly available (Wang et al., 2025c).

7.5 IN-DEPTH EVALUATION OF BINARY SPLIT ACCELERATION

We found that binary split works the best among the three proposed heuristics (Sec. 7.4.1). We now further evaluate the binary split heuristic using more classifiers, an additional quality metric, and using decision maps constructed with all inverse projection techniques that we are aware of.

7.5.1 *Using additional classifiers*

We evaluate the binary split method with additional combinations of datasets, classifiers, and (P, P^{-1}) combinations used to compute the ground-truth DBM. The datasets include FashionMNIST (Xiao et al., 2017), HAR (Anguita et al., 2012), and Iris (Fisher, 1988). Classifiers included logistic regression (LR), support vector machines (SVM), k-nearest neighbors (kNN), decision trees (DT), random forests (RF), and the neural network (NN) we used earlier for MNIST. It is important to note that the accuracy of the trained classifiers is of no concern in this experiment. If FastDBM approximates well the ground-truth DBM, FastDBM can be next used next to assess how well (or poorly) the classifiers behave.

Ground-truth DBMs were created by the DBM (t-SNE, NNInv) and DBM (UMAP, NNInv) combinations at resolution $n = 400$ pixels squared. Figure 7.5 shows the ground-truth DBMs; those created by our binary split method; 2D projections of training samples in green and the label difference encoded by red dots as

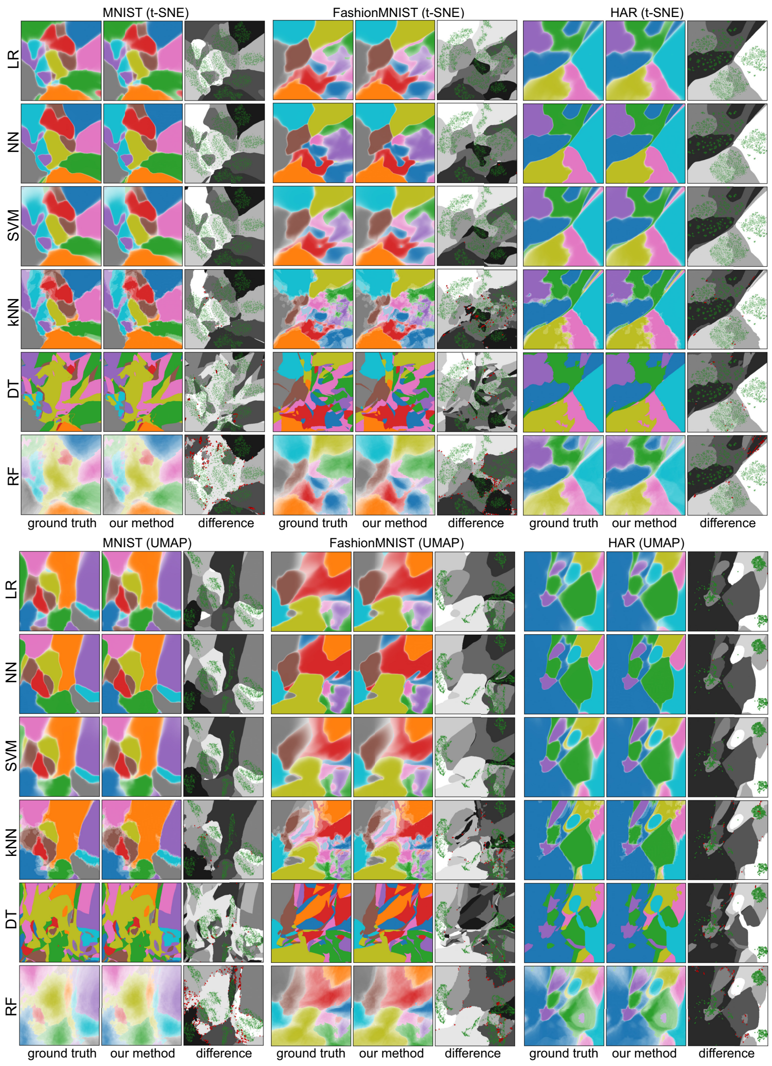


Figure 7.5: Comparison between ground-truth DBM and our binary split method for three datasets, six classifiers, t-SNE and UMAP projections.

in Fig. 7.2, for the MNIST, FashionMNIST, and HAR datasets. We see that our method yields visually almost identical label results as the ground-truth – there are only few red points in the 'difference' images. This occurs consistently for quite different DBMs, *e.g.*, the smooth decision-zone DBMs created for LR, NN, SVM, and KNN, but also the far noisier DBM created for DT, and the overall low-confidence DBM created for RF. Additional results for all other tested combinations, present in the supplementary material, confirm this observation.

7.5.2 *Consistency evaluation*

To further confirm the visual similarity between the ground-truth decision maps and the FastDBM versions shown in Fig. 7.5, we next compare the *map consistency* metric $Cons_p$ (Eqn. 4.4) computed for both cases. To recap, $Cons_p$ computes the fraction of 'consistent' pixels in a decision map image, *i.e.*, pixels whose corresponding data points (obtained by P^{-1}) have the same class label after a round-trip of projection and inverse projection (see Chapter 4). If our acceleration technique works well, then the consistency $Cons_p^{fast}$ of the images it produces should be very close to the consistency $Cons_p$ of the ground-truth decision map images. Table 12 shows, for several datasets and classifiers, the values of $Cons_p^{fast}$ computed by binary split for DBM (UMAP+NNInv) and SDBM compared to the ground-truth $Cons_p$. We see that, although both $Cons_p^{fast}$ and $Cons_p$ are less than the ideal value of one (which would imply that P^{-1} is an exact inverse of P), their values are very close to each other. That is, the quality of the images produced by FastDBM is very close to the ground-truth images.

7.5.3 *Accelerating additional direct and inverse projection techniques for creating decision maps*

So far, we computed our (accelerated) decision maps using NNInv and SSNP for the inverse projection P^{-1} since, as mentioned in Sec. 7.2, earlier work showed that NNInv and SSNP are fast and accurate for this task. Yet, other inverse projection techniques do exist, most notably iLAMP (dos Santos Amorim et al., 2012) and the inverse projection using radial basis functions (RBF) (Amorim et al., 2015). Earlier work has shown that both these techniques are slower than NNInv (Espadoto et al., 2019c). However, it is interesting to see how these techniques fare given our acceleration. Separately, iLAMP and RBF have a quite different behavior from the already-tested NNInv and SSNP. Hence, if our acceleration technique can create *accurate* approximations of decision maps using these inverse projections, this increases the claims of generality of our proposal.

Figure 7.6 shows, for the MNIST dataset, the decision maps computed by four ground-truth technique pairs (t-SNE and iL-

Table 12: $Cons_p$ of FastDBM *vs* two ground truth methods for three datasets and six classifiers. $\Delta\ Cons_p = Cons_p^{fast} - Cons_p$. See Sec. 7.4.

Model	Metric	Fashion MNIST	HAR	MNIST
DT	$Cons_p$	0.4033	0.3704	0.4718
	$Cons_p^{fast}$	0.4041	0.3659	0.4651
	$\Delta\ Cons_p$	0.0009	-0.0045	-0.0067
KNN	$Cons_p$	0.2152	0.0816	0.1414
	$Cons_p^{fast}$	0.2159	0.0735	0.1364
	$\Delta\ Cons_p$	0.0007	-0.0081	-0.0049
LR	$Cons_p$	0.2787	0.1776	0.2759
	$Cons_p^{fast}$	0.2792	0.1745	0.2727
	$\Delta\ Cons_p$	0.0005	-0.0031	-0.0032
NN	$Cons_p$	0.2959	0.1740	0.2810
	$Cons_p^{fast}$	0.2969	0.1712	0.2785
	$\Delta\ Cons_p$	0.0010	-0.0028	-0.0025
RF	$Cons_p$	0.2480	0.2477	0.3009
	$Cons_p^{fast}$	0.2464	0.2400	0.2955
	$\Delta\ Cons_p$	-0.0016	-0.0077	-0.0053
SVM	$Cons_p$	0.2153	0.1591	0.2470
	$Cons_p^{fast}$	0.2149	0.1551	0.2458
	$\Delta\ Cons_p$	-0.0004	-0.0039	-0.0013

(a) DBM (UMAP+NNInv)

Model	Metric	Fashion MNIST	HAR	MNIST
DT	$Cons_p$	0.2685	0.1515	0.3594
	$Cons_p^{fast}$	0.2678	0.1510	0.3616
	$\Delta\ Cons_p$	-0.0007	-0.0005	0.0022
KNN	$Cons_p$	0.1145	0.0778	0.0950
	$Cons_p^{fast}$	0.1149	0.0767	0.0956
	$\Delta\ Cons_p$	0.0004	-0.0011	0.0007
LR	$Cons_p$	0.0589	0.0432	0.0909
	$Cons_p^{fast}$	0.0591	0.0431	0.0910
	$\Delta\ Cons_p$	0.0003	-0.0001	0.0002
NN	$Cons_p$	0.0725	0.0320	0.0872
	$Cons_p^{fast}$	0.0726	0.0321	0.0881
	$\Delta\ Cons_p$	0.0001	0.0001	0.0009
RF	$Cons_p$	0.1455	0.0655	0.2174
	$Cons_p^{fast}$	0.1456	0.0658	0.2178
	$\Delta\ Cons_p$	0.0001	0.0002	0.0004
SVM	$Cons_p$	0.0597	0.0364	0.0893
	$Cons_p^{fast}$	0.0603	0.0364	0.0898
	$\Delta\ Cons_p$	0.0006	-0.0000	0.0006

(b) SDBM

AMP, UMAP and iLAMP, t-SNE and RBF, and UMAP and RBF) and their counterparts produced by our binary split acceleration. The ground-truth maps are noisier than those we computed so far using NNInv and SSNP for P^{-1}, in line with earlier findings (Espadoto et al., 2019c; Rodrigues et al., 2018). Our binary split method captures these ground truth images quite well – the label difference images show only a few pixels where our results differ from the ground truth, much like in Fig. 7.5. Our method speeds up the computation of most maps – see timing figures in the lower-left corners of the images. Speed up is overall much smaller except for the t-SNE and iLAMP combination. This is due to the high irregularity of the decision boundaries in this case, which generates a very large amount of cell splits – see the corresponding 'binary split process' images in Fig. 7.6.

