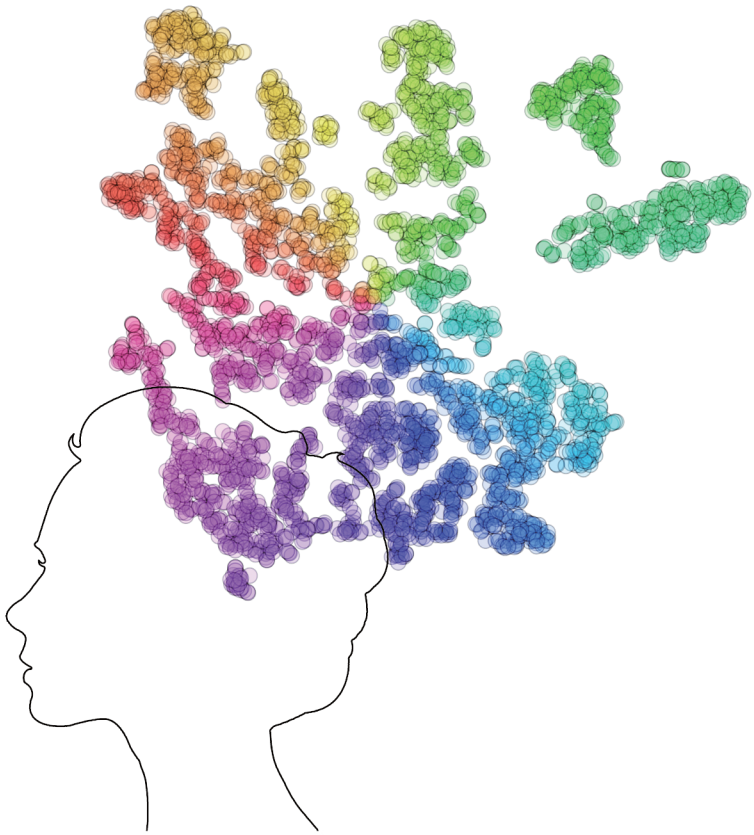
PSEUDO LABELING AND CLASSIFICATION OF HIGH-DIMENSIONAL DATA USING VISUAL ANALYTICS



BÁRBARA C. BENATO

PSEUDO LABELING AND CLASSIFICATION OF HIGH-DIMENSIONAL DATA USING VISUAL ANALYTICS

BÁRBARA CAROLINE BENATO

Cover: A colored 2D projection of high-dimensional data.

Pseudo labeling and classification of high-dimensional data using visual analytics

Bárbara Caroline Benato
PhD Thesis

This thesis is the result of a joint PhD between the University of Campinas and the Utrecht University.

Pseudo Labeling and Classification of High Dimensional-Data using Visual Analytics

Pseudoetikettering en Classificatie van Hoogdimensionale Gegevens met Visuele Analyse

(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. H.R.B.M. Kummeling,
ingevolge het besluit van het college voor promoties
in het openbaar te verdedigen op

dinsdag 9 juli 2024 des middags te 4.15 uur.

door

Bárbara Caroline Benato

geboren op 10 januari 1994
te Botucatu, Brazil

Promotors:
Prof. dr. A. C. Telea
Prof. dr. A. X. Falcão

Beoordelingscommissie:
Prof. dr. G. C. S. de Araújo
Prof. dr. H. L. Hardman
Prof. dr. A. R. Rocha
Prof. dr. A. A. Salah
Prof. dr. I. Velegrakis

This degree is awarded as part of a Joint Doctorate with University of Campinas, Brazil.

This thesis was (partly) accomplished with financial support from FAPESP (São Paulo Research Foundation, grant agreement numbers 2019/10705-8 and 2022/12668-5) and CAPES (Coordination for the Improvement of Higher Education Personnel, Financial code 001).

Dit proefschrift werd mede mogelijk gemaakt met financiële steun van FAPESP (São Paulo Research Foundation) onder subsidieovereenkomst nr. 2019/10705-8 en nr. 2022/12668-5) en CAPES (Coordination for the Improvement of Higher Education Personnel) onder subsidieovereenkomst nr. 001.

To my grandmother, Bela, who saw more courage in me than I could. And to all other COVID victims.

ABSTRACT

Machine learning (ML) works with data consisting of tens up to tens of thousands of measurements (dimensions) per sample. As the number of dimensions and/or samples grow, so does the difficulty of understanding such data and, related to that, understanding how to design ML pipelines that effectively process such data for tasks such as classification. Visualization, and in particular Visual Analytics (VA) has emerged as one of the key approaches that helps practitioners with the understanding of high-dimensional data and with ML engineering tasks. This thesis studies several novel approaches by which VA can help ML (and conversely), as follows.

Our work focuses on a visualization technique called *dimensionality reduction*, or projection, which handles efficiently and effectively large amounts of high-dimensional data. One the ML side, we consider the task of training a typical classifier for the challenging context when only a small amount of ground-truth labels is available.

We first propose a pseudo-labeling approach that explores the ability of projections to generate a reduced feature space with enough information to improve feature learning and classifier performance over iterations. We show that the 2D space generated by projections can capture very well the data structure present in high dimensions so as to support the design of high-performance feature and classifier learning models.

Secondly, we link data separation (DS), visual separation (VS), and classifier performance (CP) by pseudo-labeling and projections. We use feature spaces with high DS as input to compute high-VS projections. We use these projections to perform pseudo labeling with high propagation accuracies. Finally, we use such labels to train classifiers with a high CP. We show that the high-DS, high-VS, high-CP implication holds for several types of projection techniques. Hence, such projection techniques are suitable for the task of classifier engineering.

Thirdly, we exploit the aforementioned observation that high-VS and high-CP are correlated to propose a metric to assess the VS of labeled 2D scatterplots produced by projection techniques. Our metric computes the accuracy of label propagation in the projection space, which is simple and fast to execute. We show

that high propagation accuracies match a high VS as assessed by human subjects.

Finally, we join all our contributions to incorporate the user in the ML engineering process. We propose an interactive VA tool that assists users in manual labeling samples by providing additional information in terms of classifier decision boundary maps, projection errors, and inverse projection errors. Our results show that this approach enables users to quickly generate labeled samples that lead to higher classification performance after a few labeling iterations. This contribution shows that both algorithms and humans can exploit projections to build better classifiers.

SAMENVATTING

Machineleer (ML) werkt met gegevens bestaande uit tientallen tot tienduizenden metingen (dimensies) per datapunt. Naarmate het aantal dimensies en/of datapunten groeit, neemt ook de moeilijkheid toe om dergelijke gegevens te begrijpen en, daarmee samenhangend, om te begrijpen hoe ML-pijplijnen die dergelijke gegevens effectief verwerken voor taken als classificatie, ontworpen moeten worden. Visualisatie, en in het bijzonder Visual Analytics (VA), is naar voren gekomen als belangrijke methode die helpt bij het begrijpen van hoogdimensionale gegevens en bij ML-engineeringstaken. Dit proefschrift bestudeert verschillende nieuwe methoden waarmee VA ML kan helpen (en omgekeerd), als volgt.

Ons werk richt zich op een visualisatietechniek genaamd *dimensionality reduction*, of projectie, die grote hoeveelheden hoogdimensionale gegevens efficiënt en effectief verwerkt. Aan de ML-kant beschouwen we de taak van het trainen van een typische classificator voor het uitdagende geval waarin slechts een klein aantal *ground-truth* etiketten beschikbaar is.

We stellen eerst een pseudo-labelingsmethode voor die het vermogen van projecties onderzoekt om een kleinere *feature* ruimte te genereren met voldoende informatie om het leren van features en de classificatieprestaties over iteraties te verbeteren. We laten zien dat de door projecties gegenereerde 2D-ruimte de datastructuur in hoge dimensies zeer goed kan vastleggen om zo het ontwerp van hoogwaardige leermodellen voor features en classificaties te ondersteunen.

Ten tweede koppelen we datascheiding (DS), visuele scheiding (VS) en classificatieprestaties (CP) door pseudo-labeling en projecties. We gebruiken *feature* ruimten met hoge DS als invoer om hoge VS-projecties te berekenen. We gebruiken vervolgens deze projecties om pseudo-labeling uit te voeren met hoge propagatienauwkeurigheid. Ten slotte gebruiken we dergelijke etiketten om classificatoren met een hoge CP te trainen. We laten zien dat de implicaties van hoge DS, hoge VS en hoge CP gelden voor verschillende soorten projectietechnieken. Dergelijke projectietechnieken zijn daarom geschikt voor de taak van classificatie.

Ten derde maken we gebruik van de bovengenoemde observatie dat hoge VS en hoge CP gecorreleerd zijn om een metriek voor te stellen om de VS van gelabelde 2D-spreidingsdiagrammen die

zijn geproduceerd door projectietechnieken, te meten. Onze metriek berekent de nauwkeurigheid van de etiketpropagatie in de projectieruimte op een eenvoudige en snelle manier. We laten zien dat hoge propagatienauwkeurigheden overeenkomen met een hoge VS zoals beoordeeld door gebruikers.

Ten slotte bundelen we al onze bijdragen om de gebruiker te betrekken bij het ML-engineeringsproces. We stellen een interactieve VA-tool voor die gebruikers helpt bij het handmatig etiketteren van datapunten door aanvullende informatie te bieden in termen van classificatie-beslissingsgrenskaarten, projectiefouten en inverse projectiefouten. Onze resultaten laten zien dat deze aanpak gebruikers in staat stelt snel gelabelde datapunten te genereren die leiden tot hogere classificatieprestaties na een paar etiketteringsiteraties. Dit resultaat laat zien dat zowel algoritmen als mensen projecties kunnen gebruiken om betere classificatoren te bouwen.

RESUMO

Aprendizado de máquina (do inglês, *Machine Learning* (ML)) explora dados contendo de dezenas até dezenas de milhares de medições (dimensões) por amostra/exemplo. À medida que o número de dimensões e/ou amostras cresce, também cresce a dificuldade de compreensão do dado em questão e, relacionado a isso, a compreensão de como projetar modelos de ML que processam tais dados de forma eficaz para tarefas como classificação de dados. Visualização, e em particular analítica visual (*Visual Analytics* (VA)), tem emergido como uma das abordagens chave para ajudar profissionais no entendimento de dados de alta dimensionalidade e de tarefas de engenharia de ML. Tal tese ocupa-se em estudar diversas abordagens nas quais VA pode auxiliar ML (e vice-versa), como a seguir.

O presente trabalho foca em uma técnica de visualização denominada redução de dimensionalidade (*dimensionality reduction*), ou projeção, a qual lida eficientemente e efetivamente com grandes quantidades de dados de alta dimensionalidade. Considerando ML, considera-se a tarefa de treinar um classificador típico para o contexto desafiador onde apenas uma pequena quantidade de rótulos (*label*) verdadeiros está disponível.

Primeiramente, é proposta uma abordagem de pseudo rotulação que explora a habilidade de projeções em gerar um espaço de características reduzido com informação suficiente para melhorar a performance do aprendizado de características e do classificador ao longo das iterações. Como resultado, mostra-se que o espaço 2D gerado a partir de projeções pode capturar de forma satisfatória a estrutura do dado presente em altas dimensões de forma a auxiliar no projeto de modelos de aprendizado de características e classificação de alta performance.

Além disso, propõe-se relacionar os conceitos de separação de dados DS, separação visual (VS) e performance do classificador (CP) através da pseudo rotulação e projeções de dados. Um espaço de características com alta DS é usado como entrada para computar projeções com alta VS. Tais projeções são, então, empregadas para realizar a pseudo rotulação com altas acurácias de propagação de rótulos. Por fim, esses rótulos são utilizados para treinar um classificador com uma alta CP. A implicação entre alta DS, alta VS e alta CP é mostrada para diferentes tipos de técnicas

de projeção, as quais indicaram ser adequadas para a tarefa de engenharia de classificadores.

Adicionalmente, a constatação anteriormente mencionada de que alta VS e alta CP estão correlacionadas é explorada para propôr uma métrica para acessar a VS de gráficos de dispersão 2D resultantes de técnicas de projeções. A métrica proposta computa a acurácia da propagação de rótulos no espaço projetado, o que torna a métrica mais simples e de mais rápida de executar. As altas acurácias de propagação mostram uma correlação com uma alta VS encontrada por seres humanos.

Finalmente, as contribuições encontradas são agregadas a fim de incorporar o usuário no processo de engenharia de modelos de ML. É proposta uma ferramenta interativa de VA que auxilia o usuário na rotulação manual de amostras ao fornecer informação adicional referente a mapas de bordas de decisão de classificadores, erros de projeção, e erros de projeção inversa. Os resultados mostram que essa abordagem permite que o usuário possa rapidamente gerar novos rótulos para as amostras. Tais amostras rotuladas conduzem a maiores performances de classificação após algumas iterações apenas. Esta contribuição mostra que ambos, algoritmos e seres humanos, podem explorar projeções para a construção de melhores classificadores.

PUBLICATIONS

This thesis is based on the following publications:

- B. C. Benato, J. F. Gomes, A. C. Telea, and A. X. Falcão. Semi-supervised deep learning based on label propagation in a 2D embedded space. In *Proc. 25th Iberoamerican Congress: Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. CIARP, 2021.
- B. C. Benato, A. C. Telea, and A. X. Falcão. Iterative pseudo-labeling with deep feature annotation and confidence-based sampling. In *Proc. 34th Conference on Graphics, Patterns and Images*. SIBGRAPI, 2021.
- B. C. Benato, A. X. Falcão, and A. C. Telea. Linking Data Separation, Visual Separation, and Classifier Performance Using Pseudo-labeling by Contrastive Learning. In *Proc. 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. VISAPP, 2023.
- B. C. Benato, A. C. Telea, and A. X. Falcão. Deep feature annotation by iterative meta-pseudo-labeling on 2D projections. *Pattern Recognition*, 2023.
- B. C. Benato, A. X. Falcão, and A. C. Telea. Measuring the quality of projections of high-dimensional labeled data. *Computers & Graphics*, 2023.
- B. C. Benato, A. X. Falcão, and A. C. Telea. Linking data separation, visual separation, and classifier performance using dimensionality reduction techniques. *Book title*, pp. 00-00. Springer, 2023. (to appear)
- 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 (submitted, 2024)

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

- M. Roder, L. A. Passos, L. C. F. Ribeiro, B. C. Benato, A. X. Falcão, J. P. Papa. Intestinal parasites classification using deep belief networks. In *International Conference on Artificial Intelligence and Soft Computing*. Springer. 2020, pp. 242–251

- B. C. Benato, I. E. de Souza, F. L. Galvão, F. L., and A. X. Falcão. Convolutional neural networks from image markers. In *Beyond backpropagation: novel ideas for training neural architectures*, Workshop at NeurIPS, 2020.
- I. E. de Souza, B. C. Benato, A. X. Falcão. Feature learning from image markers for object delineation. In *Proc. 33rd Conference on Graphics, Patterns and Images*. SIBGRAPI, 2020.
- L. M. João, M. C. Abrantes, B. C. Benato, A. X. Falcão. Understanding marker-based normalization for FLIM Networks. In *Proc. 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. VISAPP, 2024. (to appear)

CONTENTS

1 INTRODUCTION

1.1 MACHINE LEARNING

Artificial intelligence (AI) is the broader field known for aiming to imitate human intelligence through a combination of software and hardware. AI plays an essential role in human lives in many ways, ranging from the usage of social media, e-commerce support, recommender systems for streaming media, and decision support systems. Nowadays, a *machine learning* (ML) algorithm is nearly always involved in most of these processes. As a subfield of AI, ML aims to enable machines to imitate how humans learn from their experience via controlled examples (samples). Giving many examples to a machine, the learner (algorithm) assembles knowledge from the same and/or distinct examples. After some time of learning (training), the learner is able to assume (predict) new and unseen examples of the real world.

Algorithms for ML can be divided mainly into *supervised* and *unsupervised* ones, based on their goal. In the *supervised learning* paradigm, the goal is to learn from data to make new predictions. The class, or label, information of an example is used to train the algorithm, in what is also known as a 'task-driven' approach. Tasks may be classification or regression. Classification problems aim to learn a function that outputs a discrete value, *i.e.*, a class label for each given example, such as deciding which animal is present in an image. For regression problems, the algorithm approximates a function that outputs a continuous value for each instance, such as predicting house prices. In the *unsupervised learning* strategy, the goal is to determine structures and patterns in data. No class information is considered in usupervised training, which is why this approach is also known as 'data-driven.'

ML models have grown alongside computational processing capability. In particular, *artificial neural networks* (ANNs) have shown excellent performance and computational scalability after breakthroughs that optimized the process of training their models. At a high level, an ANN is thought to mimic the neural structure of the brain by a mathematical formulation, thereby succeeding in learning complex functions involving millions of inputs (variables). Today, we see that ANNs can be exploited to solve tasks ranging from relatively simple ones, such as classification,

to more complex ones, such as recognizing patterns and textures, reducing feature representation, and generating new (synthetic) data. In this process, the architecture of ANNs has evolved by increasing the size and number of layers, leading to *deep neural networks* (DNNs), also known as *deep learning*.

1.2 VISUALIZATION

For the last several decades, data visualization (VIS) has grown aside and along machine learning (ML). In its early phases, visualization has been introduced to science and engineering fields by the need of understanding increasingly large (and complex) datasets generated either by measurements or by numerical simulations produced by scientific computing applications. Following this development, visualization research (and applications) have further specialized in two main sub-fields. *Scientific visualization* (scivis) has kept the focus on the original target of visualization, namely the visual exploration of spatial datasets consisting of samples of continuous quantities, such as temperature, flow, and pressure fields defined on 2D or 3D computational domains (Hansen and Johnson, 2005; Telea, 2014).

Alongside scivis, *information visualization* (infovis) has emerged as an increasingly large and important subfield. In contrast to scivis, infovis focuses on the visual exploration of data which is not defined on spatial domains and/or does not consist of (the sampling of) continuous quantities. Examples of such datasets include data tables (where rows are typically samples but position information may be missing; and/or the values of attributes, or columns, can be of categorical or text type, aside from quantitative values); networks or graphs; text data; and artifacts from software engineering such as source code and design diagrams,(Tufte, 2001; Munzner, 2014).

The differentiation between scivis and infovis applications is, however, not a hard one. Applications can generate datasets having mixed properties – consider, for instance, the field of graph visualization where the input data, the structure of a graph, is non-spatial, but the result, a graph drawing, is by definition a dataset embedded into 2D or 3D space and consisting of continuous coordinate values. Techniques used in the construction of both scivis and infovis applications share many commonalities – for example, the interaction with large datasets that supports the selection of subsets of interest and the display of additional details on demand (Shneiderman, 1996). However, even more importantly, all data visualization applications, whether scivis, infovis, or hy-

brids, share the same ultimate goal – to enable their users to gain so-called *actionable insights* into the phenomena which have generated the explored data, and, next, to use such insights to improve other aspects of the processes involved with the respective data. Recognizing this commonality between all visualization applications, a new field called *visual analytics* (VA) has emerged. VA studies the design, deployment, and use of visualization *tools* and *workflows* that empower users to gain the aforementioned actionable insights (Andrienko et al., 2020; Cook and Thomas, 2005).

1.3 INTERACTION OF MACHINE LEARNING WITH VISUALIZATION

Machine learning works, in most cases, with *high dimensional data*. By this, we mean datasets consisting of samples (also called data points or observations) which have, each, tens up to thousands of different measurements (also called dimensions, features, variables, or attributes). Indeed, consider a ML classification model which takes as input an image. In this case, the image has to be 'fed' to the model, either by considering each of its pixels as an independent variable or by extracting an (usually high) number of features from these pixels to be further fed to the model. Simplifying for the sake of exposition, an ML pipeline can be seen as a pipeline that processes data tables (of different sizes and types) whose rows consist of data samples and whose columns consist of the data attributes, respectively.

As the size – either in number of rows (samples) or columns (dimensions) – of these data tables grows, so does the difficulty of *understanding* them and, even more importantly, understanding how ML pipelines process them. As such, it is not surprising that visualization (VIS) and visual analytics (VA), with their stated aims of helping users extract actionable insights from large and complex datasets, has emerged as a key tool in assisting ML engineering. This thesis will study precisely this subset at the crossroads of ML and VA, namely how the two fields can benefit from each other.

As mentioned above, datasets in ML pipelines are, by their very nature, high dimensional. As such, visualization techniques that aim to assist ML engineering have to be able to efficiently and effectively handle such high-dimensional data. Among the family of visualizations techniques that address such goals, *dimensionality reduction* (DR) techniques, also called projections, have emerged as one of the most successful and most frequently used to assist ML engineering. Simply put, DR techniques receive as

input a high-dimensional data table or dataset having N samples, each with n dimensions, where N can typically be in the hundreds of thousands or even millions and n can be tens up to thousands, respectively. From such datasets, DR techniques create typically 2D, and more rarely 3D, scatterplots, one plot point per data sample. In this process, DR techniques aim to preserve the so-called *data structure*. That is, characteristics of the input *dataset* which are deemed important for the problem at hand, such the presence of specific data clusters and/or outliers, the distribution of samples over specific data ranges, and the relative similarities of data samples, should be encoded by similar characteristics measured by the corresponding scatterplot points. When this is achieved, one can use the scatterplots generated by DR to reason about the input data. The key added value of this proposal is that direct visual exploration of the high-dimensional data is, in general, impossible for large N and/or n values. In contrast, DR techniques have an excellent scalability in both respects – in the limit, they require a single pixel to encode a data point having any number of dimensions n.

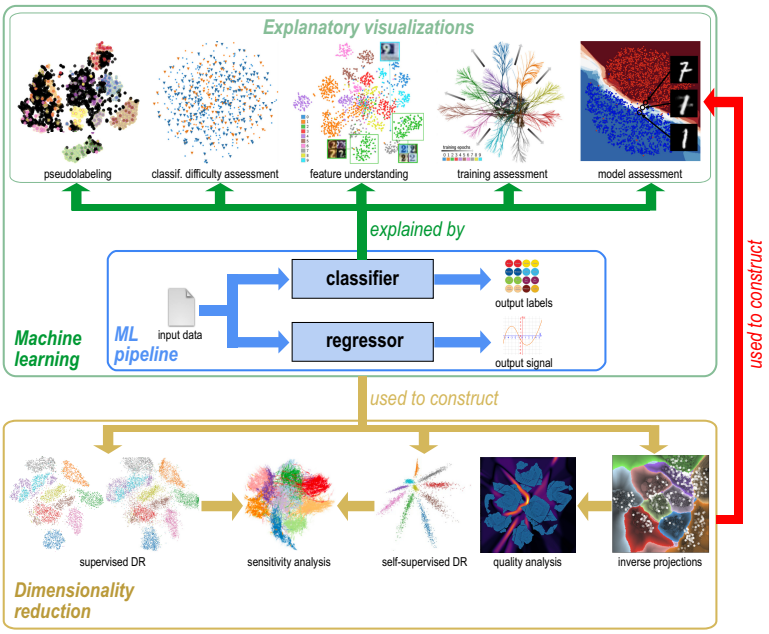


Figure 1.1: Interaction between machine learning (ML) and dimensionality reduction (DR) workflows. ML algorithms can be used to construct DR techniques. In turn, these can be used to construct explanatory visualizations for ML. Figure reproduced from Telea et al. (2024).

There are multiple ways in which high-dimensional data visualization can help ML engineering (and conversely). Figure 1.1, taken from a recent paper that addresses this topic, outlines these interactions. Here and next, we focus specifically on visualizations created using DR projections, given the aforementioned advantages of this type of visualization technique. As object of study, we consider a typical ML pipeline (blue box in the middle). This consists of a classifier or regressor which, after suitable training, is used to predict values for new data samples. In this thesis, we will mainly focus on ML classifiers. The green arrows at the top of this figure show how visualizations can be used to assist the ML engineering process at multiple phases. In the examples below, we mark in bold the topics which we also address in this thesis:

- Projections can be used to depict the distribution of given labeled samples over a dataset. When insufficient labeled samples exist for training, **pseudolabeling** can be used to create additional labels based on the structure of the projection (with or without direct human input);
- Projections can also be used to gauge the difficulty of a classification problem Rauber et al. (2017b). The degree of **visual separation** on a projection can serve as a simple-to-interpret proxy for the difficulty that a ML model will encounter when separating the data samples into different classes;
- Projections can be used to explore a data space – and in particular understand which features (or feature ranges) are key to separating different groups of samples in a dataset (Broeksema et al., 2013; Coimbra et al., 2015);
- Projections are helpful in understanding how a model evolves during typical training procedures. This way, users can get more insights on the success (or challenges) of a training sequence than when using simple aggregate performance metrics Rauber et al. (2017a);
- Finally, projections can be used to construct dense representations of the behavior of a trained ML classification model that show how the model partitions its input space into **decision** zones separated by **decision boundaries**.

Apart from the above-mentioned added value of high-dimensional visualizations (in particular, projections) for ML engineering, the opposite interaction exists too. That is, ML can be

used to create better visualizations for high-dimensional data. In a nutshell, consider a visualization as a function that transforms (high-dimensional) data to a low-dimensional image. Projections are an excellent example hereof, as they transform n-dimensional sample sets to 2-dimensional scatterplots. We can then use ML to *learn* such a function based on suitably chosen training sets and cost functions which model the types of data we want to visualize, respectively the quality criteria that the resulting visualizations should have. The yellow arrows at the bottom of Fig. 1.1 show how ML can assist the creation of projection-based visualizations. As earlier, we mark in bold the use-cases which we further address in this thesis:

- Supervised ML can be used to generate projections that mimic the appearance of those created by expensive (and/or complex) DR algorithms at a fraction of the cost of such algorithms (Espadoto et al., 2020);
- Metrics used for ML sensitivity analysis can be used to assess the robustness and stability of DR algorithms (Bredius et al., 2022);
- Self-supervised ML can simplify (and generalize) the training of regressors that mimic the behavior of given DR algorithms for cases where supervised data is not available (Espadoto. et al., 2021);
- Quality metrics used to assess the performance of ML models can be adapted to gauge the **quality** of DR projections;
- ML can help creating inverse projections which map from the 2D (visual) space back to the data space (Espadoto et al., 2023; Rodrigues et al., 2019). This allows creating visual interfaces that assist users in **manipulating** the high-dimensional data.

As Fig. 1.1 and the above explanations outline, there are numerous types of synergies between ML and DR. Our thesis will only explore a specific subset of these. To further clarify this, we show next in Fig. 1.2 the workflow for designing, implementing, and evaluating a typical ML classification model which we will next consider in our work. The red-outlined images in this workflow show the utilization of DR visualizations. The blue arrows in the figure represent the typical flow of data throughout an ML engineering pipeline. The green arrows detail the way that DR helps ML engineering – that is, they detail the interactions summarized

by the green arrows in the overview Fig. 1.1. In other words, the process described in Fig. 1.2 is the so-called *visual analytics* (VA) workflow that uses DR visualizations to assist the task of creating high-quality ML models for a given application domain. Below we describe the steps of the ML workflow depicted in Fig. 1.2.

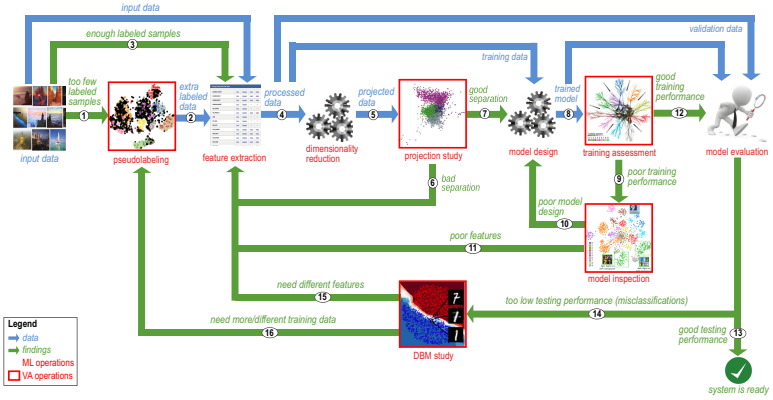


Figure 1.2: Detailed workflow of using DR techniques to assist ML which details the green arrows in the general workflow in Fig. 1.1. Visual analytics (VA) operations enabled by DR are marked by red-outlined boxes. Figure reproduced from Telea et al. (2024).

1. The ML engineering process starts – as usual – with the acquisition of a suitable input dataset with labels. If sufficient labels are available for the envisaged training, the process continues with step (3).

2. If insufficient labels are available for training, pseudolabeling can be used – optionally assisted by DR projections – to generate additional labels based on the existing ones.

3. The total set of labels (present from the input data and generated by pseudolabeling) are used to create the training and test sets for ML engineering.

4. If the data is too high-dimensional to directly train the ML model on it and/or to next visually explore the data, features can be extracted to reduce the complexity of the problem.

5. The data processed by feature extraction is next used to create a 2D projection. This projection, central to our VA workflow, will serve multiple purposes in our pipeline –

visual exploration of the overall distribution of training and test samples; visual exploration of the results of the trained model; and additional generation of pseudolabels.

6. In line with the above, a very first assessment considers how well are same-label point groups separated in the projection. When this visual separation is poor, the data is likely unsuitable to train a high-quality classifier; as such, the workflow can consider extracting different features to enhance this separation.

7. Conversely, if the visual separation is deemed good enough, the workflow continues with the usual architecting and training of the desired ML model.

8. The training of the ML model is next assessed. This can involve typical study of the (convergence of the) training loss over training epochs, for example. However, DR projections can also be used to gain more insight on specific problems encountered during the training.

9. If the training is deemed to be of insufficient quality, DR projections can next be used to investigate the trained model and highlight problems.

10. During the above step, problems related to the design of the model can be identified. In this case, the workflow loops with a model re-design step.

11. Alternatively, problems identified at step (9) can point to a poor quality of the features used by the model. In this case, the workflow loops with a new iteration of the feature extraction step.

12. When training has converged with sufficient performance, the ML model is evaluated using the typical procedure involving (unseen) testing data.

13. The model evaluation generates good-enough performance for the tasks it was designed for. In this case, the entire process ends with a ML model that is ready to be deployed in practical applications.

14. Contrary to the above, the testing step (12) shows that the model exhibits poor performance (poor generalization). In this case, DR projections – and in particular decision boundary maps – can be used to study where, in the data space, generalization problems occur.

15. The above study can yield the conclusion that the generalization problems are not related to a specific area of the data space. In this case, the problem is likely generated due to the feature *space* itself. As such, the feature extraction step (4) needs revisiting to create a better feature space.

16. In contrast, the study in step (14) can point that generalization problems relate to specific *locations* in the feature space. These can be corrected by *e.g.* adding more labeled samples to the training set – a process where pseudolabeling (3) can help.

1.4 RESEARCH QUESTIONS

Following the above considerations, our general research question can be stated quite succinctly as follows:

How can we exploit the synergy between ML and DR techniques to improve each other?

It is, however, evident that the scope and generality of the above-stated research question are too high for a (single) thesis to be able to fully answer it. As such, we further refine this general research question into two separate sub-questions, as follows:

RQ1: How to use multidimensional projections to build better models for machine learning?

RQ2: How does projection quality relate to data separation and classification performance?

Our first research question (RQ1) relates to the observation, already mentioned in Sec. 1.3, that DR projections of high quality capture well many aspects of the so-called data structure, *i.e.*, the relative organization of data samples in a high-dimensional dataset. If and when this is the case, it means that the information that a projection captures can be used as a 'proxy', or substitute, to the original information that the high-dimensional samples themselves captures. In turn, this means that one can use the projection *instead of*, and in places where, one would use the high-dimensional information, and obtain, ideally, very similar – if not better – results. The added value of this approach would be twofold. First, a projection performs, by construction, data reduction – it has, typically, only two dimensions instead of the hundreds (if not more) dimensions of a dataset. As such, it is far more compact, thus *faster* to process from a computational perspective

than the dataset it comes from. Secondly, and even more importantly, this data reduction process can remove much of the complexity and redundancy present in the high-dimensional space. If done with care, this can lead to not just a smaller dataset, but a dataset which is *easier* to process to arrive at the desired results, as compared to the original high-dimensional dataset.

Our second research question (RQ2) relates to the same observation from Sec. 1.3. Let us assume that a projection captures well the so-called data structure by means of the visual separation (VS) it exhibits in its scatterplot. Separately, it is well known from machine learning theory and practice that the performance of a trained ML classification model (classifier performance, or CP) relates to the data separation (DS) present in the dataset it works on. Simply put, it is easier to classify a dataset with high DS to reach a high CP than a dataset with a low DS. This means that all three above-mentioned factors – DS, VS, and CP – are inter-related. If such a relation is true in the sense that high values of one or two of these factors determine a high value of the third one, we can exploit such observations to *measure* and/or *optimize* one of these factors as function of the other two.

1.5 CONTRIBUTIONS

We next present the structure of our thesis and, in the same time, also explain how the various contributions presented in this thesis succeed in answering parts of the research questions RQ1 and RQ2 mentioned in the previous section.

In Chapter 2, we introduce the required background in terms of notations and related work, in both the ML and DR fields, which are required for the reader to follow our contributions presented in the following chapters.

Chapter 3 presents our first contribution which addresses RQ1, namely the utilization of DR techniques to extract useful features from a high-dimensional dataset for creating an end-to-end ML pipeline for a challenging classification problem. In more detail, we show how the (2D) embedded space created by existing DR techniques allows the efficient and effective construction of labeled datasets, starting from a small number of ground-truth labels, which are in turn needed for training the desired ML classification models. Additionally, we show that using this embedded space for pseudolabeling can lead to higher quality results – both in terms of the quality of the assigned labels and the end performance of the trained ML classification models – than using the original high-dimensional data space. We relate our contribution

with Fig. 1.2, in which steps 1, 2, 3, 4, and 5 are performed. We assume that we have a projection with a good visual separation and go straightforward to step 7 and 8, also assuming a good training performance in 12, and evaluate our model in the test set.

Chapter 4 focuses our interest on RQ2, namely the relation between data separation (DS), visual separation (VS), and classifier performance (CP). We present a number of experiments which extend earlier observed relations between DS and VS, respectively between DS and CP, and show how one can preprocess high-dimensional data, using contrastive learning, to achieve a high DS, which in turn leads to a high VS in projections of that data; finally, we use such projections to generate pseudolabels to train classification models which achieve a high CP. Our findings and proposed workflow thus generalize and connect the earlier-known evidence from DR and ML literature concerning the correlation of DS, VS, and CP and show how such correlations can be practically used in DR-assisted ML engineering. From Fig. 1.2, our goal is correlating the above-mentioned terms by evaluating the final result of steps 1, 2, 3, 4, 5, 7, 8, 12 and 13.

Chapter 5 exploits the results presented in Ch. 4 to further answer RQ2. Specifically, if, as observed in our work in Ch. 4, high-VS projections can be used to propagate labels to construct high-CP classification models, we now hypothesize that we can use existing ML quality metrics to measure the visual separation (VS) present in a given projection. We conduct extensive experiments that show how such a ML metric, measured on a wide range of projections and datasets, yields results which correlate well with VS values independently assessed by users on the respective projections. Our work thus enriches the palette of metrics available in DR literature to assess the quality of projections by a new measure which is simple and fast to compute and can be generically applied to any projection technique. Our contribution are shown in Fig. 1.2 by steps 1, 2, 3, and 5, while we use the new measure to support steps 6 and 7.

Chapter 6 revisits RQ1 by studying an additional aspect of the hypothesized added-value of projections for helping the construction of high-performance ML models. Specifically, we now consider the user-in-the-loop scenario by asking subjects to use an interactive projection-based tool to create pseudolabels, based on an existing small set of ground-truth labels and a larger given unlabeled dataset, so as to train a classification model. In this process, we also introduce extensions of the basic DR projections used so far, namely the visual exploration of projection *error met-*

rics and also the exploration of so-called *decision boundary maps which* visualize the behavior of a trained model over larger parts of its input data space than a given training or testing set. Our experiments show that projections, when complemented by the mentioned error and decision boundary map visualizations, are efficient and effective instruments that aid users in performing pseudolabeling to create training sets that increase classification performance even for difficult problems. To our knowledge, our work is the first in the literature where such error maps and decision boundary maps have been formally evaluated with respect to their effectiveness in aiding ML engineering tasks. This chapter relates with Fig. 1.2, by means of steps 1, 2, 3, 4, 5, 7, 8, 12, 14, and 16. Steps 9, 10, 11, and 15 are not in the scope of this thesis.

Finally, Chapter 7 concludes our thesis by revisiting our contributions to solving RQ1 and RQ2 and also outlining open avenues for future research based on the insights obtained during our research.

2 RELATED WORK

2.1 INTRODUCTION

As outlined in Chapter 1, the central research question of this thesis revolves around the synergies that exist between machine learning (ML) and dimensionality reduction (DR) techniques, or how these two classes of techniques can help to support and/or improve each other. To help the discussion of these synergies in the remainder of the thesis, this chapter introduces the required background and related work in MR and DL. Section 2.1 introduces the preliminaries related to ML, in particular in the context of using large datasets. Section 2.2 describes the task of pseudolabeling in the context of training ML models starting with only few ground-truth samples. Section 2.3 describes dimensionality reduction (DR). Section 2.4 discusses techniques for the visual exploration and analysis of ML models, with a focus on decision boundary maps, the particular class of such techniques on which we will focus further in our work.

2.1.1 *Machine learning preliminaries*

Notations: Let $D = \{\mathbf{x_i}\}, 1 \leq i \leq N$ be a dataset of N samples, in which $\mathbf{x} = (x^1, x^2, ..., x^n)$, $x^i \in \mathbb{R}^n$ and $1 \leq i \leq n$, a n-dimensional (nD) real sample. We call the values $x^i, 1 \leq i \leq n$, the dimensions (attributes, variables, or features) of sample $\mathbf{x}$. We can thus view D as a table with N rows (samples) and n columns (dimensions).

We call D *supervised* if there is a pair $(\mathbf{x_i}, c_i) \in D \times C$ for all $\mathbf{x}_i$ in D. The value c_i is called the label of sample $\mathbf{x}_i$. Here, C is the domain of definition of labels, which we discuss further below. In a supervised D, labels c_i are known as *true labels* and are manually assigned by a human or, alternatively, come from other trusted data sources. In an unsupervised dataset D, c_i is unknown. A label $c_i \in C$ can be attributed to a sample $\mathbf{x_i}$ by a so-called labeling process.

Classifiers and regressors: For classification problems, C is a categorical domain. Each label $c \in C$ is also known as a class label – or, more briefly, a class. A classifier for D is a function

$f : D \to C$ that maps data samples to class labels in a supervised fashion. In contrast to the above, regressors $f : D \to C$ have a continuous co-domain, that is, $C \subset \mathbb{R}^m$. In our work, we will mainly focus on the use of classifiers. For brevity of notation, we will also next use the term *labeled dataset* D to refer to a dataset D along with the labels (in C) of its data points.

Classifier training: Training a classifier f is usually an approximation or optimization problem. In more detail: Consider a given labeled dataset D. A well-trained f should, ideally, deliver the so-called ground truth labels c_i for the corresponding samples $\mathbf{x}_i \in D$. However, strictly enforcing the condition $f(\mathbf{x}_i = c_i) | \forall \mathbf{x}_i \in D$ poses several issues. First, computing a function f that strictly obeys this condition can be computationally expensive for datasets D that are large, high-dimensional, and/or have a complex distribution of samples over the embedding space $\mathbb{R}^n$. Secondly, even if this computation were possible, this may not be desirable. Indeed, the function f obtained by the strict enforcement of this condition may have undesirable behavior for additional points $\mathbf{x}_j \notin D$, leading to so-called *generalization problems*.

To tackle both above issues, training is usually done by imposing a weaker constraint of the form

$$err = \frac{1}{|D|} \sum_{\mathbf{x}_i \in D} d(f(\mathbf{x}_i), c_i) = \min, \tag{2.1}$$

where $d : \mathbf{C} \times \mathbf{C} \to \mathbb{R}^+$ is a so-called *cost function* (or loss function) defined over the classifier's co-domain C; and $|D|$ denotes the size of the set D. Using $|D|$ in Eqn. 2.1 has the effect of normalizing the actual cost by the size of the dataset used for evaluating the error, thereby making the former independent on the latter.

The cost function quantifies the difference between obtained (predicted) and real (expected) output – known as *error* – during the training. This allows (a) the construction of f to utilize various computationally effective optimization (minimization) algorithms; (b) the designer of f to choose the form (expression) of this function to *e.g.* balance computational complexity *vs* optimization accuracy; and (c) the training process to stop flexibly once a given maximal error rate *err* has been reached. Optimizing *err* in Eqn. 2.1 is typically done using iterative numerical procedures such as gradient descent or simulated annealing. These require in practice tens up to thousands of iterations, depending on the problem's complexity (difficulty), which in turn is given

by the number of samples in the dataset D; their dimensionality n; their distribution in $\mathbb{R}^n$; and the complexity of f.

The choice of d further allows modeling different scenarios. For classifiers, where C is a categorical dataset, $d(a, b)$ simply measures whether two labels a and b are identical ($d = 0$) or not ($d = 1$). For regressors, d can actually measure the distance between two samples in $\mathbb{R}^m$ using *e.g.* L_1 or L_2 metrics, leading respectively to the well-known mean absolute error (MAE) and mean square error (MSE) cost functions.

Training and testing sets: The main goal of training a classifier f is being able to classify a new set of samples than the ones on which f was trained. To explain this, we need to introduce the notions of training and test sets. A *training set* D_t has the same structure as the (labeled) dataset D described above. D_t yields the set of points used to minimize *err* in Eqn. 2.1 which leads to the computation of f. A *test set* D_T has, also, the same structure as D described above. However, $D_T \cap D_t = \emptyset$, meaning the test and training points are disjoint from each other. The test set D_T is used to evaluate the generalization performance of a trained f. That is, one evaluates *err* (Eqn. 2.1) on D_T, by using the fixed f obtained from training. The obtained error rate indicates how well f generalizes from D_t to D_T.

The values err_t and err_T, obtained respectively from Eqn. 2.1 on the training, respectively test sets D_t and D_T allow assessing several scenarios. When err_t is high, the classifier has not learned sufficiently well during training – a process also known as *underfitting*. In such cases, one typically does not proceed to testing, but reiterates the training with different parameters. When err_t is low, one next studies the test error err_T. If err_T is low, the classifier is said to have successfully *generalized* for new data. If, however, err_T is high, then generalization does not occur. In such cases, f is said to have *overfit* its training data.

Performance measuring: After constructing f by the above-mentioned training, several metrics can be used to refine the evaluation of how well the training process succeeded – that is, apart from directly evaluating Eqn. 2.1.

A commonly used metric is *accuracy*, which counts the number of correct obtained labels (predictions) over the total number of labels (samples). Its computation is given by Eqn. 2.1 with $d(c_i, c_j)$ set to the Kronecker delta function $\delta_{c_i c_j}$. Accuracy values close to 1 mean good classification performance. However, accuracy may not be a good choice in cases in which the dataset D used

for evaluating Eqn. 2.1 is *unbalanced*. An unbalanced dataset D is a dataset in which labels $c_i \in C$ do not present a uniform distribution over the class labels. If, for instance, the number of samples C_i in D of some given class c_i dominates all others, then a naive classifier which simply outputs label c_i with the probability $C_i/|D|$ (and any of the other labels with probabilities which sum up to $1 - C_i/|D|$) will yield good accuracy values.

Cohen's kappa coefficient (κ) (Fleiss and Cohen, 1973) is a metric that can be used to alleviate the classification assessment problem for unbalanced datasets. κ is defined as

$$\kappa = \frac{p_o - p_e}{1 - p_e}, \tag{2.2}$$

where p_o is the simple accuracy, *i.e.*, the number of correctly classified samples (true positives) over $|D|$ samples, and

$$p_e = \frac{1}{|D|^2} \sum_{1 \leq g \leq |C|} n_g^\alpha n_g^\beta, \tag{2.3}$$

where $|C|$ is the number of classes, $|D|$ is the number of samples, and n_g^α and n_g^β are the predicted class g given by the true label α and given label β, respectively. The κ coefficient is in a $[-1, 1]$ range, where $\kappa \leq 0$ means no agreement and $\kappa = 1$ means complete agreement between two classifiers α and β. Instead of comparing classifiers, we can use κ to measure the agreement between two different labels (true and predicted) considering a classifier. Thus, we can measure the probability of agreement between a predicted label and the true label.

In our work in Chapters 3-6, we will use both accuracy and κ for evaluating our various classification models, and comment on the suitability of each of these metrics on a case-by-case basis.

2.1.2 *The need for large datasets in machine learning*

Supervised deep neural networks (DNNs) require large annotated sets for both training and testing models in order to deliver high performance (Lin et al., 2014; Sun et al., 2017). This is inherent to the fact that the function f that a DNN aims to compute has millions (or more) of internal parameters that have to be set by the training optimization. This is not a problem if one avails of extensive collections of ground-truth labels or, alternatively, producing such labels is relatively cheap. However, there are situations when none of these conditions hold. A recurring

example of this situation which we will extensively discuss in our work in Chapters 3-6 involves the classification of biological microscopy images. While the acquisition of such images is a reasonably straightforward process, their annotation – that is, assigning ground-truth labels to the imaged information – is a very expensive process which requires specialist users (microbiologists) to carefully study each image. This quickly becomes a blocker to the scalability of using supervised ML approaches to handle such datasets.

Given its importance and frequency, several strategies aim at tackling this problem of not having sufficient labeled data. In the following, we concern ourselves only with cases where we know all classes C to infer in advance. The problem of discovering new classes in the data – while a valid and important one – is out of our scope.

Few-shot learning: Recently, few-shot learning techniques have shown the ability to deal with the absence of large supervised datasets in classification tasks using DNNs. Using very few supervised samples (roughly speaking, a few tens), few-shot learning guarantees generalization in test sets when training from one to few (one/few-shot learning) or even zero (zero-shot learning) examples per class (Wang et al., 2020). Additional unsupervised samples are not part of standard few-shot learning methods.

Semi-supervised learning: In addition to the (few) available supervised samples, semi-supervised learning considers many unsupervised samples to capture additional information during its learning process (Wang et al., 2020). Recent works have explored combining few-shot learning and semi-supervised learning (Yu et al., 2020; Li and Chao, 2021; Zhmoginov et al., 2022) by considering a fixed amount of unsupervised samples along with the few supervised ones. Aside from that, we are interested in using very few supervised samples and many unsupervised ones, in a semi-supervised setup. When some supervised samples and many unsupervised samples are available, semi-supervised learning can increase the number of labeled training samples and, consequently, improve the performance of a DNN (Lee, 2013). Given its importance to our work, we discuss variants of semi-supervised learning below in Sec. 2.2.

2.2 PSEUDO LABELING

In semi-supervised learning, deep learning (Lee, 2013; Iscen et al., 2019; Miyato et al., 2018; Jing and Tian, 2020; Pham et al., 2021) approaches have been used to propagate labels from a small set of supervised samples to a large set of unsupervised ones exploiting their feature space distribution. At a high level, the overall idea is simple: The learning algorithm extracts training information both from the (few) *supervised* samples available and also from the (usually many more) *unsupervised* samples present in the training set D_t. The scarce label information is effectively 'propagated' over the training set in suitable ways, leading to pseudolabels – that is, labels which have not been assigned by users, but are used exactly as the true labels by a training process. If we assume that the training set accurately captures the distribution of data for the problem at hand, then such information can be sufficient to train high-performance classifier models. We discuss variations of this approach below.

Pseudo labeling – a particular case of self-training – was first proposed for more effectively fine-tuning of a pre-trained model (Lee, 2013). Nevertheless, label propagation errors can negatively affect the classification performance of models trained with pseudo labels (Benato et al., 2018; Arazo et al., 2020). The confidence of the apprentice model was included in the loss function to mitigate such problems (Iscen et al., 2019; Shi et al., 2018). Recently, pseudo labeling approaches (Lee, 2013; Jing and Tian, 2020; Pham et al., 2021) essentially adopt the semi-supervised strategy with the apprentice model assigning uncertain (pseudo) labels to unsupervised samples. Those approaches has been also combined with different strategies, *e.g.*, self-supervised methods (Zhai et al., 2019; Cascante-Bonilla et al., 2020). With the same purpose, meta-pseudo-labeling (Pham et al., 2021) uses an auxiliary model (teacher) to generate pseudo labels to train the primary model (student). Still, to guarantee reasonable label propagation accuracy, such deep-learning-based methods require a training set with hundreds of supervised samples per class and a validation set with additional supervised samples for parameter optimization (Lee, 2013; Miyato et al., 2018; Jing and Tian, 2020; Pham et al., 2021). When only a few supervised samples (e.g., dozens per class) are available, this requirement is a clear obstacle for the deployment of such methods.

Pseudo labeling from extra few supervised data: Among the studies that focus on labeling unsupervised data from a few supervised samples are those based on cluster-then-label methods for classification (Das et al., 2020) and few-shot learning (Wu et al., 2022). Such methods do not use extra (data) supervision for parameter optimization and searching (Das et al., 2020; Wu et al., 2022). They deal with consistency bias by combining (i) a similarity loss and (ii) an ensemble from different Gaussian Mixture Models. However, exploring challenging datasets can be an issue when using unsupervised learning (Benato et al., 2021c) or similarity-based loss functions (Benato et al., 2023b). For such situations, strategies such as co-training, graph-consistency, and uncertainty information combined with limited supervised information can leverage pseudo-labeling approaches to avoid consistency biases (Arazo et al., 2020).

Graph-based pseudo labeling: Pseudo labels stemming from graph-based semi-supervised methods are typically faster to compute than deep learning methods. By modeling training samples as nodes of a graph whose arcs connect adjacent samples in a given feature space, one can propagate labels from supervised samples to their most strongly connected unsupervised ones. Such methods combine the connectivity aspect of graph-based methods with the learning ability of DNNs in a co-training strategy (Iscen et al., 2019; Arazo et al., 2020; Amorim et al., 2019). However, most graph-based label propagation methods need separate parameter tuning for distinct datasets. They also need a validation set with extra supervised samples. Again, the requirement of extra supervised samples is an obstacle for the deployment of such methods when only very few supervised samples are available.

Optimum path forest pseudo labeling: A particular graph-based algorithm that does not require any parameters to optimize and, also, no extra supervised set to validation is the semi-supervised Optimum Path Forest (OPFSemi) (Amorim et al., 2016) classifier. This algorithm maps both labeled and unlabeled samples to nodes of a complete graph, with edges weighted by the Euclidean distance between samples in a given feature space. The labeled samples are taken as prototypes that compete among themselves for the unlabeled ones. Each prototype 'conquers' its most closely connected unlabeled samples by offering minimum-cost paths and assigning its label to them. As a path-cost function, OPFSemi uses the maximal edge-weight along the path. By that,

OPFSemi computes a minimum-cost path forest rooted at the prototype set. The time complexity of this algorithm is $O(m^2)$ for m nodes, since the graph is complete. However, it is possible to precompute a minimum-spanning tree in $O(m^2)$ time and next perform label propagation (optimum-path forest computation) on this tree in $O(m \log m)$ time, respectively, for any randomly chosen set of prototypes. As the process is calculated over a complete graph with all samples in D, we argue that OPFSemi can capture local and global information of data distribution, instead of local information only. The ability of OPFSemi for pseudo labeling was investigated in (Benato et al., 2018; Amorim et al., 2019), showing that it can surpasses many other compared methods. We further analyze the performance of OPFSemi in our work in Chapters 3 and 4.

Embedded pseudo labeling: As explained in Chapter 1, multidimensional projections can capture well the structure of a high-dimensional dataset. This suggests the potential of executing a pseudolabeling algorithm over the embedded (2D) space of a projection obtained from a high-dimensional dataset rather than over the dataset itself. If, as DR literature claims, the projection captures well the data structure, then pseudolabeling applied to the 2D space may be better (in terms of speed but, hopefully, also in terms of quality) than pseudolabeling applied to the high-dimensional space. Benato et al. (2018) explored this hypothesis for the first time, to our knowledge. The authors showed that *manual* label propagation, performed on 2D projected (embedded) spaces computed by t-SNE, leads to better results than *automatic* label propagation, performed in a latent space with hundreds to one thousand dimensions generated by an autoencoder from the raw input data (images, in this case). Figure 2.1 illustrates this manual labeling process.

Semi-automatic pseudo labeling: As discussed above, there seem to be two distinct approaches to pseudo labeling. That is, one can use fully *automatic* methods for performing this task, such as described in (Amorim et al., 2016); or one can put the user in charge of the process to *manually* execute the labeling Benato et al. (2018). As shown by the respective works, the two approaches appear to have complementary advantages (and limitations). Automatic pseudo labeling is, of course, cheaper and simpler to execute than involving the user. However, as shown by Benato et al. (2018), manual labeling, done in the 2D projection space, can result in higher performance than automatic labeling,

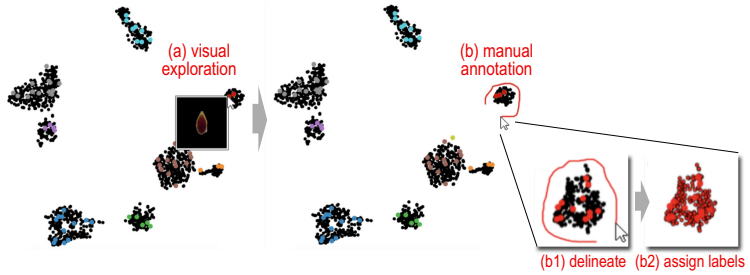


Figure 2.1: Interactive label propagation procedure for user-assisted pseudolabeling. The left image shows a given high-dimensional dataset projected to 2D using t-SNE. Colors represent ground-truth labels (black: unlabeled points). The user studies this projection to find a well-separated sample group containing a few same-label points, optionally using tooltips to view individual samples (a). Next, the user decides to select a sample group to manually label (b). For this, the desired group is encircled by manual delineation (b1). Next, the ground-truth label present in this group – the red one, in our case – is assigned to all delineated points (b2). Figure adapted from Benato et al. (2018).

even if the latter has access to the entire high-dimensional data space.

Based on these observations, Benato et al. (2021c) proposed next to *combine* the two approaches, namely automatic and user-driven labeling. In more detail, they first compared automatic labeling done in the latent feature space with the one done in the 2D projected space. In this context, the combination of t-SNE (for projection) and OPFSemi (for labeling) was considered for the first time. They found that automatic labeling performed over the 2D projected space yields better performance than the one performed in the higher dimensional spaces. This partly explains why users obtained a high performance when performing manual labeling in the same projection space (Benato et al., 2018). If both automatic and user-driven labeling work well in such a 2D projection space, this strongly suggests that the respective space preserves all the data structure needed to propagate labels effectively.

Benato et al. (2021c) next took an additional step, by actually combining automatic and manual labeling, both in the 2D projection space, as follows. Given a training set with few supervised samples and many unsupervised ones, an unsupervised autoencoder extracts features from the input dataset. Then, encoded fea-

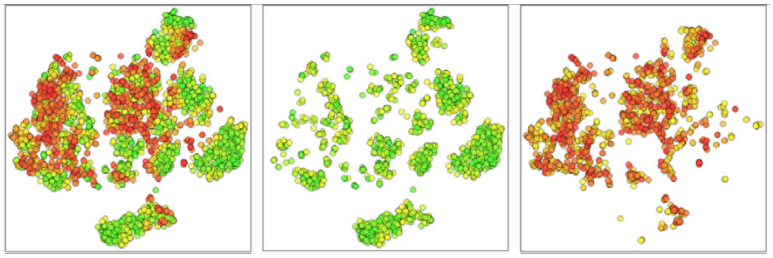


Figure 2.2: Projection of MNIST dataset, created using t-SNE, colored by classifier's confidence values. Left: All samples colored by classifier's confidence, from low (red) to high (green) confidence values. Middle: Only higher confidence samples. Right: Only low confidence samples. Samples are split in low *vs* high confidence based on a user-provided threshold. Image adapted from Benato et al. (2021c).

tures are projected to a 2D space by t-SNE. During this process, one also computes the so-called confidence of labeling, which is the probability of a classifier assigning a label to a given sample. The OPFSemi algorithm is used pseudo label the high-confidence unsupervised samples to increase the labeled set size with little effort and high quality. The low-confidence samples, in contrast, are skipped in this automatic process, and are offered to the user, who can examine them in detail, and manually label a desired subset of them, as in (Benato et al., 2018).

Figure 2.2 shows an example of a 2D projection scatterplot colored by the classifier's confidence. Based on a user-given threshold, one can separate high-confidence samples from the rest (middle image). These samples are next automatically labeled by OPFSemi. The remaining low-confidence samples (right image) are offered to the user for manual labeling. Figure 2.3 shows the labels that result from this combined approach. Starting from the supervised samples (colored points with a red border), automatic labeling is done towards the high-confidence points. This generates the light-colored points (drawn without a border). The remaining low-confidence points (black) are next offered to the user for manual labeling.

This combined pipeline showed better results than the fully-automatic and fully-manual labeling pipelines. Indeed, for the 'evident' labeling cases (high confidence), an automatic method is more efficient than asking users to manually label them; conversely, for the 'tricky' cases (low confidence), it is better to involve the user in the labeling decision than assuming an automatic algorithm can handle them. However, several questions re-

main open. First, it is unclear which type of 2D projection (apart from t-SNE) would be suitable for performing the proposed label propagation. We address this question in our work in Chapter 4. Secondly, it is unclear how the manual labeling can be further assisted (to make it faster and/or yielding better quality labels) besides showing a 2D scatterplot colored by confidence. We address this question in Chapter 6. Thirdly, the semi-automatic pseudo-labeling performance directly depends on the first learned feature space – the one provided by the autoencoder in an unsupervised way. We address this in Chapter 3 by including the few supervised information in the training and improving the feature learning over the iterations. Fourthly, the user-based labeling is performed by a single user interaction with the 2D scatterplot. In other words, there is no user in the looping of the classifier learning as in an active learning procedure. We address this In Chapter 6 by including the user in the looping of the classifier by exploring 2D projections to improve the pseudo labels assigned by the automatic pseudo labeling technique.

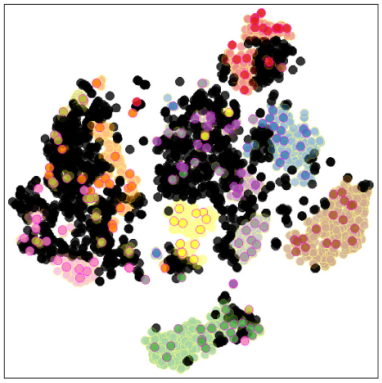


Figure 2.3: Example of semi-automatic labeled samples. Automatic labeling is performed from the supervised samples (points colored by class, saturated colors, red border) to the unsupervised and high-confidence ones (light colors, no border). Interactive labeling is realized on remaining low-confident samples (black). Image reproduced from Benato et al. (2021c).

Apart from the above, we also present additional contributions in our work, as follows. In Chapter 3, we further explore these findings from Benato et al. (2018) by using a variety of architectures to extract latent features from raw image data; pre-trained architectures for the same task; and various refinements to the basic OPFSemi label propagation presented in the initial paper.

Separately, we also study the factors which influence the success of applying OPFSemi to such embedded spaces as a function of the data separation in the feature space and the obtained visual separation in the projection space (Chapter 4).

2.3 VISUALIZING HIGH-DIMENSIONAL DATA

Multidimensional data occurs in many fields as science, engineering, medicine, and ML. As introduced in Chapter 1, such data is the object of study for both machine learning (ML) and data visualization (VIS) methods. In the last decades, many visualization methods have been proposed to depict multidimensional data – the most notable being table lenses, scatterplot matrices, parallel coordinate plots, and dimensionality reduction (DR) methods. We briefly overview these next and advocate our choice, in the rest of this thesis, for using DR techniques for the visual exploration of multidimensional data. We split this evaluation into dimension mapping techniques (Sec. 2.3.1), respectively dimension synthesis techniques (Sec. 2.3.2).

2.3.1 *Dimension mapping techniques*

Dimension mapping techniques – as their name suggests – essentially map (a subset of) the existing n dimensions of a given dataset D to so-called visual variables (Bertin, 1977, 1983). Simply put, this means that for each of the selected dimensions $n' < n$ to be displayed, a dimension of the visualization space is assigned. Next, the respective values of the data dimension (of the n' ones to map) are mapped to the respective values of the assigned visual variable. For the choice of visual variables to use in this process, one typically prefers to use first the available *spatial* variables – that is, the x and y axes of a 2D display and, if 3D visualizations are to be used, the z (depth) dimension. Additional visual variables that are used are – in order of preference – color, point size, and labels. During this process, so-called *dimension overload* can be used. That is, several data dimensions can be assigned to the same visual variable, with suitable shifting to avoid perfect overlap. Examples of this process discussed below are table lenses, parallel coordinates, and SPLOMs.

Given this one-to-one mapping, dimension mapping techniques can – as we shall next see – only handle a relatively limited number of data dimensions. As such, they are often used together with *dimension selection* techniques, which pre-select a

(small) number of data dimensions $n' \leq n$ to be visualized. Dimension synthesis techniques, discussed next in Sec. 2.3.2 avoid this selection phase.

We identify and discuss the following key dimension mapping techniques.

Table lenses use the inherent spatial layout implied by a table with N rows each having a sample (observation) with n columns (Rao and Card, 1994; Telea, 2006). They literally display the said table using a Cartesian layout. To increase visual scalability, the table is 'zoomed out' until, in a first instance, every table row becomes a pixel row where colors or bar lengths are used to depict the cell values. This process can continue further by aggregating multiple (up to hundreds) of consecutive table rows in a single pixel row. The obtained visualization can be further explored from different angles by sorting and/or clustering (ranges of) values along one or several columns and adding details-on-demand in the form of tooltips or zoom-in 'lenses' that depict selected row ranges. The key advantage of table lenses is its simple layout – most if not all targeted users are familiar with tabular data views. Table lenses are also quite scalable in the sample (row) count N.

However, such techniques do not scale well in the dimension (column) count n. Equally importantly, they only support exploration tasks which assume that samples can be *ordered* in some way. For instance, the arguably simple task of finding how many groups of similar samples exist in a dataset cannot be (easily) addressed by these techniques. Last but not least, table lenses work best for so-called *tabular data*. In information visualization, this term refers to data whose individual dimensions carry well-understood meaning by its users – such as, for instance, name, ID, age, and salary for a personnel record. In ML, one often has to work with data which does not comply with such assumptions – think, for instance, of features extracted by some latent space generation algorithm such an autoencoder; or features that correspond to the colors of individual pixels in an image. While, technically speaking, such data *can* be manipulated as a table, its columns have no intuitive meaning and are too many for individual manual exploration.

Parallel coordinate plots (PCPs) use a visual metaphor similar in many senses to table lenses (Inselberg and Dimsdale, 1990). Every dimension n of the input data gets allocated a (vertical) axis a_i to depict its values. All these axes $a_1, \ldots, a_n$ are depicted

as parallel equidistant lines. Every data point $\mathbf{x}_i \in \mathbb{R}^n$ is next depicted as a polyline that connects its specific values depicted as dots along the n axes a_i. Additionally, one extra dimension can be depicted by color coding the resulting polylines. PCPs are a quite effective instrument to depict the distribution of values in a dataset along each of its n dimensions – these become visible in terms of one-dimensional point spreads along the vertical axes a_i. Outlier values along any axis a_i appear also quite saliently in this type of visualization as polylines having outlier points along the respective axes. Large amounts of similar points are easy to identify as they create densely-packed polylines. More interestingly, direct and inverse correlations of dimensions whose axes are neighbor create visually salient patterns formed by roughly parallel line segments, respectively line segments intersecting in an X-like pattern.

Yet, PCPs suffer from the same scalability limitation as table lenses in the dimension count n as they need to allocate separate space for drawing each of their axes. Like scatterplot matrices (discussed below), individual samples are depicted as relatively complex visual objects – in this case, polylines. Finding correlations (or the lack thereof) requires one to have the dimensions to be tested *neighbor* to each other in the PCP. While this can be achieved by interactive re-arrangement of the axes, this requires a certain manual effort. Moreover, users need to explicitly perform such rearrangements before they can test for (the lack of) correlations. Note that such effort is not required for table lenses. Last but not least, PCPs are quite prone to generate a significant amount of visual clutter – much more than any of the other visualization techniques we analyze in this section. As such, they also have limited scalability in the number of samples N.

Scatterplots are, likely, one of the most frequently used technique for studying (quantitative) multidimensional data. In their standard use, one considers a by-variate dataset $D = \{\mathbf{x}_i\}$ with $\mathbf{x}_i \in \mathbb{R}^2$. Each point $\mathbf{x}_i$ is then plotted in a Cartesian framework according to its two data values to yield a scatterplot for the entire dataset D. Additional dimensions can be encoded, for each plotted point, in its color, size, and (up to some extents) label. This way, scatterplots can depict $n = 5$ data dimensions. If a third spatial coordinate is used, scatterplots can depict a total of $n = 6$ data dimensions. However, 3D scatterplots create additional interpretation problems such as choosing a good viewpoint to examine them from (Sedlmair et al., 2013). Visual

cues that assist interactive viewpoint selection can help such navigation (Coimbra et al., 2015; Zhai et al., 2022). However, even with such tools, exploring 3D scatterplots remains a challenging proposal. As such, we do not consider 3D scatterplots further in our work.

Scatterplot matrices (SPLOMS) are one of the extensions of basic 2D scatterplots that aim to alleviate their limited ability to show many data dimensions n. Simply put, a SPLOM uses a small-multiple visualization metaphor (Becker et al., 1996; Tufte, 2001) to show a 2D scatterplot for all the combinations of dimensions d_i and d_j, where $1 \leq d_i \leq n$, $1 \leq d_j \leq n$, and $d_i \neq d_j$. These plots are arranged in a matrix with plots on the same row or column sharing one common data dimension. SPLOMs have been extensively used in different fields of science under different names – for example, in astronomy, they are known as Tinsley diagrams (Tinsley, 1980). The key advantage of SPLOMs is allowing one to easily spot (direct and/or inverse) correlations between pairs of variables; and, up to a lesser extent, the presence of clusters of similar samples due to the values of a few dimensions.

However, SPLOMs have some important limitations. First and foremost, SPLOMs cannot depict more than roughly $n = 10$ dimensions as the matrix of 2D scatterplots they use grows quadratically with the data dimensionality n. Secondly, they only allow direct exploration of patterns formed by *pairs* of dimensions. If one aims to find how three or more dimensions interact, one needs to mentally compare several scatterplot cells in the SPLOM to detect possible patterns. For this reason, SPLOMs have been called 'multiple *bivariate* visualizations' (Hand et al., 2001). Finally, the fact that every data sample is essentially encoded as n^2 different *points* – one in each 2D scatterplot in the SPLOM – can be confusing, even with aiding mechanisms such as linking selection and highlighting between the scatterplots or, when considering more advanced techniques, morphing and animation between selected scatterplots (Elmqvist et al., 2008).

Scagnostics are a family of techniques that aims to alleviate the quadratic increase of SPLOMS with the dimension count n (Tukey and Tukey, 1988; Wilkinson et al., 2005b; Lehmann et al., 2012; Flores et al., 2017). The key idea is to generate all these n^2 scatterplots; then use (typically unsupervised) ML techniques to analyze all these scatterplots for the presence of patterns deemed to be interesting for a given application domain, *e.g.*, compact point

clusters, direct or inverse correlations, outliers, or clusters having specific shapes. Next, a few scatterplots that score highest along such measures of interestingness are selected and displayed for the user's perusal.

From a ML perspective, scagnostics can be seen as a *feature selection* technique type. Indeed, given the n^2 feature (dimension) combinations available for exploration, only a small set of such pairs is selected (by ML analysis) for further visual exploration. However, it is well known from ML literature that feature selection cannot, in general, yield optimal feature sets for the characterization of a complex dataset. Indeed, for such tasks, one actually needs not only to eliminate some features, but *combine* ranges of multiple ones. DR techniques, discussed below, achieve essentially this goal.

2.3.2 *Dimension synthesis techniques*

In contrast to the dimension mapping techniques discussed above, dimension synthesis techniques avoid the one-to-one mapping process between data dimensions and visual variables. Rather, they proceed by generating, or *synthesizing*, the values of visual variables based on the available data dimensions. The key advantage of this approach is that *many* data dimensions can be combined to determine the values of the few available visual variables. This alleviates the key limitation of dimension mapping techniques discussed above.

Formally speaking, dimension synthesis methods can be described as functions

$$f : \mathbb{R}^n \to \mathbb{R}^v, \tag{2.4}$$

where n gives the data dimensionality and v is the number of visual variables used for visualization. As explained earlier for dimension mapping methods (Sec. 2.3.2), v usually takes a value of 2 to 6, depending on whether we use two or three spatial dimensions in the visualization; and color, point size, and optionally text labels for depicting additional data.

Denote now $f = (f_1, \ldots, f_v)$ the expression of the v-dimensional function f in Eqn. 2.4. That is, $f_i : \mathbb{R}^n \to \mathbb{R}$ tells how the n data dimensions are used to synthesize the i^{th} visual variable. Note that, when $f_i(\mathbf{x}) = x^j$, $1 \leq j \leq n$, the above yields a dimension mapping method that maps data dimension j to visual variable i.

We next discuss the main types of dimension synthesis methods known in the literature.

Projection pursuit methods generalize the concept of scagnostics discussed earlier in Sec. 2.3.1). Consider for simplicity $v = 2$ (our target visualization is a 2D scatterplot). Let $\mathbf{n} \in \mathbb{R}^n$ be an unit vector indicating a direction in the data space. Let $\mathbf{o} \in \mathbb{R}^n$ be the centroid of the points in a given dataset D. We can then define the 2D scatterplot

$$S(D, \mathbf{n}) = \{(\mathbf{x} - \mathbf{o}) - ((\mathbf{x} - \mathbf{o}) \cdot \mathbf{n})\mathbf{n} | \mathbf{x} \in D\}, \tag{2.5}$$

that is, the projections of all points in a dataset D to the plane passing through $\mathbf{o}$ and having the normal vector $\mathbf{n}$. In this case, the two dimensions of the resulting scatterplot are a combination of all n data dimensions, given the dot product $(\mathbf{x} - \mathbf{o}) \cdot \mathbf{n}$.

One can next rotate $\mathbf{n}$ in the data space, construct $S(D, \mathbf{n})$ for various such values, and select the one(s) to be shown to the user based on an interestingness factor, much like when using scagnostics (Sec. 2.3.1). The difference with scagnostics is that, for projection pursuit, the space of possibilities to examine is not much larger – all orientations of $\mathbf{n}$ as opposed to all combinations of two dimensions from the n data ones.

Several implementations exist for projection pursuit depending on the definition of what an 'interesting' scatterplot P is, as well as on how to search the direction space $\mathbf{n}$ to find these (Friedman and Tukey, 1974; Jones and Sibson, 1987). In general, such methods are quite computationally expensive. Moreover, they have a linear and global nature – that is, they project *all* data points on a single *plane*. As we shall next see, DR methods generalize such techniques and also offer additional advantages.

2.3.3 *Multidimensional projections*

As outlined so far, visually depicting data having a large number of dimensions n is very challenging. In this respect, dimension synthesis methods have, it seems, an advantage on dimension mapping methods.

Dimensionality reduction (DR) algorithms, also called *projections*, are to date the methods of choice for this task. Projections take a dataset D and produce a scatterplot, or embedding of D, $P(D) = \{\mathbf{y}_i = P(\mathbf{x}_i) | \mathbf{y}_i \in \mathbb{R}^v\}$, where typically $v \in \{2, 3\}$. Without loss of generality, we next consider $v = 2$; that is, we project data to 2D scatterplots. We exclude 3D scatterplots from our research due to the aforementioned limitations entailing finding

good viewpoints (see Sec. 2.3.1). However, we will use additional visual variables – most notably color – to encode additional data dimensions which are not subject to the DR process.

DR algorithms fully follow the dimension synthesis model outlined by Eqn. 2.4. In technical terms, they differ in choices made for this synthesis process. For example, *linear* methods use linear functions f_i for the synthesis – and conversely for non-linear methods. Separately, *global* methods propose a single definition for each f_i that applies to all the points $\mathbf{x} \in D$. In contrast, local methods propose different definitions for f_i that apply to point subsets of D. Global and linear methods include Principal Component Analysis (PCA); local and non-linear methods include t-Stochastic Neighbor Embedding (t-SNE). Both methods are discussed further below.

Many projection methods have been proposed for P, using different underlying techniques as graphs, linear algebra, stochastic optimization, and deep learning, to mention just a few. We refer the interesting reader to recent surveys that describe these both qualitatively and quantitatively (Nonato and Aupetit, 2018; Espadoto et al., 2019a; Xie et al., 2017; Sorzano et al., 2014; Cunningham and Ghahramani, 2015; Burges, 2010; Engel et al., 2012; van der Maaten et al., 2009). From these surveys, several key points emerge, as follows:

- Multiple criteria: A 'good' DR method has to comply, ideally, with a large number of competing criteria. Without being exhaustive, these include computational speed (being scalable in both the sample count N and dimension count n), robustness to small changes in the input data, out of sample ability (being able to project additional samples atop of those used to generate a given projection), high quality (a criterion we examine separately below given its importance), and ease of implementation and use (especially with respect to hyperparameter settings);

- No absolute winner: There is no single DR method which optimally complies with all above-mentioned criteria. For example, Principal Component analysis (PCA) scores very high on all criteria except quality Jolliffe (1986); in contrast, t-Stochastic Neighbor Embedding (t-SNE) scores highest of all analyzed methods on quality, but quite poorly on scalability, stability, out of sample ability, and ease of implementation and use van der Maaten and Hinton (2008); Espadoto et al. (2019a).

- Conflicting requirements: Depending on the actual application, certain requirements (of the ones mentioned above) may range anywhere from critical to potentially nice-to-have. For instance, for visual exploration, speed, ease of use, and stability are essential; for use in ML pipelines, quality is paramount. We discuss such aspects in further detail below.

2.3.4 *Quality of projections*

Quality of projections was already named as a key criterion for a good DR technique. However, what makes such a technique of *good quality*? Surprisingly enough, there is no formal and universally accepted definition hereof. Virtually all DR literature surveys (mentioned above) globally state that a projection should preserve the so-called *data structure*. Informally put, this means that the scatterplot $P(D)$ should share properties – defined in terms of its 2D points – that can be computed as well on the corresponding data points from D. Further analyzing this topic, we find that this 'data structure' is captured in terms of various metrics. In detail, a projection-quality metric

$$M(D, P(D)) \in \mathbb{R}+ \tag{2.6}$$

evaluates how well a 2D scatterplot $P(D)$ preserves certain aspects that are measurable in the dataset D that the scatterplot was created from.

Several such metrics have been proposed in the DR literature to capture specific measurable aspects of D and $P(D)$. We discuss the most frequently used such metrics in the following.

2.3.4.1 *Classical quality metrics*

Classical quality metrics include scalar metrics, point-pair metrics, and local metrics (Espadoto et al., 2019a). Four scalar metrics frequently used in DR literature (Nonato and Aupetit, 2018) are described below. All these metrics range between 0 (worst case) and 1 (best case).

Trustworthiness (T) (Venna and Kaski, 2006): measures the fraction of points in D that are also close in $P(D)$ or how local visual patterns in a projection truly represent actual data patterns. This is related to the so-called false neighbors of a projected point (Martins et al., 2014). In the definition of T (Eqn. 2.7), $U_i^{(K)}$ is the set of points that are among the K nearest neighbors of

point i in 2D but not among the K nearest neighbors of point i in D; and $r(i,j)$ is the rank of the 2D point j in the ordered-set of nearest neighbors of i in 2D.

$$T(\mathbf{x},\mathbf{y}) = 1 - \frac{2}{NK(2n-3K-1)} \sum_{i=1}^{N} \sum_{j \in U^{(K)}_i} (r(i,j) - K) \tag{2.7}$$

Continuity (C) (Venna and Kaski, 2006): measures the fraction of points in $P(D)$ that are also close in D. This is related to the missing neighbors of a projected point (Martins et al., 2014). In the definition of C (Eqn. 2.8), $V_i^{(K)}$ is the set of points that are among the K nearest neighbors of point i in D but not among the K nearest neighbors in 2D; and $\hat{r}(i,j)$ is the rank of the $\mathbb{R}^n$ point j in the ordered-set of nearest neighbors of i in D.

$$C(\mathbf{x},\mathbf{y}) = 1 - \frac{2}{NK(2n-3K-1)} \sum_{i=1}^{N} \sum_{j \in V^{(K)}_i} (\hat{r}(i,j) - K) \tag{2.8}$$

Trustworthiness (T) and continuity (C) are typically used together. Indeed, when considered together, they essentially tell how close the sets of k nearest neighbors of a point in D, respectively $P(D)$, are to each other. An ideal projection should perfectly preserve such neighbors, *i.e.*, have maximal T and C values. When this happens, one can fully reason about the closest points $\mathbf{x}'$ to a data point $\mathbf{x}$ by examining the closest projections $\mathbf{y}'$ to $P(\mathbf{x})$.

Normalized stress (S) (Joia et al., 2011): measures the preservation of point-pairwise distances from D to $P(D)$ (see Eqn. 2.9). Euclidean distance is commonly the most used.

$$S(\mathbf{x},\mathbf{y}) = \frac{\sum_{ij} (\Delta^n(\mathbf{x_i},\mathbf{x_j}) - \Delta^q(P(\mathbf{x_i}),P(\mathbf{x_j})))^2}{\sum_{ij} \Delta^n(\mathbf{x_i},\mathbf{x_j})^2} \tag{2.9}$$

Ideally, a projection should have $S = 0$. To ease comparison with the other metrics, $1 - S$ is typically used – so that large values indicate a good projection and low values a poor one, respectively. As normalized stress (S) measures how well inter-point distances in $P(D)$ are proportional to the corresponding distances in D, a low-stress projection means that one can reason about the distances in data space by simply considering the visible distances in projection space.

Neighborhood hit (N) (Paulovich et al., 2008): measures the fraction of the K neighbors $N_i^{(}K)$ of a point i in $P(D)$ that have the same label l as point i, averaged over all points in $P(D)$ (see Eqn. 2.10). This is related to the labeled separation in a projection $P(D)$.

$$N(\mathbf{y}) = \sum_{i=1}^{N} \frac{|j \in N_i^{(K)} : l_j = l_i|}{NK} \tag{2.10}$$

Neighborhood hit (N), by definition, is only applicable to labeled data. The intuition behind this metric requires further explanation. Let us assume that, in a labeled dataset D, close points typically share the same label. While this is not guaranteed in general, this is a common assumption in many datasets used in ML, where data (including labels) comes from some continuous phenomenon. In this case, we assume that labels do not abruptly change as data slowly changes. When this is the case, a good projection is expected to show the same behavior. That is, close points in the projection (which map close points in the data space) should typically have the same labels. This is precisely the property that N aims to measure.

Scalar metrics characterize the quality of an entire projection $P(D)$ by a single value, so they are simple to interpret. However, this inherently averages quality over different parts of $P(D)$ and/or D. Point-pair metrics, *e.g.* the Shepard diagram of pairwise point distances (Joia et al., 2011), and local metrics, *e.g.* missing and false neighbor plots (Martins et al., 2014) offer finer-grained quality characterizations. These metrics are typically used to create visualizations of the quality distribution over a projection. Their main added value is to allow users to separate *parts* of a projection which they can trust (since they have relatively high quality values) from other parts which are less trustworthy (since they have relatively low quality values). We will further explore such local quality metrics in Chapter 6.

2.3.4.2 *Visual perception metrics*

Visual perception metrics have been used to assess the visual perception of different patterns present in projections.

Among them, approaches based on clustering, such as the Silhouette score, explore centroids and labels to assess group separation. Other clustering-based approaches combine information from nD and 2D spaces with labels to gauge visual perception (Marghescu, 2006). Class consistency (Sips et al., 2009) and

distance consistency (Tatu et al., 2010a) measures assess class separation via distances from defined centroids. Both combine density functions and local neighborhoods to identify class overlap. Although pseudo labels can be used as a strategy with such metrics, they still rely on suitably chosen and parameterized clustering techniques and probability density models and can have difficulties detecting (and characterizing) clusters of complex shapes — the Swiss roll dataset is a famous example. Separately, Abbas et al. (2019) explored Gaussian mixture models to measure clustering in monochrome scatterplots, but without taking into account labels. Sedlmair et al. (2012) compared cluster separability measures and human observations and concluded that grouping measures might fail to capture multiple sub-groups and groups of different sizes, shapes, and densities. Sedlmair and Aupetit (2015) used fifteen metrics and user judgment to analyze visual separability in 2D scatterplots. The authors found that the distance consistency metric (Tatu et al., 2010a) led to the best agreement with human judgement, but can vary across synthetic and real-world data scenarios. They evaluated their results using only the AUC metric, which can be affected by class unbalances. To circumvent problems related to clustering-based approaches, solutions based on graphs and minimum-spanning trees have been proposed. Wilkinson et al. (2005a); Wilkinson and Wills (2008) proposed methods to find patterns in large scatter plot matrices. Separately, Motta et al. (2015) evaluated original and projected spaces for the same purpose. The benefits of such methods include covering global and non-trivial shapes, being parameter-free, and fast to compute. However, these studies did not explore graph-based approaches in a pseudo labeling task to evaluate projections – which is our proposal. Also, they did not compare their methods in a wide experimental setup with well-known projections and datasets as we will be doing.

Human judgment has also been explored in user studies to evaluate the relation between the above-mentioned metrics and visual perception. An important contribution of these studies is designing a method to conduct the experiments and avoid hundreds of scatterplots that have to be inspected by users (Tatu et al., 2009; Sedlmair and Aupetit, 2015; Albuquerque et al., 2011). For this, scatterplots are ranked from the best to the worst, and only the top three to five are offered for user inspection (Tatu et al., 2009, 2010b). We also use this ranking in our experiments (Sec. 5.4.2). The above-mentioned studies do not use many combinations of datasets and projections. Rather, many (dozens) of metrics are compared (or a new one is proposed) for a sin-

gle (Albuquerque et al., 2011; Sedlmair and Aupetit, 2015) or a couple of datasets (Tatu et al., 2009, 2010b). Additionally, the analyzed metrics still have the main issues that we outlined before (see also (Tatu et al., 2010b)). In contrast to the above, in Chapter 5 we evaluate *many* (hundreds of) dataset-projection technique combinations, both quantitatively and by a user study.

2.3.5 *Inverse projections*

As explained above, projections P map points from a high-dimensional data space $\mathbb{R}^n$ to a (much) lower dimensional space $\mathbb{R}^v$, where typically $v \in \{2,3\}$. Given their operation, one can quite naturally wonder whether an inverse mapping – from $\mathbb{R}^v$ to $\mathbb{R}^n$ is useful and, if so, how such a mapping can be constructed. We detail both above questions in the following.

An inverse projection $P^{-1} : \mathbb{R}^v \rightarrow \mathbb{R}^n$ is implicitly defined as a function that aims to 'revert' the effect of a given direct projection mapping P. In the ideal case, given a dataset D that was projected to $P(D)$ using some user-selected projection function P, a good inverse projection obeys the condition $P^{-1}(P(\mathbf{x}_i)) = \mathbf{x}_i$, for every $\mathbf{x}_i \in D$. Key to the added value of inverse projections is that P^{-1} can be applied also to points *outside* $P(D)$. That is, in practice, one can select *any* point $\mathbf{y}$ in the 2D space where $P(D)$ lives and compute a corresponding data point $P^{-1}(\mathbf{y})$. Ideally, if P were applied to this data point, one would obtain the 2D point $\mathbf{y}$, that is, $P(P^{-1}(\mathbf{y})) = \mathbf{y}$, for all $\mathbf{y}$ in the considered $\mathbb{R}^v$ space.

While the concept of inverse projection is quite simple, it comes with several theoretical and practical challenges, as follows.

First and foremost, in most cases, it is not possible to compute a mathematical inverse of a given projection function P. Indeed, many projection functions are not injective, *i.e.*, they yield the same value $P(\mathbf{x})$ for different data values $\mathbf{x}$. As such, the condition $P^{-1}(P(x)) = \mathbf{x}$ has to be relaxed in practice, typically by aiming that $P^{-1}(P(x))$ is *close* to $\mathbf{x}$. This is similar to how ML models are trained to optimize, but not overfit, a given cost function (Sec. 2.1.1).

Secondly, we have to note that many projection techniques are *non-parametric* – t-SNE being a famous example hereof. That is, rather than providing a mapping from any point in $\mathbf{R}^n$ to a corresponding point in $\mathbb{R}^v$, they only map a *finite* set of points $\mathbf{x} \in D$ in some given dataset D to the corresponding points in $P(D)$. This means that we do not know what the projection $P(\mathbf{x}')$ would be for a point $\mathbf{x}' \in \mathbb{R}^n \setminus D$. As a consequence, we cannot design P^{-1} by directly optimizing for $P(P^{-1}(\mathbf{y})) = \mathbf{y}$ for all $\mathbf{y} \in \mathbb{R}^v$.

However, while P can only be known at the discrete set of points $P(D)$, we want to be able to evaluate P^{-1} *everywhere* over $\mathbb{R}^v$, in particular outside the set $P(D)$.

Considering the above, a practical definition of an inverse projection P^{-1}, given a direct projection $P(D)$ obtained from some dataset D and a projection algorithm D, is a function $P^{-1} : \mathbb{R}^v \rightarrow \mathbb{R}^n$ that

1. Minimizes the cost $\sum_{\mathbf{x}_i \in D} d(P^{-1}(P(x_i)) - \mathbf{x}_i)$, for some suitably chosen distance metric d;
2. Smoothly varies as its input $\mathbf{y}$ changes over $\mathbb{R}^v$.

Several techniques have been proposed to construct inverse projections based on the above model. Early on, autoencoders (AEs) have been used to this end based on the minimization of the reconstruction error (Hinton and Salakhutdinov, 2006). That is, the encoder part of the AE computes the direct projection P, whereas the decoder part computes P^{-1}. More recently, iLAMP (dos Santos Amorim et al., 2012) explored local affine transformations to compute P^{-1} for a given direction projection P, specifically LAMP (Joia et al., 2011). A different direction was proposed by NNinv (Espadoto et al., 2023) as follows. Earlier on, Espadoto et al. (2020) have shown that one can use deep learning to imitate any given projection technique. This method, called Neural Network Projection (NNP) trains a regressor to produce $P(D)$ for any given D and any user-selected projection technique P. NNInv uses the same approach, but swaps the roles of D and $P(D)$ – that is, given a 2D scatterplot $P(D)$, it aims to regress it to the corresponding dataset D. Several improvements of NNInv followed next, including Self-Supervised Neural Projection (SSNP, Espadoto. et al. (2021)), which learns both a direct and inverse projection with stronger cluster separation based on data (pseudo)labeling; and Shape-Regularized Multidimensional Projection (ShaRP, Machado et al. (2023)), which does the same by using a variational autoencoder architecture that also allows one to regularize the shapes of the obtained point clusters in the projection. Compared to iLAMP, the deep-learning-based inverse projections (AEs, SSNP, NNInv, ShaRP) are significantly faster and require no parameter setting.

Evaluating the quality of inverse projections is a significantly more complicated task than evaluating direct projection quality, for the abovementioned reason that we aim to extrapolate a function outside the area where we have ground-truth information. As such, apart from the cost minimization (1) mentioned above,

other means have been proposed to assess quality. Most notably, the smoothness condition (2) mentioned above has been visually evaluated using gradient maps (Espadoto et al., 2023). We will use similar techniques when deploying inverse projections to assist users in pseudolabeling tasks in Chapter 6.

Inverse projections are a key instrument in the construction of decision boundary maps (DBMs), a VA tool used in evaluating (and improving) the quality of ML models. We discuss DBMs in detail in Sec. 2.4.

2.3.6 *Discussion*

As already outlined above, many projection quality metrics exist – for an overview, we refer the reader to recent surveys (Nonato and Aupetit, 2018; Espadoto et al., 2019a). However, to actually determine which (if any) of these metrics are good in gauging a projection's quality, we need to first define what the projection will be *used* for, and how. Indeed: If a given projection fulfils all the tasks it was selected for, then, by definition, it should be of high quality *for those tasks* – and conversely (Nonato and Aupetit, 2018). The key problem with this approach is that the range of tasks that one can imagine to address using a projection is very wide. As such, choices are to be made in how we are going to precisely use projections next in our work – as this will tell us how we are going to evaluate their quality.

To this end, let us revisit the two research questions (RQ's) proposed in Sec. 1.4. RQ1 states that we would like to use projections to build better ML models. Let us examine, one by one, the steps implied by RQ1:

- A ML (classification) model essentially aims to capture how points, in a dataset D, are separated into different groups (distributions) and assign a different class label to each of these.
- Let us denote by DS the *data separation* between same-class points in D. Clearly, the higher DS is, the easier is to construct a good-performance classifier for D.
- Let us denote by CP the *classifier performance* of a trained ML model for a given problem (assessed using the procedures outlined in Sec. 2.1.1).
- Let us denote by VS the *visual separation* between same-class points in $P(D)$. If we aim to use a projection $P(D)$ to help

this classification process, this means that the VS of points in $P(D)$ should capture as well as possible the DS of points in D. Then, indeed, $P(D)$ can be seen as a 'proxy' for D.

Jointly put, the quantities DS, VS, and CP *seem* to be correlated. More specifically, a high DS usually involves a high CP – a well-known fact from ML literature. If we want to use projections for ML engineering (see RQ1), then DS *should* correlate with VS. Rauber et al. (2017b) examined this aspect already a while ago. More precisely, they showed that a high VS in a projection – measured by the neighborhood hit N metric – does correlate with a high CP; that is, one can easily construct a high-performance classifier for a dataset that can be projected to a projection having a high N value.

Figure 2.4 shows this. The top row shows a t-SNE projection of $N = 688$ test samples, $n = 500$ dimensions, from the $|C| = 2$ class Madelon dataset – a well-known example for ML benchmarking (Guyon et al., 2004). The two top-row images show the projection colored by classification results from a trained KNN, respectively RFC, model. The visual separation (VS) of the projection is quite poor, as indicated by the relatively low $NH = 50.97\%$ value. We see how this poor VS is reflected in the relatively low classification performance (CP), as measured by accuracy, which is 54.94% for KNN and 66.17% for RFC, respectively. The bottom row of Fig. 2.4 shows a t-SNE projection of the same samples. However, now these were reduced to only $n = 20$ dimension by a feature selection process that used extremely randomized trees (Geurts et al., 2006) to select a small set of features which are most successful into the two classes of the dataset. We easily see that this feature-selection process increased the VS in the projection ($NH = 74.15\%$). Also, the CP of both classifiers increased now to yield accuracies of 88.62% for KNN, respectively 88.92% for RFC. Additional experiments in Rauber et al. (2017b) support the claim that VS (of projections) and CP values (of the projected data being classified by trained ML models) correlate very well.