Concluding, we claim that our binary split heuristic creates accurate decision maps for all existing inverse projections we are aware of; and, for most cases except very noisy decision maps

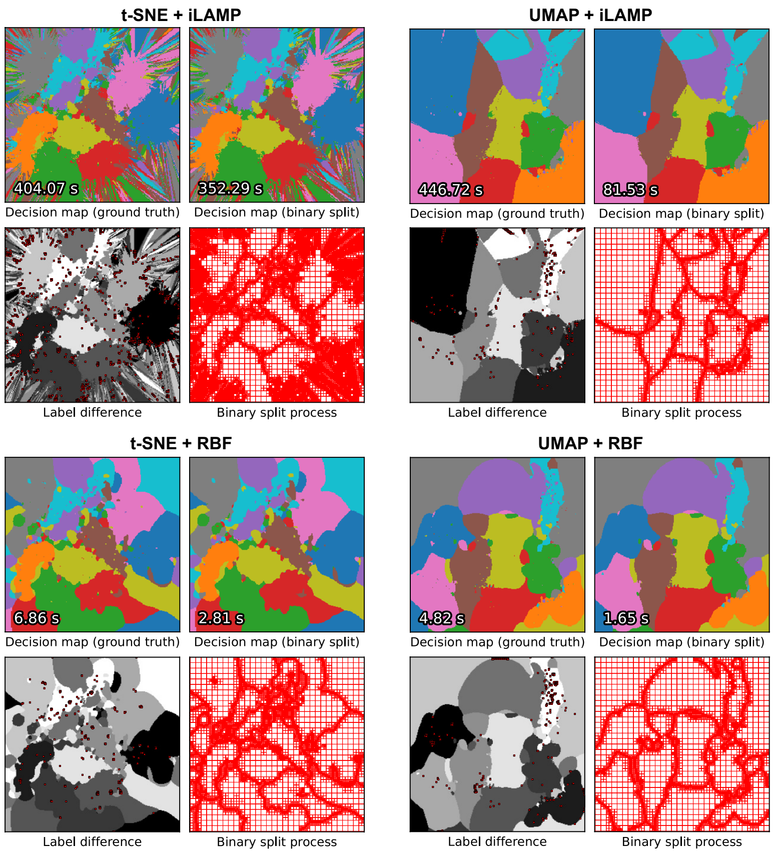


Figure 7.6: Maps computed using the iLAMP and RBF inverse projections in combination with the t-SNE and UMAP direct projections for the MNIST dataset. Resolution: 256^2. See Sec. 7.5.3.

(which are likely not useful in practice), it also accelerates the map computation by several factors.

7.6 ACCELERATING THE COMPUTATION OF CONTINUOUS MAPS

So far, our binary split method only works for maps with *label* values like classification functions $f : \mathbb{R}^n \to C$. However, several maps used in classifier visualization have continuous values, *i.e.*, are of the form $f : \mathbb{R}^n \to \mathbb{R}$. Examples are the gradient maps G (Eqn. 4.5), distance-to-closest-training sample d_D (Eqn. 4.7), and distance-to-decision boundary d_B (Eqn. 4.6). In general, one cannot reduce such continuous maps to the computation and comparison of purely categorical (label) values. Yet, we would like to accelerate their computation.

To do this, we generalize the binary split idea by replacing the label comparison (see Sec. 7.3) with a threshold comparison. For a dataset D, we compute this threshold globally as

$$T = \alpha \cdot \tau \cdot \left(\max_{\mathbf{p} \in \mathcal{B}} f(P^{-1}(\mathbf{p})) - \min_{\mathbf{p} \in \mathcal{B}} f(P^{-1}(\mathbf{p})) \right). \qquad (7.4)$$

Simply put, T is a fraction of the range of the function f over the map. $\mathcal{B}$ denotes the set of center pixels of the initial B^2 blocks. $\tau = e^{-\frac{B \cdot d}{n}}$ is a decreasing function of the block size d, *i.e.*, smaller blocks will use a higher threshold. The intuition behind this is that smaller blocks already capture f at a higher resolution so we make them harder to further split to reduce over-refinement. Conversely, if f exhibits even a small variation over large blocks, this is a reason to split these to capture further details. The last parameter α is a scaling factor. When the difference between the maximum and minimum values of the four neighbors of a block including the block itself exceeds T, we split the block.

Distance-to-boundary maps which compute d_B (Eqn. 4.6) are a first example of such continuous maps. Figure 7.7 shows the results of accelerating the computation of d_B at resolution 512^2 pixels for the three datasets as in Fig. 7.5, and for all four inverse projection techniques we are aware of (NNInv, SSNP, RBF, and iLAMP) with UMAP as the direct projection. For all inverse projection techniques, we show the ground-truth d_B map, the map computed by our binary split acceleration, and the blocks created by the binary split process. Ground-truth maps are visually almost identical from those computed by our binary split heuristic. Speed-wise, our binary split heuristic is up to roughly ten times faster than computing the ground truth, see the figures in the bottom-left corners in the respective images in Fig. 7.7. This speed-up is in line with what we visually see as amounts of cells being split in the rightmost columns in Fig. 7.7 – the largest cells in those columns indicate the original block sizes, that is, using $B = 32$ initial cells for the acceleration heuristic, as explained earlier in Sec. 7.4.

Gradient maps G (Eqn. 4.5) are a second example of continuous maps we can accelerate. Figure 7.8 shows gradient maps computed by ground truth and our generalized binary split acceleration for the same dataset-projection-inverse projection combinations as shown in Fig. 7.7. Our accelerated maps are very similar to the ground truth, while the computation time is up to

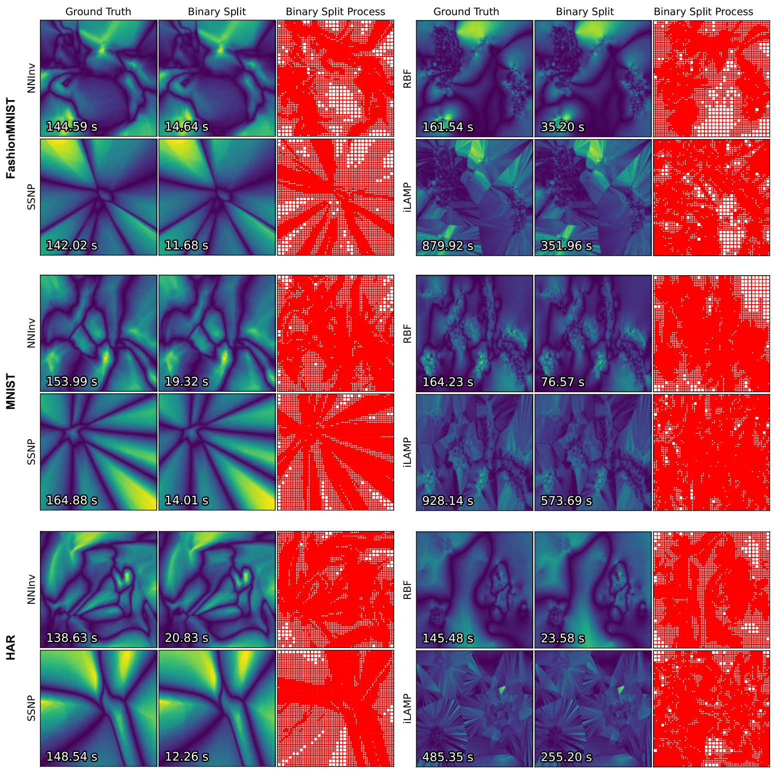


Figure 7.7: Distance-to-decision-boundary maps d_B for the generalized binary split method, three datasets (resolution: 512^2). See Sec. 7.6.

10 times lower – see timing figures in the lower-left corners of the images.

Figure 7.9 shows a final example of continuous maps, namely distance-to-closest sample maps d_D (Eqn. 4.7). As for d_B and G, our acceleration yields practically the same images with high speed-ups *vs* ground truth. Given these results and the fact that our binary split works entirely agnostically on the nature of the function f, we claim that similar results can be obtained for *any* function $f : \mathbb{R}^n \to \mathbb{R}$ that produces a real value from an inversely-projected 2D pixel. The only implicit assumption our acceleration method makes for f is that it should be locally smooth, *i.e.*, not have unbounded variations on a small spatial extent, so that we can use the threshold computed by Eqn. 7.4 to locate map areas needing subdivision.

To find a suitable choice for α, we executed a grid search over the range $[0, 0.6]$ by evaluating both computation time and MSE

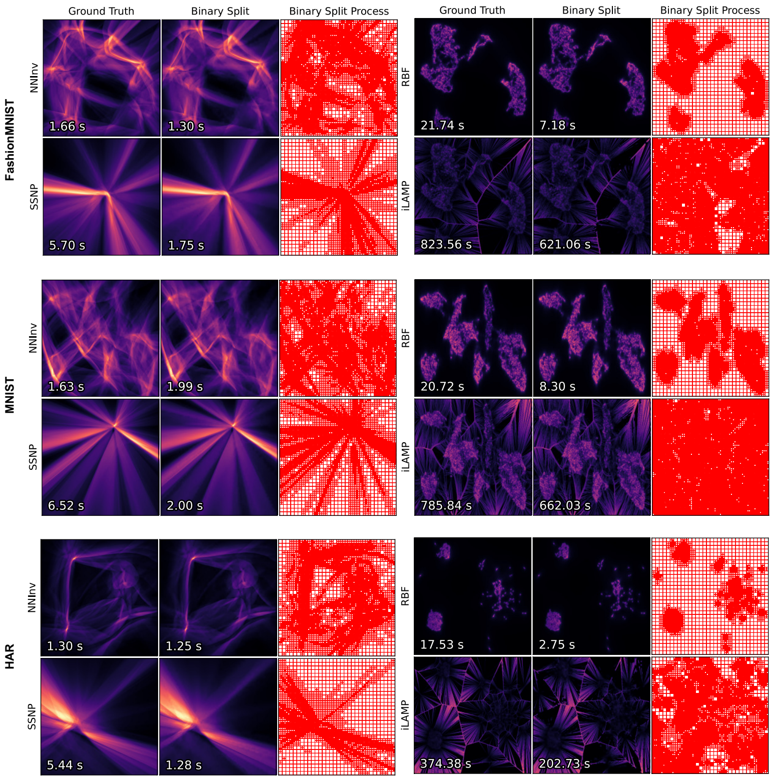


Figure 7.8: Gradient maps G for the generalized binary split, three datasets (resolution: 512^2). See Sec. 7.6.

error of our resulting map *vs* the ground-truth maps G, d_D, and d_B for the MNIST dataset. The MSE error is computed analogously to ϵ_{conf} (Eqn. 7.3). Figure 7.10 shows the search results. For larger α values, we get higher errors since the split threshold T is larger, so fewer block refinements (splits) occur; for smaller α values, the error decreases but the computational time increases, since there are more splits. We found that $\alpha \in [0.1, 0.2]$ is a good choice balancing between speed and accuracy. Specifically, we set $\alpha = 0.125$ for G, $\alpha = 0.1$ for d_D, and $\alpha = 0.15$ for d_B consistently in all our following experiments.