However, Rauber et al. (2017b) also mentioned that little can be said about the converse situation. That is, a projection with a poor N value does not *necessarily* mean that it is hard to construct a high-performance (high CP) classifier for that dataset. Moreover, they did not further evaluate VS beyond the computation of the N metric. As for all the other metrics typically used in projection quality assessment (*e.g.*, C, T, S), N has a very 'local' view on a projection – that is, it only considers patterns at a given

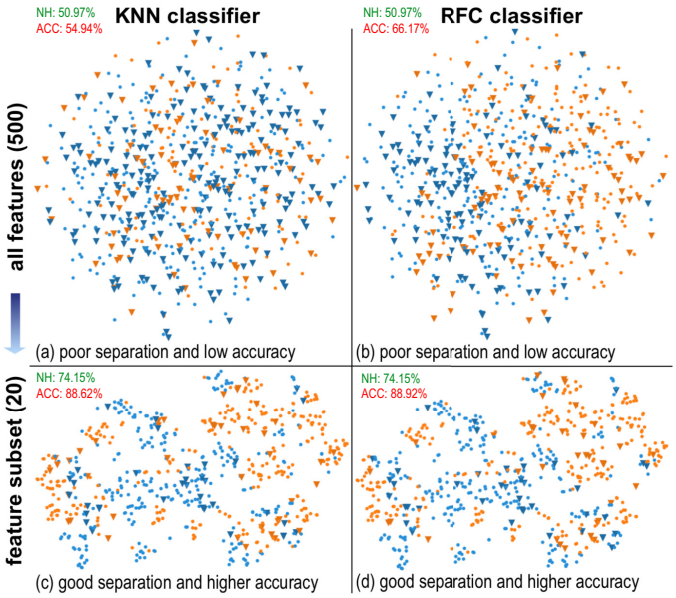


Figure 2.4: Correlation of visual separation (VS), measured by the neighborhood hit (NH) metric, and classifier performance (CP), measured as accuracy (ACC), for two different projections and two classification models. Figure reproduced from Rauber et al. (2017b).

scale, imposed by the user via the k hyperparameter (number of neighbors).

Our second research question (RQ2) aims to generalize the observations that underlie RQ1. In Chapter 4, we show how DS, VS, and CP are tightly related, by means of experiments which go beyond the relatively limited evaluation of Rauber et al. (2017b). In Chapter 5, we go one step further – we propose an actual way of measuring VS, which goes beyond the local approach of all current projection-quality metrics (using k neighborhoods) to consider the global distribution of samples. We also show that our novel approach agrees well with how users gauge VS in a projection.

2.4 VISUALIZING ML MODELS

The usage of information visualization (infovis) to understand high-dimensional data and, more specifically, the results of ML techniques is not new (Packer et al., 2013). Yet, evidence that infovis and, more specifically, its interactive hypothesis-forming-and-validation sense making loop known as VA is effective in understanding and improving ML is more recent (Rauber et al., 2017b,a; Zeiler and Fergus, 2014). The literature concerning visualizing ML models is extremely extensive. As such, in the following we will limit ourselves to discussing specific VIS and VA techniques which are closely related to our work.

Understanding ML training: Recent work shows that VA can be used to understand the training process of DL (Rauber et al., 2017a) and to infer actions that improve the network design (Zeiler and Fergus, 2014; Rauber et al., 2017b). Rauber et al. (2017b) showed that high separation of categories (labeled samples) in 2D t-SNE projections of n-D data is a strong indication of high separation of the same categories in the nD data. A strong separation also correlates well with the ease of building a high-accuracy classifier for the respective categories (van der Maaten, 2014; Rauber et al., 2017b). A recent survey of VA techniques for deep-learning engineering was proposed by Garcia et al. (2018).
Interacting with feature spaces: Other studies have investigated the use of feature space projection and user interaction to understand and design ML models (Rauber et al., 2015, 2017b, 2016; Bernard et al., 2018; Peixinho et al., 2018; Benato et al., 2018). Some of these works have addressed over-fitting and large unlabeled datasets — common issues in ML — by exploring, respectively, interactive data augmentation (Peixinho et al., 2018)

and interactive data annotation (Benato et al., 2018, 2021c) guided by feature space projections. Yet, (Benato et al., 2018, 2021c) have not considered an active learning (AL) looping, *i.e.*, using the labeled samples to re-train the classifier and use the classifier information to guide the user labeling in the next iteration. In other words, these studies use only one iteration of the user interaction. Our work in Chapter 6 extends this approach to multiple user iterations.

Decision boundary maps: A typical way to visualize the behavior of a classifier f over a test set D is to construct the projection $P(D)$ and next plot it by color-coding its points $P(\mathbf{x}_i)$ by the values $f(\mathbf{x}_i)$. Figure 2.5 shows this for a simple logistic regression classifier for a 7-class problem. This image shows quite clearly that certain classes, *e.g.* 1 (orange) and 6 (cyan) are better separated from the others. However, the image does not tell us what would happen for a data point that would project in the *white space* in the image. We can assume, for example, that points projecting somewhere inside the orange cluster will quite likely also get the label 1 (since that cluster is well separated from the rest of the projection), but we do not know this for sure.

Decision boundary maps (DBMs) aim to solve this problem by constructing a *dense* visual representation of the behavior of a trained ML model f, as follows. Given a dataset D (which can be the training set, test set, or a combination of both, used for f), a projection $P(D)$ is constructed. Next, an inverse projection P^{-1} is computed from D and $P(D)$, using any of the methods described earlier in Sec. 2.3.5. The 2D space in which $P(D)$ lives is then discretized in a pixel grid G at a user-given resolution. Next, for every pixel $\mathbf{y} \in G$, its inverse projection $P^{-1}(\mathbf{y})$ is computed. Finally, the pixel $\mathbf{y}$ is colored to depict the value of the inferred label $f(P^{-1}(\mathbf{y}))$.

Figure 2.5 (right) shows the DBM for the classifier depicted in the 2D scatterplot in the left image. Same-color regions in the DBM show the *decision zones* of the classifiers, *i.e.*, sets of points in $\mathbb{R}^n$ for which f infers the same label. Neighbor pixels having different colors in the DBM show the *decision boundaries*, *i.e.*, points where f changes value. Simply put, a DBM is a graphical representation that shows how the machine learning model separates data points into different classes based on the original feature space. For instance, in the DBM image in Fig. 2.5, we see that the orange decision zone is quite compact, except for a few small 'islands'. This confirms our earlier hypothesis that data val-

ues close to the ones which were classified as class 1 (orange) will yield the same class.

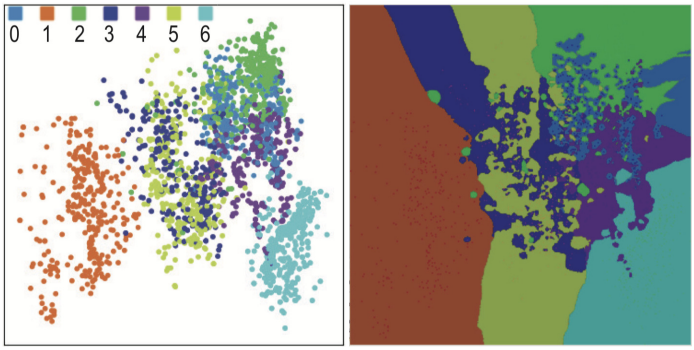


Figure 2.5: Left: Projection scatterplot colored by labels inferred by a classifier. Right: Decision map showing how the classifier operates on additional points. Same-color areas indicate the classifier's decision zones. Neighbor pixels having different colors show the classifier's decision boundaries. Image adapted from Rodrigues et al. (2018).

Several algorithms have been proposed to compute DBMs. The original method (Rodrigues et al., 2018) uses t-SNE and LAMP for P and iLAMP for P^{-1}. The method was refined by removing (from the computation of P^{-1}) projected points in $P(D)$ which have large projection errors and thus create some of the aforementioned spurious islands in the DBM, and also by encoding the confidence of the classifier f in the brightness of the DBM pixels (Rodrigues et al., 2019). A further refinement, called Supervised Decision Boundary Map (SDBM, Oliveira. et al. (2022)), leveraged the SSNP projection (see Sec. 2.3.5) to construct DBMs which exhibit far smoother decision boundaries and are thus easier to visually explore. A recent study compared several such DBM methods from the perspective of their ability to display the behavior of a given classifier with as few errors as possible (Wang et al., 2023).

Aside from the ability of DBMs to allow a good explanation of classifiers, the provided information is still not used to improve the classifier. The only exception to the latter that we know, Rodrigues (2020), explored user interaction with DBMs in an active learning looping to label samples. However, some limitations exist to this work, as follows. Classification performance is computed over validation and test sets – *i.e.*, more supervised samples are required. Also, this work does not consider combining man-

ual and automatic labeling. Our work in Chapter 6 shows how DBMs can be leveraged to account for such limitations.

3 FEATURE AND CLASSIFIER LEARNING

3.1 INTRODUCTION

The success of deep neural networks (DNNs) is evident in many applications. However, as already mentioned in Chapter 2, the need for large annotated training sets for engineering DNNs is a well-known problem (Lin et al., 2014; Sun et al., 2017). Common approaches to alleviate the problem include data augmentation (creating synthetic samples) and transfer learning common (using pre-trained weights).

To train DNNs, the set with supervised samples is usually split to generate a validation set for hyperparameter search and model evaluation. When the validation set is representative, it provides a good insight of the model's performance on unseen test sets (Bergstra and Bengio, 2012). A critical problem appears when the training set is too small (Sun et al., 2017) – *e.g.* only dozens of supervised samples per class. Assuming a set with many unsupervised samples is available, semi-supervised learning techniques can propagate pseudo labels from the supervised samples to the unsupervised ones, considerably increasing the number of labeled training samples. Yet, semi-supervised learning techniques still require hundreds to thousands of supervised samples for training and validation (Lee, 2013; Miyato et al., 2018; Jing and Tian, 2020; Pham et al., 2021).

In this chapter, we propose a meta-pseudo-labeling methodology, called *Deep Feature Annotation* (*DeepFA*), to train DNNs from very few supervised samples (e.g., 1% of a dataset) without a validation set. In *DeepFA*, the teacher (a connectivity-based semi-supervised classifier) exploits modifications of a given latent feature space of the student (a DNN) along with iterations of non-linear 2D projection for pseudo-labeling. At each iteration, the most confidently labeled samples are used to retrain the DNN, modifying its latent feature space. The semi-supervised classifier does not require parameter optimization, dismissing a validation set. The number of labeled samples is next increased to improve the DNN with pseudo-labeled training and validation sets[1].

1 This chapter is a result of the following publications: "Semi-supervised deep learning based on label propagation in a 2D embedded space" (Benato et al., 2021a); "Iterative pseudo-labeling with deep feature annotation and confidence-

For pseudo-labeling in 2D, we use a semi-supervised Optimum Path Forest (OPFSemi) classifier (Amorim et al., 2016), which has outperformed several techniques in different works (Benato et al., 2018; Amorim et al., 2019; Benato et al., 2021c,a). In (Benato et al., 2018), for instance, OPFSemi outperformed LapSVM (Sindhwani et al., 2005), achieving the highest performance when label propagation was done in the 2D embedded space created by t-SNE (van der Maaten, 2014) from the intermediary feature space of an autoencoder, in contrast to (Amorim et al., 2019) where label propagation was performed in the latent feature space.

Isolated aspects of *DeepFA* using OPFSemi on a 2D embedded space have been evaluated in conference papers. In (Benato et al., 2021a), a few iterations of the training loop with truly-and-artificially-labeled samples was enough to improve the generalization performance of a supervised DNN. The study used only 1%–5% of supervised samples. We call this version as *orig-DeepFA*, while we use *DeepFA* to refer to the entire methodology. OPFSemi's confidence was also considered when selecting unsupervised samples to train the supervised DNN (Benato et al., 2021b), reducing label propagation errors. Let us call this last version *conf-DeepFA*.

However, *conf-DeepFA*'s performance on test sets can oscillate along with the pseudo-labeling iterations such that the model obtained at the last iteration is not guaranteed to be the best model. To circumvent this problem, we propose *ext-DeepFA*, which extends *conf-DeepFA* by computing a clustering-based metric from the pseudo-labeled samples to select the optimal model for generalization among the ones generated along with the pseudo-labeling iterations. While earlier *orig-DeepFA* and *conf-DeepFA* variants have shown promising results, the methods may differ in the deep architecture used for feature learning and classification, the semi-supervised classifier for label propagation, the projection technique, and the criterion to select the model for generalization. Thoroughly exploring this 'design space' is needed to gain confidence in the results' robustness and, where possible, to find hyperparameters that lead to higher performance. The present work is then a comprehensive study on DeepFA.

We next outline six question whose answers support the exploration of *DeepFA*'s design space.

- (Q1) Can a deep neural network with pre-trained weights improve performance by self-pseudo-labeling?

based sampling" (Benato et al., 2021b); and "Deep feature annotation by iterative meta-pseudo-labeling on 2D projections" (Benato et al., 2023e).

- (Q2) Can performance be improved by using other label propagation methods than OPFSemi?
- (Q3) Can OPFSemi's confidence improve *DeepFA*'s pseudo-labeling?
- (Q4) Does *DeepFA* work for other deep architectures than the currently used VGG-16 (Simonyan and Zisserman, 2014)?
- (Q5) Can we improve *DeepFA* by choosing other layers from the network to extract a feature space?
- (Q6) How can we identify the optimal model among those computed during the *DeepFA* iteration sequence?

Table 1 summarizes our contributions, described next, as follows. Section 3.2 details *ext-DeepFA*, our extension of the existing *orig-DeepFA* and *conf-DeepFA* methods, as well as the experiments that we propose to address questions Q1..Q6. Section 3.3 details the experimental procedure. Section 3.4 shows the experimental results. Section 3.5 summarizes our answers to Q1..Q6. Section 3.6 discusses limitations of our work. Finally, Section 3.7 concludes the paper.

3.2 ITERATIVE DEEP FEATURE PSEUDOLABELING ON 2D PROJECTIONS

We next detail our extension *ext-DeepFA* of the original (*orig-DeepFA*, Benato et al. (2021a)) and confidence-based (*conf-DeepFA*, Benato et al. (2021b)) methods with the contributions listed in Tab. 1. Figure 3.1 shows our extended method called *ext-DeepFA*. We next discuss each method step and outline how the questions Q1..Q6 relate to design decisions about these steps.

3.2.1 *Deep feature learning*

We start our pipeline by training a deep feature learning algorithm by using the few available supervised samples (Fig 3.1, step 1). Obviously, since supervised samples are few, one cannot expect good results using a too deep network trained from scratch. As such, an interesting question is whether a deep neural network with pre-trained weights can improve performance by self-pseudo-labeling (Q1). A negative answer implies the need of a separate machine learning method for pseudo labeling, such

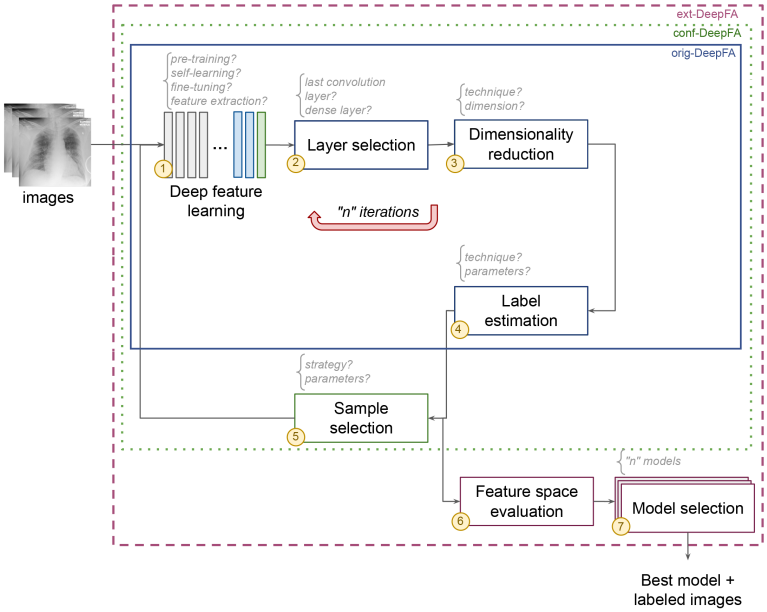


Figure 3.1: Proposed *ext-DeepFA* pipeline. Using a few supervised images, a deep feature learning algorithm is trained (1) and features are extracted from unsupervised images from a selected layer (2). The features are projected to a 2D space (3) and a semi-supervised technique propagates labels from supervised to unsupervised samples in that space (4) – the *orig-DeepFA* pipeline (inner blue box).. The most informative samples are selected (5) and their labels are used to retrain the network along with the supervised samples. Steps (1-5) are iteratively repeated – the *conf-DeepFA* pipeline (green dashed box). In each iteration, the labeled projected feature space is evaluated (6) using a quality measure to select the best model (7). The process outputs at end the optimal model and labeled samples – the *ext-DeepFA* pipeline (outer maroon dashed box).

Table 1: Comparison of how earlier *vs* our work address questions Q1..Q6 for *DeepFA*.

Question	Earlier work	Our contribution (*ext-DeepFA*)
Q1	used an autoencoder to generate the initial feature space for interactive data annotation (Benato et al., 2018, 2021c);	use a supervised, pretrained, deep architecture with few available supervised samples (Benato et al., 2021a);
Q2	compared OPFSemi label propagation only with Laplacian SVM (Benato et al., 2018, 2021c). Similar comparisons exist in (Amorim et al., 2019) but were not applied to *DeepFA*;	compare OPFSemi to two additional semi-supervised learning methods (L.Prop, L.Spread) in *DeepFA*;
Q3	OPFSemi's confidence was used to select samples for label propagation (Benato et al., 2021c), but not in *DeepFA*;	use OPFSemi's confidence to select samples to propagate labels in *DeepFA* (Benato et al., 2021b);
Q4	only VGG-16 was explored to learn the feature space (Benato et al., 2021a);	compare VGG-16 with an additional architecture (MobileNet (Sandler et al., 2018));
Q5	only the output of the last convolutional layer was used to propagate labels (Benato et al., 2021a)	use different layers of the deep network for the same purpose;
Q6	such methods either used no iterations (Benato et al., 2018, 2021c) or an upfront fixed number of iterations (Benato et al., 2021a);	propose a clustering metric to find the iteration delivering the optimal trained model.

as OPFSemi proposed by *DeepFA* and its extensions. A separate question (Q4) related to this step is which are pre-trained models that one can use to obtain high performance in the context of our application, besides the VGG-16 one proposed in earlier work (Benato et al., 2021a).

3.2.2 *Layer selection*

The feature spaces learned in the different deep layers of the network capture the structure of the data space the network is exposed to. Hence, we can use such a feature space as input for the label propagation (see Sec. 3.2.4 next). In earlier works, the last convolutional layer was used for this purpose (Benato et al., 2021a,b). An open question (Q5) is whether using deeper layers would improve performance.

3.2.3 *Dimensionality reduction*

The feature space from the selected network layer (Sec. 3.2.2) can be reduced before being used for label propagation. Earlier work

has shown that using a 2D t-SNE projection for this often yields labels of higher accuracy than when propagation is done directly in the feature space (Benato et al., 2018). A potential question is whether one could use for this step other existing projection methods than t-SNE. We believe there is evidence to the contrary: An extensive study (Espadoto et al., 2019a) showed that t-SNE has one of the highest quality, measured in terms of combined trustworthiness, neighborhood hit, continuity, Shepard correlation, and normalized stress metrics (all common quality metrics in projection literature, see also Sec. 2.3.4.1), among 45 studied projection algorithms. As such, from an application-agnostic perspective, one can say that t-SNE preserves the high-dimensional data structure better than its competitors, so it is the candidate of choice to be used.

Separately, the chosen label estimation algorithm (see Sec. 3.2.4) uses Euclidean distances *in the 2D projection space*. As such, having a compact projection where similar data points are close to each other (*i.e.*, with high neighborhood preservation), like t-SNE, favors our label propagation as opposed to projections which spread the data points more in 2D, *e.g.*, MDS variants. However, to fully prove our above point, *i.e.*, the suitability of 2D t-SNE projections, more studies are needed, *e.g.*, replacing t-SNE in our pipeline by other top-ranking projections in terms of quality from (Espadoto et al., 2019a) such as UMAP, IDMAP, or PBC.

We also argue that using t-SNE to project features in spaces higher than 2D, *e.g.*, using a 3D projection, is overall not an attractive option since (a) 3D projections score only slightly quality metrics than their 2D counterparts (Tian et al., 2021); (b) OPFSemi's use of Euclidean distances can be affected by the higher space dimensionality (and by higher Euclidean distance values as well); (c) users experience significantly higher difficulty when visualizing 3D as opposed to 2D projections, which only gets worse if one wants to use the projection to also interact with the data, *e.g.*, for manually fine-tuning the labeling.

3.2.4 *Label estimation*

The few available labels are propagated in the 2D projection to create a rich set of labeled points which is next used to refine the network training. As stated earlier, *DeepFA* uses OPFSemi for this step. OPFSemi first was proposed in (Amorim et al., 2016) and we use the same algorithm for propagating pseudo labels from the supervised samples to the unsupervised ones in a

semi-supervised way. However, OPFSemi can make mistakes and its effectiveness depends on filtering out the most likely mislabeled samples by thresholding its confidence value (Benato et al., 2021c). An open question here (Q2) is whether performance can be improved by using counterparts pseudo-labeling techniques rather than OPFSemi.

3.2.5 *Sample selection*

Earlier work (Benato et al., 2018) used all pseudo labels constructed by OPFSemi which, as noted earlier, can lead to training based on wrongly propagated labels. Separately, the confidence of OPFSemi was used to select a subset of most confident pseudolabels to use next (Benato et al., 2021c). However, this strategy has not been used in *DeepFA*. This raises the question (Q3) on how can confidence-based pseudo-label selection improve the *DeepFA* end-to-end pipeline.

3.2.6 *Model and iteration selection*

Earlier methods either applied the *DeepFA* idea for a single iteration (Benato et al., 2018, 2021c) or used a predefined number of iterations (Benato et al., 2021a,b). Both approaches are suboptimal if one is after finding the best way to train a deep model. Rather, we need to (a) execute the *DeepFA* iterations and (b) select, from the trained models computed after each iteration, the one with the highest performance.

For our scenario of few labeled samples, an inherent problem is how to gauge performance of such models. One approach is to use as a proxy the quality of the pseudo-labeled samples. Importantly, we cannot use true label information of the unsupervised samples for this purpose as this would defeat the very purpose of training with a very small supervised set (tens of samples in some cases). Rather, we need a quality metric that considers not only labels but also the separation (distance) between samples in the 2D projection space. A joint question (Q6) is which metric to use for this purpose and how to use it to determine the optimal model over all executed iterations.

3.3 EXPERIMENTAL EVALUATION

We next outline how we organized our study of the questions Q1..Q6 in Sec. 3.1 in terms of used datasets (Sec. 3.3.1, experimental setup (Sec. 3.3.2), and implementation (Sec. 3.3.3).

3.3.1 *Datasets*

We choose eight diverse datasets to perform our investigations, as follows.

MNIST: We first chose the public MNIST (LeCun and Cortes, 2010) dataset, to explore a known and easy classification task. MNIST has 0 to 9 handwritten digits grayscale images (28×28 pixels). We use 5000 random samples from the original training dataset.

Parasites: The next five datasets come from a Parasite medical image collection (Suzuki et al., 2013). This collection has three main dataset types: (i) *Helminth larvae*, (ii) *Helminth eggs*, and (iii) *Protozoan cysts*. The datasets contain color microscopy images (200×200 pixels) of the most common species of human intestinal parasites in Brazil, responsible for public health problems in most tropical countries (Suzuki et al., 2013). The datasets are challenging since they are unbalanced and contain a majority impurity class, with samples very similar to parasites, making classification hard (see Fig. 3.2). Table 2 shows the number, type, and sample count per class for the datasets (i-iii) listed above. To these datasets, we add the (iv) *Helminth eggs* and (v) *Protozoan cysts* without the impurity class datasets, yielding a total of 5 datasets.

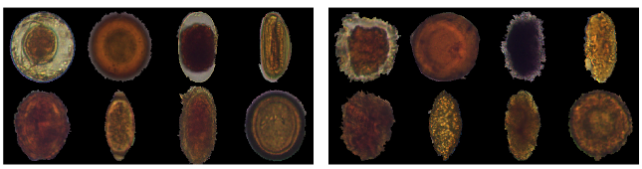


Figure 3.2: Examples of *H.Eggs* species (left) and similar images of impurities (right).

Coconut: We use a random subset of the Coconut trees dataset (Vargas-Muñoz et al., 2019) with 7,827 regions (90×90

pixels) of aerial colored images from the Kingdom of Tonga, acquired by satellite imagery in October 2017, labeled by Humanitarian OpenStreetMap. The dataset has two classes: images with (6,139) or without coconut trees (1,688).

COVID: A team of researchers from the Universities of Qatar and Dhaka have created a database (Chowdhury et al., 2020; Rahman et al., 2021) of chest X-ray images (299×299 pixels). We obtained the second update of the dataset with 21165 images split in four classes: COVID-19 positive (3616), lung opacity (non-COVID lung infections, 6012), viral pneumonia (1345), and normal (10192) cases. We use a randomly subset of this dataset with 10583 images.

Table 2: MNIST, Coconut, COVID19 (left) and three Parasites datasets (right). For each dataset, we list its number of classes, class names, and sample count per class.

Dataset	Classes	# samples
MNIST (10 classes)	zero	479
	one	563
	two	488
	tree	493
	four	535
	five	434
	six	501
	seven	550
	eight	462
	nine	495
	total	5,000
Coconut (2 classes)	coconut tree	6,139
	none	1,688
	total	7,827
COVID19 (4 classes)	COVID19	1,808
	lung opacity	3,002
	pneumonia	672
	normal	5,096
	total	10,583

Dataset	Classes	# samples
(i) *H. larvae* (2 classes)	*S.stercoralis*	446
	impurities	3068
	total	3,514
(ii) *H. eggs* (9 classes)	*H.nana*	348
	H.diminuta	80
	Ancilostomideo	148
	E.vermicularis	122
	A.lumbricoides	337
	T.trichiura	375
	S.mansoni	122
	Taenia	236
	impurities	3,444
	total	5,112
(iii) *P. cysts* (7 classes)	*E.coli*	719
	E.histolytica	78
	E.nana	724
	Giardia	641
	I.butschlii	1,501
	B.hominis	189
	impurities	5,716
	total	9,568

3.3.2 *Experimental setup*

To reproduce the scenario of few supervised samples, we define a supervised training set S with only 1% of supervised samples

from a given dataset D. The unsupervised U and test T sets have 69% and 30% of samples, respectively ($D = S \cup U \cup T$). The small S simulates the real-world scenario when one has a large D but manual effort is needed to label samples to create S. We randomly divide each dataset D into S, U, and T in a stratified manner and also generate three distinct splits for each experiment for statistical analysis. Table 3 shows the number of supervised samples in S for each of the eight datasets in Sec. 3.3.1.

Table 3: Number of supervised (S) and unsupervised (U) samples for each dataset.

	MNIST	H.eggs (w/o imp)	P. cysts (w/o imp)	H. larvae	H. eggs	P. cysts	Coconut	COVID19
S	50	17	38	35	51	95	78	105
U	3450	1220	2658	2424	3527	6602	5400	7302

We evaluate our method by the probability of the chosen deep architecture's last fully-connected layer, *i.e.*, just before the classification layer. From this, we compute accuracy and κ, which are described in Chapter 2. We evaluate label propagation accuracy by computing the number of correctly assigned labels in U.

3.3.3 *Implementation details*

As stated, we want to use *DeepFA* without a validation set whose creation would require extra user supervision (data annotation effort). For this, we fix all pipeline's parameters without any optimization step. For t-SNE, we use the default parameters in *scikit-learn*. Note that OPFSemi has no parameters.

All our neural networks were implemented in Python using Keras (Chollet et al., 2015), replacing the original fully-connected layers by two fully-connected layers with 4096 neurons and rectified linear activation followed by a decision layer with c neurons, where c is the number of classes of each dataset, and softmax activation. Models are trained by error backpropagation for a categorical cross-entropy function, using stochastic gradient descent with a linearly decaying learning rate initialized at 0.1 and momentum of 0.9, respectively. We loaded ImageNet pretrained weights and used a linear decay of 1×10^{-6} over 15 epochs. The pretrained weights for convolutional layers were fixed for the feature extraction experiments and unfrozen for fine-tuning, respectively.

3.4 EXPERIMENTAL RESULTS

We next address questions Q1..Q6 listed in Sec. 3.1 by presenting experiments, results, and discussion for each of them.

3.4.1 *Q1: Adding iterations: self pseudo labeling*

We evaluate VGG-16 pre-trained on ImageNet, with and without fine-tuning its convolutional layers (Sec. 3.3.3). We also evaluate VGG-16 performing self pseudo-labeling. We did four experiments (below, *ft* stands for fine-tuning and *fe* stands for feature extraction, respectively):

- *VGG-16*$_{ft}$: VGG-16 with fine-tuning, trained on S and tested on T;
- *self-VGG-16*$_{ft}$: VGG-16$_{ft}$ trained on $S \cup U$, being the samples in U pseudo-labeled by VGG-16$_{ft}$, and tested on T. The self training loop uses five iterations;
- *VGG-16*$_{fe}$: VGG-16 with only the last four convolutional layers used as *unfrozen* for feature extraction, trained on S and tested on T;
- *self-VGG-16*$_{fe}$: VGG-16$_{fe}$ trained on $S \cup U$, being the samples in U pseudo-labeled by VGG-16$_{fe}$, and tested on T. The self training loop uses five iterations.

3.4.1.1 *Results and discussion*

Table 4 shows the mean values of accuracy in label propagation and classification, κ, and their standard deviations over three splits, using each VGG-16-based model. We can see that feature extraction and fine-tuning do not show relevant gains in self pseudo-labeling along with the iterations. The results of the models based on feature extraction only are usually better than those of the fine-tuned models. We notice considerable gains in accuracy and κ when using *VGG-16*$_{fe}$ and *self-VGG-16*$_{fe}$, indicating that the work in Benato et al. (2021a) could have presented better results with feature extraction than using fine-tuning. As *self-VGG-16*$_{fe}$ achieved the best results, we use this pipeline in all subsequent experiments.

Table 4: (Q1) Results for four VGG-16 models considering feature extraction and fine-tuning. Best values per metric and dataset in bold.

		VGG-16 variants			
dataset	**metric**	***VGG-16$_{ft}$***	***self-VGG-16$_{ft}$***	***VGG-16$_{fe}$***	***self-VGG-16$_{fe}$***
MNIST	prop. acc	-	0.447238 ± 0.146	-	**0.586000 ± 0.007**
	acc	**0.629555 ± 0.037**	0.441334 ± 0.149	0.614444 ± 0.015	0.592222 ± 0.020
	kappa	**0.588195 ± 0.041**	0.378648 ± 0.166	0.571176 ± 0.017	0.546162 ± 0.023
H.eggs (w/o imp)	prop. acc	-	**0.758825 ± 0.088**	-	0.744004 ± 0.114
	acc	**0.790961 ± 0.050**	0.779033 ± 0.095	0.738858 ± 0.054	0.774011 ± 0.131
	kappa	**0.752807 ± 0.060**	0.735591 ± 0.113	0.693278 ± 0.060	0.734030 ± 0.153
P.cysts (w/o imp)	prop. acc	-	0.399481 ± 0.010	-	**0.648739 ± 0.111**
	acc	0.561130 ± 0.093	0.400519 ± 0.011	**0.736159 ± 0.027**	0.650230 ± 0.101
	kappa	0.324051 ± 0.175	0.020734 ± 0.021	**0.626632 ± 0.039**	0.483706 ± 0.170
H.larvae	prop. acc	-	0.897384 ± 0.031	-	**0.912837 ± 0.038**
	acc	0.874566 ± 0.001	0.886572 ± 0.017	0.893523 ± 0.017	**0.908689 ± 0.040**
	kappa	0.021406 ± 0.019	0.174158 ± 0.208	0.256836 ± 0.203	**0.385892 ± 0.402**
H.eggs	prop. acc	-	0.773803 ± 0.034	-	**0.847308 ± 0.018**
	acc	**0.858323 ± 0.013**	0.775750 ± 0.034	0.848327 ± 0.017	**0.850934 ± 0.014**
	kappa	**0.734333 ± 0.019**	0.519971 ± 0.114	0.713649 ± 0.030	0.714227 ± 0.038
P.cysts	prop. acc	-	0.730327 ± 0.022	-	**0.817978 ± 0.004**
	acc	0.758853 ± 0.077	0.734239 ± 0.028	0.818182 ± 0.004	**0.824800 ± 0.011**
	kappa	0.542967 ± 0.218	0.492070 ± 0.107	0.697633 ± 0.009	**0.705397 ± 0.022**
Coconut	prop. acc	-	**0.826153 ± 0.026**	-	0.817147 ± 0.016
	acc	0.821200 ± 0.027	0.828721 ± 0.026	**0.835249 ± 0.027**	0.813822 ± 0.031
	kappa	0.304424 ± 0.182	0.324694 ± 0.153	**0.385120 ± 0.147**	0.228646 ± 0.224
COVID19	prop. acc	-	0.659174 ± 0.022	-	**0.660816 ± 0.016**
	acc	0.627612 ± 0.034	**0.677008 ± 0.039**	0.589186 ± 0.094	0.675066 ± 0.028
	kappa	0.389689 ± 0.074	**0.480240 ± 0.068**	0.274597 ± 0.239	0.476834 ± 0.049

3.4.2 *Q2: Pseudolabeling comparison: OPFSemi* vs *others*

As related work (Benato et al., 2021a,b) only tested *orig-DeepFA* looping with OPFSemi, we evaluate other semi-supervised learning methods for label propagation over the learned (and next reduced) feature space over a few iterations. Specifically, we use the LabelPropagation (*L.Prop*) and LabelSpreading (*L.Spread*) methods, available in *scikit-learn*, with k-nearest neighbors (knn) and radial basis functions (rbf) kernels. As *L.Prop* and *L.Spread* have parameters and we want to avoid parameter searching (due the few supervised samples available), we set parameters to their default values in *scikit-learn*. The experiments done are listed below:

- *OPFSemi*: VGG-16 is trained on S. Deep features for $S \cup U$ from the last convolutional layer are projected in 2D with t-SNE and used for OPFSemi to propagate labels from S to all samples in U. VGG-16 is then retrained with $S \cup U$ and tested on T (at the last iteration of *orig-DeepFA* looping);
- *L.Prop$_{knn}$*: As above but replaces OPFSemi by *L.prop* (knn kernel);

- *L.Prop*$_{rbf}$: As above but replaces OPFSemi by *L.prop* (rbf kernel);
- *L.Spread*$_{knn}$: As above but replaces OPFSemi by *L.spread* (knn kernel);
- *L.Spread*$_{rbf}$: As above but replaces OPFSemi by *L.spread* (rbf kernel).

3.4.2.1 *Results*

Table 5 and Figure 3.7 show the results of the experiments that compare *OPFSemi* against *L.Prop*$_{rbf}$, *L.Prop*$_{knn}$, *L.Spread*$_{rbf}$, and *L.Spread*$_{knn}$ for label propagation in the *orig-DeepFA* looping. We show mean values of label propagation accuracy, classification accuracy, κ, and their standard deviation over three different splits. From the compared methods, OPFSemi yielded the best performance for *all* tested datasets. This is an important result since we want to reproduce a real scenario with a few supervised samples (Sec. 3.3.2) and OPFSemi shows that it is possible to handle diverse datasets with no parameter optimization. Interestingly, *L.Spread*$_{rbf}$ showed the highest mean label propagation accuracy in the Coconut dataset, but having the highest standard deviation. While *L.Prop* shows very low results for its default parameters, *L.Spread*$_{rbf}$ and *L.Spread*$_{knn}$ show better results depending of the dataset. For datasets with significant confusion between distinct classes in the 2D projection (P.cysts, Coconut, COVID19), *L.Spread*$_{rbf}$ surpasses *L.Spread*$_{knn}$.

3.4.2.2 *Discussion*

Using different pseudo-labeling methods within the *orig-DeepFA* looping means that the label propagation and the learned feature space can be mutually affected. To evaluate how, Fig. 3.3 shows the resulting feature space and label estimation of the two best methods found in Sec. 3.4.2, *i.e.*, *OPFSemi* and *L.Spread*$_{rbf}$. In this figure, we use datasets with similar classification values (Coconut tree, $\kappa = 0.53$) and distinct classification values (H.larvae, $\kappa \in \{0.80, 0.06\}$).

For the Coconut dataset, we see that *L.Spread*$_{rbf}$ was more conservative in propagating labels (green points concentrated in the bottom part of the projection), while *OPFSemi* was more sensitive to outliers (green points in other projection regions). This is confirmed by the f1-score metric for classes 1 (red) and 2 (green), where *OPFSemi* got higher f1 values for class 2 (0.64) compared

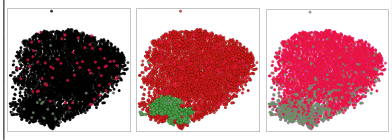
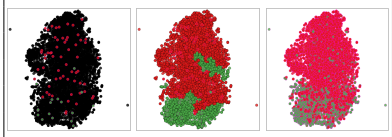
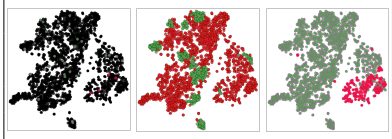
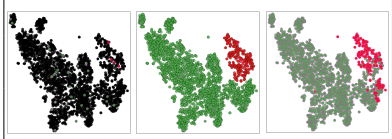
dataset	method	2D projections	metric	class 1	class 2	total
Coconut	$LSpread_{rbf}$		precision	0.87	0.79	
			recall	0.96	0.49	
			f1-score	0.92	0.60	
			accuracy			0.86
			kappa			0.53
	OPFSemi		precision	0.91	0.60	
			recall	0.88	0.67	
			f1-score	0.89	0.64	
			accuracy			0.83
			kappa			0.53
H.larvae	$LSpread_{rbf}$		precision	0.15	0.99	
			recall	0.99	0.21	
			f1-score	0.27	0.35	
			accuracy			0.31
			kappa			0.06
	OPFSemi		precision	0.82	0.97	
			recall	0.83	0.97	
			f1-score	0.82	0.97	
			accuracy			0.95
			kappa			0.80

Figure 3.3: (Q2) Comparison of *DeepFA* using $LSpread_{rbf}$ and *OPFSemi* pseudo labeling for Coconut and H.Larvae datasets, with 1% supervised samples and last iteration out of five. 2D feature-space projections of training samples ($S \cup U$) in columns per dataset (from left to right): supervised samples colored by true labels (red=1, green=2), unsupervised ones are black; samples colored by assigned pseudo labels; and samples colored by their true labels. Classification results (per class and total) are shown on the right.

Table 5: (Q2) Results from the last iteration for experiments using five label propagation methods over five iterations. Best values per dataset in bold.

		semi-supervised learning methods				
dataset	**metric**	***L.Prop***$_{rbf}$	***L.Prop***$_{knn}$	***L.Spread***$_{rbf}$	***L.Spread***$_{knn}$	***OPFSemi***
MNIST	prop. acc	0.095714 ± 0.000	0.095714 ± 0.000	0.416857 ± 0.005	0.460953 ± 0.016	**0.790000 ± 0.047**
	acc	0.096000 ± 0.000	0.096000 ± 0.000	0.402889 ± 0.013	0.451555 ± 0.015	**0.797778 ± 0.049**
	kappa	0.000000 ± 0.000	0.000000 ± 0.000	0.337752 ± 0.015	0.391369 ± 0.017	**0.775103 ± 0.054**
H.eggs (w/o imp)	prop. acc	0.146591 ± 0.088	0.197251 ± 0.000	0.872811 ± 0.036	0.602263 ± 0.108	**0.983293 ± 0.004**
	acc	0.145637 ± 0.087	0.195857 ± 0.000	0.898933 ± 0.028	0.622097 ± 0.111	**0.970496 ± 0.003**
	kappa	0.000000 ± 0.000	0.000000 ± 0.000	0.879848 ± 0.033	0.547943 ± 0.132	**0.965085 ± 0.003**
P.cysts (w/o imp)	prop. acc	0.186573 ± 0.000	0.186573 ± 0.000	0.264960 ± 0.008	0.472676 ± 0.051	**0.800569 ± 0.035**
	acc	0.186851 ± 0.000	0.186851 ± 0.000	0.256055 ± 0.023	0.472030 ± 0.039	**0.819493 ± 0.041**
	kappa	0.000000 ± 0.000	0.000000 ± 0.000	0.078200 ± 0.015	0.319221 ± 0.041	**0.756949 ± 0.054**
H.larvae	prop. acc	0.127288 ± 0.001	0.126881 ± 0.000	0.306222 ± 0.044	0.609597 ± 0.047	**0.954182 ± 0.008**
	acc	0.127014 ± 0.000	0.127014 ± 0.000	0.272986 ± 0.041	0.634123 ± 0.062	**0.955450 ± 0.002**
	kappa	0.000000 ± 0.000	0.000000 ± 0.000	0.032267 ± 0.027	0.187448 ± 0.043	**0.789743 ± 0.010**
H.eggs	prop. acc	0.069499 ± 0.002	0.052357 ± 0.030	0.482299 ± 0.049	0.621297 ± 0.062	**0.936743 ± 0.011**
	acc	0.067797 ± 0.000	0.050413 ± 0.030	0.454803 ± 0.051	0.636679 ± 0.084	**0.942634 ± 0.016**
	kappa	0.000000 ± 0.000	0.000000 ± 0.000	0.303932 ± 0.073	0.468250 ± 0.094	**0.899307 ± 0.027**
P.cysts	prop. acc	0.079389 ± 0.007	0.075307 ± 0.000	0.464935 ± 0.059	0.421283 ± 0.034	**0.732716 ± 0.056**
	acc	0.075235 ± 0.000	0.075235 ± 0.000	0.471381 ± 0.052	0.425404 ± 0.052	**0.740973 ± 0.056**
	kappa	0.000000 ± 0.000	0.000000 ± 0.000	0.302018 ± 0.061	0.264922 ± 0.050	**0.580626 ± 0.092**
Coconut	prop. acc	0.785262 ± 0.001	0.788061 ± 0.006	**0.821468 ± 0.011**	0.783741 ± 0.022	0.815930 ± 0.036
	acc	0.784163 ± 0.000	0.790691 ± 0.011	0.835107 ± 0.016	0.804030 ± 0.026	**0.839364 ± 0.017**
	kappa	0.000000 ± 0.000	0.068275 ± 0.118	0.402744 ± 0.096	0.275569 ± 0.228	**0.489274 ± 0.092**
COVID19	prop. acc	0.170919 ± 0.000	0.171729 ± 0.001	0.540390 ± 0.041	0.512263 ± 0.066	**0.589487 ± 0.039**
	acc	0.170709 ± 0.000	0.170709 ± 0.000	0.569869 ± 0.051	0.532283 ± 0.051	**0.614173 ± 0.040**
	kappa	0.000000 ± 0.000	0.000000 ± 0.000	0.363335 ± 0.057	0.328030 ± 0.054	**0.407068 ± 0.066**

to *L.Spread*$_{rbf}$ (0.60). Also, the two methods influenced the 2D projection space, which is more circular for *L.Spread*$_{rbf}$ and more more elongated for *OPFSemi*. Still, neither method could separate the feature space into distinct per-class clusters.

For H.larvae, *OPFSemi* was able to propagate labels only for regions with supervised samples of class 1 (red) and also provide a feature space (2D projection) in which class 1 is separated from other samples in the projection. The f1-score confirmed that as both classes have f1 values up to 0.8. In contrast, *L.Spread*$_{rbf}$ was more conservative in propagating labels for class 2 (green) *vs* class 1 (red) so that only the closest samples of class 2 were labeled with that class. The f1-scores for both classes were lower than 0.35, and the learned feature space presented more (and more spread) groups. At a higher level, Fig. 3.3 illustrates how distinct ways of propagating labels intervene in the learned feature space produced by the proposed *orig-DeepFA* looping.

3.4.3 *Q3: Sample selection: adding OPFSemi's confidence*

To analyze the impact of OPFSemi's confidence in the *orig-DeepFA* looping, we compared VGG-16 using different settings of OPFSemi's label propagation, with and without confidence, after five iterations, by the following experiments:

- *orig-DeepFA*: VGG-16 is trained on S. Deep features for $S \cup U$ from the last convolutional layer are projected in 2D with t-SNE and used by OPFSemi to propagate labels from S to *all* samples in U. VGG-16 is retrained from $S \cup U$ and tested on T;

- *conf-DeepFA*$_{\tau=x}$: As above but selecting pseudo-labeled samples $U_x \subset U$ with confidence $\tau \geq x$ to retrain VGG-16 from $S \cup U_x$, with $x \in \{0.7, 0.8, 0.9\}$.

- *conf-DeepFA*$_{\tau=\alpha}$: As above but starting with $\alpha = 0.8$ and increasing it by 0.04 at each iteration until a final value $\alpha = 0.96$.

3.4.3.1 *Results*

Table 6 shows the mean label propagation accuracy, classification accuracy, κ, and their standard deviation over three splits of our experiments. For all datasets, we see that selecting the most confident samples by OPFSemi during the *orig-DeepFA* looping improves the results. For *H.eggs* with and without impurities, the best results were obtained for $\tau = 0.7$. For MNIST, *H.larvae*, *P.cysts* and *COVID19*, the best results occurred for $\tau = 0.8$. For *P.cysts* without impurities, $\tau = 0.9$ and $\tau = \alpha$ led to the best (similar) results. Finally, $\tau = 0.9$ was the best choice for Coconut.

We conclude that confidence-based sampling shows clear added value in nearly all situations, as it increased κ up to 0.6 for all datasets (except *COVID19*). Setting τ is a dataset-dependent task. The adaptive ($\tau = \alpha$) confidence strategy does not seem to improve results on the test set compared to *orig-DeepFA* with no confidence sampling. One explanation can be our use of more samples in early iterations ($\tau = 0.8$) than in the later ones ($\tau = 0.96$). Testing whether the opposite strategy improves results is left for future study.

3.4.3.2 *Discussion*

Confidence-based sampling in the *orig-DeepFA* looping: Figure 3.4 shows the average κ and propagation accuracy for *DeepFA* looping with full pseudolabeling of all samples (*orig-DeepFA*), our proposed *conf-DeepFA* using OPFSemi's confidence sampling for pseudolabeling with different ways to select the confidence threshold τ, and the best result for the VGG-16 experiments (*self-VGG*$_{fe}$, see Sec. 3.4.1), for all six studied datasets. For datasets

Table 6: (Q3) Results from the last iteration for proposed experiments with full label propagation (*orig-DeepFA*), and confidence-based label propagation (*conf-DeepFA*) with confidence $\tau \in \{0.7, 0.8, 0.9\}$ and adaptive confidence ($alpha \in [0.80, 0.96]$ over 5 iterations). Best values per dataset in bold.

		***DeepFA* variants**				
dataset	**metric**	***orig-DeepFA***	***conf-DeepFA*** $_{\tau=0.7}$	***conf-DeepFA*** $_{\tau=0.8}$	***conf-DeepFA*** $_{\tau=0.9}$	***conf-DeepFA*** $_{\tau=\alpha}$
MNIST	prop. acc	0.790000 ± 0.047	0.782286 ± 0.029	**0.821714 ± 0.018**	0.750000 ± 0.028	0.795429 ± 0.007
	acc	0.797778 ± 0.049	0.788000 ± 0.030	**0.822666 ± 0.022**	0.740222 ± 0.032	0.651778 ± 0.062
	kappa	0.775103 ± 0.054	0.764348 ± 0.034	**0.802863 ± 0.024**	0.710961 ± 0.036	0.612766 ± 0.069
H.eggs (w/o imp)	prop. acc	**0.983293 ± 0.004**	0.983832 ± 0.002	0.974401 ± 0.020	0.981945 ± 0.003	0.983832 ± 0.004
	acc	0.790961 ± 0.050	**0.973007 ± 0.006**	0.971123 ± 0.013	0.938481 ± 0.056	0.806654 ± 0.126
	kappa	0.752807 ± 0.060	**0.968042 ± 0.007**	0.965848 ± 0.015	0.927708 ± 0.066	0.771216 ± 0.148
P.cysts (w/o imp)	prop. acc	0.800569 ± 0.035	0.805143 ± 0.049	0.793274 ± 0.069	0.824060 ± 0.019	**0.828141 ± 0.012**
	acc	0.819493 ± 0.041	0.826413 ± 0.039	0.814590 ± 0.060	**0.842561 ± 0.004**	0.824394 ± 0.033
	kappa	0.756949 ± 0.054	0.764035 ± 0.052	0.747127 ± 0.086	**0.785441 ± 0.006**	0.762919 ± 0.041
H.larvae	prop. acc	0.954182 ± 0.008	0.964213 ± 0.017	**0.964349 ± 0.012**	0.941846 ± 0.039	0.951471 ± 0.014
	acc	0.955450 ± 0.002	0.959558 ± 0.015	**0.965561 ± 0.004**	0.958926 ± 0.014	0.943128 ± 0.010
	kappa	0.789743 ± 0.010	0.800052 ± 0.099	**0.837948 ± 0.029**	0.804689 ± 0.082	0.705475 ± 0.069
H.eggs	prop. acc	0.936743 ± 0.011	0.936091 ± 0.005	**0.937209 ± 0.008**	0.931806 ± 0.007	0.930967 ± 0.006
	acc	0.942634 ± 0.016	**0.943938 ± 0.003**	0.942634 ± 0.009	0.908518 ± 0.022	0.853107 ± 0.025
	kappa	0.899307 ± 0.027	**0.901604 ± 0.006**	0.898922 ± 0.015	0.831488 ± 0.043	0.719695 ± 0.054
P.cysts	prop. acc	0.732716 ± 0.056	0.769748 ± 0.026	**0.780300 ± 0.018**	0.748843 ± 0.048	0.744811 ± 0.068
	acc	0.740973 ± 0.056	0.792755 ± 0.027	**0.816905 ± 0.027**	0.818066 ± 0.022	0.731104 ± 0.082
	kappa	0.580626 ± 0.092	0.652254 ± 0.051	**0.699603 ± 0.054**	0.689325 ± 0.039	0.450283 ± 0.243
Coconut	prop. acc	0.815930 ± 0.036	0.834003 ± 0.038	0.837349 ± 0.022	**0.856760 ± 0.035**	0.791165 ± 0.034
	acc	0.839364 ± 0.017	0.853696 ± 0.019	0.876827 ± 0.012	**0.880232 ± 0.006**	0.820633 ± 0.012
	kappa	0.489274 ± 0.092	0.481834 ± 0.075	0.603094 ± 0.076	**0.621104 ± 0.026**	0.346671 ± 0.058
COVID19	prop. acc	0.589487 ± 0.039	0.579767 ± 0.048	0.613249 ± 0.018	0.586112 ± 0.049	**0.624770 ± 0.114**
	acc	0.614173 ± 0.040	0.609869 ± 0.059	**0.662257 ± 0.002**	0.647454 ± 0.020	0.647874 ± 0.090
	kappa	0.407068 ± 0.066	0.416806 ± 0.076	**0.478667 ± 0.013**	0.420928 ± 0.026	0.433786 ± 0.087

yielding higher κ values, we see that *orig-DeepFA* obtained similar results to our proposed *conf-DeepFA* method. Yet, we see κ and propagation accuracy gains of almost 5% for the most challenging datasets. For *P.cysts* with impurities, κ gains actually over 10% and propagation accuracy gains over 17% – for which *orig-DeepFA* obtained worse results than VGG-16. In short, combining *DeepFA* with OPFSemi's confidence sampling (*conf-DeepFA* in Fig. 3.4) got the best results for most tested datasets.

Confidence-based sampling in *orig-DeepFA* along iterations: Figure 3.5 shows κ and propagation accuracy for one split of MNIST along five iterations of our experiments. We see that all compared approaches yielded an increase from the first to the second iteration, except *self-VGG-16*$_{fe}$. Also, we see that both κ and propagation accuracy slightly decrease after the third iteration. This may suggest that the proposed method saturates, mainly by the higher decrease in κ despite of propagation accuracy. The learned pseudolabels and the original images can be used as input for a better (known) deep architecture. Figure 3.6 shows the plot for train and validation loss and accuracy considering 20% (from S) as validation set during one split of MNIST training. The initial learning curve and the learning curves for each iteration are also shown. The learning curves show that the labeled sam-

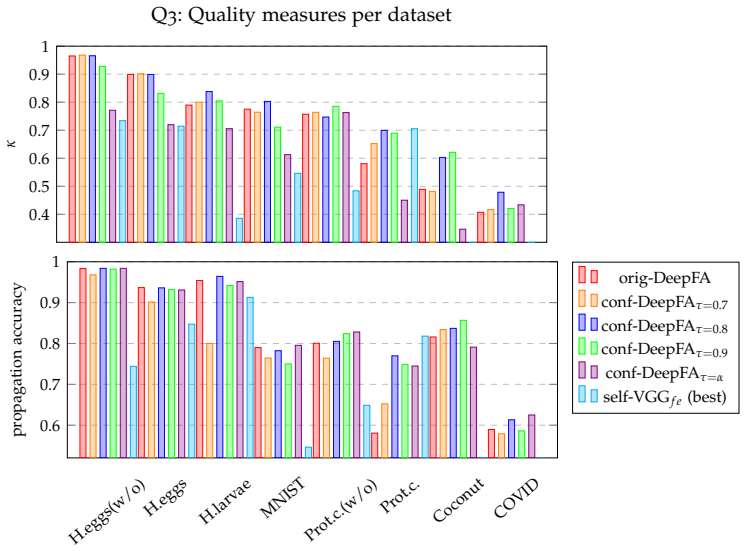


Figure 3.4: (Q3) Results of κ (top) and propagation accuracy (bottom) for the studied datasets, considering self-VGG-16$_{fe}$ (best result), *orig-DeepFA*, and *conf-DeepFA*$_{\tau}$ experiments. The datasets are ordered by higher κ values in x axis (from left to right).

ples can improve the network convergence along iterations. As future work, a different deep network can be tested at the final stage. Also, an unsupervised quality measure can be proposed to define the best feature space found at certain iterations and, hence, the best iteration of the method.

Choosing OPFSemi's confidence threshold: Adaptively selecting the confidence threshold (*conf-DeepFA*$_{\tau=\alpha}$) looks promising only for one of the tested datasets. It shows a higher decreasing in κ when compared with the experiments without changing the confidence threshold τ along the iterations. As Sec. 3.4.2 outlined, choosing OPFSemi's confidence value may depend on the dataset, its difficulty, number of samples, number of classes, and class imbalance. Figure 3.4 also shows this: It is not possible to define a *single* threshold τ for *all* chosen datasets. While this fact has been already noted in (Benato et al., 2021c), it was not studied within a looping of data annotation as we did here. Rather, in (Benato et al., 2021c), user interaction was employed to define the best confidence value based on the 2D projection guided by the data distribution and OPFSemi's confidence values (mapped to colors). We next intend to follow the same strategy to find the best confidence value for *conf-DeepFA* looping.

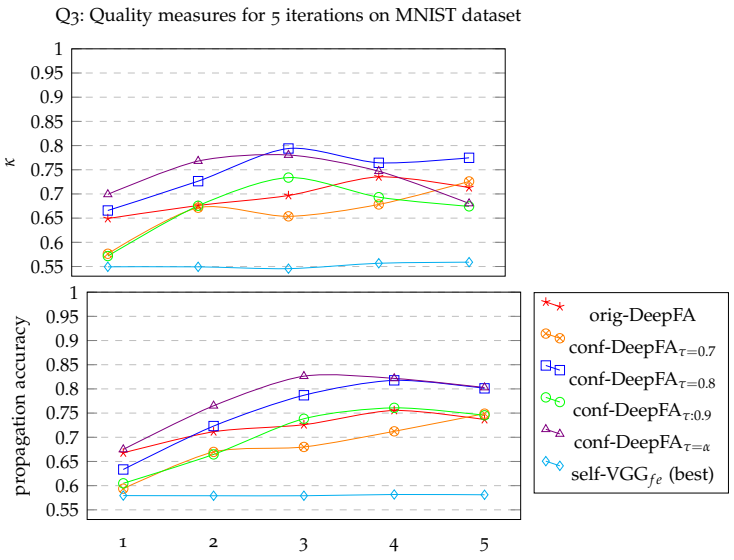


Figure 3.5: (Q3) Results of κ (top) and propagation accuracies (bottom) for the MNIST dataset in one split over 5 iterations, considering self-VGG-16$_{fe}$ (best result), *orig-DeepFA*, and *conf-DeepFA* experiments.

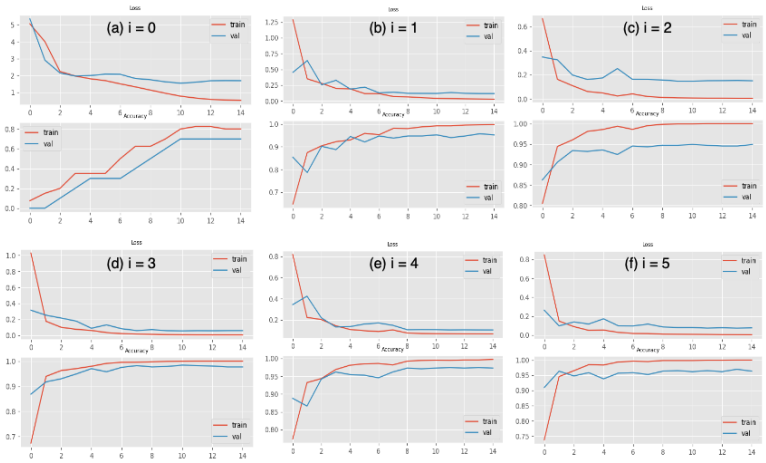


Figure 3.6: Q3) Plots of loss and accuracy for one split of MNIST. The (a) initial learning curves and per-iteration curves (b-f) are shown.

3.4.4 *Q4: Choosing the deep architecture*

As *COVID19* dataset's results showed the lower κ values in the realized experiments (Sec. 3.4.1, 3.4.2, and 3.4.3), we also aim to

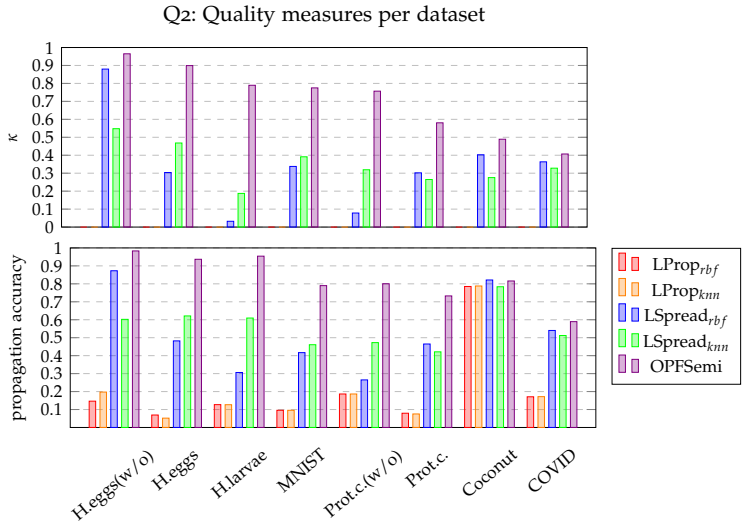


Figure 3.7: (Q2) Results of κ (top) and propagation accuracies (bottom) for the studied datasets with orig-DeepFA using $L.Prop_{rbf}$, $L.Prop_{knn}$, $L.Spread_{rbf}$, $L.Spread_{knn}$, and OPFSemi. Datasets are ordered by higher κ values on the x axis.

investigate the impact of VGG-16 architecture as feature learning strategy using labeled samples in the *DeepFA* looping. For this, we replace VGG-16 (Simonyan and Zisserman, 2014) by MobileNetV2 (Sandler et al., 2018) (which has a reduced training time and good performance) and use ImageNet's pre-trained weights. We compared the two architectures by the following experiments, each of them executed on both architectures $A \in \{VGG\text{-}16, MobileNetV2\}$:

- *orig-DeepFA*: The architecture A is trained on S. Deep features for $S \cup U$ from the last convolutional layer are projected in 2D with t-SNE and used by OPFSemi to pseudo label from S to all U samples. VGG-16 is retrained on these pseudo labels and tested on T (this is one iteration of *DeepFA* looping out of five);
- *conf-DeepFA*$_{\tau=0.8}$: As above, but OPFSemi pseudo labels from S to U_τ, for samples with confidence above $\tau = 0.8$.

3.4.4.1 *Results*

Table 7 shows the means of label propagation accuracy, classification accuracy, κ, and their standard deviation over three splits af-

ter five iterations. For COVID19, the pattern so far observed using VGG-16 was not reproduced by MobileNetV2 – i.e., a higher label propagation accuracy does not lead to the highest accuracy and κ values on the test set. Also, *conf-DeepFA* with $\tau = 0.8$ could not outperform *orig-DeepFA* using *MobileNetV2*. Different values of τ can be tested to validate that pattern. As such, for this dataset, we may conclude that MobileNetV2 was not able to learn a feature space in which OPFSemi propagates better labels than VGG-16.

Table 7: (Q4) Results from the last iteration for proposed experiments using VGG-16 and MobileNetV2 architectures with five learning iterations. Best values per dataset in bold.

dataset	metric	distinct architectures			
		VGG-16		MobileNetV2	
		orig-DeepFA	*conf-DeepFA*$_{\tau=0.8}$	*orig-DeepFA*	*conf-DeepFA*$_{\tau=0.8}$
COVID19	prop. acc	0.589487 ± 0.039	0.613249 ± 0.048	0.643580 ± 0.0385	**0.669839 ± 0.0100**
	acc	0.614173 ± 0.040	**0.662257 ± 0.002**	0.541732 ± 0.0528	0.539685 ± 0.0096
	kappa	0.407068 ± 0.066	**0.478667 ± 0.013**	0.212528 ± 0.0628	0.147655 ± 0.0369

3.4.4.2 *Discussion*

We investigate the MobileNetV2's feature space for the COVID19 dataset by showing its 2D projection and labeled samples (Fig. 3.8). We first notice that the projection presents a mixture between different classes. The supervised samples (colored points) are not clearly separated from the unsupervised ones (black), showing that this is a challenging dataset. For VGG-16, *conf-DeepFA* shows more groups having the same color, *e.g.*, the small groups around the larger one and red points grouped at the center. In contrast, *orig-DeepFA* shows a projection with more colors mixed in small groups and also red points are spread all over the larger group. For MobileNetV2, there are no clear differences in the produced feature space; for the *orig-DeepFA* experiment, the red points seem to be more grouped at the projection center. MobileNetV2 shows fewer small groups having the same color around the larger group than VGG-16. This correlates with the accuracy and κ results for both architectures. It is possible that MobileNetV2 could achieve better results if considering more supervised samples. Too few supervised samples (from 10 to 100) for training this deep architecture is still a problem for its convergence in comparison to VGG-16.

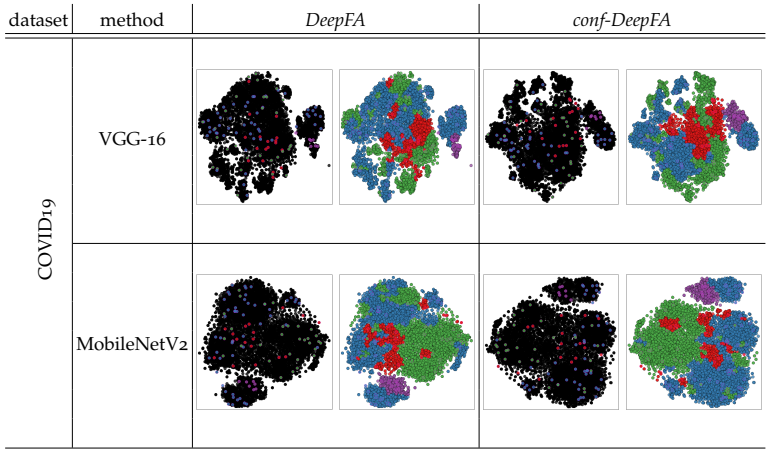


Figure 3.8: Q4) Comparison of the feature spaces generated by *DeepFA* using VGG-16 and MobileNetV2 in pseudolabeling estimation for COVID19, 1% supervised samples and last iteration. 2D feature-space projections of training samples from left to right column per dataset: supervised samples colored by true labels, unsupervised ones are black; and samples colored by assigned pseudolabels.

3.4.5 *Q5: Choosing the layer in the deep architecture*

In the earlier experiments, we only investigate our proposed deep feature learning looping using the last convolutional layer's output. However, Rauber et al. (Rauber et al., 2017b) showed that the multilayer perceptron (mlp) layers, located after the convolutional layers, can create a 2D projected space with better separation among different classes for shallow architectures and without loading pre-trained weights. To test this for our *DeepFA* looping, we compare the result of deep features provided by the output of (i) the last convolutional layer and (ii) the last mlp layer. To facilitate our analysis, we use a fixed value of τ for *conf-DeepFA* and five iterations for all datasets. Instead of showing the propagation accuracy (which does not consider the class unbalance) we compute the propagation κ for the labeled samples in training set. Our experiments are described below:

- L_{conv}: VGG-16 is trained on S. Features from the last convolutional layer for samples in $S \cup U$ are projected in 2D with t-SNE and used by OPFSemi to pseudo labels from S to U_τ, for $\tau = 0.8$. VGG-16 is retrained on these pseudo labels and

tested on T (this is one iteration of *conf-DeepFA* looping out of five);

- L_{mlp}: As above, but using features from the last hidden fully-connected layer in the DNN.

3.4.5.1 *Results and discussion*

Table 8 shows the mean label propagation accuracy, classification accuracy, κ, and their standard deviation over three splits after five iterations. For MNIST, H.eggs, and Coconut, using the last convolutional layer obtained the best results. In contrast, for H.eggs without impurities, P.cysts without impurities, H.larvae, and COVID, the last mlp layer obtained the best results. For P.cysts, the results of using either L_{conv} or L_{mlp} were similar when considering the standard deviation. The choice of the layer to be used in our *conf-DeepFA* looping also affect the κ results on the test set. All in all, the choice of which layer to use seems to depend on the dataset.

3.4.6 *Q6: Choosing the best* DeepFA *model and iteration*

As in earlier work, all our experiments so far used only a fixed number of (five) iterations. However, as Sec. 3.4.3 and Fig. 3.5 show, there is no guarantee that the last iteration delivers the best model. Following Sec. 3.2.6, we propose to evaluate the produced feature space in each iteration by using an unsupervised clustering metric. For this, we select the Calinski-Harabasz Index (CHI) (Caliński and Harabasz, 1974) which calculates the ratio of the sum of between-cluster and sum of within-cluster dispersion for all clusters, where cluster dispersion is defined as the sum of squared distances over the cluster points. Formally put

$$CHI = \frac{\frac{\sum_{k=1}^{K} n_k ||c_k - c||^2}{K-1}}{\frac{\sum_{k=1}^{K} \sum_{i=1}^{n_k} ||x_i - c_k||^2}{N-K}}, \tag{3.1}$$

where x_i a sample in a dataset with N samples; K the number of clusters; n_k is the number of samples in cluster k; c_k is the centroid of cluster k; and c is the global centroid of all samples. As CHI is high when the obtained clusters are dense and well separated from each other, we propose to choose the best label estimation in the produced feature space by selecting that one which achieves the best CHI value after several iterations of our method.

Table 8: Q5) Results from the last iteration for experiments comparing the label propagation in the t-SNE projected space of distinct layers L_{conv} (the last convolutional layer's output) and L_{mlp} (the last MLP layer's output) during five iterations of conf-DeepFA looping with $\tau = 0.8$. Best values per dataset in bold.

		distinct output layers	
dataset	**metrics**	L_{conv}	L_{mlp}
MNIST	prop. kappa	**0.560920 ± 0.317**	0.523341 ± 0.058
	acc	**0.605778 ± 0.282**	0.584000 ± 0.057
	kappa	**0.561751 ± 0.314**	0.537411 ± 0.063
H.eggs (w/o imp)	prop. kappa	**0.980513 ± 0.001**	0.977636 ± 0.003
	acc	0.976146 ± 0.008	**0.977401 ± 0.000**
	kappa	0.971776 ± 0.009	**0.973232 ± 0.000**
P.cysts (w/o imp)	prop. kappa	0.773862 ± 0.015	**0.807765 ± 0.043**
	acc	0.851211 ± 0.011	**0.861592 ± 0.033**
	kappa	0.797588 ± 0.018	**0.810277 ± 0.048**
H.larvae	prop. kappa	0.819094 ± 0.043	**0.824582 ± 0.029**
	acc	0.958610 ± 0.002	**0.961137 ± 0.005**
	kappa	0.807432 ± 0.017	**0.823241 ± 0.040**
H.eggs	prop. kappa	**0.904969 ± 0.003**	0.861352 ± 0.033
	acc	**0.945024 ± 0.006**	0.930465 ± 0.015
	kappa	**0.902488 ± 0.010**	0.875675 ± 0.025
P.cysts	prop. kappa	0.631756 ± 0.057	**0.652409 ± 0.060**
	acc	0.808545 ± 0.036	**0.812609 ± 0.026**
	kappa	**0.682719 ± 0.070**	0.681252 ± 0.051
Coconut	prop. kappa	**0.623280 ± 0.057**	0.514769 ± 0.114
	acc	**0.887186 ± 0.018**	0.872996 ± 0.024
	kappa	**0.652869 ± 0.060**	0.577084 ± 0.116
COVID19	prop. kappa	0.430679 ± 0.036	**0.516374 ± 0.034**
	acc	0.672126 ± 0.041	**0.699108 ± 0.033**
	kappa	0.490593 ± 0.059	**0.546634 ± 0.043**

To ease our analysis, we use a fixed $\tau = 0.8$ and *ten* total iterations for all datasets. Also, we compare the result of deep features provided by distinct layers (L) from the output of (i) the last convolutional layer (L_{conv}) and (ii) the last multilayer-perceptron layer (L_{mlp}). We compute the propagation κ instead of propagation accuracy for the labeled samples in the training set. The experiments are described below:

- *conf-DeepFA*$_{iter=5}$: VGG-16 is trained on S. Deep features for $S \cup U$ from L are projected in 2D with t-SNE, and used for OPFSemi's pseudo-labeling from S to U_τ, for samples with confidence above $\tau = 0.8$. OPFSemi's pseudo labels are used to retrain VGG-16, and the network is tested on T (this is one iteration of *conf-DeepFA* looping out of *five*);
- *conf-DeepFA*$_{iter=best}$: As above, but we compute CHI on the 2D projections (over *ten* iterations) and select the model obtained from the iteration with the highest CHI value;
- *conf-DeepFA*$_{iter=worst}$: As above, but we select the iteration with the lowest CHI value.

3.4.6.1 *Results*

Table 9 shows the mean label propagation accuracy, classification accuracy, κ, and their standard deviation over three splits after five iterations, for the best among ten iterations, and for the worst among ten iterations. First, we consistently see that the highest CHI values lead to the best classification results (accuracy and κ) on the test set, except for P.cysts with and without impurities (for L_{conv}) and MNIST (for L_{mlp}). Also, the lowest CHI values lead to the worst classification results on the test set. The iteration with the best CHI value yields better results than those that we have found so far in our earlier experiments (last of five iterations). For MNIST, H.eggs with and without impurities, Coconut, and COVID19, the best iteration using L_{conv} obtains better classification results than when using L_{mlp}. For this dataset, using L_{conv} with the highest-CHI iteration selection strategy increases κ by 0.15 as compared to using the last iteration.

3.4.6.2 *Discussion*

Using CHI to select the labeled feature space: Figure 3.9 shows CHI and κ for the last five, best, and worst iterations for L_{conv} and L_{mlp}. The worst values (red) are always lower than the last-five iterations (orange) and also lower than the best values (green) for

Table 9: Q5) Results for experiments comparing the label propagation in the t-SNE projected space of distinct layers (a) L_{conv} (last convolutional layer's output), and (b) L_{mlp} (last MLP layer's output) during *ten iterations* of *conf-DeepFA* looping with $\tau = 0.8$. The result for five iterations, and both the best and worst iterations chosen by the CHI value are presented. Best values per dataset and per layer L in bold.

dataset	metric	L_{conv}			L_{mlp}		
		iter= 5	iter=worst	iter=best	iter= 5	iter=worst	iter=best
MNIST	CHI	3805.08 ± 2544.0	1582.67 ± 1173.2	**5430.51 ± 2356.6**	2807.49 ± 1172.5	1104.40 ± 198.9	**4055.42 ± 83.5**
	prop. kappa	0.560920 ± 0.317	0.423657 ± 0.310	**0.704985 ± 0.112**	0.523341 ± 0.058	0.379667 ± 0.056	**0.599035 ± 0.023**
	acc	0.605778 ± 0.282	0.533556 ± 0.282	**0.743556 ± 0.078**	**0.584000 ± 0.057**	0.462000 ± 0.069	0.548000 ± 0.163
	kappa	0.561751 ± 0.314	0.479521 ± 0.316	**0.714611 ± 0.087**	**0.537411 ± 0.063**	0.401889 ± 0.076	0.496539 ± 0.183
H.eggs (w/o imp)	CHI	6161.35 ± 766.6	4200.59 ± 646.2	**6729.25 ± 413.4**	**7599.88 ± 308.6**	3434.74 ± 740.5	**7599.88 ± 308.6**
	prop. kappa	0.980513 ± 0.001	0.978596 ± 0.006	**0.981472 ± 0.002**	**0.977636 ± 0.003**	0.961359 ± 0.008	**0.977636 ± 0.003**
	acc	0.976146 ± 0.008	0.975518 ± 0.007	**0.979284 ± 0.005**	**0.977401 ± 0.000**	0.972379 ± 0.007	**0.977401 ± 0.000**
	kappa	0.971776 ± 0.009	0.971012 ± 0.008	**0.975469 ± 0.006**	**0.973232 ± 0.000**	0.967319 ± 0.008	**0.973232 ± 0.000**
P.cysts (w/o imp)	CHI	2447.19 ± 209.7	1453.61 ± 511.9	**3098.45 ± 219.9**	2786.55 ± 198.5	1854.04 ± 253.9	**3534.84 ± 260.9**
	prop. kappa	**0.773862 ± 0.015**	0.672889 ± 0.062	0.684681 ± 0.032	**0.807765 ± 0.043**	0.680152 ± 0.076	0.791739 ± 0.065
	acc	**0.851211 ± 0.011**	0.793829 ± 0.036	0.791811 ± 0.025	0.861592 ± 0.033	0.816609 ± 0.041	**0.863898 ± 0.031**
	kappa	**0.797588 ± 0.018**	0.712134 ± 0.059	0.729677 ± 0.026	0.810277 ± 0.048	0.750887 ± 0.055	**0.817631 ± 0.039**
H.larvae	CHI	774.15 ± 214.7	499.97 ± 47.4	**920.94 ± 340.6**	5546.44 ± 8201.1	570.23 ± 313.8	**8664.83 ± 12964.6**
	prop. kappa	0.819094 ± 0.043	0.775107 ± 0.050	**0.824502 ± 0.038**	0.824582 ± 0.029	0.796519 ± 0.009	**0.830402 ± 0.042**
	acc	0.958610 ± 0.002	0.952607 ± 0.009	**0.959242 ± 0.006**	0.961137 ± 0.005	0.955134 ± 0.006	**0.964613 ± 0.005**
	kappa	**0.807432 ± 0.017**	0.755035 ± 0.068	0.806115 ± 0.031	0.823241 ± 0.040	0.785666 ± 0.054	**0.840656 ± 0.035**
H.eggs	CHI	1041.17 ± 8.9	724.86 ± 54.2	**1108.63 ± 37.9**	1187.05 ± 47.2	846.19 ± 31.8	**1319.95 ± 72.3**
	prop. kappa	**0.904969 ± 0.003**	0.837934 ± 0.031	0.885585 ± 0.014	0.861352 ± 0.033	0.810120 ± 0.012	**0.870839 ± 0.019**
	acc	0.945024 ± 0.006	0.917427 ± 0.022	**0.945676 ± 0.005**	0.930465 ± 0.015	0.915906 ± 0.014	**0.931986 ± 0.014**
	kappa	0.902488 ± 0.010	0.851161 ± 0.040	**0.904525 ± 0.009**	0.875675 ± 0.025	0.847946 ± 0.027	**0.878822 ± 0.024**
P.cysts	CHI	842.15 ± 212.7	737.011 ± 256.6	**1329.19 ± 115.3**	1590.07 ± 128.3	1155.53 ± 289.7	**1861.46 ± 197.0**
	prop. kappa	**0.631756 ± 0.057**	0.626103 ± 0.075	0.631550 ± 0.063	0.652409 ± 0.060	0.632259 ± 0.017	**0.668595 ± 0.040**
	acc	0.808545 ± 0.036	**0.811796 ± 0.021**	0.792523 ± 0.039	0.812609 ± 0.026	0.800882 ± 0.032	**0.813886 ± 0.020**
	kappa	**0.682719 ± 0.070**	0.682633 ± 0.055	0.665641 ± 0.056	0.681252 ± 0.051	0.668188 ± 0.052	**0.696163 ± 0.029**
Coconut	CHI	4043.65 ± 1101.3	1338.66 ± 707.1	**5028.79 ± 1068.9**	2811.05 ± 1232.2	930.54 ± 762.9	**4469.81 ± 1436.5**
	prop. kappa	0.623280 ± 0.057	0.448743 ± 0.125	**0.623487 ± 0.105**	0.514769 ± 0.114	0.370199 ± 0.228	**0.571377 ± 0.092**
	acc	**0.887186 ± 0.018**	0.863914 ± 0.016	0.886760 ± 0.015	0.872996 ± 0.024	0.834823 ± 0.041	**0.879239 ± 0.013**
	kappa	0.652869 ± 0.060	0.537551 ± 0.071	**0.669792 ± 0.040**	0.577084 ± 0.116	0.379283 ± 0.274	**0.634328 ± 0.043**
COVID19	CHI	1112.59 ± 512.8	511.67 ± 110.9	**1516.63 ± 270.7**	3712.00 ± 951.4	1827.87 ± 677.0	**4830.30 ± 882.7**
	prop. kappa	0.430679 ± 0.036	0.410851 ± 0.046	**0.494343 ± 0.025**	0.516374 ± 0.034	0.360516 ± 0.014	**0.538565 ± 0.034**
	acc	0.672126 ± 0.041	0.646614 ± 0.046	**0.713491 ± 0.012**	0.699108 ± 0.033	0.634016 ± 0.032	**0.699318 ± 0.033**
	kappa	0.490593 ± 0.059	0.459456 ± 0.056	**0.555074 ± 0.019**	0.546634 ± 0.043	0.436049 ± 0.044	**0.548049 ± 0.044**

both CHI and κ. We conclude that using CHI values computed from the produced 2D feature space in training set and using pseudo-labels (even prone to errors) to select the best and worst iterations yields good classification results on the test set. This supports the choice of the best iteration of *conf-DeepFA* looping even for different datasets. Also, we notice that higher CHI values lead to higher κ values. For example, for MNIST, both CHI and κ are higher for the L_{conv} setup than for the L_{mlp} setup. We also see that L_{mlp} yields to higher CHI (and κ) values than L_{conv} for all datasets, except for MNIST.

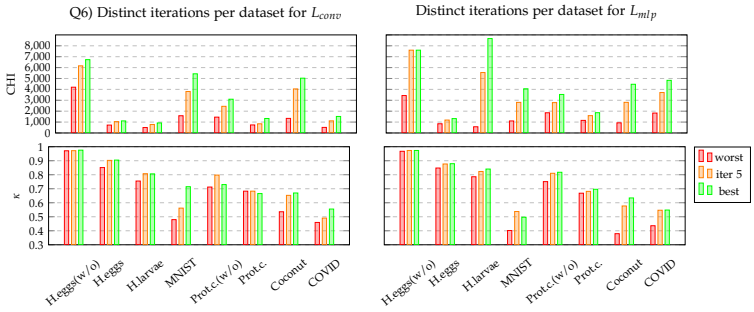


Figure 3.9: Q6) Results of CHI (top) and κ (bottom) for the studied datasets, considering the last convolutional layer (left) and the last mlp layer (right) for *conf-DeepFA* with fixed certainty value ($\tau = 0.8$). The worst (red) and the best (green) iterations out of *ten* iterations were chosen based in the worst and best CHI values on the training set respectively. The results after five iterations (orange) are also shown. The datasets are ordered by higher κ values in x axis (from left to right).

Added-value of CHI evaluation in *conf-DeepFA:* Figure 3.10 plots, for each dataset, the best κ chosen by the best CHI value divided by the highest κ obtained on the test set for both L_{conv} and L_{mlp}. We see that the CHI-based selection approach achieves at least 95% of the best possible result in the test set for all datasets. This confirms the added-value of using CHI for choosing the best iteration of *conf-DeepFA*.

3.5 ANSWERS TO THE STUDIED QUESTIONS

Figure 3.11 summarizes our targeted questions, performed experiments, and best obtained results per question. We evaluate (Q1) the feature space generated by VGG-16 by feature extraction and fine-tuning strategies and compare (Q2) OPFSemi label propaga-

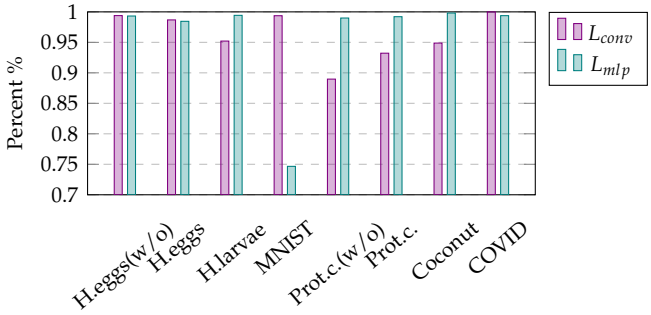


Figure 3.10: Q6) Effectiveness of using CHI for choosing the best iteration in *ext-DeepFA* given by the best κ chosen by the best CHI in each iteration over the best possible κ in the test set.

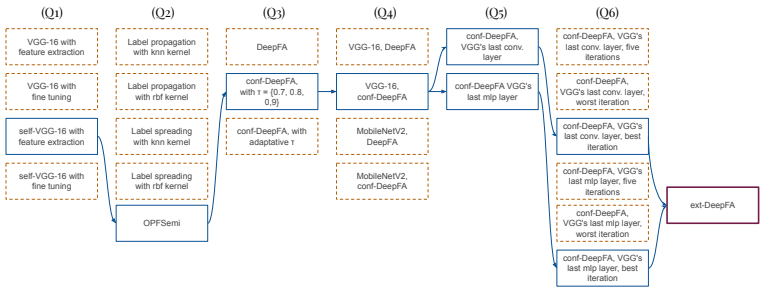


Figure 3.11: Brief summary of the raised questions. For each addressed question, the compared experiments (orange, dashed) and the best result (blue, solid) are presented. Each resulting answer leads and composes the final *ext-DeepFA*.

tion with other semi-supervised methods within the annotation looping. Next, we include (Q3) a confidence sampling strategy to OPFSemi's pseudo-labeling to define the most confident samples for training VGG-16 and evaluate (Q4) a feature space produced by MobileNetV2, a recent deep architecture with much fewer parameters to optimize. We also compare (Q5) two feature spaces provided by distinct layers of the deep network and investigate (Q6) the usage of a clustering metric to choose the best iteration in the proposed looping.

Considering the proposed experimental setup for *orig-DeepFA*, *conf-DeepFA*, and *ext-DeepFA* evaluation, we show that (Q1) self-trained VGG-16 models based on feature extraction only are usually better than fine-tuned models. (Q2) OPFSemi label propaga-

tion yielded the best performance on the 2D projected space for all tested datasets. (Q3) Confidence-based sampling showed clear added value, with different confidence values for each dataset though. (Q4) MobileNetV2 was not able to learn a feature space in which OPFSemi propagated better labels than VGG-16. (Q5) Propagating labels in different layers of the deep architecture led to different label propagation and classification accuracies depending on the dataset. (Q6) High values of the chosen clustering metric consistently led to the best propagation accuracy and classification results.

3.6 LIMITATIONS

We can split our limitations into those related to the experimental validation and the proposed technique. In validating our work, we explored eight datasets (from toy to real scenarios), two deep-learning approaches (VGG-16 and MobileNetV2), five iterations of the deep annotation loop, and one projection method (t-SNE). Exploring more combinations of such techniques is definitely of extra added value. Using more than five looping iterations could help to understand how much the learned feature space can be improved.

Related to the limitations of the proposed technique, we see that the selection of the confidence threshold τ and layer depth may vary on the dataset. To solve that observed dataset dependency, we plan to include user knowledge to select those values and layers. Although we have shown that the 2D projection provided by the t-SNE algorithm is suitable to offer relevant information to OPFSemi's label propagation, the t-SNE projection errors were not considered to improve the performed label propagation and feature learning. Additionally, using the t-SNE algorithm can be an issue due to its scalability: projecting more than dozens of thousands of samples can cost minutes. We intend to analyze further the impact of the t-SNE projection errors and other projection techniques in our proposed pipeline.

3.7 CONCLUSION

In this chapter, we have proposed a methodology, called *DeepFA*, for labeling unsupervised samples for creating high-quality classifiers for image data. For this, we designed and evaluated several variants of the *DeepFA* approach, as follows. Our first ver-

sion, called *orig-DeepFA*, labels unsupervised samples to increase the quality of image classification and extracted feature spaces by using very few supervised samples and many unsupervised ones for training. To cover some of the limitations *orig-DeepFA*, we designed two subsequent variants, called *conf-DeepFA* and *ext-DeepFA*. We drove our design by evaluating six questions that aim to explore the space of possible improvements.

Our investigation led to several findings. First, we showed that OPFSemi's label propagation by minimum graph paths conducted over t-SNE projections is better than other label propagation methods, even those using graphs, neighborhood distances, or kernel tricks. Using a confidence sampling strategy in *conf-DeepFA*, we selected the best-labeled samples to retrain the model and minimized the effect of wrongly assigned labels in the learned feature space. Next, in *ext-DeepFA*, we used – to our knowledge – for the first time the Calinski-Harabasz Index (CHI) to evaluate the learned feature space and pseudo-labels of t-SNE projected features in a 2D space. CHI reveals a correlation among the best 2D feature space, best pseudo-labels, and best classification results, even if these are pseudo-labels prone to errors. When using CHI values to choose the best iteration, we achieve at least 95% of the best κ value achievable in the test set. This insight opens new ways for using CHI as an evaluation strategy in the learned and projected feature space and its pseudo labels.

To solve the observed dataset dependency in the selection of the confidence threshold τ and layer depth, one can next include user knowledge to provide a semi-automatic pseudo-labeling along the lines in (Benato et al., 2021c), but now considering the proposed looping of the deep feature annotation method. Also, considering that CHI can provide relevant information about the deep annotated features, this can be used to support the user in labeling in a semi-automatic fashion. In this respect, a direction for future work is to consider self-supervised learning strategies in deep feature learning to improve the learned feature space. Additionally, an interesting idea is to evaluate other projection methods (beyond t-SNE) with high neighborhood preservation qualities for potential label propagation improvement. Ultimately, we expect that our automatic-and-interactive deep feature-based pseudo-labeling combination will lead to higher quality and more explainable deep learning methods.

4

LINKING DATA SEPARATION, VISUAL SEPARATION, AND CLASSIFIER PERFORMANCE

4.1 INTRODUCTION

While supervised learning has achieved great success, using datasets with either (i) few data points or (ii) few supervised, *i.e.* labeled, points, is fundamentally hard, and especially critical in *e.g.* medical contexts where obtaining (labeled) points is expensive. For (i), methods such as few-shot learning (Sung et al., 2018; Sun et al., 2017), transfer-learning (Russakovsky et al., 2015), and data augmentation have been used to increase the sample count. For (ii), solutions include semi-supervised learning (Iscen et al., 2019; Wu and Prasad, 2018), pseudo-labeling (Lee, 2013; Jing and Tian, 2020), and meta-learning (Pham et al., 2021)[1].

As introduced in Sec. 2.2, pseudo-labeling, also called self-training, takes a training set with few supervised and many unsupervised samples and assigns pseudo-labels to the latter samples – a process known as data annotation – and re-trains the model with all (pseudo)labeled samples. Yet, as the name suggests, pseudo-labels are not perfect, as they are *extrapolated* from actual labels, which can affect training performance (Benato et al., 2018; Arazo et al., 2020). Also, pseudo-labeling methods still require training and validation sets with thousands of supervised samples per class to yield reasonable results (Miyato et al., 2018; Jing and Tian, 2020; Pham et al., 2021).

Both pseudo-labeling, and broader, the success of training a classifier, depend on a key aspect – how easy is the data *separable* into different groups of similar points. Projections, or dimensionality reduction methods, are well known techniques that aim to achieve precisely this (Nonato and Aupetit, 2018; Espadoto et al., 2019a). Two key observations were made in this respect (as introduced in Sec. 2.3.6 and further discussed in detail in Sec. 4.2):

1 This chapter is a result of the following publications: "Linking data separation, visual separation, and classifier performance using pseudo-labeling by contrastive learning" (Benato et al., 2023c); and "Linking data separation, visual separation, and classifier performance using dimensionality reduction techniques" (Benato et al., 2023b).

O1 Visual separability (VS) in a projection mimics the data separability (DS) in the high dimensional space;

O2 Data separability (DS) is key to achieving high classifier performance (CP);

These observations have been used in several directions, *e.g.*, using projections to assess DS (VS→DS, van der Maaten et al. (2009)); using projections to find which samples get misclassified (VS→CP, Nonato and Aupetit (2018)); increasing DS to get easier-to-interpret projections (DS→VS, Kim et al. (2022b)); using projections to assess classification difficulty (VS→CP, Rauber et al. (2017b,a)); and using projections to build better classifiers (VS→CP, Benato et al. (2018, 2021a)). However, to our knowledge, no work so far has explored the relationship between DS, VS, *and* CP in the context of using pseudo-labeling for machine learning (ML).

We address the above by studying how to generate a high DS using *contrastive learning* approaches which have shown state-of-the-art results (Chen et al., 2020; Grill et al., 2020; He et al., 2020; Khosla et al., 2020) and have surpassed results of (self-and-semi-) supervised methods and even known supervised loss functions such as cross-entropy (Chen et al., 2020). We compare two contrastive learning models, SimCLR (Chen et al., 2020) and SupCon (Khosla et al., 2020), and propose a hybrid approach that combines both. We evaluate DS by measuring CP for a classifier trained with only 1% supervised samples. Then, we evaluate VS fed with the encoder's output of our trained contrastive models. Lastly, we investigate CP by using our above pseudo-labeling to train a deep neural network. We perform all our experiments in the context of a challenging medical application (classifying human intestinal parasites in microscopy images).

Our main contributions are as follows:

C1: We use contrastive learning to reach high DS;

C2: We show that projections constructed from contrastive learning methods (with good DS) lead to a good VS between different classes;

C3: We train classifiers with pseudo-labels generated via good-VS projections to achieve a high CP;

C4 We identify projection techniques for which DS strongly correlates with VS and also techniques for which this does not happen;

C5 We show that good-VS projections are essential for training classifiers that reach a high CP.

Jointly taken, our work brings more evidence that links the observations O1 and O2 mentioned above, *i.e.*, that VS, DS, and CP are strongly correlated and that this correlation, and 2D projections of high-dimensional data, can be effectively *used* to build higher-CP classifiers for the challenging case of training-sets having very few supervised (labeled) points.