Figure 7.11 shows the computation times and normalized MSE errors of our generalized binary split method for different image sizes and for all the three maps d_D, G, and d_B. The results are quite similar with the binary split used for label-based maps (Fig. 7.3c): Our method is roughly linear in the map resolution (as compared to quadratic in resolution for the brute-force ground truth computation), while errors decrease inversely quadratically

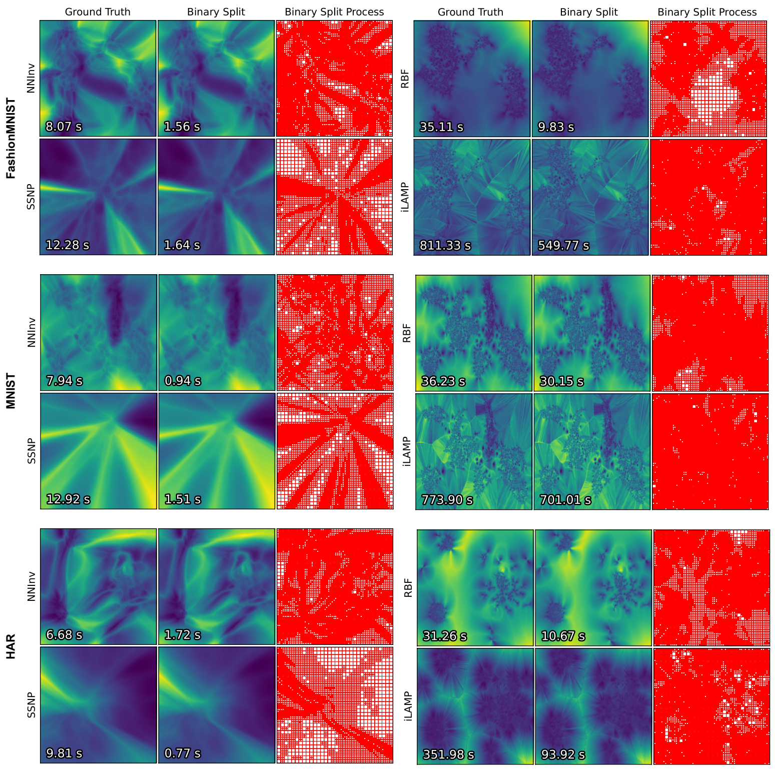


Figure 7.9: Distance to nearest sample maps d_D for the generalized binary split method, three datasets (resolution: 512^2). See Sec. 7.6.

with resolution. All in all, the above results show us that the generalized binary split is a computationally effective and accurate way to accelerate the construction of continuous maps.

7.7 DISCUSSION

We next discuss several aspects of our method.

Genericity: Our acceleration method based on the binary split can accommodate the construction of classifier maps for *any* function $f : \mathbb{R}^n \to \mathbb{R}$. This covers, but is not restricted to, the actual classification label F, gradient maps G, distance-to-boundary maps d_B, and distance-to-closest-training sample maps d_D. We can accelerate the computation of all such functions, while preserving their accuracy, in a black-box manner, *i.e.*, without knowing anything additional about what the respective functions cap-

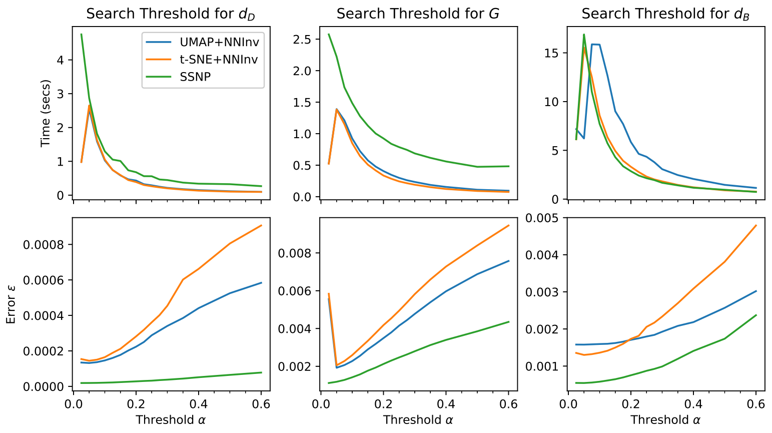


Figure 7.10: Generalized binary split method: Search for the best threshold α for the MNIST dataset. Columns show different classifier maps (d_D, G, d_B). Top row shows computation time. Bottom row shows computation error.

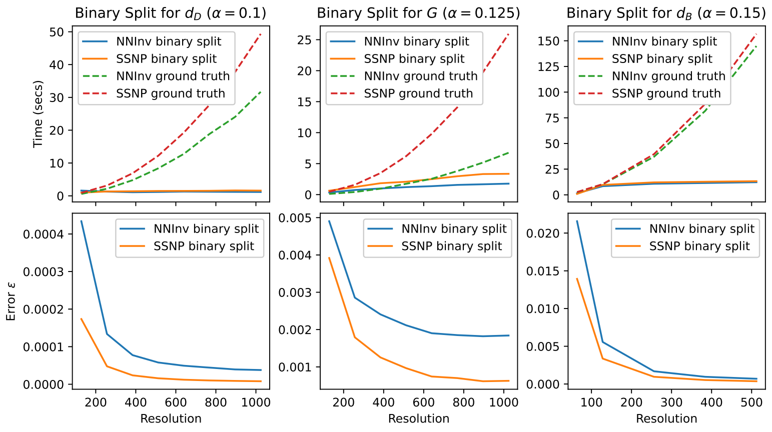


Figure 7.11: Performance of the generalized binary split method with varying grid resolutions for NNInv and SSNP on the MNIST dataset. Columns show different classifier maps (d_D, G, d_B) with optimized thresholds α. Top row shows computation time for binary split and ground truth methods. Bottom row shows the error as the image resolution increases.

ture or how they are computed. The only constraint we have is that such functions are smooth.

Performance: Our experiments showed a consistent performance gain of our binary split heuristic of roughly 5 times with respect to the brute-force computation of the decision maps at each pixel. This figure varies mainly as a function of the *smoothness* of the inverse projection method P^{-1} being used: NNInv, SSNP, and RBF are relatively speaking smooth mappings so they require less block splits to capture their variation, thus, yield higher speed-ups. In contrast, iLAMP is far less smooth so it requires many more block splits, thereby reducing our speed-ups to roughly 1 to 2 times. Practically, for the inverse projection methods NNInv and SSNP, which were earlier found to be the most reliable for computing decision maps, our acceleration yields roughly linear times *vs* the map resolution as compared to quadratic time for the brute-force computation.

Quality: All our experiments showed that we can obtain the maps virtually identical visually to the ground-truth ones no matter which type of function we visualize. Moreover, the quality, measured in terms of normalized MSE *vs* ground truth, only increases with the image resolution. This means that our accelerated maps can be safely substituted for the brute-force-computed ones for all practical reasons.

Ease of use: Our acceleration method is essentially parameter-free – the only two parameters B (initial block count, see Sec. 7.3.1) and α (controlling the split threshold for continuous mappings, see Eqn. 7.4) have well-tested presets which are independent on the classification model, choice of direct and inverse projections P and P^{-1}, and type of map being computed.

Limitations: The key assumption behind our acceleration is that the *combination* of function f we aim to visualize with the inverse projection function P^{-1} is smooth and has bounded variation over $\mathbb{R}^2$. While this is true of all f and P^{-1} we know of, and is also in line with the well-known smoothness assumption underlying most machine learning methods for f, that cases could exist where smoothness would not hold. In such cases, it is possible that our acceleration does not yield worthwhile speed-ups and/or the accelerated maps have visible errors as compared to the ground truth ones. Separately, we believe that the confidence split method (Sec. 7.3.2) has not yet reached its true potential. Better interpolation schemes than our current linear one should be able to decrease the number of generated cells and thereby achieve higher performance at the same quality level as compared to the so far currently best-ranked binary split. Another open challenge lies in scaling decision map visualizations to a large number of classes. In that case, encoding class values in categorical colors will not work well. This limitation is broadly shared by many visualizations that use categorical color maps to encode class values. Potential solutions can group class values hierarchically to reduce the needed color count and offer detail-on-demand interactively. Note that this scalability problem only affects decision maps (which depict class value) and not the classifier maps (which depict real-valued quantities).

7.8 conclusion

We have presented FastDBM, a technique for accelerating the computation of maps that describe the working of general-purpose classification models. Our technique is agnostic of the exact type of maps being computed as shown by its application to create maps of classification label, classification confidence, distance-to-classification-boundary, distance-to-closest-training sample, and gradient maps. We show that our technique can be applied to not only to decision maps but also real-valued maps; and also show high speed-ups and accuracy for more combinations of direct and inverse projection methods used to compute the maps. Practically, we show that our method can compute classifier maps that are visually almost identical to ground-truth ones with a speed-up of one order of magnitude on average. This allows the further deployment of such visualizations in interactive visual analytics workflows for classifier engineering. Our method depends on just two free parameters for which we provide good preset values. Our method can accelerate any current classifier map computation technique, and can be applied to any trained classifier model, as it only requires access to the inverse projection function this technique uses, respectively to the black-box execution of the trained model.

Future work aims to explore our acceleration technique to compute additional classifier maps. Also, we consider speeding up our method by more advanced sampling and interpolation schemes, GPU execution, and evaluating it on novel direct and inverse projection methods which arrive in the infovis arena. In parallel, measuring the added value of computing near-real-time classifier maps for classifier engineering, *e.g.*, in the context of visual active learning, is a key goal we aim at.

8 CONCLUSIONS

We close this thesis by revisiting our contributions and discussing future research directions.

In Chapter 1, we introduced our key research question:

How can we improve decision map methods to understand and control which parts of the data space they sample, leading to desirable quality values, with high computational performance?

This question was next refined into four more specific research questions that address **quality**, **coverage**, **control** and **interactivity** of decision maps, respectively:

RQ1: How can we define such quality metrics and how do current decision map methods fare with respect to each other from the perspective of such metrics?

RQ2: How do decision map methods differ in their sampling of the data space? Does this sampling depend on the data dimensionality or depicted classification model?

RQ3: How can we enable users to control the parts of the data space depicted by a decision map in simple, interactive, ways?

RQ4: Can we significantly improve this computation time for creating decision maps for any classification model and for any type and dimensionality of datasets?

Next, in Chapter 2, we presented related work related to dimensionality reduction (direct and inverse projections) and decision maps. We highlighted how ML methods, in particular deep learning, can assist (inverse) projection and decision map computation; and also how projections and decision maps are key instruments for the exploration of ML models and related datasets. Apart from elaborating on the natural synergy between DL and MR in Chapter 2, our main thesis contributions come in the next chapters, as follows.

8.1 DECISION MAPS IN PRACTICE

In Chapter 3, we presented an application of decision maps for understanding a mineral genesis classification problem in practical setting. Specifically, we created a decision map for a classifier trained on pyrite trace elements data to classify its formation environment. We demonstrated two use cases of the decision map: (1) providing extra evidence for a mineral deposit whose genetic environment is in debate; (2) visually identifying the most important trace elements for the classification.