4.2 RELATED WORK

4.2.1 *Relationship between data separation, visual separation, and classifier performance*

Relations between VS, DS, and CP have been *partially* explored.

Rauber et al. (2017b) used the VS of a t-SNE (van der Maaten, 2014) projection to gauge the difficulty of a classification task (CP). As described in more detail in Sec. 2.3.6, they found that VS and CS are positively correlated when VS is medium to high but could not infer actionable insights for low-VS projections. Also, they did not address the task of *building* higher-CP classifiers using t-SNE.

In a related vein, Rodrigues et al. (2019) used the VS in projections to construct so-called decision boundary maps (DBMs, see Sec. 2.4). While they did not mention the VS created by projections, the implication is clearly there. Indeed, a good projection of a dataset that exhibits reasonable data separation (DS) should create clusters of 2D points which have reasonably well separated labels, just as in the data. This is also the intuition behind the neighborhood hit projection-quality metric (see Sec. 2.3.4.1). When this happens, *i.e.*, when we have a good-VS projection, then we can construct a DBM which shows (reasonably) well separated decision zones, one per class label. Next, one can use this DBM for interpreting classification performance (CP) for different sample groups. However, just as Rauber et al. (2017b), it is not clear what a low-VS projection does imply, *i.e.*, if this indicates that the visualized classifier (by the DBM) has a poor CP, or simply that the underlying projection and/or inverse projection used to construct the DBM have poor quality.

Kim et al. (2022b,a) showed that one can improve VS by increasing DS, the latter being done by mean shift (Comaniciu and Meer, 2002). However, their aim was to generate easier-to-interpret projections and not use these to build higher-CP classifiers. More-

over, their approach actually changed the input data in ways not easy to control, which raises question as to the interpretability of the resulting projections.

Finally, Benato et al. (2018, 2021c) used the VS of t-SNE projections to create pseudo-labels and train higher-CP classifiers from them (as discussed in Chapter 3). They showed that label propagation in the 2D projection space can lead to higher-CP classifiers than when propagating labels in the data space. Yet, they did not study how correlations between DS and VS can affect CP.

4.2.2 *Self-supervised learning*

Self-supervised contrastive methods in representation learning have been the choice for learning representations without using any labels (Chen et al., 2020; Grill et al., 2020; He et al., 2020; Khosla et al., 2020). Such methods work by using a so-called *contrastive loss* to pull similar pairs of samples closer while pushing apart dissimilar pairs. To select (dis)similar samples without using label information, one can generate multiple views of the data via transformations. For image data, SimCLR (Chen et al., 2020) used transformations such as cropping, Gaussian blur, color jittering, and grayscale bias. MoCo (He et al., 2020) explored a momentum contrast approach to learn a representation from a progressing encoder while increasing the number of dissimilar samples. BYOL (Grill et al., 2020) used only augmentations from similar examples. SimCLR has shown significant advances in (self-and-semi-) supervised alearning and achieved a new record for image classification with few labeled data. Supervised contrastive learning (SupCon) (Khosla et al., 2020) generalized both SimCLR and N-pair losses and was proven to be closely related to triplet loss. SupCon surpasses cross-entropy, margin classifiers, and other self-supervised contrastive learning techniques.

4.3 LINKING DATA SEPARATION, VISUAL SEPARATION, AND CLASSIFIER PERFORMANCE

Following the above, we propose to improve DS in the feature space that EPL takes as input by using two *contrastive learning* models (SimCLR (Chen et al., 2020) and SupCon (Khosla et al., 2020), used both separately and combined) and without using ground-truth labels. The feature space to input in EPL comes from the encoder's output from these contrastive models. During the process, outlined in Fig. 4.1, we test our three claims (Sec. 4.1),

i.e., that DS has improved (C1); that this has led to an improved VS in the 2D projections used by EPL (C2); and finally that the generated pseudo-labels by EPL can be used to train a classifier with high CP (C3). Our method is detailed next.

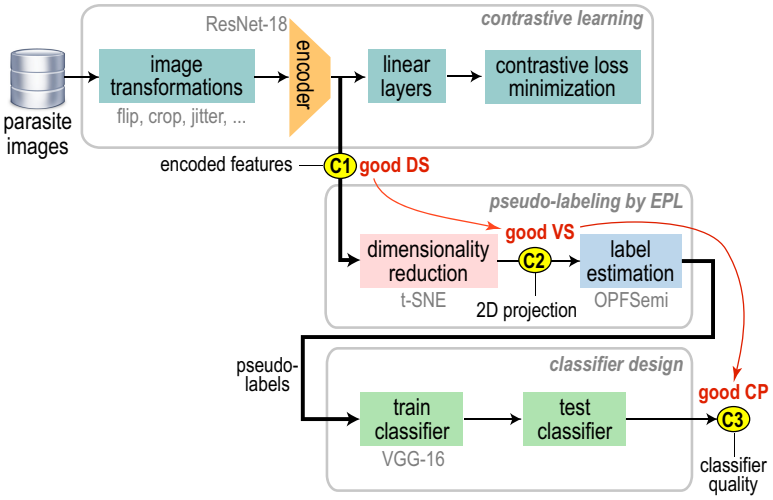


Figure 4.1: We train a model from image transformations of the original data with a contrastive learning loss. Next, we project the latent features from the encoder's output to 2D and pseudo-label the resulting points. Finally, we use these pseudo-labels to train a classifier.

4.3.1 *Contrastive learning*

We generate the latent space to be used by EPL (Fig. 4.1, top box) in three different ways: (a) from the many unsupervised samples available by using SimCLR (Chen et al., 2020); (b) using our 1% supervised samples with SupCon (Khosla et al., 2020); and (c) by combining the SimCLR and SupCon methods.

4.3.2 *Pseudolabeling by EPL*

We use ResNet-18 (He et al., 2016) as bottleneck for both SimCLR and SupCon strategies. The encoder's output of ResNet-18 (hundreds of dimensions) is projected in a lower dimensional space (2D) using different projection techniques (Fig. 4.1, middle gray box). Propagating pseudo-labels in a 2D space was first observed to lead to high-CP classifiers particularly using t-SNE by EPL in (Benato et al., 2021a,b). Here, we explore the same EPL's function

of exploring 2D points to propagate the (few) true labels to all unsupervised points for chosen projection techniques. A detailed description of EPL is given in Chapter 3. Finally, we measure the VS of the 2D scatterplots created by the tested projection techniques by measuring the success of pseudolabeling.

4.3.3 *Classifier training with pseudo-labels*

To finally test the quality of our generated pseudo-labels, we train a deep neural network, namely VGG-16 with ImageNet pretrained weights, and test it on our parasite datasets (Fig. 4.1, bottom box). This architecture was shown to have the best results for our datasets (Osaku et al., 2020).

4.4 EXPERIMENTAL EVALUATION

4.4.1 *Projection methods*

As outlined in Sec. 4.6, we want to evaluate the impact of different projection techniques on the measured DS-VS-CP correlations. For this, we chose the 10 most accessible and easy-to-implement projection techniques from the projection-quality benchmark proposed in (Espadoto et al., 2019a). Table 10 shows our selection, with all techniques available in *scikit-learn*, except UMAP which has a separate Python implementation. Our selection covers linear *vs* nonlinear, and global *vs* local, projections; and also projections taking samples *vs* sample-distances as input. For all techniques, we fixed their parameters to the default values proposed by each author in *scikit-learn*.

4.4.2 *Datasets*

As outlined in Sec. 4.1, we apply our proposed approach in the medical context. We explored (i) *Helminth larvae* (H.larvae); (ii) *Helminth eggs* (H.eggs); and (iii) *Protozoan cysts* (P.cysts). To evaluate different difficulty levels, we also explore (ii) and (iii) without the impurity class, which form our last two datasets. A detailed description of this dataset is provided in Chapter 3.

4.4.3 *Data layout for validation*

As outlined in Sec. 4.1, our main goal it to build a classifier for the chosen datasets exploring only a small set of supervised sam-

Table 10: Projection techniques chosen for our evaluation. For each one, we list the linearity, type of input, and whether the technique is local or global.

projection	linearity	input	local or global
FA Jolliffe (1986)	linear	samples	global
FICA Hyvarinen (1999)	linear	distances	global
ISO Tenenbaum et al. (2000)	nonlinear	distances	local
KPCA Schölkopf et al. (1997)	nonlinear	samples	global
LLE Roweis and Saul (2000)	nonlinear	samples	local
MDS Torgerson (1958)	nonlinear	samples	global
MLLE Zhang and Wang (2006)	nonlinear	samples	local
PCA Jolliffe (1986)	linear	samples	global
t-SNE van der Maaten and Hinton (2008)	nonlinear	samples	local
UMAP McInnes et al. (2018)	nonlinear	distances	local

ples. For this, we split each of the five considered datasets D (Sec. 4.4.2) into a supervised training-set S containing 1% supervised samples from D, an unsupervised training-set U with 69% of the samples in D, and a test set T with 30% of the samples in D (hence, $D = S \cup U \cup T$). We repeat the above division randomly and in a stratified manner to create three distinct splits of D in order to gain statistical relevance when evaluating results next. Table 11 shows the sizes $|S|$ and $|U|$ for each dataset.

Table 11: Number of samples in S and U for each dataset.

	H.eggs (w/o imp)	P. cysts (w/o imp)	H. larvae	H. eggs	P. cysts
S	17	38	35	51	95
U	1220	2658	2424	3527	6602

To measure quality, we compute accuracy and κ (since our datasets are unbalanced). Both metrics are detailed in Chapter 2.

4.4.4 *Implementation details*

We next outline our end-to-end implementation.

Contrastive learning: We implemented SimCLR and SupCon in Python using Pytorch. We generate two augmented images (views) for each original image by random horizontal flip, resized crop (96×96), color jitter (brightness$=0.5$, contrast$=0.5$, saturation$=0.5$, hue$=0.1$) with probability of 0.8, gray-scale

with probability of 0.2, Gaussian blur (9×9), and a normalization of 0.5.

Latent space generation: We replace ResNet-18's decision layer by a linear layer with 4,096 neurons, a ReLU activation layer, and a linear layer with 1,024 neurons respectively. We train the model by backpropagating errors of NT-Xent and SupCon losses for SimCLR and SupCon, respectively, with a fixed temperature of 0.07. We use the AdamW optimizer with a learning rate of 0.0005, weight decay of 0.0001, and a learning rate scheduler using cosine annealing, with a maximum temperature equal to the epochs and minimum learning rate of 0.0005/50. We use 50 epochs and select the best model through a checkpoint obtained from the lowest validation loss during training. Finally, we use the 512 features of the ResNet-18's encoder to obtain our latent space.

Classifier using pseudo-labels: We replace the original VGG-16 classifier with two linear layers with 4,096 neurons followed by ReLU activations and a softmax decision layer. We train the model with the last four layers unfixed by backpropagating errors using categorical cross-entropy. We use stochastic gradient descent with a linear decay learning rate initialized at 0.1 and momentum of 0.9 over 15 epochs.

Parameter setting: OPFSup and OPFSemi, used for pseudo-labeling (Sec. 4.3.2), have no parameters. For Linear SVM and t-SNE (Sec. 4.5.1.1), we use the default parameters provided by scikit-learn.

For replication purposes, all our code and results are made openly available (Benato, B.C., 2022).

4.5 EXPLORING A PROJECTION WITH A GOOD VISUAL SEPARATION

4.5.1 *Proposed experiments*

To describe our experiments, we first introduce a few notations. S, U, and T are the supervised (known labels), unsupervised (to be pseudo-labeled), and test sets (see Sec. 4.4.3). Let I be the images in a given dataset having true labels L and pseudo-labels P. Let F be the latent features obtained by the three contrastive learning methods; and let F' be the features' projection to 2D via t-SNE.

We use subscripts to denote on which subset I, L, P, and F are computed, *e.g.* F_S are the latent features for samples in S. Finally, let A be the initialization strategy for training a classifier C.

Figure 4.2 shows the several experiments we executed to explore the claims C1-C3 listed in Sec. 4.1. These experiments are detailed next.

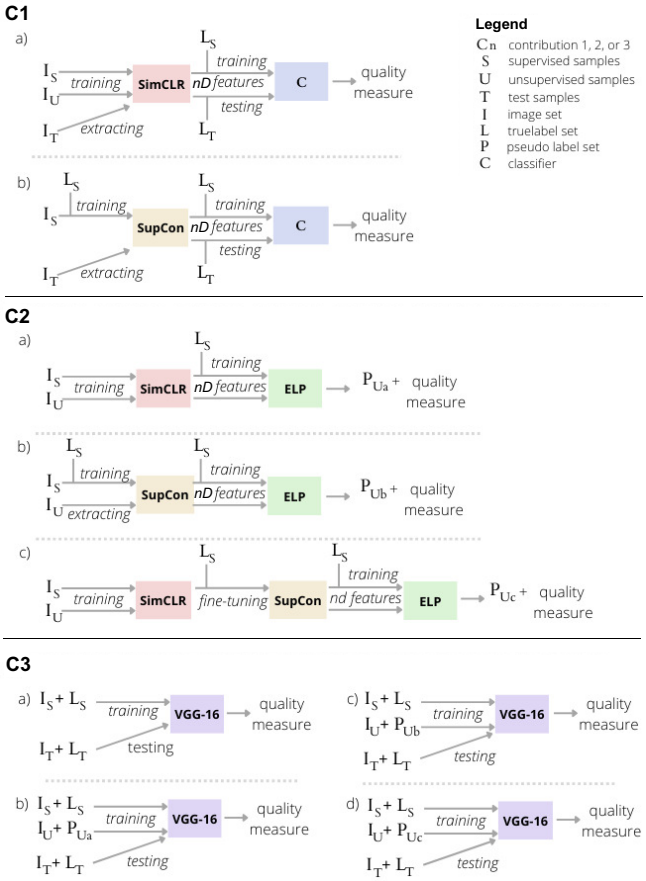


Figure 4.2: Summary of the proposed experiments.

4.5.1.1 *Experiment for testing C1*

Our first claim C1 is the following: contrastive learning methods produce high separability of classes (*i.e.*, DS) in the learned feature space. Also, we noticed that using contrastive learning increased the propagation accuracy in up to 20% *vs* using a simpler feature learning method, *i.e.*, generating the latent space via

autoencoders (Benato et al., 2018). Since the concept of data separability is not uniquely and formally defined (see Sec. 4.1), directly measuring DS is a difficult task. As such, we assess DS by a 'proxy' method: We train two distinct classifiers C, both using 1% supervised samples. For this, we use Linear SVM, a simple linear classifier used to check the linear separability of classes in the latent space; and OPFSup (Papa and Falcão, 2009), an Euclidean distance-based classifier. Our assumption is if these classifiers yield high quality, then DS is high, and conversely. We measure classifiers' quality by accuracy and κ over correctly classified samples in T.

With the above, we conduct three experiments – one per method of latent space generation (see Sec. 4.3.1):

a) *SimCLR:* Train with A on $I_{S\cup U}$; extract features F_S and F_T; train C on F_S and L_S; test on F_T and L_T.

b) *SupCon:* Train with A on I_S and L_S; extract features F_S and F_T; train and test as above.

c) *SimCLR+SupCon:* Train SimCLR with A on $I_{S\cup U}$; fine-tune with SupCon on I_S and L_S; extract features F_S and F_T; train and test as above.

4.5.1.2 *Experiment for testing C2*

Similarly to C1, evaluating the VS of projections to test C2 can be done in many ways since visual separation of clusters in a 2D scatterplot is a broad concept. In DR literature, several metrics have been proposed for this task (see surveys of Espadoto et al. (2019a) and Nonato and Aupetit (2018) and also Sec. 2.3.4). Yet, such metrics are typically used to gauge the projection quality when explored by a *human*. Rather, in our context, we use projections *automatically* to drive pseudo-labeling and improve classification (Sec. 4.3.2). As such, it makes sense to evaluate our projections' VS by how well they can do this label propagation. For this, we compare the computed pseudo-labels with the true, supervised, labels by computing accuracy and κ for the correctly computed pseudo-labels over U. We do this via three experiments:

a) *SimCLR:* Train with A on $I_{S\cup U}$; extract features $F_{S\cup U}$; compute 2D features F' with t-SNE from $F_{S\cup U}$; propagate labels L_S with OPFSemi from F'_S to F'_U;

b) *SupCon:* Train with A on I_S and L_S; extract features $F_{S\cup U}$; compute 2D features F' with t-SNE from $F_{S\cup U}$; propagate labels as above;

c) *SimCLR+SupCon:* Train SimCLR with A on $I_{S \cup U}$; fine-tune with SupCon on I_S and L_S; extract features $I_{S \cup U}$; compute 2D features F' with t-SNE from $F_{S \cup U}$; propagate labels as above.

4.5.1.3 *Experiment for testing C3*

Finally, we use the computed pseudo-labels to train and test a DNN classifier, namely VGG-16, to test C3, *i.e.*, gauge how CS is correlated (or not) with VS and DS. For this, we do the following experiments:

a) *baseline:* train with I_S and L_S; test on I_T and L_T;

b) *SimCLR:* train with $I_{S \cup U}$ and $L_{S \cup P_U}$, with pseudo-labels P_U from (Sec. 4.5.1.2,a); test as above;

c) *SupCon:* train with $I_{S \cup U}$ and $L_{S \cup P_U}$, with pseudo-labels P_U from (Sec. 4.5.1.2,b); test as above;

d) *SimCLR+SupCon:* train with $I_{S \cup U}$ and $L_{S \cup P_U}$, with pseudo-labels P_U from (Sec. 4.5.1.2,c); test as above.

4.5.2 *Results*

We present the results of the experiments in Sec. 4.5.1 and along our claims C1-C3.

4.5.2.1 *C1: Contrastive learning yields high DS*

Table 12 shows the classification results for the experiments in Sec. 4.5.1.1 in terms of accuracy and κ (mean and standard deviation) for the trained Linear SVM and OPFSup classifiers. To ease interpretation, we next summarize these results by averaging classification values using a heatmap in Fig. 4.3.

We first discuss the contrastive learning methods trained from scratch *vs* using ImageNet pre-trained weights. The best accuracy and κ for all datasets exceed 0.70 and 0.59, respectively. Linear SVM obtained the best results for most datasets, showing that these latent spaces have a reasonable *linear separation* between classes even when classified with only 1% supervised samples. In contrast, OPFSup seems to suffer from the dimensionality curse as it uses Euclidean distances in the latent space. This further motivates the latent space's dimensionality reduction when using an OPF classifier. In cases where OPFSemi values are higher than

Table 12: C1: DS assessment of SimCLR's, SupCon's, and SimCLR+SupCon's latent spaces using Linear SVM and OPFSup on T. The three methods are compared trained from scratch and with pre-trained weights during 50 epochs. Best values per dataset and model initialization are in bold.

datasets	metrics	from scratch					
		SimCLR		SupCon		SimCLR+SupCon	
		LinearSVM	OPFSemi	LinearSVM	OPFSemi	LinearSVM	OPFSemi
H.eggs*	acc	0.842436 ± 0.047	0.861268 ± 0.019	0.811048 ± 0.048	0.752668 ± 0.058	0.890144 ± 0.015	**0.902072 ± 0.006**
	κ	0.810976 ± 0.057	0.835316 ± 0.023	0.770732 ± 0.058	0.705686 ± 0.067	0.868792 ± 0.018	**0.883541 ± 0.007**
P.cysts*	acc	**0.722607 ± 0.034**	0.678489 ± 0.025	0.668973 ± 0.094	0.660611 ± 0.016	0.719723 ± 0.010	0.673876 ± 0.032
	κ	0.616310 ± 0.047	0.569074 ± 0.031	0.559129 ± 0.121	0.553795 ± 0.019	**0.616431 ± 0.011**	0.562813 ± 0.039
H.larvae	acc	0.935545 ± 0.003	0.905213 ± 0.003	0.929858 ± 0.009	0.901422 ± 0.037	**0.934913 ± 0.015**	0.932070 ± 0.017
	κ	**0.710109 ± 0.040**	0.564216 ± 0.050	0.673113 ± 0.020	0.554242 ± 0.165	0.708803 ± 0.064	0.683580 ± 0.094
H.eggs	acc	0.772056 ± 0.005	0.710778 ± 0.012	0.657975 ± 0.004	0.561930 ± 0.037	**0.783572 ± 0.022**	0.730335 ± 0.025
	κ	0.565300 ± 0.041	0.524010 ± 0.017	0.122930 ± 0.116	0.259556 ± 0.045	**0.595363 ± 0.062**	0.543848 ± 0.038
P.cysts	acc	0.733078 ± 0.028	0.627772 ± 0.009	0.628701 ± 0.017	0.527342 ± 0.015	**0.766284 ± 0.018**	0.677464 ± 0.026
	κ	0.561195 ± 0.025	0.409251 ± 0.010	0.254190 ± 0.056	0.260470 ± 0.017	**0.600513 ± 0.037**	0.482015 ± 0.033

datasets	metrics	pre-trained					
		SimCLR		SupCon		SimCLR+SupCon	
		LinearSVM	OPFSemi	LinearSVM	OPFSemi	LinearSVM	OPFSemi
H.eggs*	acc	0.809793 ± 0.031	0.834903 ± 0.032	0.854990 ± 0.017	0.842436 ± 0.018	0.839297 ± 0.013	**0.880728 ± 0.020**
	κ	0.773903 ± 0.037	0.803842 ± 0.038	0.825372 ± 0.021	0.811695 ± 0.023	0.809970 ± 0.015	**0.858649 ± 0.024**
P.cysts*	acc	0.685410 ± 0.039	0.580450 ± 0.012	**0.742215 ± 0.012**	0.697232 ± 0.005	0.690312 ± 0.030	0.614764 ± 0.011
	κ	0.579214 ± 0.048	0.444268 ± 0.019	**0.652441 ± 0.012**	0.595298 ± 0.006	0.571982 ± 0.039	0.483336 ± 0.013
H.larvae	acc	0.949447 ± 0.007	0.947551 ± 0.016	0.950079 ± 0.008	0.949447 ± 0.010	**0.952607 ± 0.007**	0.951343 ± 0.005
	κ	**0.779016 ± 0.039**	0.767287 ± 0.080	0.755562 ± 0.061	0.748063 ± 0.072	0.777646 ± 0.049	0.775075 ± 0.045
H.eggs	acc	0.772490 ± 0.022	0.755976 ± 0.040	0.780965 ± 0.053	0.703390 ± 0.065	0.765971 ± 0.038	**0.782051 ± 0.017**
	κ	0.606751 ± 0.031	0.586526 ± 0.055	0.566952 ± 0.146	0.469467 ± 0.108	0.591410 ± 0.056	**0.617267 ± 0.022**
P.cysts	acc	0.616278 ± 0.083	0.615117 ± 0.020	**0.721932 ± 0.013**	0.635551 ± 0.007	0.709857 ± 0.043	0.681528 ± 0.024
	κ	0.378195 ± 0.125	0.370532 ± 0.043	0.505566 ± 0.043	0.419729 ± 0.011	**0.526674 ± 0.052**	0.486552 ± 0.030

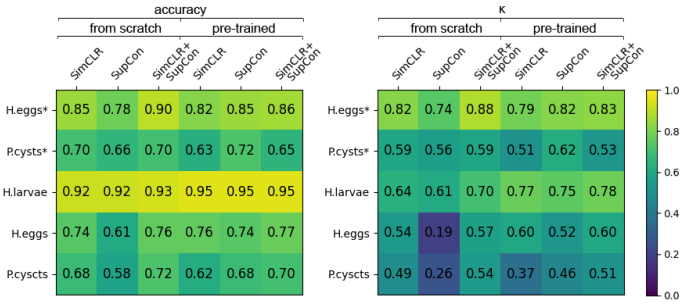


Figure 4.3: C1: Values from Tab. 12 averaged per contrastive learning technique.

LinearSVM, both are relatively close, i.e., for *H.eggs** using SimCLR+SupCon. Separately, we see that the ImageNet pre-trained weights helped the three compared methods for most methods and datasets, while the best values for *H.eggs** and *P.cysts* are obtained when trained from scratch. SimCLR had an increase of around 0.10 in κ for *H.larvae* with pre-trained weights. SupCon also had an extra 0.10 accuracy and κ for all datasets with pre-trained weights. SimCLR+SupCon achieved its best results compared to SimCLR and SupCon for most datasets with pre-trained weights. We also see this in Fig. 4.3 – for the κ plot, brighter cells are mainly in the SimCLR+SupCon columns. We also see that higher accuracy values do not always reflect higher κ values, e.g., for the *H.larvae* dataset. This can be justified by the unbalancing of classes presented in such datasets, as discussed in Sec. 4.4.2.

Although contrastive learning yields quite high DS values, we see both in Tab. 12 and Fig. 4.3 that there exists quite some DS variation across datasets. This will help us next explore how different projection methods map these values to visual separation (VS).

4.5.2.2 *C2: t-SNE projections of contrastive latent spaces yield high VS*

Table 13 show the results for the experiments in Sec. 4.5.1.2, *i.e.*, the mean propagation accuracy and κ in pseudo-labeling for the correctly assigned labels in U for EPL run on latent spaces created by SimCLR, SupCon, and SimCLR+SupCon.

The best results were obtained when using the ImageNet pre-trained weights. This shows that the pseudo-labeling on the contrastive latent space is favored by such pre-trained weights. SupCon gained almost 0.20 in κ compared with SimCLR for H.eggs and P.cysts without impurity. SupCon obtained the best results for the H.Eggs and P.cysts without impurities, while the SimCLR+SupCon obtained the best results for the same datasets with impurities. SimCLR+SupCon improved the results of SimCLR for those datasets. For H.larvae, the results of the three methods were similar.

4.5.2.3 *C3: Classifiers trained by pseudo-labels obtained from high-VS projections have a high CP*

Table 14 shows the results of classification for VGG-16 trained from the pseudo-labeling performed on latent spaces from SimCLR, SupCon, and SimCLR+SupCon.

Table 13: C2: Propagation results for pseudo-labeling U on the projected SimCLR's and SupCon's latent spaces, from scratch and using ImageNet pre-trained weights. Best values per dataset are in bold.

		trained from scratch			with ImageNet pre-trained weights		
		a) SimCLR	b) SupCon	c) SimCLR+SupCon	a) SimCLR	b) SupCon	c) SimCLR+SupCon
H.eggs	acc	0.861493 ± 0.012	0.713554 ± 0.077	0.896255 ± 0.041	0.795203 ± 0.129	**0.951765 ± 0.041**	0.830234 ± 0.123
(w/o imp)	κ	0.561568 ± 0.009	0.473379 ± 0.025	0.567093 ± 0.020	0.756312 ± 0.153	**0.942519 ± 0.049**	0.797482 ± 0.148
P.cysts	acc	0.652324 ± 0.027	0.641073 ± 0.038	0.650470 ± 0.027	0.568991 ± 0.036	**0.706973 ± 0.092**	0.565282 ± 0.091
(w/o imp)	κ	0.537704 ± 0.043	0.531090 ± 0.040	0.533208 ± 0.031	0.428962 ± 0.036	**0.619738 ± 0.102**	0.439581 ± 0.103
H.larvae	acc	0.898739 ± 0.033	0.886539 ± 0.003	0.941169 ± 0.013	**0.959062 ± 0.007**	0.946184 ± 0.010	0.954724 ± 0.005
	κ	0.532710 ± 0.179	0.404983 ± 0.119	0.694591 ± 0.029	**0.817274 ± 0.030**	0.777621 ± 0.020	0.792838 ± 0.009
H.eggs	acc	0.710173 ± 0.035	0.585802 ± 0.026	0.741755 ± 0.065	0.719862 ± 0.077	0.751723 ± 0.052	**0.780418 ± 0.080**
	κ	0.357514 ± 0.044	0.178536 ± 0.031	0.374099 ± 0.108	0.532788 ± 0.120	0.553654 ± 0.094	**0.624724 ± 0.113**
P.cysts	acc	0.607884 ± 0.049	0.530785 ± 0.019	0.666119 ± 0.027	0.670898 ± 0.051	0.577025 ± 0.049	**0.705042 ± 0.035**
	κ	0.380969 ± 0.066	0.235849 ± 0.018	0.457391 ± 0.056	0.430201 ± 0.022	0.320479 ± 0.057	**0.513962 ± 0.043**

We notice that the results of VGG-16's classification follow the same pattern as the propagation results (Tab. 13). The best results were found by the methods using the ImageNet pre-trained weights. Also, SupCon obtained the best results for H.Eggs and P.cysts without impurities, while SimCLR+SupCon obtained the best results for the same datasets with impurities. SupCon showed a gain of almost 0.20 in κ for H.eggs without impurity and H.larvae, and 0.15 for P.cysts without impurity when compared with the baseline. In short, the results show that VGG-16 can learn from the pseudo-labels since it provided good classification accuracies and κ – higher than 0.85 and 0.76, respectively – for H.eggs and P.cysts without impurity and H.Larvae. However, the compared methods could not surpass the baseline for H.eggs and P.cysts with impurities. We discuss this aspect next.

Table 14: C3: VGG-16's classification results on T when using pseudo labels from SimCLR's, SupCon and SimCLR+SupCon latent spaces, from scratch and with ImageNet pre-trained weights. Best values per dataset are in bold.

		a) baseline	trained from scratch			with ImageNet pre-trained weights		
			b) SimCLR	c) SupCon	d) SimCLR+SupCon	a) SimCLR	b) SupCon	c) SimCLR+SupCon
H.eggs	acc	0.812932 ± 0.059	0.435028 ± 0.400	0.714375 ± 0.088	0.925926 ± 0.035	0.823603 ± 0.138	**0.961080 ± 0.039**	0.858129 ± 0.127
(w/o imp)	κ	0.775954 ± 0.073	0.292310 ± 0.506	0.662603 ± 0.098	0.912482 ± 0.041	0.790296 ± 0.164	**0.953710 ± 0.047**	0.831107 ± 0.152
P.cysts	acc	0.757209 ± 0.015	0.589965 ± 0.174	0.662053 ± 0.064	0.606113 ± 0.188	0.752905 ± 0.183	**0.857411 ± 0.085**	0.740945 ± 0.216
(w/o imp)	κ	0.651933 ± 0.023	0.383736 ± 0.334	0.558104 ± 0.071	0.408416 ± 0.354	0.622887 ± 0.185	**0.766192 ± 0.043**	0.608864 ± 0.200
H.larvae	acc	0.930806 ± 0.026	0.903950 ± 0.034	0.888784 ± 0.009	0.942496 ± 0.015	**0.956714 ± 0.004**	**0.952607 ± 0.008**	**0.957978 ± 0.001**
	κ	0.613432 ± 0.233	0.538558 ± 0.196	0.406452 ± 0.168	0.738656 ± 0.061	**0.809830 ± 0.019**	**0.803148 ± 0.018**	**0.807574 ± 0.021**
H.eggs	acc	**0.862234 ± 0.015**	0.728814 ± 0.059	0.606693 ± 0.042	0.779444 ± 0.073	0.737723 ± 0.068	0.780095 ± 0.060	0.806389 ± 0.073
	κ	**0.740861 ± 0.028**	0.566056 ± 0.064	0.286646 ± 0.063	0.627849 ± 0.099	0.553855 ± 0.114	0.592800 ± 0.116	0.661330 ± 0.103
P.cysts	acc	**0.850691 ± 0.018**	0.687333 ± 0.028	0.379775 ± 0.020	0.703820 ± 0.020	0.725648 ± 0.036	0.645304 ± 0.052	0.737258 ± 0.036
	κ	**0.751667 ± 0.028**	0.429244 ± 0.179	0.184170 ± 0.023	0.522443 ± 0.027	0.540847 ± 0.049	0.395300 ± 0.079	0.565966 ± 0.045

4.5.3 *Discussion*

We next discuss several aspects pertaining to our results.

4.5.3.1 *Visual separation* vs *classifier performance*

Figure 4.4.i shows the 2D t-SNE projections of the three computed latent spaces for all five studied datasets. For each dataset, the top row (a) shows the few (1%) supervised labels (colored points) thinly spread among the vast majority of unsupervised (black) samples; the bottom row (b) shows samples colored by the computed pseudo-labels.

We see in all images a good correlation of the visual separation VS (point groups separated from each other by whitespace) with the lack of label mixing in such groups. For H.eggs without impurity, all three latent space projections show a clear VS, and we see that this leads to almost no color mixing in the propagated pseudo-labels. For the H.eggs dataset, we see how the visually separated groups show almost no color mixing, whereas the parts of the projection where no VS is present show color mixing. For P.cysts without impurity, there is a clearly separated group at the bottom in all three projections which also has a single color (label). The remaining parts of the projections, which have no clear VS into distinct groups, show a mix of different colors. For P.cysts, the projections have even less VS, and we see how labels get even more mixed – for instance, the impurity class (brown) is spread all over the projection. For H.larvae, the larvae class (red) is better separated from the big group of impurities (green), and this correlates with the larvae samples being all located in a tail-like periphery of the projection – thus, better visually separated from the rest.

All in all, these results show that a good VS leads to a low mixing of the propagated labels, and conversely. In turn, a low mixing will lead to a high classification performance (CP), and conversely, *i.e.*, our claim C3. Figure 4.4.ii shows this by comparing the results for the baseline and for VGG-16 trained with the generated pseudo-labels. We see a gain of almost 0.20 in κ from baseline (red) to the proposed pseudo-labeling method (green) for those datasets with a clear VS and little label mixing in the projections. Conversely, we see the CP results are are below to baseline for the datasets with poor VS and color-mixing in their projections.

4.5.3.2 *Contrastive learning from few supervised samples*

Our experiments with ImageNet pre-trained weights and higher results show that SimCLR – even trained with *thousands* of unsupervised samples (69%) – and having more information on the

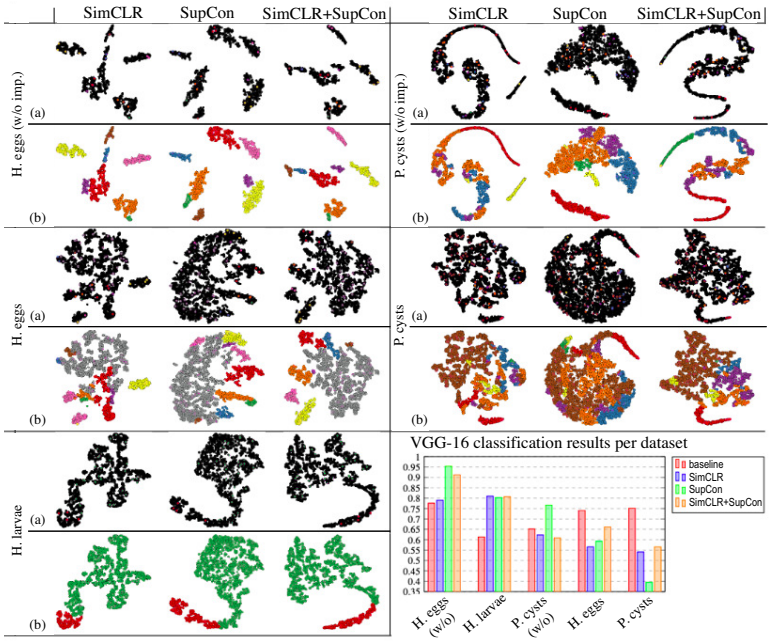


Figure 4.4: (i) top left: t-SNE projections of the three contrastive latent spaces (SimCLR, SupCon, SimCLR+SupCon) for the six studied datasets. In (a), black points are the unsupervised samples U and colored points the supervised ones S. In (b), colors show the computed pseudo-labels. (ii) bottom right: κ values for baseline, SimCLR, SupCon, and SimCLR+SupCon experiments. Datasets are ordered on higher κ values from left to right.

data distribution of the original space – did not surpass SupCon which used only *dozens* of supervised samples (1%). Our explanation for this is that the latent space generated when SupCon was used to fine-tune SimCLR (SupCon+SimCLR) had a *better DS* than the one created by SimCLR – this is by Tab. 12. This shows the benefit of using SupCon with supervised data restriction as compared to SimCLR, a finding that up to our knowledge is novel. Separately, the fact that a higher DS lead to a higher CP further supports our claim C3.

4.5.4 *Summary of findings: C1-C3*

Let us summarize our findings so far involving contributions C1-C3. We proposed a method to create high-quality classifiers for image datasets from training-sets having only very few super-

vised (labeled) samples. For this, we used two contrastive learning approaches (SimCLR and SupCon) as well as a combination of the two to generate latent spaces. Next, we projected these spaces to 2D using t-SNE, propagated labels in the projection, and finally used these pseudo-labels to train a final deep-learning classifier for a challenging problem involving the classification of human intestinal parasite images.

Our results so far show that SupCon performed better than SimCLR when only 1% of supervised samples were available, even though SimCLR uses thousands of distinct samples of the data distribution. We showed label propagation accuracies up to 95% for the studied datasets without impurities (an adversarial class) and up to 70% for datasets with impurities, respectively.

Additionally, our experiments show that a high data separation (DS) in the latent space leads to a high visual separation (VS) in the 2D t-SNE projection which, in turn, leads to high classifier performance (CP). While partial results of this kind have been presented by earlier infovis and machine learning papers, our work is, to our knowledge, the first time that DS, VS, and CP are all linked in the context of an application involving the generation of rich training-sets by pseudo-labeling.

4.6 EXPLORING MULTIPLE PROJECTIONS: C4-C5

We explored in Section 4.5 the links of DS, VS, and CP in the context of using pseudolabeling. However, the end-to-end relationships between DS, VS, and CP – especially when using different projection techniques – were still not fully studied. The key limitation was using a *single* projection technique, t-SNE (van der Maaten and Hinton, 2008), which gave good VS results. This thus only shows that, if DS and VS are both high, then CP is also high. This does not explain the *full* link between DS, VS, and CP. For instance: Does a high DS *always* imply a high VS? Does a high VS *always* imply a high CP? What are these correlations when one uses a different projection technique than t-SNE?

To answer such questions, we extend our studies to evaluate 10 projection techniques (apart from t-SNE) which produce a wide range of VS values. Apart from the work presented so far in Sec. 4.5, which covered claims C1-C3, we now cover our last two contributions:

C4 We identify projection techniques for which DS strongly correlates with VS and also techniques for which this does not happen;

C5 We show that good-VS projections are essential for training classifiers that reach a high CP.

Our work brings evidence that DS, VS, and CP are strongly correlated for a specific class of projection techniques. For these projections, one can use our proposed pipeline to design high-CP classifiers, specially for training sets with very few supervised (labeled) points. We also identify a class of projection techniques which lead to poor VS regardless of the available DS. We argue that these projections are less useful instruments in classifier design tasks and, more broadly, any infovis task where assessing DS via VS is important.

4.6.1 *Proposed experiments*

When describing, next, our different experiments, we used the same notations as defined earlier in Section 4.5.1. Figure 4.5 shows the several experiments we performed to explore the claims C4 and C5 listed in Sec. 4.6. We next detail these experiments.

4.6.1.1 *Experiment for testing C4*

For testing C4, many strategies could be used to evaluate the VS of projections since visual separation of clusters in a 2D scatterplot is a broad concept. Several metrics have been proposed for this task in DR literature – see surveys (Espadoto et al., 2019a; Nonato and Aupetit, 2018). However, existing metrics are usually used to gauge the projection quality when explored by a *human*. Rather, in our context, we use projections *automatically* to drive pseudo labeling and improve classification (Sec. 4.3.2) – and, in this process, we aim to find which projection techniques are best for this task. In this way, evaluating our projections' VS by how well they can do this label propagation is a good assessment for this purpose. We compare the computed pseudo labels with the true, supervised, labels by computing accuracy and κ for the correctly computed pseudo-labels over U. This comparison is performed using distinct projection techniques P (Sec. 4.4.1) to understand how P is directly related to the different VS resulted from each projection. We proposed three experiments aim to achieve precisely this:

a) *SimCLR:* Train with A on $I_{S \cup U}$; extract features $F_{S \cup U}$; compute 2D features F' with P from $F_{S \cup U}$; propagate labels L_S with OPFSemi from F'_S to F'_U;

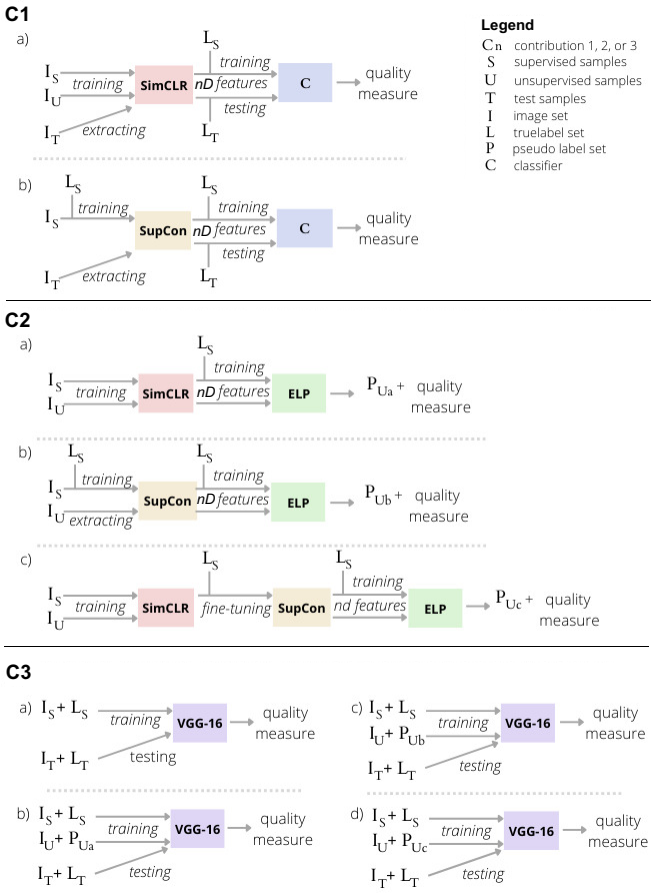


Figure 4.5: Summary of the proposed experiments for testing our claims C1, C2 (C4), and C3 (C5). We added C1.c) to complement the experimental setup first proposed in (Benato et al., 2023c).

b) *SupCon:* Train with A on I_S and L_S; extract features $F_{S\cup U}$; compute 2D features F' with P from $F_{S\cup U}$; propagate labels as above;

c) *SimCLR+SupCon:* Train SimCLR with A on $I_{S\cup U}$; fine-tune with SupCon on I_S and L_S; extract features $I_{S\cup U}$; compute 2D features F' with P from $F_{S\cup U}$; propagate labels as above.

4.6.1.2 *Experiment for testing C5*

Finally, the computed pseudo labels are used to train and test a DNN classifier, in this case VGG-16, to test how CP is correlated (or not) with VS and DS. For this, we do the following experiments:

a) *baseline:* train with I_S and L_S; test on I_T and L_T;

b) *SimCLR:* train with $I_{S\cup U}$ and $L_{S\cup P_U}$, with pseudo-labels P_U from (Sec. 4.6.1.1,a); test as above;

c) *SupCon:* train with $I_{S\cup U}$ and $L_{S\cup P_U}$, with pseudo-labels P_U from (Sec. 4.6.1.1,b); test as above;

d) *SimCLR+SupCon:* train with $I_{S\cup U}$ and $L_{S\cup P_U}$, with pseudo-labels P_U from (Sec. 4.6.1.1,c); test as above.

4.6.2 *Results*

We next show the results of the experiments in Sec. 4.6.1 along our claims C4 and C5.

4.6.2.1 *C4: Correlation between different projections and VS*

Table 15 shows the results for the experiments in Sec. 4.6.1.1, *i.e.*, the mean propagation accuracy and κ in pseudolabeling for the correctly assigned labels in U for EPL run using the selected P on latent spaces created by SimCLR, SupCon, and SimCLR+SupCon. Figure 4.6(left) shows the same data using a heat map for easier interpretation.

We got the best results using the ImageNet pre-trained weights – see the lightgreen-yellow cells in the three rightmost columns marked *pre-trained* in Fig. 4.6(left). This shows that such pre-trained weights favor the pseudolabeling on the contrastive latent space. Additionally, SimCLR+SupCon obtained the best results among compared the compared projections – see the brighter colors in the rightmost column in the same figure.

More interestingly, we see a clear pattern of similar horizontal colors in Fig. 4.6(left). For κ, these 'color bands' match very well the projection techniques. For instance, the dark-blue block of cells in the figure tells that FICA, ISO, KPCA, LLE, and MLLE score very poor values, *i.e.*, create a poor VS. In contrast, t-SNE, UMAP, FA, and MDS have brighter cells, so, they create better VS. We will explore this further in Sec. 4.6.3.