Our work contributes to both the information visualization community and the mineralogy community by showing how decision maps, extended by additional features, can be efficient and effective instruments to interpret classification models and arrive at a better understanding of the involved data dimensions. While this work showed the added value of decision maps in research where the target users are not ML experts, it also highlighted the need for a comprehensive evaluation of existing decision maps, and the need for several enhancements of the by-then existing decision map techniques so as to make them more flexible and more informative. At a high level, our work here did not directly answer **RQ1 – RQ4** but demonstrated that, since decision maps are useful instruments in practice, answering these questions is next important to further increase the efficiency and effectiveness of decision maps.

8.2 QUALITY OF DECISION MAPS

In Chapter 4, we conducted a series of comprehensive assessments for the three state-of-the-art decision maps methods, namely DBM, SDBM, and DeepView, combined with several common classifiers, on a range of both synthetic and real-world datasets, which addresses **RQ1 (quality of decision maps)**. In this work, we proposed six global metrics and four local metrics to measure the global and local quality of decision maps, respectively.

Our results showed that none of the evaluated decision map techniques consistently outperforms the others in all measured aspects. Separately, our analysis exposed several previously unknown properties and limitations of decision map techniques. In particular we saw that all the studied decision map methods have inherent limitations in various quality aspects; and that these limitations can fluctuate significantly depending on the dataset and/or classifier being explored. To support practitioners, we also propose a workflow for selecting the most appropriate decision map technique for given datasets, classifiers, and requirements of the application at hand.

8.3 COVERAGE OF DECISION MAPS

One of our most surprising discoveries in Chapter 4 is that all the three examined decision map methods produce surface-like structures on a synthetic 3D dataset. At the time of the respective work, this indicated a potential limitation in the *coverage* of the data space by inverse projections and decision maps. In Chapter 5, we elaborated to elucidate this potential limitation. For this, we further investigated the coverage of inverse projections and decision maps in more scenarios, *i.e.*, additional datasets of different dimensionalities, more classifiers for creating decision maps, and more direct and inverse projection method combinations to create the maps, thereby addressing **RQ2 (coverage of decision maps)**. We studied the said methods by using both visual assessment and quantitative assessment based on the intrinsic dimensionality of the backprojected surfaces.

Our results demonstrate that, despite differences in the behavior of various inverse projection methods, all the studied techniques essentially capture only two-dimensional structures embedded within the data space, regardless of the dimensionality of the data space or the classifier used. This work – which we consider to be the first key result of this thesis – highlights the fundamental limitations of all studied decision map techniques, particularly in terms of the extent to which they represent a classifier's behavior and the specific regions and ways in which they capture this behavior. Understanding these limitations is crucial both for selecting an appropriate technique to construct decision maps and for accurately interpreting the resulting maps in practice. Additionally, we also found the dimensionality of the projection space q has an impact on the intrinsic dimensionality of inverse projections.

8.4 CONTROLLABLE INVERSE PROJECTIONS

In Chapter 6, we proposed LCIP, to our knowledge the first controllable inverse projection method – which we consider to be the second key result of this thesis. Our method overcomes the surface dimensionality of backprojections studied in Chapter 5 and thereby addresses **RQ3 (control of decision maps)**. To achieve this control, we first found a latent space which is disentangled from the 2D projection space, and then designed a control mechanism to manipulate this latent space. The inverse projection, computed via deep learning, is then updated based on the manipulation of the latent space, as our proposed inverse projection takes both the latent space and the 2D projection space as input. Additionally, we managed to found a reasonable initial status for the latent space and thereby the inverse projection, named as 'LCIP (fixed)' in Chapter 6.

Our evaluations showed that, without conducting the control, LCIP (fixed) are already comparable to existing inverse projection methods. When adding control, LCIP can hit higher intrinsic dimensionalities, and more importantly, can generate more diverse and meaningful inverse projections that no existing inverse projection methods can achieve. We also demonstrated the application of LCIP in style transfer application.

8.5 FAST DECISION MAP COMPUTATION

In Chapter 7, we presented FastDBM, a set of techniques for accelerating the computation of classifier maps (decision maps and other maps related to inverse projections) thereby addressing **RQ4 (interactivity of decision maps)**. Our core idea is based on an initially sparse sampling of the image space, followed by local recursive refinement of areas marked as containing rapid changes of the depicted signal.

Our evaluations showed that FastDBM can compute classifier maps that are visually almost identical to ground-truth ones with a speed-up of over one order of magnitude. Importantly, our method can accelerate any current classifier map computation technique, *e.g.* decision boundary maps, gradient maps, or distance to data maps; and can be applied to any trained classifier model, as it only requires access to the inverse projection function this technique uses, respectively to the black-box execution of the trained model (or property to be evaluated on that model). Importantly, FastDBM works with both discrete maps (*e.g.* decision boundary maps) and continuous maps (all other aforementioned examples) in similar ways and with similar computational effort. FastDBM is, we believe, an important step towards the application of classifier maps in interactive visual analytics scenarios where users need to rapidly recompute and examine such maps.

8.6 DIRECTIONS FOR FUTURE WORK

Based on our research, we foresee several new directions for future work.

Enhancements of LCIP: In Chapter 6, we proposed LCIP, a controllable deep-learning-based inverse projection method. All the enhancements in Section 2.3.4.2, such as gradient maps or distance to boundary maps, can benefit from the extra freedom that our user control offers. More interestingly, additional enhancements can be particularly designed for LCIP particularly, based on its dynamic control. For example, since we know the inverse projection of a pixel image is a surface-like structure, we can visualize the distance of each data sample to this surface. This enhancement would (1) provide clues to users about the selections of target points before the interaction; and (2) intuitively show how far the inversely projected 'surface' moves when users conduct the interactions. Additionally, combining LCIP with Fast-DBM would significantly increase the interactivity rate which is key to LCIP.

Applications of LCIP: In Chapter 6, we have only demonstrated one application of LCIP – style transfer. We envision that LCIP can be applied to a wide range of applications in the context of VIS4ML, such as data augmentation, building controllable decision maps, and active learning. Future work could explore these applications in more details. One of the obvious low-hanging fruits is to apply LCIP to the data augmentation task. Rodrigues (2020) have shown that the original DBM method can be used to generate new samples for a training set and therefore improve the performance of a classifier – an approach further verified in a different application context by Benato et al. (2024). As our work showed that all previous inverse projection methods can only cover a surface-like structure in the data space, the samples generated by these methods are limited. In contrast, we believe that LCIP can generate more diverse samples for the training set, and therefore improve the performance of classifier training more effectively.

More advanced architectures for neural network (inverse) projection methods: So far, all the existing deep learning-based projection and inverse projection methods, including our proposed LCIP, use only simple fully connected neural networks. Future work could explore more advanced architectures, such as adding attention mechanisms to enhance the ability to focus on critical features for learning (inverse) projections, or leveraging diffusion models for more robust inverse projection processes. We foresee that using such advanced techniques can open new perspectives for decision maps and make them even more suitable and effective for assisting machine learning tasks.

Handling modern AI models: Our work showed that decision maps can be computed to bring useful insights for classification models. However, much of the current AI research moves into the direction of developing and deploying regression models such as the Large Language Models (LLMs) or image transformers well known via ChatGPT or DALL-E, to mention just a few. As such models start claiming an increasingly large share of the interest of both developers and end users of AI, a key question is whether decision maps can be adapted to convey insights in their working, much as we showed they can do for classification models. While we did not attack this question in this thesis, we believe that our contributions form a starting basis for this exploration: Our research questions **RQ1 – RQ4**, and our methodology to answer them, are directly applicable to the study of future decision-map-like methods designed for regressors. Additionally, the results we obtained for decision maps, most notably the limited coverage of control-free methods; the possibility for adding interactivity by capturing information lost by projections; and the possibility of accelerating map construction by hierarchical division of the image space, can be direct starting points for future researchers aiming to design regressor maps.

BIBLIOGRAPHY

G. Alicioglu and B. Sun. A survey of visual analytics for Explainable Artificial Intelligence methods. *Computers & Graphics*, 102 (C):502–520, 2022.

B. Alsallakh, A. Jourabloo, M. Ye, X. Liu, and L. Ren. Do convolutional neural networks learn class hierarchy? *Ieee tvcg*, 24(1): 152–162, 2018.

E. Amorim, E. Vital Brazil, J. Mena-Chalco, L. Velho, L. G. Nonato, F. Samavati, and M. Costa Sousa. Facing the high-dimensions: Inverse projection with radial basis functions. *Computers & Graphics*, 48:35–47, 2015.

W. Amorim, A. Falcão, J. Papa, and M. Carvalho. Improving semi-supervised learning through optimum connectivity. *Pattern Recognition*, 60(C):72–85, 2016.

N. Andrienko, G. Andrienko, G. Fuchs, A. Slingsby, C. Turkay, and S. Wrobel. *Visual analytics for data scientists*. Springer, 2020.

D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *Proc. Intl. Workshop on ambient assisted living*, pages 216–223. Springer, 2012.

A. Ansuini, A. Laio, J. H. Macke, and D. Zoccolan. Intrinsic dimension of data representations in deep neural networks. In *Proc. NeurIPS*, volume 32, 2019.

G. Appleby, M. Espadoto, R. Chen, S. Goree, A. Telea, E. W. Anderson, and R. Chang. HyperNP: Interactive Visual Exploration of Multidimensional Projection Hyperparameters. *arXiv:2106.13777 [cs]*, 2021.

S. Ö. Arik and T. Pfister. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 6679–6687, 2021.

M. Aumüller and M. Ceccarello. The Role of Local Intrinsic Dimensionality in Benchmarking Nearest Neighbor Search, 2019. URL http://arxiv.org/abs/1907.07387. arXiv:1907.07387 [cs].

M. Aupetit. Visualizing distortions and recovering topology in continuous projection techniques. *Neurocomputing*, 10(7–9): 1304–1330, 2007.

C. Azodi, J. Tang, and S. Shiu. Opening the black box: Interpretable machine learning for geneticists. *Trends Genet*, 36(6): 442–455, 2020.

J. Bac, E. M. Mirkes, A. N. Gorban, I. Tyukin, and A. Zinovyev. Scikit-Dimension: A Python Package for Intrinsic Dimension Estimation. *Entropy*, 23(10):1368, 2021.

N. Bahadur and R. Paffenroth. Dimension Estimation Using Autoencoders, 2019. URL http://arxiv.org/abs/1909.10702. arXiv:1909.10702 [cs, stat].

Z. U. Bajwah, P. K. Seccombe, and R. Offler. Trace element distribution, Co:Ni ratios and genesis of the big cadia iron-copper deposit, new south wales, australia. *Mineral. Deposita*, 22(4): 292–300, 1987.

M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: a geometric frame- work for learning from labeled and unlabeled examples. *J. Mach. Learn. Res.*, 7:2399–2434, 2006.

I. Belousov, R. R. Large, S. Meffre, L. V. Danyushevsky, J. Steadman, and T. Beardsmore. Pyrite compositions from VHMS and orogenic Au deposits in the Yilgarn Craton, Western Australia: Implications for gold and copper exploration. *Ore Geology Reviews*, 79:474–499, 2016.

E. A. Belousova, W. L. Griffin, S. Y. O'Reilly, and N. I. Fisher. Apatite as an indicator mineral for mineral exploration: trace-element compositions and their relationship to host rock type. *Journal of Geochemical Exploration*, 76(1):45–69, 2002a.

E. A. Belousova, W. L. Griffin, S. Y. O'Reilly, and N. L. Fisher. Igneous zircon: trace element composition as an indicator of source rock type. *Contributions to mineralogy and petrology*, 143 (5):602–622, 2002b.

B. C. Benato, C. Grosu, A. X. Falcão, and A. C. Telea. Human-in-the-loop: Using classifier decision boundary maps to improve pseudo labels. *Computers & Graphics*, 124:104062, 2024.

B. C. Benato, A. C. Telea, and A. X. Falcao. Semi-Supervised Learning with Interactive Label Propagation Guided by Feature Space Projections. In *Proc. SIBGRAPI*, pages 392–399. IEEE, 2018.

B. C. Benato, J. F. Gomes, A. C. Telea, and A. X. Falcão. Semi-Automatic Data Annotation guided by Feature Space Projection. *Pattern Recognition*, 109:107612, 2021.

R. Bennett. The intrinsic dimensionality of signal collections. *IEEE Trans. Inform. Theory*, 15(5):517–525, 1969.

K. J. Bergen, P. A. Johnson, M. V. de Hoop, and G. C. Beroza. Machine learning for data-driven discovery in solid Earth geoscience. *Science*, 363(6433):eaau0323, 2019.

J. Bernard, M. Hutter, M. Zeppelzauer, D. Fellner, and M. Sedlmair. Comparing visual-interactive labeling with active learning: An experimental study. *IEEE TVCG*, 24(1):298–308, 2017.

D. Blumberg, Y. Wang, A. Telea, D. A. Keim, and F. L. Dennig. Inverting Multidimensional Scaling Projections Using Data Point Multilateration. In *EuroVis workshop on visual analytics (EuroVA)*. The Eurographics Association, 2024.

R. Bodnar, P. Lecumberri-Sanchez, D. Moncada, and M. Steele-MacInnis. 13.5 - Fluid Inclusions in Hydrothermal Ore Deposits. In *Treatise on Geochemistry (Second Edition)*, pages 119–142. Elsevier, 2014.

I. Borg and P. J. F. Groenen. *Modern multidimensional scaling: theory and applications*. Springer series in statistics. Springer, 2nd ed edition, 2005.

R. Borgo, J. Kehrer, D. H. S. Chung, E. Maguire, R. S. Laramee, H. Hauser, M. Ward, and M. Chen. Glyph-based visualization: Foundations, design guidelines, techniques and applications. In *Eurographics 2013 - state of the art reports*. The Eurographics Association, 2013.

A. Botchkarev. Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology. *Interdisciplinary J. of Information, Knowledge, and Management*, 14:45–79, 2019.

A. Bralia, G. Sabatini, and F. Troja. A revaluation of the Co/Ni ratio in pyrite as geochemical tool in ore genesis problems. *Mineral. Deposita*, 14(3):353–374, 1979.

C. Bredius, Z. Tian, and A. Telea. Visual Exploration of Neural Network Projection Stability. In *Proc. MLVis*, 2022.

L. Breiman. Random Forests. *Mach. Learn.*, 45(1):5–32, 2001.

K. Breiter, J. Ďurišová, and M. Dosbaba. Chemical signature of quartz from S-and A-type rare-metal granites–A summary. *Ore Geology Reviews*, 125:103674, 2020.

F. Camastra. Data dimensionality estimation methods: a survey. *Pattern Recognit.*, 36(12):2945–2954, 2003.

P. Campadelli, E. Casiraghi, C. Ceruti, and A. Rozza. Intrinsic dimension estimation: Relevant techniques and a benchmark framework. *Math. Probl. Eng.*, 2015:1–21, 2015.

D. V. Carvalho, E. M. Pereira, and J. S. Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8):832, 2019.

A. Chattopadhay, A. Sarkar, P. Howlader, and V. Balasubramanian. Grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks. In *Proc. IEEE WACV*, 2018.

A. Chatzimparmpas, R. M. Martins, I. Jusufi, and A. Kerren. A survey of surveys on the use of visualization for interpreting machine learning models. *Information Visualization*, 19(3):207–233, 2020a.

A. Chatzimparmpas, R. M. Martins, and A. Kerren. t-viSNE: Interactive assessment and interpretation of t-SNE projections. *IEEE TVCG*, 26(8):2696–2714, 2020b.

A. Chatzimparmpas, K. Kucher, and A. Kerren. Visualization for trust in machine learning revisited: The state of the field in 2023. *IEEE Computer Graphics and Applications*, 44(3):99–113, 2024.

A. Chatzimparmpas, R. M. Martins, A. C. Telea, and A. Kerren. DeforestVis: Behavior Analysis of Machine Learning Models with Surrogate Decision Stumps, 2023. URL http://arxiv.org/abs/2304.00133. arXiv:2304.00133 [cs].

N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *JAIR*, 16:321–357, 2002.

Y. Choi, Y. Uh, J. Yoo, and J. W. Ha. StarGAN v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2020.

D. B. Coimbra, R. M. Martins, T. T. Neves, A. C. Telea, and F. V. Paulovich. Explaining three-dimensional dimensionality reduction plots. *Information Visualization*, 15(2):154–172, 2016.

T. H. Colding and W. P. Minicozzi. Shapes of embedded minimal surfaces. *Proceedings of the National Academy of Sciences*, 103(30): 11106–11111, 2006.

D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE TPAMI*, 24(5):603–619, 2002.

K. A. Cook and J. J. Thomas. Illuminating the path: The research and development agenda for visual analytics. Technical report, Pacific Northwest National Laboratory (PNNL), Richland, WA (US), 2005.

C. Cortes and V. Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, 1995.

D. R. Cox. Two further applications of a model for binary regression. *Biometrika*, 45(3/4):562–565, 1958.

J. Cunningham and Z. Ghahramani. Linear dimensionality reduction: Survey, insights, and generalizations. *J. Mach. Learn. Res.*, 16:2859–2900, 2015.

A. P. Deditius, M. Reich, S. E. Kesler, S. Utsunomiya, S. L. Chryssoulis, J. Walshe, and R. C. Ewing. The coupled geochemistry of Au and As in pyrite from hydrothermal ore deposits. *Geochimica et Cosmochimica Acta*, 140:644–670, 2014.

V. Dibia and C. Demiralp. Data2Vis: Automatic generation of data visualizations using sequence-to-sequence recurrent neural networks. *IEEE Computer Graphics & Applications*, 39(5):33–46, 2019.

F. Duarte, F. Sikanski, F. Fatore, S. Fadel, and F. V. Paulovich. Nmap: a novel neighborhood preservation space-filling algorithm. *IEEE TVCG*, 20(12):2063–2071, 2014.

I. El Moudden, S. El Bernoussi, and B. Benyacoub. Modeling human activity recognition by dimensionality reduction approach. In *Proc. IBIMA*, pages 1800–1805, 2016.

J. L. Elman. Finding structure in time. *Cognitive science*, 14(2): 179–211, 1990.

S. van den Elzen, G. Andrienko, N. Andrienko, B. Fisher, R. Martins, J. Peltonen, A. Telea, and M. Verleysen. The flow of trust: A visualization framework for externalizing, exploring and explaining trust in ML applications. *IEEE CG &A*, 43(2):78–88, 2023.

M. Espadoto, R. Martins, A. Kerren, N. Hirata, and A. Telea. Toward a quantitative survey of dimension reduction techniques. *IEEE TVCG*, 27(3):2153–2173, 2019a.

M. Espadoto, F. C. M. Rodrigues, and A. Telea. Visual analytics of multidimensional projections for constructing classifier decision boundary maps. In *Proc. IVAPP*. SCITEPRESS, 2019b.

M. Espadoto, F. C. M. Rodrigues, N. S. T. Hirata, and R. Hirata Jr. Deep Learning Inverse Multidimensional Projections. In *Proc. EuroVA*. The Eurographics Association, 2019c.

M. Espadoto, N. S. T. Hirata, and A. C. Telea. Deep learning multidimensional projections. *Information Visualization*, 19(3):247–269, 2020.

M. Espadoto, G. Appleby, A. Suh, D. Cashman, M. Li, C. E. Scheidegger, E. W. Anderson, R. Chang, and A. C. Telea. UnProjection: Leveraging Inverse-Projections for Visual Analytics of High-Dimensional Data. *IEEE TVCG*, 29(2):1559–1572, 2021a.

M. Espadoto, N. Hirata, and A. Telea. Self-supervised Dimensionality Reduction with Neural Networks and Pseudo-labeling. In *Proc. IVAPP*, pages 27–37. SciTePress, 2021b.

M. Espadoto, F. Rodrigues, N. Hirata, and A. Telea. OptMap: Using Dense Maps for Visualizing Multidimensional Optimization Problems. In *Proc. IVAPP*, pages 123–132. SCITEPRESS - Science and Technology Publications, 2021c.

M. Espadoto, F. C. M. Rodrigues, N. S. T. Hirata, and A. C. Telea. Visualizing high-dimensional functions with dense maps. *SN Computer Science*, 4(230), 2023.

E. Facco, M. d'Errico, A. Rodriguez, and A. Laio. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Sci Rep*, 7(1):12140, 2017.

M. Fan, N. Gu, H. Qiao, and B. Zhang. Intrinsic dimension estimation of data by principal component analysis, 2010. URL http://arxiv.org/abs/1002.2050. arXiv:1002.2050 [cs].

R. A. Fisher. Iris Plants Database, 1988. UCI Machine Learning Repository.

R. Garcia, A. Telea, B. da Silva, J. Torresen, and J. Comba. A task-and-technique centered survey on visual analytics for deep learning model engineering. *Computers and Graphics*, 77:30–49, 2018.

Z. Geng and Y. Wang. Physics-guided deep learning for predicting geological drilling risk of wellbore instability using seismic attributes data. *Engineering Geology*, 279:105857, 2020.

P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE, 2018.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. *arXiv:1406.2661 [cs, stat]*, 2014.

D. D. Gregory, M. J. Cracknell, R. R. Large, P. McGoldrick, S. Kuhn, V. V. Maslennikov, M. J. Baker, N. Fox, I. Belousov, M. C. Figueroa, J. A. Steadman, A. J. Fabris, and T. W. Lyons. Distinguishing Ore Deposit Type and Barren Sedimentary Pyrite Using Laser Ablation-Inductively Coupled Plasma-Mass Spectrometry Trace Element Data and Statistical Analysis of Large Data Sets. *Economic Geology*, 114(4):771, 2019.

C. Grosu, Y. Wang, and A. Telea. Computing fast and accurate decision boundary maps. In *Proc. EuroVA*. The Eurographics Association, 2024.