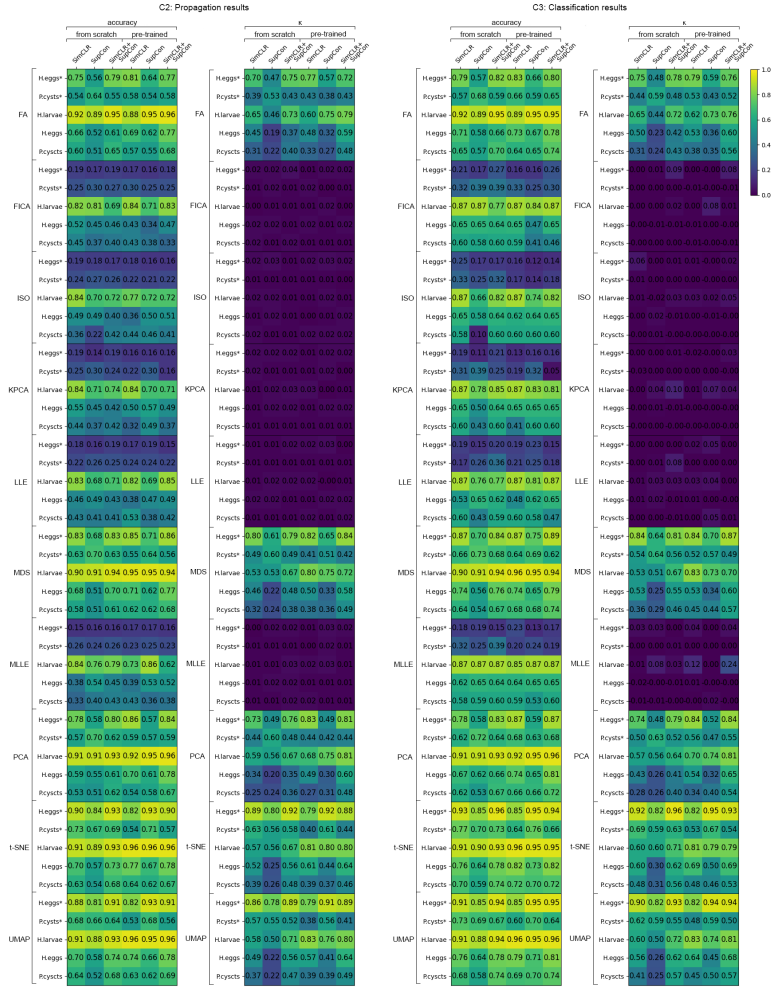


Figure 4.6: *"C4: Propagation results"* and *"C5: Classification results"*. For the proposed experiments, results of accuracy and κ are shown for pseudolabeling using 10 projection techniques P on SimCLR's, SupCon's, and SimCLR+SupCon's latent spaces trained from scratch or with pre-trained weights for five datasets.

Table 15: C2: Propagation results for pseudo-labeling U on the projected SimCLR's and SupCon's latent spaces, from scratch and using ImageNet pre-trained weights. Best values per dataset are in bold.

techniques	datasets	metric	from scratch			pre-trained		
			SimCLR	SupCon	SimCLR+SupCon	SimCLR	SupCon	SimCLR+SupCon
FA	H.eggs*	acc	0.747777 ± 0.068	0.560496 ± 0.050	0.791431 ± 0.040	**0.807060 ± 0.026**	0.641337 ± 0.071	0.766370 ± 0.065
		κ	0.697796 ± 0.082	0.473382 ± 0.062	0.750259 ± 0.047	**0.767607 ± 0.032**	0.586708 ± 0.086	0.721839 ± 0.077
	P.cysts*	acc	0.539688 ± 0.073	**0.644288 ± 0.054**	0.552300 ± 0.022	0.576286 ± 0.039	0.537092 ± 0.038	0.576781 ± 0.038
		κ	0.384467 ± 0.085	**0.530016 ± 0.059**	0.427268 ± 0.046	0.428285 ± 0.047	0.383229 ± 0.065	0.428245 ± 0.059
	H.larvae	acc	0.920835 ± 0.021	0.889115 ± 0.015	0.947540 ± 0.012	0.882201 ± 0.056	0.950386 ± 0.005	**0.955809 ± 0.008**
		κ	0.643369 ± 0.093	0.455761 ± 0.078	0.726970 ± 0.092	0.599440 ± 0.164	0.751991 ± 0.044	**0.785414 ± 0.053**
	H.eggs	acc	0.659959 ± 0.030	0.518726 ± 0.016	0.607322 ± 0.040	0.693032 ± 0.014	0.618129 ± 0.078	**0.768586 ± 0.049**
		κ	0.445596 ± 0.026	0.187524 ± 0.014	0.368271 ± 0.066	0.480564 ± 0.027	0.319186 ± 0.101	**0.587166 ± 0.078**
	P.cysts	acc	0.596337 ± 0.031	0.513513 ± 0.018	0.649445 ± 0.038	0.573540 ± 0.004	0.550197 ± 0.029	**0.675427 ± 0.018**
		κ	0.305130 ± 0.020	0.219125 ± 0.034	0.399547 ± 0.084	0.331404 ± 0.048	0.271900 ± 0.046	**0.482847 ± 0.007**
FICA	H.eggs*	acc	0.191054 ± 0.003	0.174616 ± 0.034	**0.191592 ± 0.017**	0.167071 ± 0.005	0.164915 ± 0.028	0.185005 ± 0.001
		κ	0.015913 ± 0.004	0.015189 ± 0.009	0.056603 ± 0.009	0.011312 ± 0.021	0.015913 ± 0.008	**0.020143 ± 0.009**
	P.cysts*	acc	0.245920 ± 0.032	0.295128 ± 0.008	0.269288 ± 0.017	**0.301681 ± 0.043**	0.248393 ± 0.043	0.247464 ± 0.045
		κ	0.013333 ± 0.009	0.017266 ± 0.010	0.012307 ± 0.005	**0.018144 ± 0.009**	0.004888 ± 0.005	0.009506 ± 0.009
	H.larvae	acc	0.818219 ± 0.030	0.814152 ± 0.027	0.688762 ± 0.132	**0.839637 ± 0.026**	**0.711265 ± 0.153**	0.829606 ± 0.017
		κ	0.001788 ± 0.003	0.009610 ± 0.012	0.013793 ± 0.026	**0.014248 ± 0.006**	0.017165 ± 0.013	0.003038 ± 0.022
	H.eggs	acc	**0.518539 ± 0.048**	0.452674 ± 0.064	0.456307 ± 0.046	0.433016 ± 0.026	0.343767 ± 0.100	0.472331 ± 0.077
		κ	0.016130 ± 0.002	**0.015664 ± 0.006**	0.013574 ± 0.005	0.018069 ± 0.001	0.010873 ± 0.002	0.016699 ± 0.001
	P.cysts	acc	**0.450948 ± 0.031**	0.367179 ± 0.079	0.395749 ± 0.031	0.429745 ± 0.072	0.380469 ± 0.129	0.330695 ± 0.070
		κ	0.018332 ± 0.004	0.013987 ± 0.007	**0.020642 ± 0.008**	0.014538 ± 0.008	0.014562 ± 0.003	0.007237 ± 0.003
ISO	H.eggs*	acc	**0.194826 ± 0.011**	0.182700 ± 0.014	0.172460 ± 0.017	0.176772 ± 0.028	0.162220 ± 0.028	0.164915 ± 0.032
		κ	0.019858 ± 0.007	0.028010 ± 0.004	0.009387 ± 0.003	0.021445 ± 0.006	**0.034176 ± 0.008**	0.023816 ± 0.006
	P.cysts*	acc	0.235534 ± 0.020	**0.268917 ± 0.056**	0.260880 ± 0.024	0.224901 ± 0.008	0.211795 ± 0.035	0.219955 ± 0.010
		κ	0.011823 ± 0.004	**0.013657 ± 0.005**	0.013012 ± 0.001	0.008791 ± 0.003	0.010802 ± 0.006	0.002970 ± 0.003
	H.larvae	acc	**0.844923 ± 0.015**	0.698116 ± 0.224	0.724278 ± 0.156	0.774976 ± 0.041	0.717365 ± 0.188	0.722380 ± 0.168
		κ	0.022980 ± 0.016	**0.023176 ± 0.015**	0.009617 ± 0.008	0.010355 ± 0.018	0.017956 ± 0.010	0.009347 ± 0.011
	H.eggs	acc	0.486398 ± 0.067	0.489379 ± 0.125	0.399665 ± 0.023	0.364263 ± 0.079	0.503168 ± 0.112	**0.505496 ± 0.012**
		κ	0.014483 ± 0.013	**0.020610 ± 0.005**	0.010909 ± 0.007	0.004325 ± 0.007	0.013514 ± 0.004	0.019615 ± 0.003
	P.cysts	acc	0.159365 ± 0.008	0.223732 ± 0.014	0.419790 ± 0.027	0.435717 ± 0.016	**0.463541 ± 0.022**	0.406202 ± 0.034
		κ	0.015452 ± 0.008	0.009935 ± 0.001	0.014798 ± 0.002	0.016392 ± 0.006	**0.019200 ± 0.012**	0.017231 ± 0.004
KPCA	H.eggs*	acc	**0.190245 ± 0.009**	0.143088 ± 0.018	0.188858 ± 0.007	0.158717 ± 0.041	0.155753 ± 0.023	0.161837 ± 0.023
		κ	**0.022804 ± 0.011**	0.017998 ± 0.006	0.020763 ± 0.004	0.009970 ± 0.008	0.016143 ± 0.015	0.016003 ± 0.003
	P.cysts*	acc	0.252596 ± 0.046	**0.304772 ± 0.056**	0.243571 ± 0.047	0.222552 ± 0.008	0.302423 ± 0.049	0.164565 ± 0.012
		κ	0.020874 ± 0.015	0.010778 ± 0.007	0.008015 ± 0.008	0.007448 ± 0.007	**0.021246 ± 0.002**	0.002761 ± 0.003
	H.larvae	acc	0.838824 ± 0.006	0.709231 ± 0.179	**0.741223 ± 0.126**	0.837332 ± 0.029	0.697438 ± 0.162	0.709773 ± 0.181
		κ	0.010105 ± 0.017	0.023394 ± 0.012	**0.034326 ± 0.015**	0.025248 ± 0.005	0.003343 ± 0.009	0.007443 ± 0.004
	H.eggs	acc	0.550307 ± 0.052	0.452022 ± 0.185	0.421837 ± 0.030	0.499348 ± 0.054	**0.574064 ± 0.022**	0.490963 ± 0.064
		κ	0.014382 ± 0.005	0.019979 ± 0.009	**0.020720 ± 0.008**	0.007635 ± 0.006	0.019011 ± 0.006	0.019884 ± 0.007
	P.cysts	acc	0.441292 ± 0.024	0.367080 ± 0.125	0.417849 ± 0.047	0.323727 ± 0.088	**0.490817 ± 0.017**	0.371310 ± 0.053
		κ	0.008013 ± 0.009	0.013086 ± 0.006	0.012913 ± 0.001	**0.014376 ± 0.008**	0.011941 ± 0.007	0.008518 ± 0.004
LLE	H.eggs*	acc	0.177580 ± 0.009	0.163029 ± 0.034	**0.194557 ± 0.003**	0.165185 ± 0.011	0.189976 ± 0.014	0.153058 ± 0.026
		κ	0.012537 ± 0.006	0.010076 ± 0.003	**0.024128 ± 0.006**	0.016414 ± 0.006	0.033328 ± 0.010	0.000590 ± 0.012
	P.cysts*	acc	0.217483 ± 0.016	**0.260633 ± 0.039**	0.252367 ± 0.020	0.238501 ± 0.070	0.244931 ± 0.047	0.220450 ± 0.002
		κ	0.005612 ± 0.002	0.009387 ± 0.002	0.011296 ± 0.008	**0.011855 ± 0.013**	0.007083 ± 0.005	0.006551 ± 0.003
	H.larvae	acc	0.853401 ± 0.006	0.677918 ± 0.162	0.712214 ± 0.188	0.817812 ± 0.038	0.688762 ± 0.160	**0.847499 ± 0.029**
		κ	**0.013540 ± 0.026**	0.011881 ± 0.010	0.018322 ± 0.004	0.016302 ± 0.005	-0.004704 ± 0.009	0.006490 ± 0.016
	H.eggs	acc	0.455798 ± 0.129	0.485374 ± 0.075	0.430501 ± 0.116	0.379356 ± 0.110	0.466604 ± 0.138	**0.492520 ± 0.072**
		κ	0.017533 ± 0.002	**0.022688 ± 0.005**	0.012804 ± 0.007	0.010157 ± 0.005	0.018747 ± 0.003	0.015604 ± 0.008
	P.cysts	acc	0.431039 ± 0.069	0.408242 ± 0.118	0.408541 ± 0.081	**0.526106 ± 0.011**	0.375890 ± 0.077	0.418446 ± 0.102
		κ	0.013445 ± 0.006	0.009230 ± 0.004	0.014226 ± 0.008	**0.018359 ± 0.007**	0.012149 ± 0.006	0.015147 ± 0.004
MDS	H.eggs*	acc	0.831851 ± 0.010	0.679332 ± 0.055	0.825923 ± 0.028	0.852331 ± 0.047	0.710051 ± 0.024	**0.861110 ± 0.016**
		κ	0.797731 ± 0.012	0.614286 ± 0.068	0.789018 ± 0.035	0.823455 ± 0.056	0.647216 ± 0.028	**0.835812 ± 0.019**
	P.cysts*	acc	0.627473 ± 0.040	**0.699060 ± 0.014**	0.629204 ± 0.028	0.550198 ± 0.050	0.633943 ± 0.017	0.563922 ± 0.031
		κ	0.494613 ± 0.061	**0.595492 ± 0.015**	0.490610 ± 0.043	0.408429 ± 0.058	0.514499 ± 0.022	0.416427 ± 0.040
	H.larvae	acc	0.856570 ± 0.008	0.907279 ± 0.021	0.938728 ± 0.012	**0.952691 ± 0.010**	0.949573 ± 0.006	0.944286 ± 0.014
		κ	0.530915 ± 0.025	0.532948 ± 0.100	0.670950 ± 0.103	**0.799463 ± 0.039**	0.746419 ± 0.050	0.716183 ± 0.115
	H.eggs	acc	0.677008 ± 0.002	0.510062 ± 0.035	0.699273 ± 0.032	0.711384 ± 0.042	0.620179 ± 0.041	**0.774734 ± 0.035**
		κ	0.455775 ± 0.042	0.223363 ± 0.025	0.480271 ± 0.045	0.501590 ± 0.055	0.325114 ± 0.050	**0.584801 ± 0.064**
	P.cysts	acc	0.577920 ± 0.018	0.508685 ± 0.010	0.666142 ± 0.063	0.615699 ± 0.013	0.616694 ± 0.016	**0.677517 ± 0.043**
		κ	0.318017 ± 0.034	0.241293 ± 0.016	0.383062 ± 0.068	0.382044 ± 0.036	0.350747 ± 0.024	**0.489937 ± 0.053**
MLLE	H.eggs*	acc	0.154136 ± 0.039	0.169415 ± 0.022	0.155753 ± 0.036	**0.173269 ± 0.005**	0.168418 ± 0.026	0.158717 ± 0.028
		κ	0.004386 ± 0.008	0.024440 ± 0.004	0.019896 ± 0.003	0.012537 ± 0.004	**0.030118 ± 0.005**	0.018684 ± 0.001
	P.cysts*	acc	**0.261746 ± 0.027**	0.244313 ± 0.039	0.258902 ± 0.006	0.225766 ± 0.023	0.259495 ± 0.057	0.228116 ± 0.005
		κ	0.012912 ± 0.008	0.011391 ± 0.002	0.007871 ± 0.006	0.004547 ± 0.003	**0.015666 ± 0.006**	0.010145 ± 0.005
	H.larvae	acc	0.836248 ± 0.009	0.764810 ± 0.062	0.790159 ± 0.048	0.728074 ± 0.064	**0.862001 ± 0.011**	0.621933 ± 0.135
		κ	0.012424 ± 0.011	0.014140 ± 0.029	0.025816 ± 0.018	0.016783 ± 0.015	**0.027929 ± 0.008**	0.005776 ± 0.008
	H.eggs	acc	0.379635 ± 0.038	**0.539035 ± 0.034**	0.449972 ± 0.049	0.389510 ± 0.077	0.531582 ± 0.065	0.520682 ± 0.031
		κ	0.013558 ± 0.010	**0.023266 ± 0.003**	0.016370 ± 0.005	0.013985 ± 0.012	0.008807 ± 0.006	0.014190 ± 0.003
	P.cysts	acc	0.331492 ± 0.042	0.400080 ± 0.050	**0.429994 ± 0.023**	0.427356 ± 0.085	0.362152 ± 0.077	0.377731 ± 0.022
		κ	0.012621 ± 0.003	0.014195 ± 0.002	**0.019642 ± 0.003**	0.017896 ± 0.004	0.007093 ± 0.002	0.009973 ± 0.004
PCA	H.eggs*	acc	0.776341 ± 0.022	0.575047 ± 0.072	0.801132 ± 0.043	**0.855026 ± 0.052**	0.569616 ± 0.061	0.838857 ± 0.030
		κ	0.730955 ± 0.027	0.485557 ± 0.093	0.760041 ± 0.053	**0.826294 ± 0.063**	0.485068 ± 0.065	0.808110 ± 0.035
	P.cysts*	acc	0.572700 ± 0.042	**0.696588 ± 0.023**	0.615356 ± 0.055	0.585188 ± 0.019	0.573318 ± 0.033	0.590999 ± 0.040
		κ	0.438663 ± 0.050	**0.597055 ± 0.026**	0.483103 ± 0.066	0.442557 ± 0.022	0.420640 ± 0.051	0.440307 ± 0.057
	H.larvae	acc	0.910804 ± 0.005	0.910804 ± 0.026	0.931951 ± 0.017	0.915006 ± 0.033	0.950522 ± 0.007	**0.958520 ± 0.001**
		κ	0.587131 ± 0.050	0.563387 ± 0.151	0.668527 ± 0.105	0.676664 ± 0.116	0.751620 ± 0.053	**0.811768 ± 0.001**
	H.eggs	acc	0.587572 ± 0.038	0.545929 ± 0.022	0.612167 ± 0.033	0.703186 ± 0.041	0.612540 ± 0.088	**0.777809 ± 0.042**
		κ	0.340487 ± 0.048	0.201855 ± 0.015	0.351600 ± 0.053	0.493198 ± 0.055	0.302314 ± 0.122	**0.598595 ± 0.072**
	P.cysts	acc	0.530536 ± 0.034	0.558576 ± 0.005	0.615599 ± 0.039	0.544124 ± 0.031	0.585092 ± 0.007	**0.669563 ± 0.018**
		κ	0.245823 ± 0.084	0.241375 ± 0.011	0.361185 ± 0.080	0.273647 ± 0.076	0.314512 ± 0.030	**0.475270 ± 0.058**
t-SNE	H.eggs*	acc	0.904069 ± 0.022	0.831616 ± 0.026	0.931016 ± 0.025	0.824306 ± 0.027	**0.933980 ± 0.015**	0.901374 ± 0.015
		κ	0.885518 ± 0.026	0.804574 ± 0.031	0.918209 ± 0.030	0.791693 ± 0.031	**0.921045 ± 0.018**	0.882814 ± 0.017
	P.cysts*	acc	**0.726014 ± 0.060**	0.674085 ± 0.025	0.691642 ± 0.027	0.537834 ± 0.030	0.711064 ± 0.007	0.572082 ± 0.028
		κ	**0.631292 ± 0.078**	0.584841 ± 0.040	0.585401 ± 0.043	0.397794 ± 0.033	0.608087 ± 0.016	0.441656 ± 0.026
	H.larvae	acc	0.906059 ± 0.013	0.852233 ± 0.039	0.925037 ± 0.022	**0.957299 ± 0.008**	0.955532 ± 0.002	0.955537 ± 0.005
		κ	0.572966 ± 0.086	0.556742 ± 0.151	0.670628 ± 0.099	**0.809426 ± 0.042**	0.795117 ± 0.014	0.801494 ± 0.016
	H.eggs	acc	0.703279 ± 0.032	0.570989 ± 0.040	0.731787 ± 0.013	0.772592 ± 0.009	0.669648 ± 0.085	**0.783492 ± 0.030**
		κ	0.524161 ± 0.043	0.248194 ± 0.041	0.558122 ± 0.030	0.608138 ± 0.021	0.436442 ± 0.124	**0.636986 ± 0.046**
	P.cysts	acc	0.630481 ± 0.012	0.538948 ± 0.013	**0.678413 ± 0.011**	0.644368 ± 0.035	0.616644 ± 0.019	0.667214 ± 0.028
		κ	0.393737 ± 0.002	0.263230 ± 0.010	**0.478606 ± 0.032**	0.389898 ± 0.062	0.366433 ± 0.047	0.460750 ± 0.030
UMAP	H.eggs*	acc	0.880625 ± 0.020	0.811372 ± 0.056	0.907033 ± 0.026	0.823767 ± 0.070	**0.926435 ± 0.018**	0.908619 ± 0.011
		κ	0.857432 ± 0.024	0.777552 ± 0.066	0.889462 ± 0.032	0.790407 ± 0.084	**0.911833 ± 0.022**	0.891483 ± 0.013
	P.cysts*	acc	**0.679896 ± 0.045**	0.661597 ± 0.023	0.640826 ± 0.034	0.527201 ± 0.042	0.678783 ± 0.012	0.564293 ± 0.028
		κ	**0.566764 ± 0.059**	0.554236 ± 0.022	0.518054 ± 0.051	0.382443 ± 0.048	0.562840 ± 0.029	0.414946 ± 0.045
	H.larvae	acc	0.907957 ± 0.011	0.877999 ± 0.043	0.931815 ± 0.017	**0.961095 ± 0.004**	0.949980 ± 0.006	0.958113 ± 0.001
		κ	0.584348 ± 0.059	0.499324 ± 0.130	0.710935 ± 0.049	**0.810896 ± 0.019**	0.763058 ± 0.053	0.803964 ± 0.011
	H.eggs	acc	0.695840 ± 0.021	0.578349 ± 0.006	0.743432 ± 0.051	0.742035 ± 0.044	0.660145 ± 0.061	**0.784796 ± 0.037**
		κ	0.488702 ± 0.012	0.217511 ± 0.009	0.563820 ± 0.081	0.567092 ± 0.056	0.410998 ± 0.121	**0.637002 ± 0.051**
	P.cysts	acc	0.635707 ± 0.011	0.518342 ± 0.024	0.677318 ± 0.041	0.634214 ± 0.024	0.623264 ± 0.008	**0.687223 ± 0.031**
		κ	0.368501 ± 0.023	0.215635 ± 0.010	0.470794 ± 0.064	0.388012 ± 0.058	0.387465 ± 0.006	**0.491069 ± 0.035**

4.6.2.2 *C5: Classifiers trained by pseudo-labels obtained from high-VS projections have a high CP*

Table 16 shows the results of classification for VGG-16 trained from the pseudolabeling performed on latent spaces from SimCLR, SupCon, and SimCLR+SupCon. Figure 4.6(right) shows the same data as a heatmap plot for interpretation ease. We see that the *"C2: Propagation results"* (left) and *"C3: Classification results"* (right) subfigures in Fig. 4.6 are very similar – hence, CP is indeed highly correlated with VS.

We got the best CP results by the methods using the ImageNet pre-trained weights – see the three rightmost columns in Fig. 4.6(right) which have brighter cells. Also, SimCLR+SupCon obtained the best results for most datasets, as shown by the lighter color in the last column. Projection-wise, we see again that t-SNE, UMAP, and MDS lead to the highest overall CP values. Separately, κ shows a bigger gap between projection techniques with high values (lighter/yellow cells) and low values (darker/purple cells). These results show that VGG-16 can learn from the pseudo-labels since it provided good classification accuracies and κ – higher than 0.80 and 0.70, respectively – among the studied datasets and projections.

4.6.3 *Discussion*

We next discuss the main findings emerging from our results.

4.6.3.1 *Data separation* vs *visual separation depends on the projection technique*

Figures 4.3 and 4.6(left) give us two main insights.

First, we see a similar pattern in κ (lighter/yellow or darker/purple colors) for distinct *datasets*. DS and VS are correlated for some projections P, *e.g.*, MDS, t-SNE, and UMAP; and somewhat less for FA and PCA. For other projections, *e.g.*, FICA, ISO, KPCA, LLE, and MLLE, this correlation is absent. For these last projections, while there is some variation in κ for distinct datasets in DS, VS is close to 0 for *all* datasets. This tells that some projections can map DS to VS quite well, whereas others cannot and tend to create a low VS no matter how high DS is. In short, VS *strongly depends* on the projection technique P.

Secondly, we see a distinct pattern for those projections that capture well DS in their VS. Datasets with medium-high DS (accuracy in $[0.8, 1]$, κ in $[0.7, 1.00]$) tends to yield also medium-

Table 16: C3: VGG-16's classification results on T when using pseudo labels from SimCLR's, SupCon and SimCLR+SupCon latent spaces, from scratch and with ImageNet pre-trained weights. Best values per dataset are in bold.

techniques	datasets	metric	baseline	from scratch			pre-trained		
				SimCLR	SupCon	SimCLR+SupCon	SimCLR	SupCon	SimCLR+SupCon
FA	H.eggs*	acc	0.812932 ± 0.059	0.791588 ± 0.071	0.569994 ± 0.077	0.816070 ± 0.037	**0.826114 ± 0.043**	0.660389 ± 0.087	0.797238 ± 0.074
		κ	0.775954 ± 0.073	0.750532 ± 0.086	0.481551 ± 0.096	0.780221 ± 0.044	**0.790394 ± 0.052**	0.589562 ± 0.106	0.758894 ± 0.088
	P.cysts*	acc	**0.757209 ± 0.015**	0.574395 ± 0.132	0.684544 ± 0.056	0.590542 ± 0.015	0.663783 ± 0.037	0.586217 ± 0.049	0.648212 ± 0.033
		κ	**0.651933 ± 0.023**	0.435072 ± 0.157	0.585071 ± 0.064	0.482646 ± 0.019	0.533400 ± 0.052	0.426891 ± 0.093	0.516758 ± 0.055
	H.larvae	acc	0.930806 ± 0.026	0.917535 ± 0.033	0.889099 ± 0.015	0.948183 ± 0.018	0.891627 ± 0.057	0.947551 ± 0.010	**0.951343 ± 0.014**
		κ	0.613432 ± 0.230	0.645066 ± 0.129	0.441950 ± 0.135	0.723843 ± 0.126	0.624029 ± 0.171	0.731107 ± 0.071	**0.758448 ± 0.094**
	H.eggs	acc	**0.862234 ± 0.010**	0.708822 ± 0.015	0.582790 ± 0.047	0.655367 ± 0.035	0.730552 ± 0.010	0.667753 ± 0.058	0.782269 ± 0.057
		κ	**0.740861 ± 0.028**	0.500625 ± 0.016	0.230899 ± 0.071	0.415120 ± 0.056	0.525028 ± 0.024	0.358934 ± 0.112	0.602154 ± 0.095
	P.cysts	acc	**0.850691 ± 0.018**	0.650528 ± 0.010	0.569604 ± 0.016	0.698595 ± 0.039	0.638221 ± 0.013	0.646697 ± 0.024	0.736445 ± 0.012
		κ	**0.751667 ± 0.028**	0.312113 ± 0.026	0.241764 ± 0.048	0.430761 ± 0.136	0.380899 ± 0.060	0.345129 ± 0.061	0.556038 ± 0.038
FICA	H.eggs*	acc	**0.812932 ± 0.059**	0.209040 ± 0.010	0.165725 ± 0.075	0.271186 ± 0.088	0.161331 ± 0.008	0.161958 ± 0.058	0.259887 ± 0.056
		κ	**0.775954 ± 0.073**	0.002840 ± 0.004	0.009936 ± 0.022	0.092821 ± 0.104	0.002707 ± 0.037	-0.001412 ± 0.005	0.077978 ± 0.066
	P.cysts*	acc	**0.757209 ± 0.015**	0.315744 ± 0.106	0.390138 ± 0.000	0.388408 ± 0.002	0.333045 ± 0.081	0.247981 ± 0.101	0.299884 ± 0.081
		κ	**0.651933 ± 0.023**	0.000569 ± 0.001	0.000000 ± 0.000	0.002938 ± 0.004	-0.008647 ± 0.011	-0.000579 ± 0.001	-0.001702 ± 0.015
	H.larvae	acc	**0.930806 ± 0.026**	0.871722 ± 0.002	0.872986 ± 0.000	0.756668 ± 0.141	0.872670 ± 0.000	0.841706 ± 0.039	0.872670 ± 0.001
		κ	**0.613432 ± 0.230**	0.001145 ± 0.002	0.000000 ± 0.000	0.024169 ± 0.009	0.003042 ± 0.006	0.080839 ± 0.073	0.006713 ± 0.005
	H.eggs	acc	**0.862234 ± 0.010**	0.649935 ± 0.005	0.647762 ± 0.004	0.644937 ± 0.009	0.646893 ± 0.005	0.467623 ± 0.257	0.651456 ± 0.003
		κ	**0.740861 ± 0.028**	-0.004383 ± 0.005	-0.005765 ± 0.003	-0.007738 ± 0.012	-0.005335 ± 0.004	0.000909 ± 0.006	-0.001991 ± 0.002
	P.cysts	acc	**0.850691 ± 0.018**	0.595147 ± 0.003	0.584814 ± 0.015	0.597353 ± 0.000	0.592825 ± 0.006	0.408452 ± 0.267	0.462441 ± 0.191
		κ	**0.751667 ± 0.028**	-0.001485 ± 0.002	0.003837 ± 0.010	0.000000 ± 0.000	-0.005944 ± 0.008	-0.002462 ± 0.003	-0.008557 ± 0.012
ISO	H.eggs*	acc	**0.812932 ± 0.059**	0.248588 ± 0.062	0.173886 ± 0.028	0.173258 ± 0.033	0.164470 ± 0.058	0.118644 ± 0.051	0.137477 ± 0.050
		κ	**0.775954 ± 0.073**	0.057848 ± 0.080	0.003443 ± 0.003	0.010141 ± 0.012	0.009016 ± 0.013	-0.000439 ± 0.000	0.018688 ± 0.012
	P.cysts*	acc	**0.757209 ± 0.015**	0.326701 ± 0.098	0.247981 ± 0.101	0.315455 ± 0.106	0.173299 ± 0.010	0.141003 ± 0.065	0.180219 ± 0.010
		κ	**0.651933 ± 0.023**	0.008227 ± 0.007	0.000000 ± 0.000	0.000000 ± 0.000	0.000000 ± 0.000	-0.000103 ± 0.000	0.000000 ± 0.000
	H.larvae	acc	**0.930806 ± 0.026**	0.872986 ± 0.000	0.662243 ± 0.298	0.822117 ± 0.072	0.872670 ± 0.001	0.740916 ± 0.189	0.822749 ± 0.072
		κ	**0.613432 ± 0.230**	0.014444 ± 0.013	-0.015757 ± 0.030	0.031433 ± 0.044	0.031303 ± 0.015	0.016744 ± 0.020	0.048051 ± 0.059
	H.eggs	acc	**0.862234 ± 0.010**	0.649283 ± 0.006	0.577357 ± 0.107	0.642329 ± 0.009	0.621252 ± 0.044	0.643850 ± 0.013	0.650152 ± 0.004
		κ	**0.740861 ± 0.028**	0.003225 ± 0.005	0.020891 ± 0.030	-0.006907 ± 0.003	0.001426 ± 0.003	-0.006799 ± 0.010	-0.003234 ± 0.004
	P.cysts	acc	**0.850691 ± 0.018**	0.586634 ± 0.023	0.599036 ± 0.006	0.596424 ± 0.001	0.596424 ± 0.001	0.597237 ± 0.000	0.596308 ± 0.001
		κ	**0.751667 ± 0.028**	-0.003610 ± 0.005	0.009051 ± 0.002	-0.001265 ± 0.001	-0.001084 ± 0.002	-0.000161 ± 0.000	-0.001397 ± 0.001
KPCA	H.eggs*	acc	**0.812932 ± 0.059**	0.193974 ± 0.003	0.112994 ± 0.031	0.209340 ± 0.010	0.133199 ± 0.054	0.157584 ± 0.063	0.163214 ± 0.059
		κ	**0.775954 ± 0.073**	0.000112 ± 0.003	0.000674 ± 0.001	0.012136 ± 0.016	-0.018456 ± 0.013	-0.000097 ± 0.000	0.002955 ± 0.052
	P.cysts*	acc	**0.757209 ± 0.015**	0.330561 ± 0.119	0.390138 ± 0.000	0.249423 ± 0.100	0.187139 ± 0.000	0.322376 ± 0.096	0.051038 ± 0.002
		κ	**0.651933 ± 0.023**	-0.025975 ± 0.037	0.000000 ± 0.000	0.003966 ± 0.005	0.000000 ± 0.000	0.000000 ± 0.000	-0.002450 ± 0.003
	H.larvae	acc	**0.930806 ± 0.026**	0.872670 ± 0.001	0.780411 ± 0.132	0.848657 ± 0.036	0.872354 ± 0.002	0.825908 ± 0.070	0.807899 ± 0.091
		κ	**0.613432 ± 0.230**	0.003063 ± 0.007	0.037027 ± 0.027	0.104310 ± 0.066	0.013242 ± 0.010	0.066086 ± 0.022	0.040867 ± 0.052
	H.eggs	acc	**0.862234 ± 0.010**	0.650804 ± 0.004	0.500217 ± 0.217	0.644285 ± 0.007	0.651021 ± 0.002	0.651890 ± 0.002	0.652977 ± 0.001
		κ	**0.740861 ± 0.028**	-0.003327 ± 0.004	0.006654 ± 0.010	-0.008322 ± 0.006	-0.001763 ± 0.002	-0.001528 ± 0.001	-0.001004 ± 0.001
	P.cysts	acc	**0.850691 ± 0.018**	0.596772 ± 0.001	0.427029 ± 0.240	0.596773 ± 0.001	0.411471 ± 0.260	0.597237 ± 0.000	0.597005 ± 0.000
		κ	**0.751667 ± 0.028**	-0.000805 ± 0.001	0.001823 ± 0.003	-0.000298 ± 0.000	-0.000119 ± 0.002	-0.000161 ± 0.000	-0.000483 ± 0.001
LLE	H.eggs*	acc	**0.812932 ± 0.059**	0.192090 ± 0.003	0.152542 ± 0.075	0.199623 ± 0.010	0.185185 ± 0.028	0.227872 ± 0.045	0.151287 ± 0.075
		κ	**0.775954 ± 0.073**	0.000017 ± 0.000	0.002360 ± 0.002	0.000179 ± 0.000	0.018815 ± 0.017	0.054332 ± 0.071	0.000028 ± 0.000
	P.cysts*	acc	**0.757209 ± 0.015**	0.173299 ± 0.010	0.262976 ± 0.090	0.355824 ± 0.126	0.209054 ± 0.140	0.247981 ± 0.101	0.180219 ± 0.010
		κ	**0.651933 ± 0.023**	0.000000 ± 0.000	0.006750 ± 0.010	0.080931 ± 0.115	0.000000 ± 0.000	0.000000 ± 0.000	0.000000 ± 0.000
	H.larvae	acc	**0.930806 ± 0.026**	0.872354 ± 0.002	0.763349 ± 0.154	0.774723 ± 0.141	0.870774 ± 0.006	0.806319 ± 0.093	0.872986 ± 0.000
		κ	**0.613432 ± 0.230**	0.009639 ± 0.007	0.034258 ± 0.030	0.027194 ± 0.019	0.027385 ± 0.021	0.035078 ± 0.027	0.000000 ± 0.000
	H.eggs	acc	**0.862234 ± 0.010**	0.533681 ± 0.165	0.645589 ± 0.009	0.621903 ± 0.041	0.478705 ± 0.225	0.617557 ± 0.051	0.650804 ± 0.002
		κ	**0.740861 ± 0.028**	0.005994 ± 0.013	0.019156 ± 0.029	-0.006581 ± 0.005	0.008658 ± 0.022	0.001140 ± 0.002	-0.002347 ± 0.002
	P.cysts	acc	**0.850691 ± 0.018**	0.597353 ± 0.000	0.427245 ± 0.240	0.586690 ± 0.011	0.597353 ± 0.000	0.577383 ± 0.016	0.473906 ± 0.176
		κ	**0.751667 ± 0.028**	0.000000 ± 0.000	0.000985 ± 0.002	-0.008861 ± 0.012	0.000303 ± 0.001	0.054789 ± 0.067	0.014402 ± 0.019
MDS	H.eggs*	acc	0.812932 ± 0.059	0.870057 ± 0.007	0.698054 ± 0.053	0.839924 ± 0.030	0.868801 ± 0.056	0.750786 ± 0.044	**0.889517 ± 0.009**
		κ	0.775954 ± 0.073	0.844017 ± 0.009	0.633636 ± 0.066	0.806060 ± 0.038	0.843305 ± 0.067	0.696556 ± 0.054	**0.867837 ± 0.011**
	P.cysts*	acc	**0.757209 ± 0.015**	0.660035 ± 0.042	0.729815 ± 0.018	0.679642 ± 0.008	0.675026 ± 0.019	0.688004 ± 0.049	0.623414 ± 0.047
		κ	**0.651933 ± 0.023**	0.535152 ± 0.066	0.636650 ± 0.017	0.558141 ± 0.010	0.452008 ± 0.046	0.574029 ± 0.067	0.485190 ± 0.040
	H.larvae	acc	0.930806 ± 0.026	0.896366 ± 0.010	0.907425 ± 0.021	0.941864 ± 0.022	**0.961137 ± 0.003**	0.948183 ± 0.014	0.944392 ± 0.020
		κ	0.613432 ± 0.230	0.527845 ± 0.047	0.512390 ± 0.132	0.670216 ± 0.171	**0.828970 ± 0.016**	0.732767 ± 0.094	0.702961 ± 0.166
	H.eggs	acc	**0.862234 ± 0.010**	0.735767 ± 0.006	0.556280 ± 0.010	0.755324 ± 0.023	0.736636 ± 0.040	0.649283 ± 0.049	0.787701 ± 0.039
		κ	**0.740861 ± 0.028**	0.527193 ± 0.052	0.250879 ± 0.023	0.552969 ± 0.033	0.529298 ± 0.057	0.342224 ± 0.071	0.598315 ± 0.075
	P.cysts	acc	**0.850691 ± 0.018**	0.640660 ± 0.022	0.541159 ± 0.012	0.672820 ± 0.066	0.639273 ± 0.041	0.684663 ± 0.022	0.742947 ± 0.021
		κ	**0.751667 ± 0.028**	0.360158 ± 0.031	0.293691 ± 0.017	0.460289 ± 0.091	0.516147 ± 0.048	0.437128 ± 0.046	0.572537 ± 0.027
MLLE	H.eggs*	acc	**0.812932 ± 0.059**	0.176397 ± 0.069	0.186440 ± 0.007	0.149404 ± 0.053	0.231638 ± 0.028	0.133710 ± 0.051	0.172630 ± 0.061
		κ	**0.775954 ± 0.073**	0.029830 ± 0.024	0.030248 ± 0.055	0.003127 ± 0.004	0.038996 ± 0.038	0.000489 ± 0.001	0.039277 ± 0.051
	P.cysts*	acc	**0.757209 ± 0.015**	0.322664 ± 0.095	0.254902 ± 0.096	0.394175 ± 0.005	0.198673 ± 0.015	0.240773 ± 0.106	0.187139 ± 0.000
		κ	**0.651933 ± 0.023**	0.000096 ± 0.000	0.000000 ± 0.000	0.009169 ± 0.009	0.001291 ± 0.006	0.000000 ± 0.000	0.000000 ± 0.000
	H.larvae	acc	**0.930806 ± 0.026**	0.872618 ± 0.001	0.872670 ± 0.003	0.870774 ± 0.002	0.854029 ± 0.032	0.872986 ± 0.000	0.870774 ± 0.007
		κ	**0.613432 ± 0.230**	0.008591 ± 0.012	0.008602 ± 0.071	0.027161 ± 0.028	0.121338 ± 0.009	0.000000 ± 0.000	0.236101 ± 0.148
	H.eggs	acc	**0.862234 ± 0.010**	0.619513 ± 0.121	0.651456 ± 0.002	0.658418 ± 0.007	0.636679 ± 0.012	0.651456 ± 0.001	0.651456 ± 0.001
		κ	**0.740861 ± 0.028**	-0.019102 ± 0.011	-0.002448 ± 0.003	-0.010811 ± 0.001	0.006862 ± 0.014	-0.000265 ± 0.000	-0.001735 ± 0.000
	P.cysts	acc	**0.850691 ± 0.018**	0.583769 ± 0.015	0.585743 ± 0.016	0.590889 ± 0.000	0.592128 ± 0.007	0.528387 ± 0.095	0.596076 ± 0.001
		κ	**0.751667 ± 0.028**	-0.013466 ± 0.014	-0.008483 ± 0.012	-0.000645 ± 0.001	-0.003423 ± 0.005	0.019087 ± 0.030	-0.001528 ± 0.001
PCA	H.eggs*	acc	0.812932 ± 0.059	0.782800 ± 0.022	0.575644 ± 0.086	0.827997 ± 0.033	0.865035 ± 0.065	0.593220 ± 0.072	**0.868801 ± 0.027**
		κ	0.775954 ± 0.073	0.739026 ± 0.028	0.483446 ± 0.110	0.792577 ± 0.041	0.837979 ± 0.079	0.515894 ± 0.080	**0.844098 ± 0.031**
	P.cysts*	acc	**0.757209 ± 0.015**	0.619666 ± 0.083	0.719147 ± 0.025	0.641580 ± 0.059	0.677047 ± 0.019	0.626586 ± 0.027	0.679354 ± 0.048
		κ	**0.651933 ± 0.023**	0.496203 ± 0.104	0.632033 ± 0.030	0.520132 ± 0.069	0.556229 ± 0.027	0.474462 ± 0.041	0.551953 ± 0.072
	H.larvae	acc	0.930806 ± 0.026	0.906793 ± 0.005	0.912480 ± 0.029	0.931438 ± 0.019	0.923539 ± 0.026	0.949447 ± 0.014	**0.957662 ± 0.002**
		κ	0.613432 ± 0.230	0.567678 ± 0.037	0.557973 ± 0.171	0.644459 ± 0.162	0.696070 ± 0.105	0.737386 ± 0.095	**0.807875 ± 0.003**
	H.eggs	acc	**0.862234 ± 0.010**	0.668622 ± 0.046	0.618644 ± 0.054	0.664711 ± 0.029	0.744242 ± 0.027	0.647327 ± 0.066	0.812690 ± 0.032
		κ	**0.740861 ± 0.028**	0.433312 ± 0.065	0.260372 ± 0.051	0.411077 ± 0.062	0.543626 ± 0.048	0.321544 ± 0.130	0.647622 ± 0.057
	P.cysts	acc	**0.850691 ± 0.018**	0.615000 ± 0.017	0.531638 ± 0.005	0.668899 ± 0.031	0.658307 ± 0.032	0.664229 ± 0.001	0.720771 ± 0.027
		κ	**0.751667 ± 0.028**	0.284629 ± 0.104	0.263112 ± 0.026	0.395488 ± 0.115	0.344865 ± 0.153	0.403226 ± 0.034	0.535129 ± 0.032
t-SNE	H.eggs*	acc	0.812932 ± 0.059	0.932831 ± 0.032	0.846202 ± 0.025	**0.964846 ± 0.022**	0.850596 ± 0.051	0.954802 ± 0.023	0.938481 ± 0.017
		κ	0.775954 ± 0.073	0.919898 ± 0.038	0.816217 ± 0.030	**0.958337 ± 0.026**	0.822694 ± 0.059	0.946002 ± 0.027	0.927126 ± 0.020
	P.cysts*	acc	0.757209 ± 0.015	**0.771626 ± 0.050**	0.697232 ± 0.020	0.732122 ± 0.023	0.644464 ± 0.047	0.756920 ± 0.012	0.658881 ± 0.028
		κ	0.651933 ± 0.023	**0.688570 ± 0.066**	0.594524 ± 0.035	0.633941 ± 0.042	0.534253 ± 0.057	0.666824 ± 0.024	0.544564 ± 0.034
	H.larvae	acc	0.930806 ± 0.026	0.931501 ± 0.013	0.902685 ± 0.029	0.933365 ± 0.017	**0.956714 ± 0.003**	0.953871 ± 0.003	0.952291 ± 0.004
		κ	0.613432 ± 0.230	0.859203 ± 0.111	0.595899 ± 0.123	0.785465 ± 0.079	**0.810965 ± 0.018**	0.786613 ± 0.022	0.787406 ± 0.014
	H.eggs	acc	**0.862234 ± 0.010**	0.761191 ± 0.027	0.636028 ± 0.025	0.777705 ± 0.013	0.824641 ± 0.003	0.725772 ± 0.070	0.819426 ± 0.024
		κ	**0.740861 ± 0.028**	0.599112 ± 0.036	0.299141 ± 0.077	0.621008 ± 0.032	0.685570 ± 0.014	0.500910 ± 0.118	0.686552 ± 0.042
	P.cysts	acc	**0.850691 ± 0.018**	0.698711 ± 0.003	0.592476 ± 0.025	0.715748 ± 0.019	0.719378 ± 0.007	0.701614 ± 0.011	0.719959 ± 0.032
		κ	**0.751667 ± 0.028**	0.484463 ± 0.017	0.313296 ± 0.022	0.559176 ± 0.038	0.483280 ± 0.034	0.460516 ± 0.054	0.527912 ± 0.030
UMAP	H.eggs*	acc	0.812932 ± 0.059	0.913999 ± 0.039	0.846202 ± 0.055	0.937225 ± 0.033	0.849341 ± 0.088	**0.950498 ± 0.024**	0.945386 ± 0.004
		κ	0.775954 ± 0.073	0.897254 ± 0.047	0.817901 ± 0.065	0.925412 ± 0.039	0.820393 ± 0.105	**0.940671 ± 0.028**	0.935039 ± 0.005
	P.cysts*	acc	**0.757209 ± 0.015**	0.728862 ± 0.044	0.686851 ± 0.023	0.666667 ± 0.040	0.601499 ± 0.098	0.702422 ± 0.022	0.636678 ± 0.004
		κ	**0.651933 ± 0.023**	0.621347 ± 0.063	0.590353 ± 0.021	0.547625 ± 0.062	0.480729 ± 0.113	0.590577 ± 0.043	0.500455 ± 0.014
	H.larvae	acc	0.930806 ± 0.026	0.911848 ± 0.010	0.883096 ± 0.034	0.935861 ± 0.015	**0.959242 ± 0.006**	0.947867 ± 0.013	0.958610 ± 0.005
		κ	0.613432 ± 0.230	0.860489 ± 0.075	0.503848 ± 0.098	0.718764 ± 0.056	**0.826163 ± 0.025**	0.739471 ± 0.099	0.805239 ± 0.026
	H.eggs	acc	**0.862234 ± 0.010**	0.755107 ± 0.009	0.644285 ± 0.019	0.781399 ± 0.045	0.790960 ± 0.025	0.707084 ± 0.063	0.814863 ± 0.036
		κ	**0.740861 ± 0.028**	0.561245 ± 0.010	0.261623 ± 0.039	0.616187 ± 0.079	0.633562 ± 0.039	0.449519 ± 0.148	0.680416 ± 0.051
	P.cysts	acc	**0.850691 ± 0.018**	0.684547 ± 0.012	0.575061 ± 0.010	0.743295 ± 0.036	0.691165 ± 0.025	0.703819 ± 0.017	0.741089 ± 0.014
		κ	**0.751667 ± 0.028**	0.408555 ± 0.082	0.251174 ± 0.026	0.561552 ± 0.073	0.445321 ± 0.084	0.500526 ± 0.021	0.566037 ± 0.016

high VS accuracy and κ (compare Figs. 4.3 and 4.6). *H.eggs** and *H.larvae* are examples of these datasets. We see a similar pattern for low-medium accuracy and κ: For DS accuracy in $[0, 0.5]$ and κ in $[0, 0.4]$, we get medium-high accuracy and κ for VS (see Fig. 4.6 for *P.cysts**, *P.cysts*, or *H.eggs*). This tells that some projection techniques can not only correlate VS with DS, but also keep differences for distinct datasets, while others do not.