I. Guyon, S. G. S, and A. Ben-Hur. Result analysis of the NIPS 2003 feature selection challenge. In *Advances in neural information processing systems*, pages 545–552, 2004.

N. Hadad, L. Wolf, and M. Shahar. A two-step disentanglement method. In *Proc. IEEE CVPR*, pages 772–780, 2018.

L. Hamel. Visualization of support vector machines with unsupervised learning. In *2006 IEEE symposium on computational intelligence and bioinformatics and computational biology*, pages 1–8, 2006.

C. D. Hansen and C. R. Johnson. *The visualization handbook*. Elsevier, 2005.

X. He, W. Su, N. Shen, X. Xia, and F. Wang. In situ multiple sulfur isotopes and chemistry of pyrite support a sedimentary source-rock model for the Linwang Carlin-type gold deposit in the Youjiang basin, southwest China. *Ore Geology Reviews*, 139: 104533, 2021.

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

F. Hohman, M. Kahng, R. Pienta, and D. H. Chau. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE TVCG*, 25(8):2674–2693, 2019.

T. Hong, X. W. Xu, J. Gao, S. G. Peters, J. Li, M. Cao, P. Xiang, C. Wu, and J. You. Element migration of pyrites during ductile deformation of the Yuleken porphyry Cu deposit (NW-China). *Ore Geology Reviews*, 100:205–219, 2018.

B. Hu, L. P. Zeng, W. Liao, G. Wen, H. Hu, M. Y. H. Li, and X. F. Zhao. The Origin and Discrimination of High-Ti Magnetite in Magmatic-Hydrothermal Systems: Insight from Machine Learning Analysis. *Economic Geology*, 2022.

A. Inselberg and B. Dimsdale. Parallel coordinates: A tool for visualizing multi-dimensional geometry. In *Proc. IEEE VIS*, pages 361–378, 1990.

A. Jaiswal, R. Y. Wu, W. Abd-Almageed, and P. Natarajan. Unsupervised adversarial invariance. In *Advances in neural information processing systems*, volume 31, 2018.

A. Jaiswal, Y. Wu, W. AbdAlmageed, and P. Natarajan. Unified Adversarial Invariance, 2019. URL http://arxiv.org/abs/1905.03629. arXiv:1905.03629 [cs, stat].

T. Jiang, J. Gradus, and A. Rosellini. Supervised machine learning: a brief primer. *Behavior Therapy*, 51(5):675–687, 2020.

P. Joia, D. Coimbra, J. A. Cuminato, F. V. Paulovich, and L. G. Nonato. Local affine multidimensional projection. *IEEE TVCG*, 17(12):2563–2571, 2011.

I. T. Jolliffe and J. Cadima. Principal component analysis: a review and recent developments. *Phil. Trans. Royal Soc. A*, 374(2065), 2016.

I. Jolliffe. *Principal component analysis*. Springer, 2002.

A. Karpatne, I. Ebert-Uphoff, S. Ravela, H. A. Babaie, and V. Kumar. Machine Learning for the Geosciences: Challenges and Opportunities. *IEEE Transactions on Knowledge and Data Engineering*, 31(8):1544–1554, 2019.

T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and Improving the Image Quality of StyleGAN. *arXiv:1912.04958 [cs, eess, stat]*, 2020.

H. Kaur, H. Nori, S. Jenkins, R. Caruana, H. Wallach, and J. Wortman Vaughan. Interpreting Interpretability: Understanding Data Scientists' Use of Interpretability Tools for Machine Learning. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2020.

D. Keim, G. Andrienko, J. D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon. Visual analytics: Definition, process, and challenges. In *Information visualization – human-centered issues and perspectives*, volume 4950, pages 154–175. Springer, 2008.

M. Keith, K. M. Haase, A. R. Chivas, and R. Klemd. Phase separation and fluid mixing revealed by trace element signatures in pyrite from porphyry systems. *Geochimica et Cosmochimica Acta*, 329:185–205, 2022.

Y. Kim, M. Espadoto, S. Trager, J. Roerdink, and A. Telea. SDR-NNP: Sharpened dimensionality reduction with neural networks. In *Proc. IVAPP*, 2022.

J. Kowalewski and A. Ray. Predicting novel drugs for SARS-CoV-2 using machine learning from a> 10 million chemical space. *Heliyon*, 6(8), 2020.

R. R. Large, V. V. Maslennikov, F. Robert, L. V. Danyushevsky, and Z. Chang. Multistage Sedimentary and Metamorphic Origin of Pyrite and Gold in the Giant Sukhoi Log Deposit, Lena Gold Province, Russia. *Economic Geology*, 102(7):1233–1267, 2007.

R. R. Large, L. Danyushevsky, C. Hollit, V. Maslennikov, S. Meffre, S. Gilbert, S. Bull, R. Scott, P. Emsbo, H. Thomas, B. Singh, and J. Foster. Gold and Trace Element Zonation in Pyrite Using a Laser Imaging Technique: Implications for the Timing of

Gold in Orogenic and Carlin-Style Sediment-Hosted Deposits. *Economic Geology*, 104(5):635–668, 2009.

Y. LeCun, C. Cortes, and C. Burges. MNIST handwritten digit database, 2010. URL http://yann.lecun.com/exdb/mnist. http://yann.lecun.com/exdb/mnist.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521 (7553):436–444, 2015.

D. J. Lehmann, G. Albuquerque, M. Eisemann, M. Magnor, and H. Theisel. Selecting coherent and relevant plots in large scatterplot matrices. *Computer Graphics Forum*, 31(6):1895–1908, 2012.

S. Lespinats and M. Aupetit. CheckViz: Sanity check and topological clues for linear and Nonlinear mappings. *Computer Graphics Forum*, 30(1):113–125, 2011.

C. Li, E. M. Ripley, and Y. Tao. Magmatic Ni-Cu and Pt-Pd sulfide deposits in China. *SEG Special Publications*, 2019.

Q. L. Liang, Z. Xie, X. Y. Song, R. Wirth, Y. Xia, and J. Cline. EVOLUTION OF INVISIBLE Au IN ARSENIAN PYRITE IN CARLIN-TYPE Au DEPOSITS. *Economic Geology*, 116(2):515–526, 2021.

S. Lin, K. Hu, J. Cao, T. Bai, Y. Liu, and S. Han. An in situ sulfur isotopic investigation of the origin of Carlin-type gold deposits in Youjiang Basin, southwest China. *Ore Geology Reviews*, 134: 104187, 2021.

Z. C. Lipton. The Mythos of Model Interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

H. Liu, J. Harris, R. Sherlock, P. Behnia, E. Grunsky, M. Naghizadeh, K. Rubingh, G. Tuba, E. Roots, and G. Hill. Mineral prospectivity mapping using machine learning techniques for gold exploration in the Larder Lake area, Ontario, Canada. *Journal of Geochemical Exploration*, 253:107279, 2023.

J. Liu, W. Li, X. Zhu, J. X. Zhou, and H. Yu. Ore genesis of the Late Cretaceous Larong porphyry W-Mo deposit, eastern Tibet: Evidence from in-situ trace elemental and S-Pb isotopic compositions. *Journal of Asian Earth Sciences*, 190:104199, 2020.

M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu. Towards better analysis of deep convolutional neural networks. *IEEE TVCG*, 23(1):91–100, 2016.

S. Liu, D. Maljovec, B. Wang, P. T. Bremer, and V. Pascucci. Visualizing high-dimensional data: Advances in the past decade. *IEEE TVCG*, 23(3):1249–1268, 2015.

F. Luan, S. Paris, E. Shechtman, and K. Bala. Deep photo style transfer. In *Proc. IEEE CVPR*, 2017.

L. van der Maaten. Learning a parametric embedding by preserving local structure. In *Proc. Intl. Conf. on artificial intelligence and statistics*, pages 384–391, 2009.

L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *J. Mach. Learn. Res.*, 9(11):2579–2605, 2008.

A. Machado, A. Telea, and M. Behrisch. ShaRP: Shape-regularized multidimensional projections. In *Proc. EuroVA*. The Eurographics Association, 2023.

A. Machado, M. Behrisch, and A. Telea. Exploring classifiers with differentiable decision boundary maps. *Computer Graphics Forum*, 43(3):e15109, 2024.

G. Mamani, F. Fatore, L. Nonato, and F. Paulovich. User-driven feature space transformation. *Computer Graphics Forum*, 32(3): 291–299, 2013.

E. T. Mansur, S. J. Barnes, and C. J. Duran. An overview of chalcophile element contents of pyrrhotite, pentlandite, chalcopyrite, and pyrite from magmatic Ni-Cu-PGE sulfide deposits. *Miner Deposita*, 56(1):179–204, 2021.

W. E. Marcilio and D. M. Eler. Explaining dimensionality reduction results using Shapley values, 2021.

I. Marin, S. Gotovac, M. Russo, and D. Božić-Štulić. The effect of latent space dimension on the quality of synthesized human face images. *Journal of Communications Software and Systems*, 17 (2):124–133, 2021.

R. Martins, D. Coimbra, R. Minghim, and A. C. Telea. Visual analysis of dimensionality reduction quality for parameterized projections. *Computers & Graphics*, 41:26–42, 2014.

M. F. Mathieu, J. J. Zhao, J. Zhao, A. Ramesh, P. Sprechmann, and Y. LeCun. Disentangling factors of variation in deep representation using adversarial training. In *Advances in neural information processing systems*, volume 29, 2016.

C. Mavrogonatos, P. Voudouris, F. Zaccarini, S. Klemme, J. Berndt, A. Tarantola, V. Melfos, and P. Spry. Multi-Stage Introduction of Precious and Critical Metals in Pyrite: A Case Study from the Konos Hill and Pagoni Rachi Porphyry/Epithermal Prospects, NE Greece. *Minerals*, 10(9):784, 2020.

L. McInnes, J. Healy, and J. Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction, 2018. URL http://arxiv.org/abs/1802.03426. arXiv:1802.03426 [cs, stat].

M. A. Migut, M. Worring, and C. J. Veenman. Visualizing multidimensional decision boundaries in 2D. *Data Mining and Knowledge Discovery*, 29(1):273–295, 2015.

R. Minghim, F. V. Paulovich, and A. A. Lopes. Content-based text mapping using multi-dimensional projections for exploration of document collections. In *Proc. SPIE*, 2006.

T. S. Modrakowski, M. Espadoto, A. X. Falcão, N. S. T. Hirata, and A. Telea. Improving Deep Learning Projections by Neighborhood Analysis. In *Computer Vision, Imaging and Computer Graphics Theory and Applications*, volume 1474, pages 127–152. Springer International Publishing, 2022.

C. Molnar. *Interpretable Machine Learning*. Lean Publishing, 2020.

R. Monarch. *Human-in-the-Loop Machine Learning: Active Learning and Annotation for Human-Centered AI*. Simon and Schuster, 2021.

S. M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. DeepFool: a simple and accurate method to fool deep neural networks, 2016. URL http://arxiv.org/abs/1511.04599. arXiv:1511.04599 [cs].