4.6.3.2 *Assessing the quality of visual separation*

As explained earlier, we use κ as a measure of visual separation of (same-labeled) points in a projection. While this is arguably intuitive – one can propagate labels easier when surrounding points are unlabeled or have the same label than when surrounding points would have many different labels – we would like to directly test how κ and VS *as perceived by humans* agree.

To assess the correlation between κ and perceived VS, we ranked our results as follows. We computed the average accuracy and κ for each P (over all contrastive learning methods, initialization strategies, and datasets). Next, we sorted the projections P over accuracy and κ (see Tab 17). Finally, we show in Fig. 4.7 the actual projection results for the best, medium, and worst projection following the above ranking.

Table 17: Average values of propagation results (C2) for accuracy and κ and studied P. P is ordered in increasing order for the respective metrics.

metrics	**techniques ordered by metric**									
acc	ISO	M-LLE	LLE	FICA	K-PCA	FA	PCA	MDS	UMAP	t-SNE
	0.400488	0.406233	0.411292	0.413671	0.414182	0.690367	0.697184	0.721874	0.752606	0.761406
κ	LLE	FICA	M-LLE	K-PCA	ISO	FA	PCA	MDS	UMAP	t-SNE
	0.013256	0.014352	0.014943	0.014971	0.015558	0.497711	0.505837	0.538983	0.583886	0.600066

The best-ranked projection in Fig. 4.7 is t-SNE. In line with this ranking, we indeed see a quite good separation of points having a label from those having different labels (or gray, *i.e.*, unlabeled). For *H.eggs**, all three latent space projections (SimCLR, SupCon, and SimCLR+SupCon) show a clear VS, and we see that this leads to almost no color mixing in the propagated pseudo-labels. For *P.cysts**, there is a clearly separated group (red) in all three projections which also has a single color (label). The remaining projections, which have no clear VS in terms of distinct groups, show a mix of different colors. For *H.larvae*, the larvae class (red) is better separated from the big group of impurities (gray), and this correlates with the larvae samples being all located in a tail-like

periphery of the projection – thus, better visually separated from the rest. For *H.eggs*, we see how the visually separated groups show almost no color mixing, whereas the parts of the projection where no VS is present show color mixing. For *P.cysts*, the projections have even less VS, and we see how labels get even more mixed – for instance, the impurity class (gray) is spread all over the projection.

The medium-ranked projection technique in Fig. 4.7 is FA. Its scatterplots show a less clear correlation between VS and lack of label mixing in distinct groups. For *H.eggs**, we notice some VS for SimCLR and SupCon. Some groups are better clustered (red and yellow) than others (brown and gray), but with few whitespace among those groups. No clear VS can be seen for all other datasets and contrastive learning approaches – the points are condensed in a single group with similar colors close to each other in this group.

Finally, LLE scored as the worst method in our ranking. Figure 4.7 shows, indeed, no clear separation of points into groups having the same color – not only are colors intermixed all over the projection, bit it is often hard to even visually 'split' the projection into distinct point groups.

All above results show, first of all, that the κ ranking of projections is indeed in line with our perception of visual separation. This empirically validates our decision to measure the latter by computing the former. Also, our results show that a good VS leads to a low mixing of the propagated labels, and conversely. In turn, a low mixing leads to a high classification performance (CP), and conversely, *i.e.*, our claim C5. Table 17 shows this by ranking the average results of each projection by each metric for the baseline and VGG-16 trained with the generated pseudo-labels. We see the best κ value for t-SNE with a clear VS and little label mixing in the projections. Conversely, we see the medium-low κ values $[0, 0.6]$ for FA and LLE with poor VS and color-mixing in their projections.

4.6.3.3 *Data separation* vs *visual separation* vs *classifier performance*

Section 4.6.3.1 discussed how a good projection P can map high DS to high VS. Section 4.6.3.2 showed that VS can be measured by κ. Section 4.6.2.2 showed that CP is correlated with VS. Let us now put together all these observations.

Figure 4.8 shows plots of the correlation between (i) DS and VS and (ii) VS and CP, using both accuracy or κ. To simplify the plot, we averaged over initialization strategies (pre-trained,

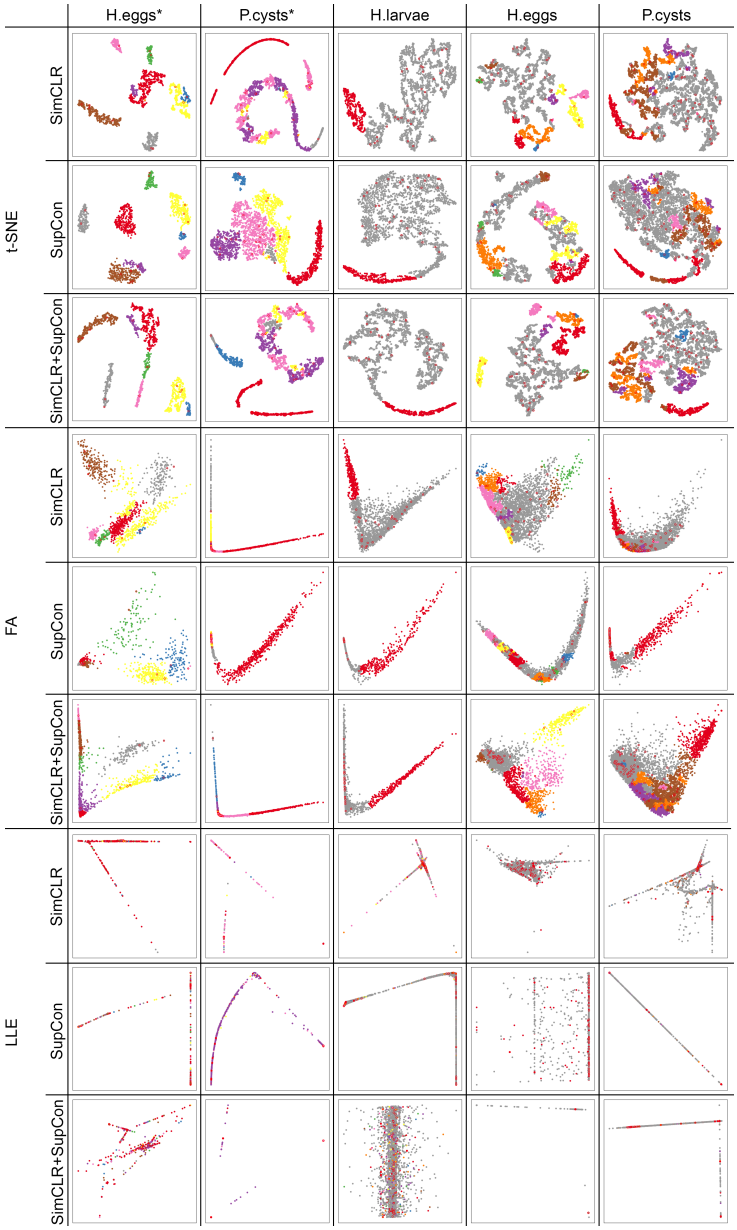


Figure 4.7: 2D projections of the best (t-SNE), medium (FA), and worst (LLE) P for the three contrastive latent spaces (SimCLR, SupCon, SimCLR+SupCon) and the six studied datasets (columns). Colored points show the computed pseudo-labels for distinct classes. Outlined points in red represent the supervised points.

scratch) per contrastive learning method. As such, every point in a plot is a (dataset, contrastive learning method, projection technique) combination. Points are colored to indicate projection techniques. For all same-projection points (15 of them), we also plot a trend line showing their correlation.

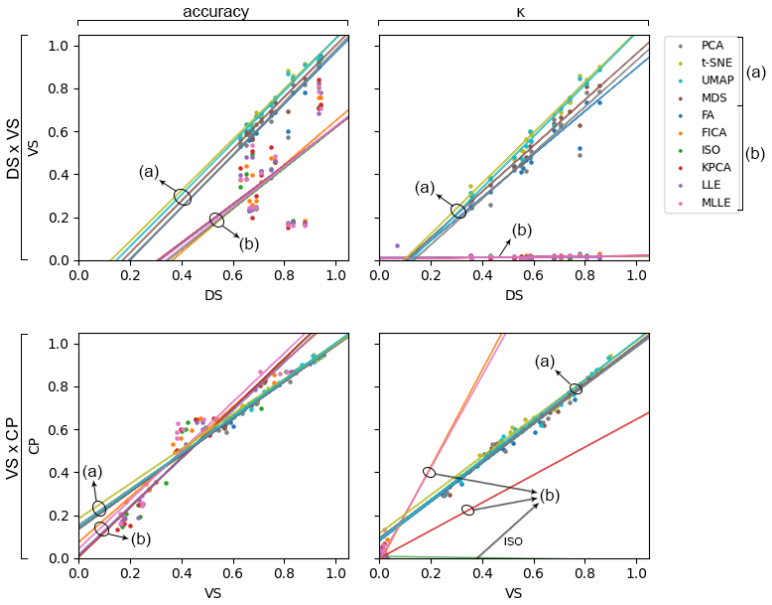


Figure 4.8: Correlation plots *DS-VS* and *VS-CP* for accuracy and κ. In each plot, a point corresponds to the average of initialization strategies (scratch or pre-trained) for each dataset and contrastive learning method projected by a distinct P (in different colors). Each line represents the trend of same-P (same-color) points. Projections P are grouped into those which (a) map well DS to VS and (b) create poor VS regardless of DS.

Figure 4.8 gives several insights. The DS-VS correlation plots confirm our findings in Sec. 4.6.3.1. For accuracy, all trend lines have an increasing slope which tells a positive DS-VS correlation, i.e., good DS leads to good VS. We notice two main groups of trend lines: one for (a) UMAP, t-SNE, MDS, PCA, and FA, and another for (b) all other projection techniques. Indeed, Fig. 4.6(left) showed a pattern of darker/blue points for (b) (see e.g. the *H.eggs**'s row). For κ, we see an increasing slope for trend lines only for (a), and horizontal lines for (b). This says that only projections in group (a) have a positive DS-VS correlation. Figure 4.6(left) confirms this by showing a stronger pattern of darker/blue cells, i.e., low κ, for *all datasets* in (b). We conclude that

projection techniques in group (a) yield a strong DS-VS correlation, whereas projections in group (b) do not do that but rather generate poor-VS results for any DS value.

The VS-CP correlation plots strengthens our earlier findings from Fig. 4.6 in Sec. 4.6.2.2. For accuracy, all trend lines show an increasing slope and thus a positive VS-CP, with a slight difference between projections in group (a) and group (b). For κ, we also see the positive VS-CP correlation except for ISO which has an almost flat line. Additionally, we see that the trend lines in the middle of the plot correspond for projections in group (a). For these projections, both VS and CP spread widely over the $[0, 1]$ range. In contrast, lines for projections in group (b) have values lower than 0.05 for VS and CP. This is the same pattern found in our earlier analysis for DS *vs.* VS, i.e., the presence of two groups of projection techniques with distinct correlation values. The existence of these groups highlights the importance of the chosen projection technique in supporting the DS, VS, and CP links. In detail, for UMAP, t-SNE, MDS, PCA, and FA (group (a)), we confirm a strong DS-VS and VS-CP, thus DS-VS-CP, correlation. In contrast, for FICA, ISO, KPCA, LLE, and MLLE (group (b)), we find a VS-CP *relation* in the sense that both these values are low. However, we did not find any DS-VS *correlation* since these projections map any DS values in $[0, 1]$ to very poor VS values (close to zero).

The above insights connect to our qualitative and ranked analysis of the scatterplots in Fig. 4.7 and Tab. 17. The presence of two groups in the correlation plots is also connected to the quality of the perceived projection scatterplots. Scatterplots with a clear VS among groups of different classes/colors separated by whitespace come from t-SNE, a projection in the group (a). Conversely, scatterplots with no VS among groups of different classes/colors and without any whitespace separation come from LLE, a projection in group (b).

Our key finding – that projections split in two groups – (a) and (b), with good, respectively poor, DS-VS correlation – complements and extends the largest former quantitative comparison of projection techniques (Espadoto et al., 2019a). Our set (a), i.e., projections which best preserve DS-VS and are best for building high-CP classifiers, matches quite closely the projections found best in (Espadoto et al., 2019a). However, our quality criteria – VS measured by pseudolabeling performance and CP measured by classifier performance are *completely* different than the criteria used in (Espadoto et al., 2019a) to measure projection quality. The latter criteria include metrics such as trustworthiness, conti-

nuity, normalized stress, neighborhood hit, and Shepard correlation. As outlined in Sec. 4.2, such metrics only measure how a projection preserves *local* data structure. This is typically done by using small-sized neighborhoods of under 10 points in both the data and projection space. Visual separation (VS), however, occurs at much larger scales in a scatterplot. Moreover, we measure VS in a completely different way, namely, as the performace of a ML algorithm (pseudolabeling) that handles all the scatterplot points globally rather than in terms of small-scale, independent, neighborhoods. As such, the fact that we 'rank' projections quite similarly to (Espadoto et al., 2019a), but using completely different metrics, has several *potentially* far-reaching implications (all to be tested by future work):

- Projections assessed by local quality metrics (Espadoto et al., 2019a) can be used to predict classifier performance;
- Higher-level properties like visual separation (VS) can be predicted by lower-level metrics (Espadoto et al., 2019a);
- the VS of a projection, measured by κ or accuracy during pseudolabeling, can be an additional quality metric for generic projection assessment. Apart from their ability to gauge a projection more globally, such metrics are also much faster to compute than neighborhood-based metrics such as trustworthiness, continuity, normalized stress, or Shepard correlation.

4.7 CONCLUSION

We presented a detailed study of the link between [C1] data separation (DS) in a high-dimensional partially-labeled dataset, [C2] the visual separation (VS) in a 2D projection of that dataset, and [C3] the performance of a classifier (CP) constructed by pseudo-labeling the abovementioned projection.

In our work, we used two contrastive learning approaches (SimCLR and SupCon) as well as their combination. We projected the latent spaces produced by these methods to 2D using ten projection techniques (FA, FICA, ISO, KPCA, LLE, MDS, MLLE, PCA, t-SNE, and UMAP). We propagated labels in these projections and finally used these pseudo-labels to train a deep-learning classifier for a challenging problem involving the classification of human intestinal parasite images.

Our results show that SimCLR+SupCon performed better than using only SimCLR or SupCon to create a data space with strong

DS. In turn, this allowed us to construct an end-to-end classifier with higher accuracy and κ values than earlier reported for the respective datasets in the literature.

Separately, we showed that [C4] the 10 studied projection techniques can be split into two groups. Projections in the first group (FICA, ISO, KPCA, LLE, and MLLE) yield very poor VS results for any DS values of their input data, and consequently also very poor CP results. These projections are hence not useful for our classifier engineering pipeline and, arguably, they will also have challenges for other infovis applications where VS is important. Projections in the second group (FA, MDS, PCA, t-SNE, and UMAP) show a good DS-VS correlation and, next, [C5] a good VS-CP correlation. These projections are thus ideal for our classifier engineering task and, arguably, for other infovis applications where VS is important. Our work shows, to our knowledge, for the first time how *specific* projection techniques preserve a strong DS-VS-CP correlation (or not). Our findings can assist additional applications in infovis or machine learning where projections are used.

Several future work directions are possible. First, the connection between pseudolabeling quality and visual separation could be further exploited to e.g. design new metrics for visual separation using labeling algorithms or, conversely, to guide labeling by existing visual quality metrics. Secondly, user experiments could be designed and executed to assess more formally the relationship between pseudolabeling quality and perceived visual separation. Finally, we aim to involve users in the loop to assist the automatic pseudolabeling process by *e.g.* adjusting some of the automatically propagated labels based on the human assessment of VS. We believe that this will lead to even more accurate pseudolabels and, ultimately, more accurate classifiers for the problem at hand.

5 MEASURING VISUAL SEPARATION IN PROJECTIONS

5.1 INTRODUCTION

As already introduced in Chapter 2 and supported by various applications in Chapters 3 and 4, projections are a method of choice for capturing the structure of high-dimensional data in a low-dimensional embedding. This low dimensional embedding can be used by the human user for visual exploration but also by automatic algorithms for performing tasks such as pseudolabeling[1].

The success of such tasks involving projections depends on the *visual separation* (VS) of the projection used to depict it. If a dataset exhibits clear *data separation* (DS) into samples of different classes, then analysts should be able to gauge this by seeing a corresponding *visual* separation in the projection, in terms of densely-packed, ideally non-overlapping, groups of points with the same label (within a given group). Conversely, if a dataset exhibits poor data separation, its projection should also show poor visual separation. The relations between VS, DS, and classifier performance (CP) have been discussed in detail in Chapter 4.

Many projection methods have been proposed, using different underlying techniques as graphs, linear algebra, optimization, and neural networks (Nonato and Aupetit, 2018). Such techniques generate a wide variety of scatterplots for the same give dataset, especially when one changes their various hyperparameters. Several *metrics* have been proposed to quantify a projection's quality. However, the most used metrics in the DR literature – Trustworthiness (T) (Venna and Kaski, 2006), Continuity (C) (Venna and Kaski, 2006), Normalized stress (S) (Joia et al., 2011), and Neighborhood hit (N) (Paulovich et al., 2008), introduced in Sec. 2.3.4.1, do not directly measure visual separation at a global projection level but rather more local properties (as discussed further in Sec. 5.3.3).

Figure 18 shows this by a simple example of three DR techniques (t-distributed stochastic neighbor embedding [t-SNE] (van der Maaten and Hinton, 2008), stochastic proximity embedding

1 This chapter is a result of the published paper: "Measuring the quality of projections of high-dimensional labeled data" (Benato et al., 2023a).

[SPE] (Agrafiotis, 2003), and uniform manifold approximation and projection [UMAP] (McInnes et al., 2018)) and their T, C, S, and N metrics, all ranging between 0 (worst quality) and 1 (best quality). The SPE plot has high metric values but arguably much poorer visual separation (of the 9 color-coded classes) into distinct, same-color, point groups than the t-SNE plot which has much lower metric values. UMAP and SPE have similar (high) metric values but, we argue, visual separation is much stronger in the UMAP than the SPE plot. All in all, this shows that these four metrics do not capture visual separation well.

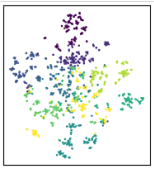
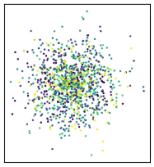
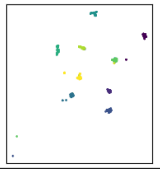
projection	T	C	S	N	scatterplot
t-SNE	0.516584	0.649824	0.809278	0.256746	
SPE	0.833158	0.937247	0.777273	0.839947	
UMAP	0.798405	0.871780	0.762065	0.978571	

Table 18: Values of T, C, S, and N (Sec. 5.3.3) and scatterplots of a dataset (cnae9, Sec. 5.3.1) for three projection techniques (t-SNE, SPE, and UMAP, Sec. 5.3.2). S values are set as $1 - \textit{normalized stress}$ for easy interpretation. We see that the metrics do not correlate well with the perceived visual separation of the label-colored points in the projection.

Recent ML studies have explored the VS information of 2D projection spaces to assess data separation in high dimensions (van der Maaten et al., 2009); understand deep learning classifiers (Rauber et al., 2017b); find misclassified samples (Nonato and Aupetit, 2018); investigate decision boundaries of classifiers (Rodrigues et al., 2019); build better classifiers (Benato et al., 2018, 2021c, 2023e); and investigate the correlation among high-dimensional separability, VS, and classifier performance (Benato et al., 2023c). These studies have also made the object of the work we presented in Chapters 3 and 4. Apart from the above,

some studies have investigated 1-near-neighbor classifiers (van der Maaten and Hinton, 2008; Zhou et al., 2022) and Gaussian mixture models (Abbas et al., 2019) to estimate the quality of clustering. However, they did not aim to specifically measure the correlation between classes *and* clusters. To our knowledge, ML approaches were not directly used to measure this VS relation in projections.

In this chapter, we propose a new VS quality assessment approach based on ML techniques. We exploit earlier findings that studied VS in t-SNE projections to propagate labels, also called pseudo labeling. Projections with high VS (as assessed qualitatively by users) led to good label propagation results (Benato et al., 2023c). Our hypothesis is that the converse is also true: If we measure a good label propagation score, then the projection will have a high VS. For label propagation, we use the semi-supervised optimum path forest algorithm (OPFSemi) (Amorim et al., 2016) in the 2D projection space provided by DR methods. OPFSemi was shown to lead to very good label propagation accuracies in both high-dimensional and low-dimensional spaces (Amorim et al., 2016; Benato et al., 2023e) and as such is a good candidate for this task. We evaluate the label propagation by computing the coefficient of agreement of Cohen's Kappa (κ) (Fleiss and Cohen, 1973) between true and pseudo labels, a simple but fast and effective way to perform this task which works well also for unbalanced labeled datasets. We assess our proposal on 39 projection algorithms for 18 labeled datasets and show that our method correlates with perceived VS (measured by a user study) better than well-known metrics for projection quality used in the DR literature. As such, we argue that our metric is an additional useful way to characterize the quality of a projection, atop of existing projection quality metrics.

Summarizing, we propose a method to quantify VS separation in projections which

a) yields better global and local quantification of VS when compared to four popular metrics in DR;

b) generically handles any high-dimensional quantitative labeled dataset and any projection technique;

c) is easy to use as it is parameter-free;

d) is fast to compute and simple to implement.

5.2 MEASURING VISUAL SEPARATION BY PSEUDO LABELING

Our work builds atop of the findings from Chapter 4 by hypothesizing that high performance in label propagation indicates a high separability of same-label groups in the projection space. Fig. 5.1 illustrates this.

We use OPFSemi in the 2D projection space. As the algorithm explores a complete graph with all samples in a given dataset (Sec. 5.2.2), we argue that OPFSemi can capture local and global information of data distribution instead of local information only – as the neighbor-based metrics T, C, and N do. Other advantages of OPFSemi are that the method is free of parameters and does not make assumptions about the shapes of the classes (Amorim et al., 2016).

Labels are propagated from supervised samples (colored) to unsupervised samples (black). When there is poor VS in a given projection (a), pseudo labels are wrongly assigned, something which we can measure as described next in Sec. 5.2.3. When there is good VS in the projection (b), pseudo labels are accurately assigned.

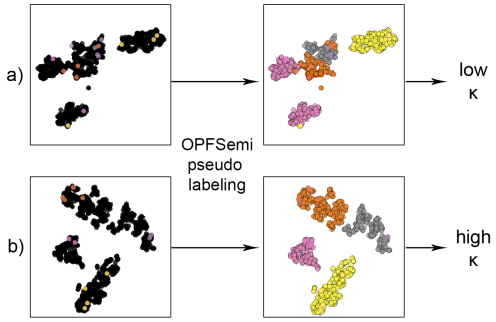


Figure 5.1: Pseudo labeling as a measure of visual separation (see Sec. 5.2).

Fig. 5.2 shows our VS measurement pipeline which is detailed next.

5.2.1 *Sample selection*

We start with a 2D projection $P(D)$ of some labeled dataset D, computed by any desired projection algorithm P. We next randomly split $P(D)$ into a ground-truth dataset A and test dataset

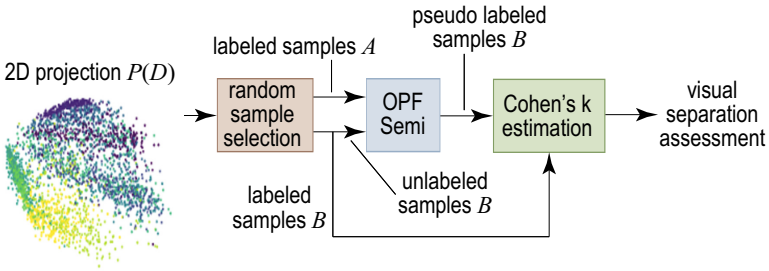


Figure 5.2: Pipeline of our approach to assess VS in projections.

B. In our experiments, we take 50% of the samples in $P(D)$ in each of A and B (different fractions can be considered).

5.2.2 *Using OPFSemi for pseudo labeling*

We pseudo label the samples in B by propagating the true labels from A using OPFSemi (detailed in Chapter 3). Its time complexity is $O(m^2)$ for m nodes, since the graph is complete, but it is possible to precompute a minimum-spanning tree in $O(m^2)$ and perform label propagation (optimum-path forest computation) on this tree in $O(m \log m)$ for any randomly chosen set of prototypes in the case of our application. As the process is calculated over a complete graph with all samples in D, we argue that OPFSemi can capture local and global information of data distribution, instead of local information only.

5.2.3 *Pseudo labeling effectiveness measurement*

To assess the quality of pseudo labeling, we measure the agreement between the true labels (original labels of samples in B) and the pseudo labels assigned to B by OPFSemi. This agreement could be measured by accuracy, f1 score, or AUC, for example. However, such metrics do not take into account the number of false positives and false negatives, which can highly impact the results for datasets having significant class imbalance. Earlier studies showed the advantage of using κ over accuracy to measure the agreement in pseudo labeling (Benato et al., 2023c). In Chapter 2, κ is presented in detail.

5.3 EXPERIMENTAL EVALUATION

To evaluate our usage of OPFSemi to gauge visual separation, we designed several experiments based on the projection-quality benchmark proposed in (Espadoto et al., 2019a) to our knowledge, the largest public such benchmark for DR. All our results and code are openly available (Benato et al., 2023d).

dataset	type	samples	dimensions	labels
bank (Moro et al., 2014)	tables	2059	63	ordinal
cifar10 (Krizhevsky et al., 2009)	images	3250	1024	categorical (10)
cnae9 (Ciarelli and Oliveira, 2009)	text	1080	856	categorical (9)
coil20 (Nene et al., 1996)	images	1440	400	categorical (20)
epileptic (Andrzejak et al., 2001)	tables	5750	178	ordinal
fashion_mnist (Xiao et al., 2017)	images	3000	784	categorical (10)
fmd (Sharan et al., 2009)	images	997	1536	ordinal
har (Anguita et al., 2012)	tables	735	561	categorical (30)
hatespeech (Davidson et al., 2017)	text	3222	100	ordinal
hiva (Guyon et al., 2007)	tables	3076	1617	ordinal
imdb (Maas et al., 2011)	text	3250	700	ordinal
orl (Samaria and Harter, 1994)	images	400	396	categorical (40)
secom (McCann and Johnston, 2008)	tables	1567	590	ordinal
seismic (Sikora et al., 2010)	tables	646	24	ordinal
sentiment (Kotzias et al., 2015)	text	2748	200	ordinal
sms (Almeida et al., 2011)	text	836	500	ordinal
spambase (M. Hopkins and Suermondt, 1999)	text	4601	57	ordinal
svhn (Netzer et al., 2011)	images	733	1024	categorical (9)

Table 19: The 18 datasets used in our evaluation and their characteristics.

5.3.1 *Datasets*

From the benchmark, we chose 18 datasets which are often used in many ML and DR evaluations. Importantly, they are all labeled and, since they are used in ML benchmarks, we know that labels and features are correlated. These datasets come from different application domains and have different sample and dimension counts. Table 19 shows the type of data, sample count, dimension count, and the type and number of labels for each dataset (for more details, see (Espadoto et al., 2019a)).

5.3.2 *Projection algorithms*

From the 44 projection techniques evaluated in (Espadoto et al., 2019a), we used 39 techniques (see Table 20). The remaining 5 techniques were excluded since their code, as provided in (Espadoto et al., 2019a), was hard to understand and run. All

these techniques are well known in the DR literature and practice. Among them are examples of linear and non-linear and global and local, projections. Also, we consider projections that input the high-dimensional samples themselves and projections which only require a similarity (distance) matrix of the samples. We fixed the parameters of all projection techniques to the default values proposed by each author. More details about the chosen projection techniques and their default parameter values can be found in (Espadoto et al., 2019a).

5.3.3 *Metrics*

As we are proposing a new metric to evaluate visual separation (Sec. 5.2), an immediate question is how this metric compares to well established metrics for measuring projection quality. To assess this, we consider, for the latter, the four scalar metrics T, C, N, and S described in Sec. 2.3.4.1. These are also among the metrics considered by the projection benchmark in (Espadoto et al., 2019a). For brevity, we next refer to these four metrics as 'standard' metrics. We compute T, C, and N using $K = 7$ nearest neighbors (see Eqns. 2.7, 2.8, and 2.10), in line with (Espadoto et al., 2019a).

5.3.4 *Experimental design*

We executed two types of evaluations, as follows:

a) Quantitative analysis (Sec. 5.4.1):

(i) *Correlation plots:* We plot the correlation between our proposed assessment of VS (κ) approach and each standard metric. This yields one scatterplot for each of the four standard metrics. In such a plot, each point is a dataset projected by a projection technique, with all dataset-technique combinations considered. The aim of this analysis is to see whether our new metric correlates or not with existing metrics. If so (which we will show it is not the case), then our new metric does not bring any added value. If not (which is the case), then they cannot *both* gauge visual separation equally well – either our new metric or the standard ones are better for this measurement, but not both of them. We analyze this aspect further via our qualitative analysis described below.

projection	linearity	input	local or global
DM (Coifman and Lafon, 2006)	nonlinear	samples	local
FA (Jolliffe, 1986)	linear	samples	global
FMAP (Faloutsos and Lin, 1995)	nonlinear	samples	global
GPLVM (Lawrence, 2003)	nonlinear	samples	global
F-ICA (Hyvarinen, 1999)	linear	distances	global
IDMAP (Minghim et al., 2006)	nonlinear	distances	local
ISO (Tenenbaum et al., 2000)	nonlinear	distances	local
L-ISO (Chen et al., 2006)	nonlinear	samples	local
LAMP (Joia et al., 2011)	nonlinear	samples	local
LE (Belkin and Niyogi, 2001)	nonlinear	samples	local
LLC (Teh and Roweis, 2002)	nonlinear	samples	local
LLE (Roweis and Saul, 2000)	nonlinear	samples	local
H-LLE (Donoho and Grimes, 2003)	nonlinear	distances	local
M-LLE (Zhang and Wang, 2006)	nonlinear	samples	local
LPP (He and Niyogi, 2003)	linear	samples	global
LSP (Paulovich et al., 2008)	nonlinear	samples	local
LTSA (Zhang and Zha, 2004)	nonlinear	samples	local
L-LTSA (Zhang et al., 2007)	linear	samples	local
MC (Brand, 2002)	nonlinear	samples	local
MDS (Torgerson, 1958)	nonlinear	samples	global
L-MDS (De Silva and Tenenbaum, 2004)	nonlinear	samples	global
N-MDS (Kruskal, 1964)	nonlinear	samples	global
L-MVU (Weinberger et al., 2005)	nonlinear	distances	global
NMF (Lee and Seung, 2000)	linear	distances	global
PBC (Paulovich and Minghim, 2006)	nonlinear	samples	local
PCA (Jolliffe, 1986)	linear	samples	global
I-PCA (Lim et al., 2004)	linear	samples	global
K-PCA-P (Schölkopf et al., 1997)	nonlinear	samples	global
K-PCA-R (Schölkopf et al., 1997)	nonlinear	samples	global
K-PCA-S (Schölkopf et al., 1997)	nonlinear	samples	global
P-PCA (Tipping and Bishop, 1999)	linear	samples	global
S-PCA (Zou et al., 2006)	linear	samples	global
PLSP (Paulovich et al., 2011)	nonlinear	samples	global
G-RP (Dasgupta, 2000)	nonlinear	samples	global
S-RP (Dasgupta, 2000)	nonlinear	samples	global
t-SNE (van der Maaten and Hinton, 2008)	nonlinear	samples	local
SPE (Agrafiotis, 2003)	nonlinear	samples	global
T-SVD (Halko et al., 2009)	linear	samples	global
UMAP (McInnes et al., 2018)	nonlinear	distances	local

Table 20: The 39 projection techniques used in our evaluation. We list the linearity, input type, and whether the technique is local or global.

(ii) *Statistical analysis:* We present the main statistical information for our new metric and the standard metrics (minimum, maximum, mean, standard deviation, median, and mode).

b) Qualitative analysis (Sec. 5.4.2): We qualitatively study a subset of such combinations, aiming to find out which metrics – our new one and/or the standard ones – agrees with the perceived visual separation in the projection scatterplots. For this, we perform four qualitative analyses, as follows.

(i) *Random analysis:* We select 8 datasets randomly from the 18 studied ones. For each dataset, we analyze 3 scatterplots of distinct projections and the respective correlation plots.

(ii) *Ranked analysis:* For each dataset, we rank the projections by each quality metric. For the three best and worst projections in terms of this ranking, we study their visual separation *vs* the computed metric values.

(iii) *Correlation plot and ranked analysis:* We plot the same as in (a,i), highlighting the best and worse cases in (b,ii) in terms of good and poor visual separation, respectively.

(iv) *User study:* We ask 108 participants to rank a total of 2916 projections in terms of visual separation and compute the correlation of their rankings with κ.

To use any quality metric in practice, one needs to interpret its values. In our concrete case, all metrics range between 0 (worst case) and 1 (best case). Assuming that a given metric encodes some quality aspect, it is clear that values very close to 1 will indicate 'good' projections in that respect, whereas values close to 0 will indicate 'poor' projections. To simplify the analysis, we next proceed by binning the $[0, 1]$ range in three bins, as follows. Metric values above a superior boundary sb, *i.e.*, in the range $[sb, 1]$, are considered to indicate good projections. Metric values below an inferior boundary (ib), *i.e.*, in the range $[0, ib]$, are considered to indicate poor projections. Metric values in the range $[ib, sb]$ will indicate projections with average quality. Setting these thresholds, thus, allows us to split the study of projection quality in three categories. For T, C, S, and N, we set $ib = 0.4$ and $sb = 0.8$, following earlier studies on how these metrics capture a projection's quality from the respective four viewpoints (Espadoto et al., 2019a). For κ, which measures our proposed visual separation, we set $ib = 0.4$ and $sb = 0.7$ based on our empirical observation of visual separation in projections discussed next in Sec. 5.4.

Practically put, this leads to the following automated workflow for usage of κ in practice: For a given projection, a computational pipeline measures κ following Sec. 5.2.3). If $\kappa > sb$, the projection has good visual separation – so, it can be further shown to its intended users. If $\kappa < ib$, the projection has poor visual separation, so it should not be offered for visual exploration to the users. If $\kappa \in [ib, sb]$, we cannot automatically determine if the projection is 'fit for visual consumption' from the perspective of visual separation, so other metrics or factors should be considered in its assessment. This workflow can be used *e.g.* by a system that computes many projections of a given dataset, *e.g.*, using several algorithms or hyperparameter grid-search, and uses κ to find the best one to serve to its users.

5.4 RESULTS

We next detail the results of our experiments and our observations in terms of how κ surpasses the standard metrics for VS assessment in projections.

5.4.1 *Quantitative analysis*

5.4.1.1 *Correlation plots*

As outlined in Sec. 5.3.4, we have $18 \times 39 = 702$ dataset-projection combinations, each assessed by five metrics (T, C, S, N, κ). Analyzing this information in table form is not very insightful. As such, we aggregate it in terms of four correlation plots. Each plot compares the values of one of the standard metrics with κ (Fig. 5.3a). In each plot, we set κ on the x-axis and the standard metric on the y-axis. The plotted points are the 702 dataset-projection combinations. Each red line represents the trend of same-dataset points – there are thus 18 such lines. We see that, for all the four correlation plots in Fig. 5.3a, blue points are concentrated in the middle-right regions of the plots, indicating that all quality metrics score mostly average or high values. More interestingly, we do not see any positive (or negative) correlation between κ and the standard metrics. Also, for T, C, and S, there are more horizontal red lines than increasing or decreasing trend lines, while, for N, we see more increasing trend lines. This suggests no clear correlation (strictly positive or negative) between κ and the standard metrics. For example, if there was a strong pattern of increasing lines for all datasets in one of the four plots, then κ and the standard metric for that plot would agree, *i.e.*, they would gauge visual

separation similarly. Conversely, if there was a strong pattern of decreasing lines, then κ and the standard metric for that plot would still capture the same information – high values for one metric would tell the same as low values for the other metric. In these cases, κ would not bring clear added value atop of the respective standard metric. However, our plots do not show this. Since κ and the standard metrics do not show a clear correlation, two situations can happen:

- if the considered projections all have *similar* VS, then all metrics are equally poor predictors, since their values vary a lot while the actual VS is not;
- if the considered projections have *different* VS, then either κ is correlated with it and the standard metrics are not (thus, κ measures VS better than the standard metrics), or the standard metrics are correlated with it and κ is not (thus, the standard metrics measure VS better than κ).

Section 5.4.2 studies the above hypotheses in further detail by considering the actual perceived VS in the projections.

Fig. 5.3 (b) shows the binning of the metric ranges using the *ib* and *sb* thresholds introduced in Sec. 5.3.4). The nine cells are colored to indicate the fraction of the total amount of projection-dataset combinations that fall within each range. T, S, and N present medium-right regions with higher percentages of points (densely populated), while C presents right-medium regions with higher percentages of points. This shows, in short, that interpreting κ is easier than the standard metrics because it has a higher variance. The fact that the standard metrics have a low variance – thus small changes in their values – means that it is harder to use their values in practice to determine the quality of a projection. This is also visible in the selected examples in Table 18. More interestingly, earlier work has observed that projections with very different visual separation yield standard metrics of quite similar values (Espadoto et al., 2019a; Castelein et al., 2023). We show next in Sec. 5.4.2 that κ's variance is connected to the perceived visual separation in the projections.

5.4.1.2 *Statistical analysis*

Table 21 refines these insights on the variance of the compared metrics. We show here the minimum, maximum, mean, standard deviation (std), median, and mode values for all metrics computed for the 702 dataset-projection combinations. Minimum and

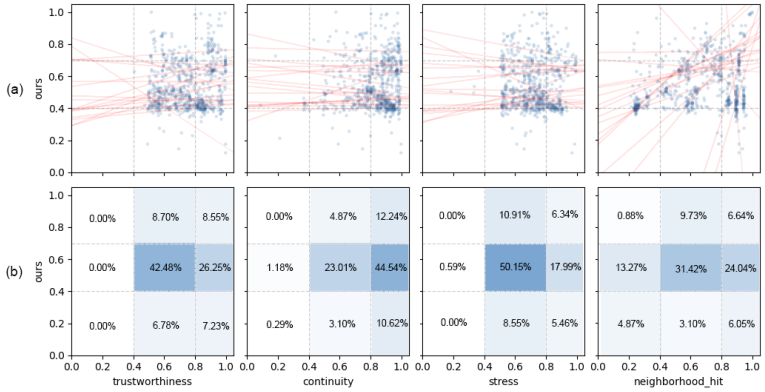


Figure 5.3: Correlation plots of κ (ours) and the standard metrics (trustworthiness T, continuity C, stress S, and neighborhood hit N). Stress values are set as $1 - S$ for easy interpretation. In the plots in (a), each blue point is one of the 702 dataset-projection algorithm combinations. Red lines trend all points for the *same* dataset. In (b), we show the fraction of dataset-technique combinations that fall within the nine categories created by the *ib* and *sb* thresholds.

maximum values are similar for all metrics, except for T, which shows a minimum value of 0.4. Mean and median values are higher than 0.72 for T, C, and S. For N, mean and median values are higher than 0.65 with the highest standard deviation of all considered metrics. κ presents values between 0.49 and 0.55 for mean and median. Mode values are higher than 0.8 for T, C, S, and N, while around of 0.7 for κ. Summarizing, T, C, S, and N mostly assign high values for projections, while κ suggests a wider range of values. This supports our earlier observation that κ may be easier to interpret than the standard metrics. Interestingly, mean, median, and mode are in the same range of the most densely populated regions highlighted in the scatterplots of Fig. 5.3.

5.4.2 *Qualitative analysis*

Our quantitative analysis showed that κ is not correlated with the standard metrics (Sec. 5.4.1). We further study if κ better reflects visual separation by several qualitative analyses.

metric	minimum	maximum	mean	std	median	mode
κ	0.120832	1.000000	0.541826	0.150	0.499891	0.696969
T	0.407621	0.998752	0.753015	0.145	0.762368	0.820627
C	0.087105	0.999063	0.833071	0.143	0.876587	0.940296
S	0.185023	1.000000	0.723541	0.128	0.729920	0.817838
N	0.203571	1.000000	0.653787	0.237	0.686142	0.914286

Table 21: Minimum, maximum, mean, standard deviation (std), median and mode values for each metric. Values are calculated over all datasets and projection techniques.

5.4.2.1 *Random analysis*

Since it is not practical to study all 702 projection-dataset scatterplots, we first randomly select several such scatterplots to show the diversity of VS among different projection techniques (Fig. 5.4). For this, we first randomly choose eight of the 18 datasets. For each standard metric T, C, S, and N, we next randomly select three different projection techniques per dataset, yielding a total of 24 projection scatterplots considered for each standard metric. We next show the correlation plot between κ and each metric. In this correlation plot, the points associated to the three selected projection techniques are highlighted (blue). In each column of Fig. 5.4, projections are sorted left-to-right by decreasing κ.

Several things are visible in this figure, as follows. We first notice that dense correlation plots, *i.e.*, datasets for which the projections have small ranges of both κ and the compared standard metric, presented projections with *poor* VS – see, for example, the projections for the *imdb*, *secom*, *seismic*, and *svhn* datasets. The selected projections for these datasets have values of T, C, S, and N higher than 0.8. For these projections, κ ranges from 0.3 to 0.5. Thus, here, the low κ agrees with the perceived poor visual separation, while the standard metrics do not. In the leftmost columns for each metric, we notice *average* visual separation – see *e.g.* the datasets *bank*, *cnae9*, and *hatespeech*. For these cases, κ values exceed 0.6. The standard metrics for these cases range from very low (0.4) to very high (0.9). This is a second indication that κ reflects perceived visual separation better than the standard metrics. Finally, we can not see any projection with *average* or *good* visual separation in the third (last but one) row of each metric. These are the scatterplots with the lowest κ among the selected ones. This also shows a good agreement between κ and the perceived visual separation.

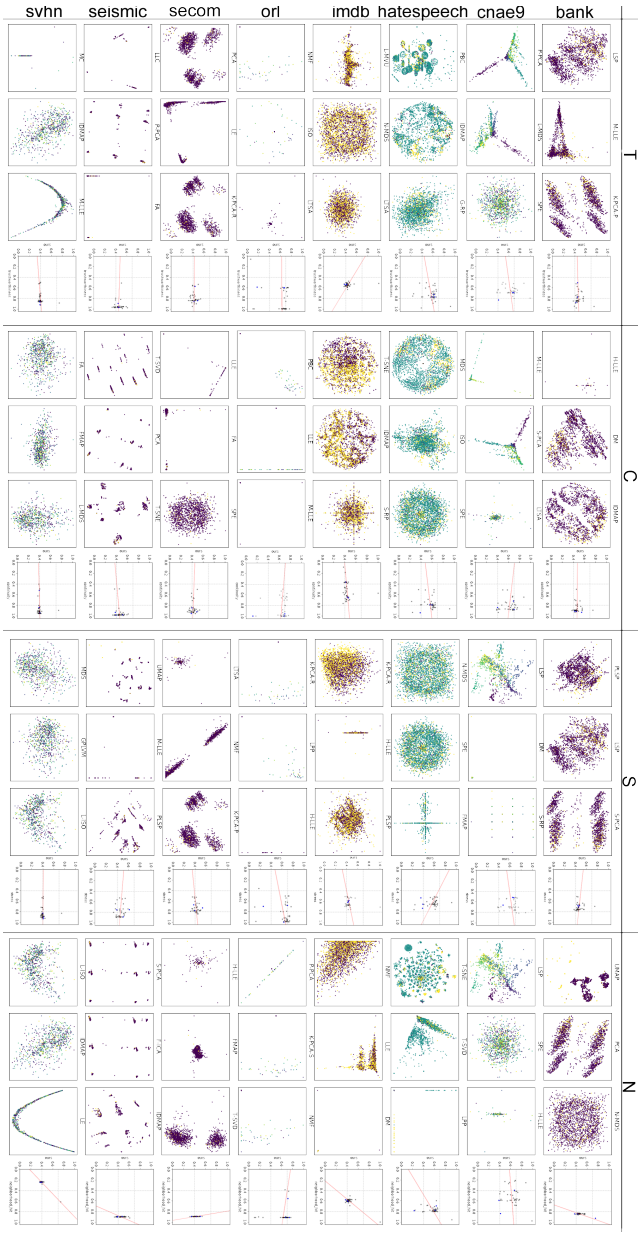


Figure 5.4: Randomly chosen projections. Columns contain eight randomly selected datasets. For each of the T, C, S, and N metrics and for each dataset, we show three randomly chosen projections sorted left-to-right on decreasing κ. The fourth row (for each of the four metrics) shows the correlation plot between κ and the metric. Blue points in this plot indicate the three selected projections.