D. Moyer, S. Gao, R. Brekelmans, A. Galstyan, and G. Ver Steeg. Invariant Representations without Adversarial Training. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

T. Munzner. *Visualization analysis and design: Principles, techniques, and practice*. CRC Press, 2014.

T. T. T. Neves, R. M. Martins, D. B. Coimbra, K. Kucher, A. Kerren, and F. V. Paulovich. Fast and reliable incremental dimensionality reduction for streaming data. *Computers & Graphics*, 102: 233–244, 2022.

L. Nonato and M. Aupetit. Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and layout enrichment. *IEEE TVCG*, 25:2650–2673, 2018.

A. Oliveira, M. Espadoto, R. Hirata, N. Hirata, and A. Telea. Improving self-supervised dimensionality reduction: Exploring hyperparameters and pseudo-labeling strategies. In *Communications in computer and information science*, volume 1691, pages 135–161. Springer, 2023a.

A. A. A. M. Oliveira, M. Espadoto, R. Hirata, and A. C. Telea. Stability Analysis of Supervised Decision Boundary Maps. *SN COMPUT. SCI.*, 4(3):226, 2023b.

A. A. A. M. Oliveira, M. Espadoto, R. Hirata Jr, and A. C. Telea. SDBM: Supervised Decision Boundary Maps for Machine Learning Classifiers. In *Proc. IVAPP*, pages 77–87, 2022.

G. O'Sullivan, D. Chew, G. Kenny, I. Henrichs, and D. Mulligan. The trace element composition of apatite and its application to detrital provenance studies. *Earth-Science Reviews*, 201:103044, 2020.

M. Padala, D. Das, and S. Gujar. Effect of Input Noise Dimension in GANs. In *Neural Information Processing*, Lecture Notes in Computer Science, pages 558–569. Springer International Publishing, 2021.

X. Pan, A. Tewari, T. Leimkühler, L. Liu, A. Meka, and C. Theobalt. Drag Your GAN: Interactive Point-based Manipulation on the Generative Image Manifold, 2023. URL http://arxiv.org/abs/2305.10973. arXiv:2305.10973 [cs].

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library, 2019. URL http://arxiv.org/abs/1912.01703. arXiv:1912.01703 [cs, stat].

F. V. Paulovich and R. Minghim. Text map explorer: a tool to create and explore document maps. In *Proc. IEEE IV*, pages 245–251, 2006.

F. V. Paulovich, L. G. Nonato, R. Minghim, and H. Levkowitz. Least square projection: A fast high-precision multidimensional projection technique and its application to document mapping. *IEEE TVCG*, 14(3):564–575, 2008.

F. V. Paulovich, D. M. Eler, J. Poco, a. C. P. Botha, R. Minghim, and L. G. Nonato. Piecewise laplacian-based projection for interactive data exploration and organization. *Computer Graphics Forum*, 30(3):1091–1100, 2011.

W. Paulsen. A Peano-based space-filling surface of fractal dimension three. *Chaos, Solitons & Fractals*, 168, 2023.

G. Peano. Sur une courbe, qui remplit toute une aire plane. *Mathematische Annalen*, 36(1):157–160, 1890.

J. A. Pearce and J. R. Cann. Tectonic setting of basic volcanic rocks determined using trace element analyses. *Earth and planetary science letters*, 19(2):290–300, 1973.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, and V. Dubourg. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830, 2011.

E. Pekalska, D. de Ridder, R. P. W. Duin, and M. A. Kraaijveld. A new method of generalizing Sammon mapping with application to algorithm speed-up. In *Proc. ASCI*, volume 99, pages 221–228, 1999.

M. Petrelli. *Introduction to Python in Earth Science Data Analysis: From Descriptive Statistics to Machine Learning*. Springer Textbooks in Earth Sciences, Geography and Environment. Springer International Publishing, 2021.

M. Petrelli and D. Perugini. Solving petrological problems through machine learning: the study case of tectonic discrimination using geochemical and isotopic data. *Contrib Mineral Petrol*, 171(10):81, 2016.

N. Pezzotti, T. Höllt, J. Van Gemert, B. P. F. Lelieveldt, E. Eisemann, and A. Vilanova. Deepeyes: Progressive visual analytics for designing deep neural networks. *IEEE TVCG*, 24(1):98–108, 2017.

K. F. Qiu, H. C. Yu, J. Deng, D. McIntire, Z. Y. Gou, J. Z. Geng, Z. S. Chang, R. Zhu, K. N. Li, and R. Goldfarb. The giant Zaozigou Au-Sb deposit in West Qinling, China: magmatic-or metamorphic-hydrothermal origin? *Mineralium Deposita*, pages 1–18, 2020.

R. Rao and S. K. Card. The Table Lens: Merging graphical and symbolic representations in an interactive focus+context visualization for tabular information. In *Proc. ACM SIGCHI*, pages 318–322, 1994.

P. Rauber, A. Falcao, and A. Telea. Visualizing time-dependent data using dynamic t-SNE. In *Proc. EuroVis – short papers*, pages 43–49, 2016.

P. Rauber, S. G. Fadel, A. Falcão, and A. Telea. Visualizing the hidden activity of artificial neural networks. *IEEE TVCG*, 23(1): 101–110, 2017a.

P. E. Rauber, A. X. Falcão, and A. C. Telea. Projections as visual aids for classification system design. *Information Visualization*, 17(4):282–305, 2017b.

R. Ren, Z. Liu, Y. Li, W. X. Zhao, H. Wang, B. Ding, and J. R. Wen. Sequential Recommendation with Self-Attentive Multi-Adversarial Network. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 89–98. ACM, 2020.

M. K. Revan, Y. Genç, V. V. Maslennikov, S. P. Maslennikova, R. R. Large, and L. V. Danyushevsky. Mineralogy and trace-element geochemistry of sulfide minerals in hydrothermal chimneys from the Upper-Cretaceous VMS deposits of the eastern Pontide orogenic belt (NE Turkey). *Ore Geology Reviews*, 63:129–149, 2014.

M. Ribeiro, S. Singh, and C. Guestrin. "Why Should I Trust You?": Explaining the predictions of any classifier, 2016.

F. C. M. Rodrigues, M. Espadoto, R. Hirata, and A. C. Telea. Constructing and Visualizing High-Quality Classifier Decision Boundary Maps. *Information*, 10(9):280, 2019.

F. C. M. Rodrigues, R. Hirata, and A. C. Telea. Image-based visualization of classifier decision boundaries. In *Proc. SIBGRAPI*, pages 353–360. IEEE, 2018.

F. C. M. Rodrigues. *Visual Analytics for Machine Learning*. PhD Thesis, University of Groningen, 2020.

B. Rottier and V. Casanova. Trace element composition of quartz from porphyry systems: a tracer of the mineralizing fluid evolution. *Mineralium Deposita*, pages 1–20, 2020.

B. Rusk. Cathodoluminescent Textures and Trace Elements in Hydrothermal Quartz. In *Quartz: Deposits, Mineralogy and Analytics*, pages 307–329. Springer Berlin Heidelberg, 2012.

D. Sacha, M. Kraus, D. A. Keim, and M. Chen. VIS4ML: An Ontology for Visual Analytics Assisted Machine Learning. *IEEE TVCG*, 25(1):385–395, 2019.

R. Salakhutdinov and I. Murray. On the quantitative analysis of deep belief networks. In *Proc. ICML*, 1998.

G. Salton and M. J. McGill. *Introduction to modern information retrieval*. McGraw-Hill computer science series. McGraw-Hill, 1986.

E. P. dos Santos Amorim, E. V. Brazil, J. Daniels, P. Joia, L. G. Nonato, and M. C. Sousa. iLAMP: Exploring high-dimensional spacing through backward multidimensional projection. In *Proc. IEEE VAST*, pages 53–62, 2012.

A. Schulz, A. Gisbrecht, and B. Hammer. Using discriminative dimensionality reduction to visualize classifiers. *Neural Process. Lett.*, 42:27–54, 2015.

A. Schulz, F. Hinder, and B. Hammer. DeepView: Visualizing Classification Boundaries of Deep Neural Networks as Scatter Plots Using Discriminative Dimensionality Reduction. In *Proc. IJCAI*, pages 2305–2311, 2020.

Scikit-learn developers. Support vector machine classification of the iris dataset, 2024. URL https://scikit-learn.org/stable/auto_examples/svm/plot_iris_svc.html.

Y. M. Sheng. Distal gold mineralization associated with porphyry system: The case of Hongzhuang and Yuanling deposits, East Qinling, China. *Ore Geology Reviews*, page 16, 2022.

R. Shwartz-Ziv and N. Tishby. Opening the black box of deep neural networks via information, 2017.

R. da Silva, P. Rauber, R. Martins, R. Minghim, and A. C. Telea. Attribute-based visual explanation of multidimensional projections. In *Proc. EuroVA*, 2015.

M. Sips, B. Neubert, J. Lewis, and P. Hanrahan. Selecting good views of high-dimensional data using class consistency. *Comp Graph Forum*, 28(3):831–838, 2009.

D. Smilkov and S. Carter. Playground Tensorflow: Visualization of simple decision maps for a neural network, 2024.

J. T. Sohns, C. Garth, and H. Leitte. Decision Boundary Visualization for Counterfactual Reasoning. *Comput. Graph. Forum*, 42 (1):7–20, 2023.

I. Sommerville. *Software engineering*. O'Reilly Publishing, 2015.

C. O. S. Sorzano, J. Vargas, and A. P. Montano. A survey of dimensionality reduction techniques, 2014. arXiv:1403.2877 [stat.ML].

H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush. LSTMVis: a tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE TVCG*, 24(1):667–676, 2017.

J. X. Sui, J. W. Li, A. H. Hofstra, H. O'Brien, Y. Lahaye, D. Yan, Z. K. Li, and X. Y. Jin. Genesis of the Zaozigou gold deposit, West Qinling orogen, China: Constraints from sulfide trace element and stable isotope geochemistry. *Ore Geology Reviews*, 122:103477, 2020.

T. Suwannaphong, S. Chavana, S. Tongsom, D. Palasuwan, T. H. Chalidabhongse, and N. Anantrasirichai. Parasitic egg detection and classification in low-cost microscopic images using transfer learning. *SN Computer Science*, 5(1):82, 2023.

L. Tang, Y. Zhao, S. T. Zhang, L. Sun, X. K. Hu, Y. M. Sheng, and T. Zeng. Origin and evolution of a porphyry-breccia system: Evidence from zircon U-Pb, molybdenite Re-Os geochronology, in situ sulfur isotope and trace elements of the Qiyugou deposit, China. *Gondwana Research*, 89:88–104, 2021.

A. C. Telea. Combining extended table lens and treemap techniques for visualizing tabular data. In *Proc. EuroVis*, pages 120–127, 2006.

A. Telea, A. Machado, and Y. Wang. Seeing is Learning in High Dimensions: The Synergy Between Dimensionality Reduction and Machine Learning. *SN Computer Science*, 5(3):279, 2024.

A. C. Telea. *Data visualization: principles and practice*. CRC Press, second ed edition, 2014.

J. Thijssen, Z. Tian, and A. Telea. Scaling up the explanation of multidimensional projections. In *Proc. EuroVA*, 2023.

J. Thiyagalingam, M. Shankar, G. Fox, and T. Hey. Scientific machine learning benchmarks. *Nature Reviews Physics*, 4:413–420, 2022.

M. Thoma. The reuters dataset, 2017. URL https://martin-thoma.com/nlp-reuters.

Z. Tian, X. Zhai, D. van Driel, G. van Steenpaal, M. Espadoto, and A. Telea. Using multiple attribute-based explanations of multidimensional projections to explore high-dimensional data. *Comput. Graph.*, 98:93–104, 2021.

W. S. Torgerson. Multidimensional scaling: I. Theory and method. *Psychometrika*, 17:401–419, 1952.

E. R. Tufte. *The visual display of quantitative information, 2^n? edition*. Graphics Press, 2001.

E. R. Tufte. The visual display of quantitative information Graphics Press. *Cheshire, Connecticut*, 6410, 1983.

F. Y. Tzeng and K. L. Ma. Opening the black box – data driven visualization of neural networks. In *Proc. IEEE visualization*, 2005.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

J. Venna and S. Kaski. Visualizing gene interaction graphs with local multidimensional scaling. In *Proc. ESANN*, pages 557–562, 2006.

E. Vernier, R. Garcia, I. da Silva, J. Comba, and A. Telea. Quantitative evaluation of time-dependent multidimensional projection techniques. *Computer Graphics Forum*, 39(3):241–252, 2020.

E. Vernier, J. Comba, and A. Telea. Guided stable dynamic projections. *Computer Graphics Forum*, 40(3):87–98, 2021.

P. J. Verveer and R. P. W. Duin. An evaluation of intrinsic dimensionality estimators. *IEEE PAMI*, 17(1):81–86, 1995.

H. Wang, J. Li, A. Telea, J. Kosinka, and Z. Wu. USTNet: Unsupervised Shape-to-Shape Translation via Disentangled Representations. *Computer Graphics Forum*, 41(7):141–152, 2022a.

X. Wang, C. Q. Liu, Y. Yi, M. Zeng, S. L. Li, and X. Niu. Machine Learning Predicts the Methane Clumped Isotopologue (^{12} CH $_{2}$ D $_{2}$) Distributions Constrain Biogeochemical Processes and Estimates the Potential Budget. *Environ. Sci. Technol.*, 57(46): 17876–17888, 2023a.

Y. Wang, F. Dennig, M. Behrisch, and A. Telea. LCIP: Loss-controlled inverse projection of high-dimensional data, 2025a. submitted to IEEE TVCG.

Y. Wang, F. Dennig, M. Behrisch, and A. Telea. LCIP implementation source code, 2025b. URL https://github.com/wuyuyu1024/lcip.

Y. Wang, C. Grosu, and A. Telea. Generalized FastDBM implementation source code, 2025c. URL https://github.com/yuwang-vis/generalized_fastDBM.

Y. Wang, H. Yao, and S. Zhao. Auto-encoder based dimensionality reduction. *Neurocomputing*, 184:232–242, 2016.

Y. Wang and A. Telea. Fundamental Limitations of Inverse Projections and Decision Maps. In *Proc. IVAPP*, volume 1, pages 571–582, 2024.

Y. Wang and A. Telea. Investigating Desirable Properties of Inverse Projections and Decision Maps. *Communications in Computer and Information Science*, 2025.

Y. Wang, K. Qiu, A. Müller, Z. Hou, Z. Zhu, and H. Yu. Machine Learning Prediction of Quartz Forming-Environments. *Journal of Geophysical Research: Solid Earth*, 126(8):e2021JB021925, 2021.

Y. Wang, K. Qiu, Z. Hou, and H. Yu. Quartz Ti/Ge-P discrimination diagram: A machine learning based approach for deposit classification. *Acta Petrol. Sin*, 38(1):281–290, 2022b.

Y. Wang, A. Machado, and A. Telea. Quantitative and Qualitative Comparison of Decision Map Techniques for Explaining Classification Models. *Algorithms*, 16(9):438, 2023b.

Y. Wang, K. F. Qiu, A. C. Telea, Z. L. Hou, T. Zhou, Y. W. Cai, Z. J. Ding, H. C. Yu, and J. Deng. Interpreting mineral deposit genesis classification with decision maps: A case study using pyrite trace elements. *American Mineralogist*, 109(12):2116–2126, 2024.

Y. Wang, C. Grosu, and A. Telea. Computing fast and accurate maps for explaining classification models. *Computers & Graphics*, 129:104230, 2025d.

M. Wattenberg, F. Viégas, and I. Johnson. How to use t-sne effectively. *Distill*, 1(10):e2, 2016.

J. Weickert and H. Hagen. *Visualization and processing of tensor fields*. Springer, 2005.

L. Wilkinson, A. Anand, and R. Grossman. Graph-theoretic scagnostics. In *Proc. IEEE InfoVis*, pages 21–21. IEEE Computer Society, 2005.

H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms, 2017. arXiv 1708.07747 [cs.LG].

Q. Xie, Z. Dai, Y. Du, E. Hovy, and G. Neubig. Controllable Invariance through Adversarial Feature Learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Z. Xie, Y. Xia, J. S. Cline, M. J. Pribil, A. Koenig, Q. Tan, D. Wei, Z. Wang, and J. Yan. Magmatic Origin for Sediment-Hosted Au Deposits, Guizhou Province, China: In Situ Chemistry and Sulfur Isotope Composition of Pyrites, Shuiyindong and Jinfeng Deposits. *Economic Geology*, 113(7):1627–1652, 2018.

A. Yates, A. Webb, M. Sharpnack, H. Chamberlin, K. Huang, and R. Machiraju. Visualizing multidimensional data with glyph SPLOMs. *Computer Graphics Forum*, 33(3):301–310, 2014.

I. K. Yeo and R. A. Johnson. A new family of power transformations to improve normality or symmetry. *Biometrika*, 87(4): 954–959, 2000.

J. Yuan, C. Chen, W. Yang, M. Liu, J. Xia, and S. Liu. A survey of visual analytics techniques for machine learning. *Comp. Visual Media*, 7(1):3–36, 2021.

P. Zhang, X. W. Huang, B. Cui, B. C. Wang, Y. F. Yin, and J. R. Wang. Re-Os isotopic and trace element compositions of pyrite and origin of the Cretaceous Jinchang porphyry Cu-Au deposit, Heilongjiang Province, NE China. *Journal of Asian Earth Sciences*, 129:67–80, 2016.

J. Zheng, H. Shen, J. Yang, X. Tang, M. Chen, H. Yu, J. Guo, and X. Wei. Autoencoders with Intrinsic Dimension Constraints for Learning Low Dimensional Image Representations, 2023. URL http://arxiv.org/abs/2304.07686. arXiv:2304.07686 [cs].

Z. Zheng and L. Sun. Disentangling Latent Space for VAE by Label Relevant/Irrelevant Dimensions. In *Proc. IEEE CVPR*, pages 12184–12193. IEEE, 2019.

R. Zhong, Y. Deng, W. Li, L. V. Danyushevsky, M. J. Cracknell, I. Belousov, Y. Chen, and L. Li. Revealing the multi-stage ore-forming history of a mineral deposit using pyrite geochemistry and machine learning-based data interpretation. *Ore Geology Reviews*, 133:104079, 2021a.

R. Zhong, Y. Deng, and C. Yu. Multi-layer perceptron-based tectonic discrimination of basaltic rocks and an application on the Paleoproterozoic Xiong'er volcanic province in the North China Craton. *Computers & Geosciences*, 149:104717, 2021b.

T. Zhou, K. Qiu, Y. Wang, H. Yu, and Z. Hou. Apatite Eu/Y-Ce discrimination diagram: A big data based approach for provenance classification. *Acta Petrol. Sin*, 38(1):291–299, 2022.

T. Zhou, Y. Cai, M. An, F. Zhou, C. Zhi, X. Sun, and M. Tamer. Visual interpretation of machine learning: Genetical classification of apatite from various ore sources. *Minerals*, 13(4):491, 2023.

Z. Zhu, F. Zhou, Y. Wang, T. Zhou, Z. Hou, and K. Qiu. Machine learning-based approach for zircon classification and genesis determination. *Earth Science Frontiers*, 29(5):464, 2022.

M. van der Zwan, V. Codreanu, and A. Telea. CUBu: Universal real-time bundling for large graphs. *IEEE TVCG*, 22(12):2550–2563, 2016.

BIOGRAPHY

Yu Wang was born on May 9 1997 in Hebei, China. He received his Bachelor's degree in Product Design from the School of Gemmology at China University of Geosciences, Beijing (CUGB) in 2019. During his undergraduate studies, he also obtained the Gemmology Diploma (FGA) from the Gemmological Association of Great Britain (Gem-A).

Following this, he began a six-year graduate program in Geology at CUGB. Yu's research in this direction focuses on the application of machine learning to classify mineral genetic types as well as using various explainable AI (XAI) techniques to improve the design and performance of such models.

In 2022, Yu met prof. Alexandru Telea and began collaborating on research related to decision maps. In March 2023, Yu joined the Visualization and Graphics (VIG) group within the Department of Information and Computing Sciences at Utrecht University. While working with VIG, Yu expanded his research scope to include high-dimensional data visualization, inverse projection techniques, and decision maps for classifiers.

Yu's research interests include interpretable machine learning, dimensionality reduction, deep-learning-based inverse projection methods, and generative AI. He is passionate about bridging algorithmic design and human understanding through visual analytics.

ACKNOWLEDGMENTS

First and foremost, I would like to express my heartfelt gratitude to my supervisor, prof. Alex Telea. Thank you for accepting me as his PhD student. Thank you for your patient guidance, strong support, and for helping shape the person I am today. You've given me a new direction in life, and I will always be grateful for that.

I also sincerely thank my second supervisor, dr. Michael Behrisch, for his support and for making my PhD journey smoother and more rewarding.

My appreciation extends to all members of the Visualization and Graphics (VIG) group for the valuable feedback, engaging discussions, and the friendly and supportive atmosphere throughout my time here.

I also want to thank my friend, Zhihai Zhu, who first guided me onto the path of programming and computer science, and who has patiently provided support along the way.

To my climbing friends – thank you for bringing fun, color, and adventure to my life in the Netherlands. Your friendship has meant so much to me.

A special thanks to Lisa for designing the cover of this thesis, volunteering as the photographer for my defense, and offering many other forms of support along the way.

Last but certainly not least, I want to thank my family and friends for their constant love, encouragement, and understanding throughout this journey. Your support has been my foundation.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

`http://code.google.com/p/classicthesis/`

Final Version as of June 13, 2025 (`classicthesis`).