5.4.2.2 *Ranked analysis*

Fig. 5.5 shows projections ranked by each metric for all 18 studied datasets. In each row (metric) per dataset, we show the best three projections (three left columns in each dataset, surrounded in green) and the worst three projections (three right columns in each dataset, surrounded in red). Within each group of three such projections, the projections are sorted left-to-right on decreasing values of the respective metric. A larger version of Fig. 5.5 showing more details is given in the appendix ad the end of this chapter.

An immediate observation is that projections having the highest (respectively lowest) κ values are also the best, respectively worst, in terms of perceived visual separation. We see many projections having similar VS that are ranked either best or worst by the standard metrics, see *e.g.* the *bank*, *cifar10*, *hiva*, and *imdb* datasets. So, standard metrics are not good predictors of VS. Projections with *average* VS are ranked as worst by at least one of the standard metrics – see *e.g.* the *coil20*, *fashion_mnist*, *fmd*, and *har* datasets. An interesting point concerns N: When both N and κ agree in the (first) best projection, the second-best N value actually has poor VS – see *e.g.* the *cnae9*, *coil20*, *fashion_mnist*, *fmd*, *hatespeech*, and *imdb* datasets. This matches the fact that N shows more increasing trend lines for some datasets in the correlation plots (Sec. 5.4.2) compared to the other standard metrics. Hence, N is also not a good predictor of VS. Also, we see that one of the best three projections in terms of κ is seen as the worst projection by the standard metrics for the *cnae9*, *coil20*, *fmd*, *har*, *hiva*, and *sentiment* datasets. All in all, we consistently see that κ has high values for high perceived VS and low values for poor perceived VS, while the standard metrics do not correlate with VS.

5.4.2.3 *Correlation plot and ranked analysis*

Fig. 5.3 showed that there is no correlation of κ with the standard quality metrics (Sec. 5.4.1). However, this figure did not show whether there is a correlation between κ and the perceived visual separation. To do this, we select, from the best three ranked projections by κ in Fig. 5.5, those with convincingly good visual separation as perceived by ourselves. These are UMAP (*bank, cnae9, coil20, fashion_mnist, fmd, har, hatespeech, imdb, seismic, sentiment, spambase*); t-SNE (*cnae9, coil20, fashion_mnist*); Projection by Clustering (PBC) (Paulovich and Minghim, 2006) (*coil20, fashion_mnist*); and Interactive Document Maps (IDMAP) (Minghim

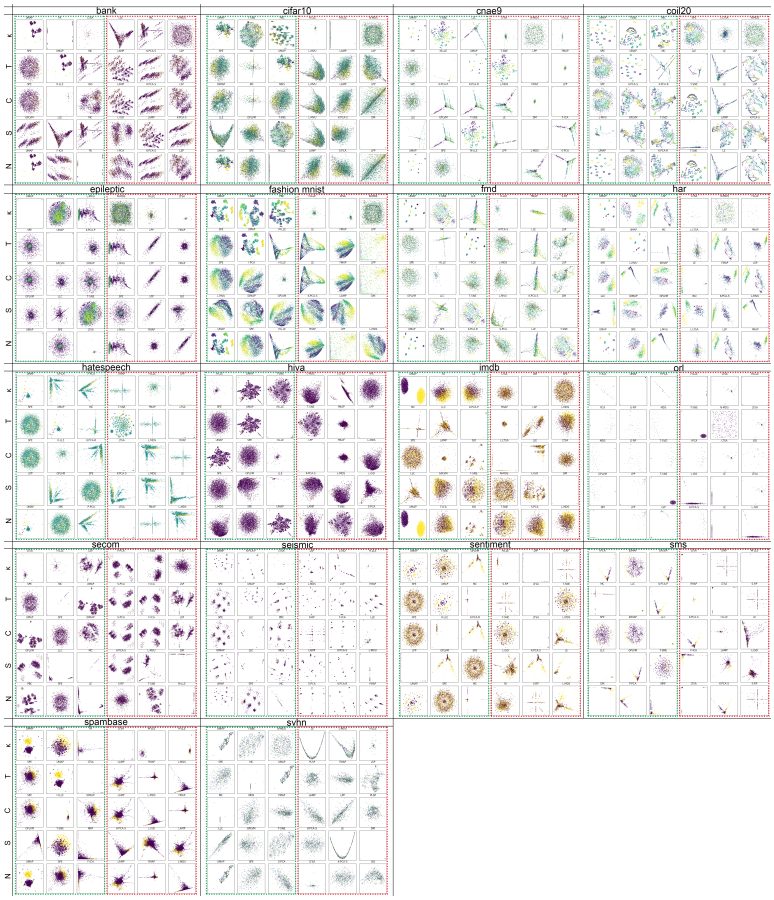


Figure 5.5: Projections ranked by each metric (κ, T, C, S, and N) for all 18 studied datasets. In each row per dataset, the first three columns show the best three (dashed green) projections; the last three columns (dashed red) show the worst three projections according to each metric. (Figure in better quality is presented in Sec. 5.7.)

et al., 2006) (*sms*). Note that, for some datasets, we did not find any projection with a convincingly good visual separation. Separately, we take all the worst-three-ranked projections by κ in Fig. 5.5 which we visually confirm that have a very poor visual separation.

Figure 5.6 shows the correlation plots between κ and the standard metrics – same as Fig. 5.3, but with the projections selected as best, respectively worst, marked in green, respectively red. We see that the green and red points are far apart from each other along the vertical (κ) axis. The green points clearly at the top, above $\kappa = 0.7$. The red points are nearly all below $\kappa = 0.4$, with and all below $\kappa = 0.5$. Hence, κ correlates very well with our perception of visual separation. However, we see that both green and red points spread quite uniformly along the entire range of the co-plotted standard metric (horizontal axis). For example, there are many red points with $\kappa < 0.4$ which have standard metric values above 0.8 and even close to 1; and there are also many green points with standard metric values below 0.7. All in all, this shows that the standard metrics do not correlate in any significant way with the perceived visual separation.

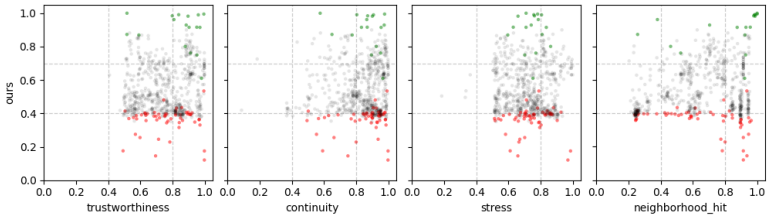


Figure 5.6: Correlation plots of κ with the standard metrics with points denoting projections with good perceived VS in green and poor perceived VS in red, respectively. We see that perceived VS correlates very well with κ but not with any of the standard metrics.

5.4.3 *User evaluation*

We further check the correlation of κ with perceived visual separation by a user evaluation[2].

As explained before (Sec. 5.3), we consider 18 (labeled) datasets and 39 projection techniques, resulting in a total of 702 possible combinations. For each dataset, we use a minimum of 34 of

2 We acknowledge the help of Carlijne Govers, Utrecht University, in the organization and execution of the user evaluation.

the total of 39 projection techniques. Some combinations were excluded as they yielded extremely poor results, meaning the assessment of the VS of the respective projections was further not interesting. All in all, we have a total of 678 projection scatterplots which we next aim to assess. For each of these scatterplots, we compute κ as described in Sec. 5.2.3.

5.4.3.1 *Data preprocessing*

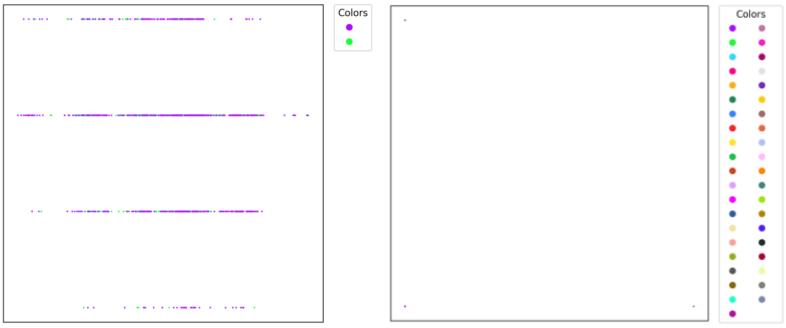
Since the key idea of this study is to measure the visual separation of colored labeled groups from other colored labeled groups in a projection scatterplot, the choice of the color palette is very important. We used the *alphabet* palette derived from *Polychrome 36* (Coombes et al., 2019), a categorical palette with 26 high-contrasting colors. Since we have a maximum of 41 labels in our considered 18 datasets, we manually picked the remaining 15 colors from the *Dark 24*, *oldsky.colors*, and *sky.colors* palettes (Coombes et al., 2019), so that all 41 colors differ as much as possible from each other. Figure 5.7 shows our color palette. Next, in preparation for the user study, we render each scatterplot to a high resolution image (300 dpi) with round dot markers (for the scatterplot points) of about 0.08% of the image size. The color palette is drawn to the right of the scatterplot.

label	hex code	color	20	#B10DA1
0	#AA0DFE		21	#C075A6
1	#16FF32		22	#FC1CBF
2	#2ED9FF		23	#B00068
3	#FA0087		24	#E2E2E2
4	#FEAF16		25	#782AB6
5	#1C8356		26	#FFCB00
6	#3283FE		27	#9B6C6A
7	#F6222E		28	#EB663B
8	#FBE426		29	#B1BEFA
9	#1CBE4F		30	#FEBDFF
10	#C4451C		31	#F98500
11	#DEA0FD		32	#3D8484
12	#FE00FA		33	#9FE600
13	#325A9B		34	#B68100
14	#F7E1A0		35	#511CFB
15	#F8A19F		36	#222A2A
16	#90AD1C		37	#AF0038
17	#565656		38	#E5FFA3
18	#85660D		39	#7F7F7F
19	#1CFFCE		40	#778AAE

Figure 5.7: Chosen 41-colors palette for the user study.

Secondly, we manually label scatterplots considered as *outliers*. We define outliers as plots in which there is significant, and confusing, overplotting of points having different labels. When this takes place, assessing the *data* that is actually plotted is hard; as such, assessing whether the plot has a good (or poor) VS is also confusing.

Figure 5.8 presents two examples of outlier scatterplots. Image (a) shows a scatterplot having only two labels (colors), with points overlapping along four horizontal line patterns. A closer



(a) (b)

Figure 5.8: Examples of outlier scatterplots for (a) *bank-S-RP* and (b)*orl-M-LLE* dataset-projection combinations. In (a), points having two possible labels (green and orange) overlap along four lines. In (b), Points having 41 labels in total overlap in only three points located close to the scatterplot's corners.

inspection of the image shows that the green points can be easily missed since the purple ones dominate the horizontal line patterns. As such, and due to the clear separation in four lines, a user can inadvertently consider this plot as having a high VS (due to the four well-separated lines), whereas it actually has a low VS (due to the intermixing green-purple points along each line). Image (b) shows a different outlier case. Here, the scatterplot has 41 labels (as visible in the right legend). However, the projection yields an extreme case where only three distinct points are visible. Clearly, there is a very high amount of overplotting; and, due to the label count, there is clearly poor VS. Yet, users who *e.g.* do not consider the label count shown by the legend could inadvertently consider that this plot has a very high VS due to the presence of the three different dots visible in the image. We further use the 'outlier' labels we assign to the scatterplots when interpreting the users' inputs (Sec. 5.4.3.4).

Finally, we bin the computed κ value – which ranges from 0.120833 to 1.0 over all 678 scatterplots – in 9 bins, each of size 0.1. Figure 5.9 shows the κ distribution after binning. Further, instead of an actual κ value, we use the index of the bin this value falls within. We call this next the 'round κ'. This further simplifies analyzing the correlation of computed values (round κ) with values provided by users in the evaluation (which are also integral values to simplify the users' task).

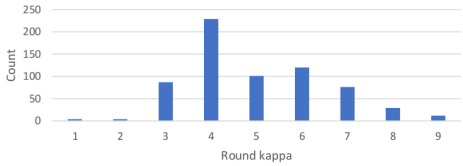


Figure 5.9: Round κ values computed from actual ones via binning.

5.4.3.2 *Study setup*

To evaluate how users perceive VS in projection scatterplots, we conducted an online study using Qualtrics. The study is a walk-through scenario containing five phases, as follows.

1. *Introduction:* We shortly explain the overall context of the study, that is, assessing the visual separation in scatterplots used for 'data science' purposes, and asks for the informed consent of the participants before proceeding.

2. *Explanation* We explain what VS is by means of several examples including plots which we consider as (clearly) having high, respectively low VS. In this explanation, we refer to VS values using a Likert scale with 5 values ("extremely bad", "bad", "neutral", "good", and "extremely good"). However, we give no definition of what 'good' or 'bad' means, apart from the aforementioned scatterplot examples.

3. *Control:* In this phase, users are asked to actually score themselves four different scatterplots using the above-mentioned Likert scale. After they score, we provide to them the correct answer (given our interpretation of VS) and explain why this is the correct one. The examples include situations containing multiple labels as well as overplotting, so users become aware of such effects. The purpose of this phase is to ensure a good (and uniform across participants) understanding of what we mean by VS. In turn, this should reduce potential experience-induced biases across participants during the evaluation.

4. *Sample selection:* Clearly, we cannot ask participants to evaluate all our 678 scatterplots. Rather, we randomly sample this set based on the distribution in Fig. 5.9, with a total of 27 scatterplots to evaluate, 3 for each round κ value. The sampling is done in such a way so that scatterplots belong-

ing to the same round κ values are uniformly (and randomly) spread over the participants – that is, no particular scatterplot for that round κ value gets being seen by more participants than other plots in the same group. All in all, this sampling ensures that (a) all users see, and must assess, scatterplots having all the round κ values (from very bad to very good); (b) users get different scatterplots to score; and (c) we score relatively more scatterplots for the more frequent κ values, due to the aforementioned uniform sampling per round κ value.

5. *Scoring:* Users receive each their set of 27 plots to score, which they do using the aforementioned Likert scale. The plots are offered in random order with respect to round κ to avoid possible biases. Each plot is presented separately in the survey page to avoid users to compare multiple plots when providing their answers. Users can zoom in-and-out the shown images in case they want to examine in more detail some dense point agglomerations in a scatterplot. To simplify execution (and evaluation of the results), we do not allow going back to change previous responses. A progress bar indicates how many of the 27 plots have been evaluated. Finally, participants were allowed to terminate the test early if so desired. Figure 5.10 shows an example of the survey form.

6. *Demographics:* The study is concluded by asking various questions on the users' demographics (age, physical conditions, experience with scatterplots).

5.4.3.3 *Participants*

We distributed the survey online via several media channels to a wide variety of potential participants. No pre-selection was done as we consider that the concept of visual separation in a scatterplot should be easy enough to grasp for a wide population. A total of 108 persons completed the study. Table 22 shows an overview of the participants' self-reported demographics. As visible, the distribution of participants across the measured attributes (gender, age, education, experience, and type of device used for the study) is quite balanced.

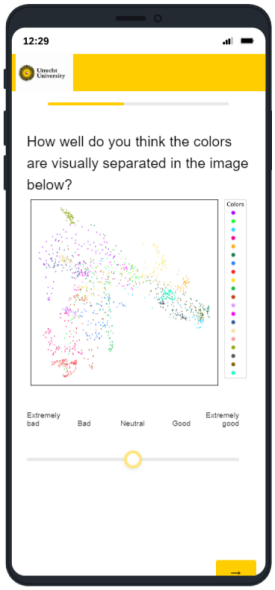


Figure 5.10: Example of a question page in the survey on a mobile device.

attributes	values	count	percentage
gender	female	37	34%
	male	64	59%
	non-binary/third genre	2	2%
	prefer not to say	5	5%
age	18-20	16	15%
	21-30	62	57%
	31-40	9	8%
	41-50	4	4%
	51-60	12	11%
	> 60	15	5%
education	secondary school	37	34%
	intermediate/technical study	2	2%
	bachelor	44	41%
	masters	21	19%
	doctoral	4	4%
experience	< 1	13	12%
	1	7	6%
	2	15	14%
	3	5	5%
	4	3	3%
	≥ 5	5	5%
	no experience	60	55%
device	laptop or desktop	51	47%
	mobile	57	53%
total		108	100%

Table 22: Profile of the 108 study participants by gender, age, education, experience, and device used during the study.

5.4.3.4 *Study results*

As we recruited $P = 108$ participants and each was asked to rank $T = 27$ projection scatterplots, we expected to have $P \cdot T = 2916$ evaluated projection scatterplots. Yet, three of the answers were missing due to participants terminating the test early. We thus have a total of $S = 2913$ evaluated scatterplots which we analyze next.

Figure 5.11 shows the number of different answers for each round κ. For round $\kappa \geq 3$, we notice a trend: Low round κ values ($3 \geq$ round $\kappa \geq 4$) correlate with a high number of "extremely bad" or "bad" user assessments. Similarly, high round *kappa* values ($\kappa \geq 8$) correlate with a high number of "good" or "extremely good" user assessments. This shows a positive correlation between our measured *kappa* and the users' assessments. An exception for this trend is for round $\kappa \leq 2$. Here, we see a more balanced amount of positive ("good", "extremely good") *vs* negative ("bad", "extremely bad") assessments. This seems to indicate a possible doubt of the users.

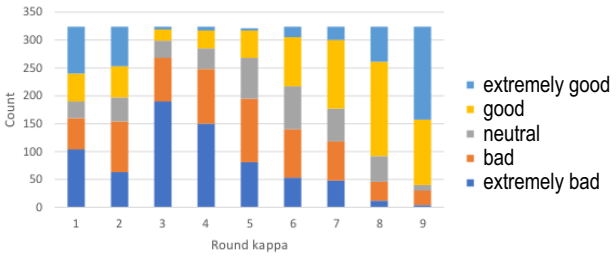


Figure 5.11: Number of different answers for each round κ value.

To understand the situation for low κ values, we plot in Fig. 5.12 the number of outliers, *i.e.*, plots which we ourselves assessed as confusing in terms of VS (see Sec. 5.4.3.1), against their measured round κ values. We see that most of outliers occur for round $\kappa \leq 2$. This explains the exception in the trend observed in Fig. 5.11. In more detail: The two leftmost bars in Fig. 5.11 tell that about half of the participants perceived overlapping samples (which are characteristic to outlier plots as explained earlier) as good VS. The other half would perceive outlier plots as having poor VS. Interestingly, this is fully in line with how one can assess such an outlier plot: Without additional information explaining what data is actually overplotted (*i.e.*, how many samples, and of how many classes, overplot), one cannot say anything about the VS of such a scatterplot. That is, the respective scatterplot can

have a good VS (or not) with a likelihood of 50%, which is in line with the users' assessments expressed by the abovementioned two leftmost bars.

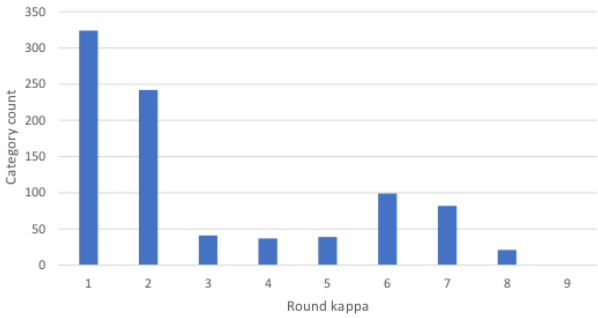


Figure 5.12: Distribution of outliers per round κ. A scatterplot is defined as an outlier concerning overlapping points of distinct colors (labels).

Finally, we analyze the relationship of our measured κ with perceived visual separation (inferred from the user study) through Pearson correlation. Figure 5.13 plots the users' recorded scores *vs* the κ values for the $S = 2913$ evaluated projection scatterplots. To reduce visual clutter, we averaged scores computed over the same scatterplot by multiple users. The Pearson correlation of visual separation scores with κ is 0.55. Moreover, if we leave out the projections with round $\kappa \leq 2$ – that is, scatterplots considered as outliers – we get a correlation score of 0.64. While, as expected, we see some spread of the user scores for the same κ value (and conversely), Fig. 5.13 and the computed correlation factors mentioned above tell us that κ is in good agreement with the perceived visual separation.

5.5 DISCUSSION

We next discuss our main findings.

5.5.1 *Assessing VS by existing metrics*

Our experiments show that the T, C, S, and N projection-quality metrics cannot be (easily) used to predict visual separation of same-label clusters in projections. Our statistical analysis indicates that these metrics have high mean, median, and mode values – they tend to assign values above 0.8 to many projections of many datasets. Hence, even high values of these metrics can

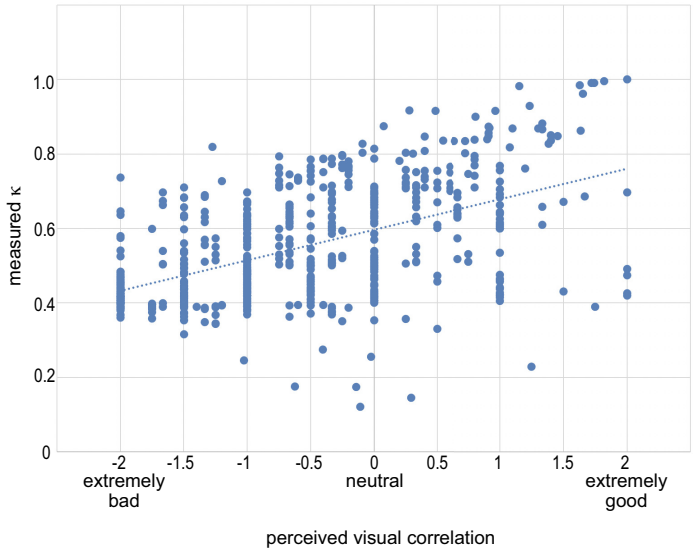


Figure 5.13: Correlation plot of measured κ with perceived visual separation scores (measured by 108 users on 2916 projections).

lead to poor or indistinct VS. Our qualitative analysis shows that projections which have narrow ranges of these metrics have poor or indistinct VS. Also, for a given dataset projected by several methods, the one having the best VS does not necessarily have the highest value of all (or some) of the standard metrics. Conversely, we see cases in which the highest metric value leads to one of the worst-VS projections for a given dataset projected by several methods.

5.5.2 *Our approach to assess VS*

Using κ to gauge OPFSemi's performance in label propagation on projected spaces – was consistently shown to better capture VS of projections than the aforementioned four metrics. Our statistical analysis indicates that κ shows reasonable values for mean and median when considering all compared datasets. A mean and mode around 0.5 and 0.7, respectively, suggests that our approach evaluated a large number of projections with values around 0.7, but also a significant amount of low values to compensate the mean. Our qualitative analysis shows that κ values can better capture the extreme cases: Values of κ roughly above 0.7 all have good perceived VS; values of κ below 0.4 correspond to projections where no discernible VS is present. Values of κ in

the range $[0.4, 0.7]$ indicate projections with an average amount of VS.

In our analysis, UMAP, t-SNE, and PBC consistently score high VS values for all datasets. These were also the best techniques found by the independent study of Espadoto et al. (Espadoto et al., 2019a) which used the average of T, C, S, and N. Importantly, this does *not* mean that the said average can be used to measure visual separation. As shown in Table 18 and Fig. 5.6 projections can have high T, C, S, and N values and *still* poor VS. The said four metrics measure how well a projection captures data patterns (neighborhoods and distances); our κ measures how well a projection is visually separated into different same-label groups. Hence, a good projection should have ideally high values of *all* T, C, S, N, and κ. Our κ is an additional quality factor that complements, but does not replace, existing quality metrics.

5.5.3 *Computational cost to assess VS*

Measuring VS by our method is fast and requires no parameter settings. For example, for the *hiva* dataset – $N = 3076$ samples, $n = 1617$ dimensions, the largest among the evaluated datasets (Sec. 5.3.1) – computing κ took only 0.1216 seconds on a consumer-grade laptop on average for all the considered 39 projections (0.1149 seconds to run OPFSemi; 0.0067 seconds to compute Cohen's Kappa). In contrast, assessing the four standard metrics requires an expensive grid-search procedure to factor out their hyperparameter values and is quite slow to compute (minutes per dataset (Espadoto et al., 2019a)).

5.5.4 *Limitations*

We measure OPFSemi's performance by propagating labels from 50% of the samples in a dataset to the remaining ones. It is not currently clear how our results – and the ability of κ to measure visual separation – depends on this data split. Yet, earlier work has shown that OPFSemi has consistent performance even when using far fewer labels (Benato et al., 2018, 2021c, 2023e). Using this fraction as a parameter is interesting to consider as this would define a *multiscale* visual separation metric. We aim to study this aspect in future work, together with a comparison of our kappa score with Silhouette coefficient based metrics computed for a wide range of clustering methods and clustering hyperparameter settings.

A separate aspect relates to the interpretation of visual separation. A projection having poor visual separation is not necessarily a 'bad' one – the labels may be intrinsically mixed up in the high-dimensional space, in which case it is hard to assume that a projection can separate them well. Conversely, if we know that a projection is poor in terms of its T, C, S, and N quality metrics, the fact that it has (or not) a good visual separation is of little relevance to its actual usefulness for data exploration tasks – in general, one should not further use such a projection since it does not represent well the data structure. However, for datasets where we *know* that labels are well separated in the data *and* we know that the projection has high data-structure-preserving quality, we expect the projection to keep this aspect. In these cases, we can use our approach to gauge the quality of the projection. As such, visual separation should be used *in conjunction* with other information to judge the suitability of a projection for visual exploration tasks – a conclusion drawn from different viewpoints also by earlier authors (Nonato and Aupetit, 2018).

Lastly, while our user study (Sec. 5.4.2) shows that κ correlates with perceived visual separation, extra analysis is needed to show how this depends on projection techniques, datasets, and user experience. We aim to cover this in future work.

5.6 CONCLUSION

We proposed a novel approach to assess the visual separation quality of 2D projections. Our approach is based on assessing the performance of a graph-based semi-supervised classifier in propagating labels in the projection (2D) space. If high label propagation performance is achieved, *i.e.*, few wrongly labels are assigned then the projected space is well separated into distinct groups of same-label samples. To evaluate our proposal, we executed both quantitative and qualitative analyses using 18 datasets and 39 projection techniques in line with the benchmark of (Espadoto et al., 2019a). We showed that our proposed approach can better gauge visual separation in projections than common projection-quality metrics. Up to our knowledge, this is the first time that the visual separation quality of 2D projections is assessed through label propagation task for many projection techniques.

We next aim to evaluate the impact of different amounts of labels in the classifier to assess visual separation in projections. Also, we aim to explore the OPFSemi classifier to evaluate projection quality in reducing the high-dimensional space while pre-

serving patterns of the original data, by combining optimum path forests computed in both high and low dimensional spaces.

5.7 APPENDIX

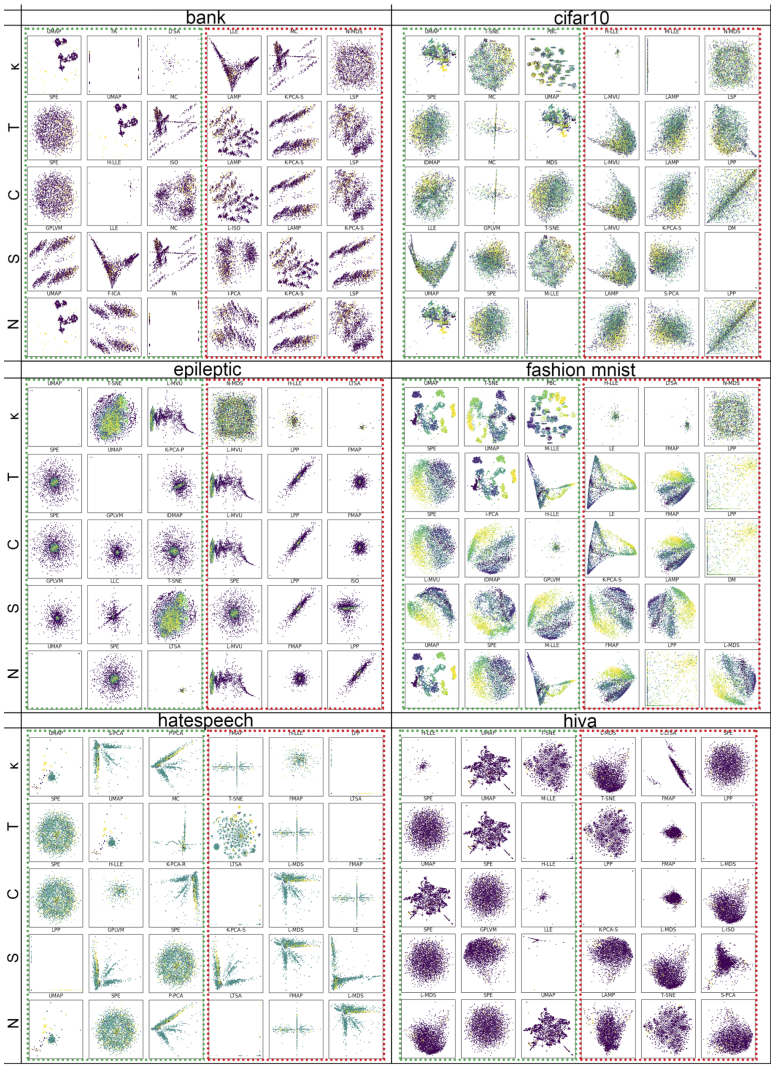


Figure 5.14: Details of Fig. 5.5 for the first 6 of 18 datasets.

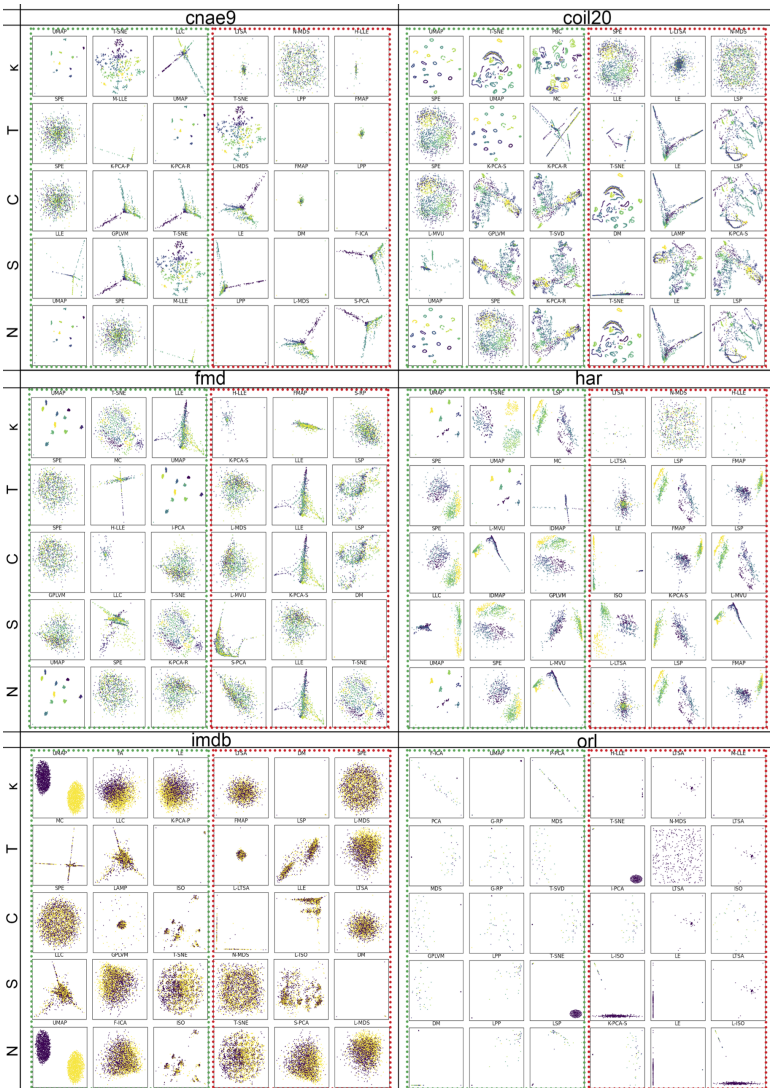


Figure 5.15: Details of Fig. 5.5 for the second 6 of 18 datasets.

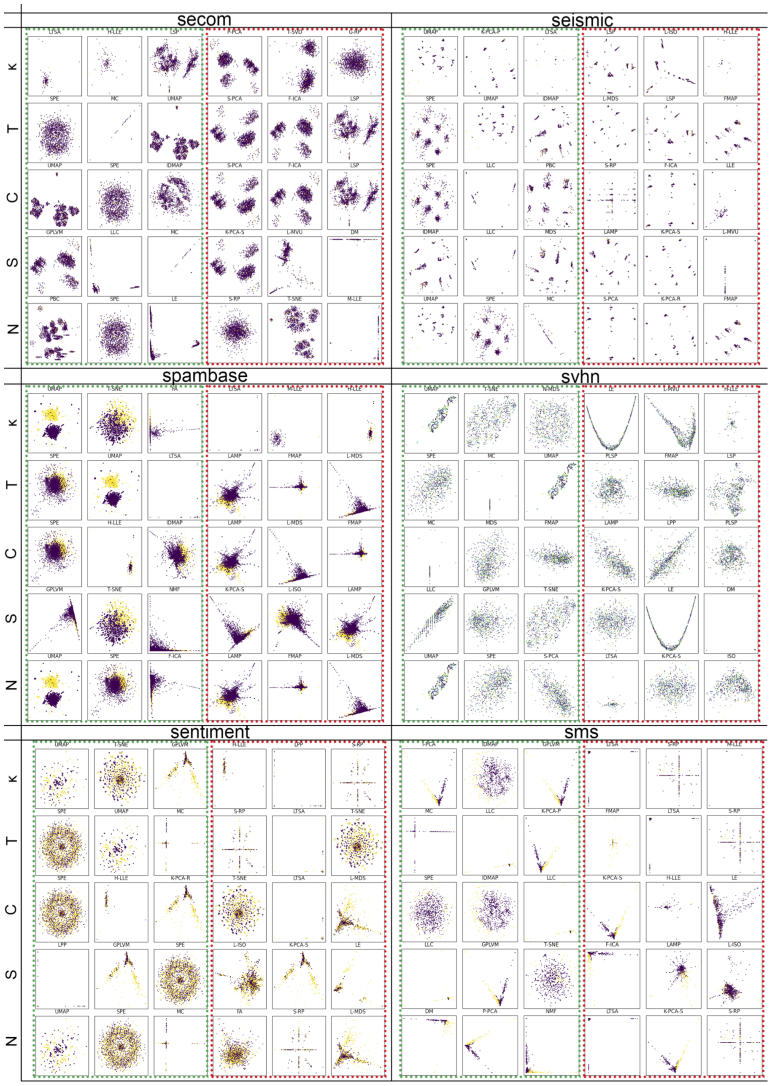


Figure 5.16: Details of Fig. 5.5 for the last 6 of 18 datasets.

6

ACTIVE LEARNING USING DECISION BOUNDARY MAPS

6.1 INTRODUCTION

Our work so far has shown the added value of projection techniques for the generation of pseudolabels for constructing high-performance classification models. Specifically, in Chapter 3, we explored OPFSemi's ability to propagate labels in 2D projection spaces leading to the aforementioned high classification performance. In Chapter 4, we use such pseudo labels to further explore the correlation between data separation, visual separation, and classifier performance, showing that these three aspects are closely connected. In Chapter 5, we further showed that pseudo labels constructed by OPFSemi in a 2D projection space can be used to measure the projection's visual separation as perceived by users[1]

All in all, the above show that projections, and label propagation algorithms in projection spaces, are useful instruments for *automating* both ML and VA tasks. However, this potentially conveys the incorrect message that the *user* has no actual place in such pipelines. We argue that this is by far not the case. As mentioned in Chapter 2, the human ability to manually propagate labels in a 2D projection was first assessed in (Benato et al., 2018). The respective results showed that this ability can *surpass* automatic label propagation in the data space. Further on, this manual approach was combined with automatic label propagation techniques by essentially doing automatic propagation in high-confidence areas while leaving the low-confidence areas to be manually annotated by the user (Benato et al., 2021c).

However, the quality of the pseudo labels produced in Benato et al. (2021c) was constrained by the initial feature space. If the feature learning step produces a feature space with poor visual separation, then both automatic and manual label propagation techniques would fail. We circumvented this problem in Chapter 3 by proposing to improve the feature learning over iterations of pseudo labeling using 2D projections. However, this approach left the user (manual annotation) out of the loop.

1 This chapter is a result of the paper "Decision boundary maps for supporting user-drive pseudo labeling" (Benato et al., 2024).

This chapter aims to complete our quest for assessing the added-value of manual labeling by incorporating more advanced VA techniques in support of this user task. Specifically, we consider using Decision Boundary Maps (DBMs, introduced in Sec. 2.4) and direct-and-inverse projection errors (introduced in Secs. 2.3.4.1 and 2.3.5, respectively) as visual aids to help users decide where, in a projection, they should concentrate their manual labeling efforts. If the user is able to get insights about the classifier's decision, then the user can (arguably) successfully intervene in its training by manually propagating labels as a way to leverage both propagation and classifier performance. We achieve this by adding an active learning looping in the classifier step of our proposed DeepFA pipeline introduced in Chapter 3. Figure 6.1 clarifies the above by comparing the pipeline we have used in our previous DeepFA workflows with the interactive pipeline that uses DBMs and error maps (further described in this chapter).

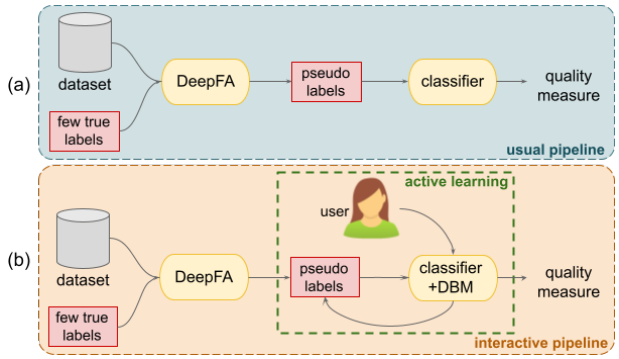


Figure 6.1: (a) Generic pipeling using DeepFA. (b) Interactive pipeline in an active learning scenario using DeepFA and VA tools.

6.2 RELATED WORK

6.2.1 *Active learning*

Since, as mentioned in the previous section, our approach is related to Active Learning (AL), we next briefly describe the setting of AL approaches. Classical AL pipelines work as follows: An algorithm selects a set of samples based on specified criteria and passes them to a user for labeling or inspection. The user-provided information is given to the learner so the learner can improve itself by it (Settles, 2009). The process iterates until some

pre-established stopping criteria are reached, *e.g.*, a desired target classification performance or a maximal user effort being spent.

Modern classifiers use predominantly deep learning techniques such as deep CNNs (Wang et al., 2017; Yu et al., 2015), deep restricted Boltzmann machines (Zhou et al., 2013), Bayesian CNNs (Gal et al., 2017) with dropout as stochastic regularization, and also DBNs (Zou et al., 2015; Liu et al., 2017) for the AL task. Apart from that, some studies explored the learning of feature representation and classifiers together, such as incremental CNN learning (Wang et al., 2017) and incremental dictionary learning (Liu et al., 2017) without many layers. Given our DeepFA framework presented in Chapter 3, which targets labeling image datasets based on CNNs, we concern ourselves next with studies that consider AL and deep CNNs.

There are three main problems concerning deep AL strategies (Ren et al., 2021): (1) DL models still require many supervised samples for training the deep model in the first AL iteration; (2) using the softmax layer's output from the DL model is unreliable for computing the confidence for sample selection (Wang and Shang, 2014); (3) fine-tuning DL models during an AL looping may be difficult.

Solutions to alleviate (1) have considered user interaction using projected spaces. A recent study (Luus et al., 2019) aimed to *simulate* user labeling in AL looping. For this, they used an improved semi-supervised extension of the t-SNE (van der Maaten and Hinton, 2008) projection, in which t-SNE plays the role of user labeling. A deep classifier is periodically trained with all available labeled samples and the softmax prediction vector decides which samples to label next. However, this study did not actually involve the user in the looping.

Iwata *et al.* (Iwata et al., 2013) proposed an AL framework for interactive visualization which selects objects for the user to relocate in order to obtain a desired visualization. However, the main goal of this work is to obtain better visualizations using AL and not provide labeled datasets with high classification performance. Bernard *et al.* (Bernard et al., 2018) compared the performance of AL labeling and visual-interactive labeling using projections. Their findings suggest that visual-interactive labeling can *outperform* AL if the dimensionality reduction step separates well the class distributions. Our work in Chapter 4 outlines similar findings by exposing the correlation between DS, VS, and CP. Later, in another study, Bernard *et al.* (Bernard et al., 2018) presented a systematic quantitative analysis of different user strategies (called computational building blocks) for selecting label-

ing samples via visual-interactive learning interfaces. While such studies compared AL and visual-interactive labeling, they did not *combine* these techniques, as we will be doing. Finally, all above-mentioned studies only used relatively simple datasets. Real-world complex datasets provide more complex feature spaces which are more challenging to handle via AL (with or without VA-based techniques). In our work next, we consider both a relatively simple dataset, and a complex, real-world, one.

6.3 VISUAL ANALYTICS FOR ACTIVE LEARNING AND PSEUDO LABELING

We next describe in detail our proposed pipeline that combines VA techniques and manual labeling in an active learning looping (see also Figure 6.2).

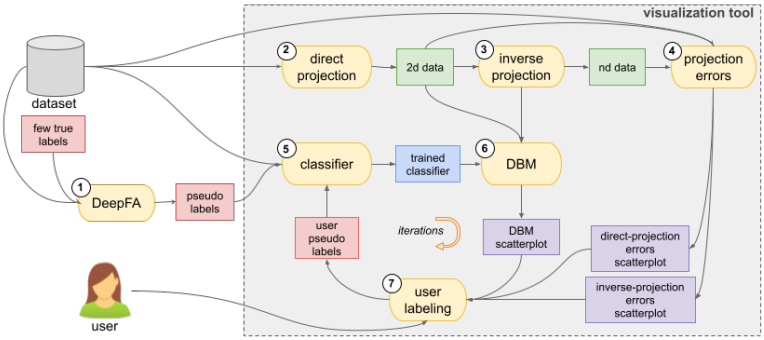


Figure 6.2: Pipeline of the proposed interactive approach.

1) DeepFA: We start with a dataset with only a few true labels available. Since we would like to have a fully labeled dataset in the end, we use DeepFA to generate pseudo labels. This will minimize the user effort in the labeling process. When executing DeepFA, we consider only 1% of supervised samples. Produced pseudo labels are used to train the classifier in step 5.

2) Direct projection: To let users label data in a meaningful and intuitive way, we need a visual representation for it. We achieve this by using a projection technique that maps the data to a 2D scatterplot. We evaluated several different techniques for this task: autoencoders (Hinton and Salakhutdinov, 2006), PCA (Jolliffe, 1986), t-SNE (van der Maaten and Hinton, 2008),

SSNP (Espadoto. et al., 2021), and UMAP (McInnes et al., 2018).

3) Inverse projection: To compute the DBM of a given classifier, we need an inverse projection able to map 2D points to the data space, as outlined in Sec. 2.4. For this, we evaluated the use of the autoencoders (Hinton and Salakhutdinov, 2006), SSNP (Espadoto. et al., 2021), and NNinv (Espadoto et al., 2023) techniques.

4) Visualization errors: We use the data points, their projections, and the computed inverse projection function to evaluate errors of both the direct and the inverse projection functions (see Secs. 6.3.1 and 6.3.2).

5) Classifier training: We train a classifier using 70% of samples (including the pseudolabels assigned by DeepFA) and keeping 30% for calculating classification performance metrics. Any classifier can be used here as our pipeline regards it as a black box. In our concrete experiments, we will use a deep neural network for this step (see Sec. 6.4).

6) DBM computation: We depict the trained classifier using the DBM techniques discussed earlier in Sec. 2.4.

7) Manual labeling: The user can examine the projection scatterplot (of the input dataset), the direct and inverse projection errors (computed at step 4), and the DBM of the trained classifier (computed at step 6) to decide where to manually label data samples (see Sec. 6.3.3). The newly-generated pseudo labels are used to retrain the classifier – that is, the pipeline re-starts at step 5. Steps 5-7 thus constitute the active learning loop. The looping ends when the classifier has achieved a desired target performance or when the user decides that enough manual labeling effort has been invested in the process.

In the next sections, we explain the technical details involved in the operations contained in the above steps.

6.3.1 *Direct projection errors*

Showing *local* projection errors can help users decide where they next channel their labeling effort. In more detail: Assume a group of points in the projection is marked as having high errors. Then, the information displayed in that area can be misleading. This includes class labels and decision zones and decision boundaries shown by DBM techniques. Let us refine both these cases. Con-

sider an area in a projection scatterplot showing inferred labels by color coding. Such an area can contain a mix of many colors – see *e.g.* the left image in Fig. 2.5 and related discussion in Sec. 2.4. This can lead the user to believe that the classifier has some odd behavior in the respective area. However, the abovementioned color mix can be an artifact of the projection – the classifier can work perfectly in the respective area. The same is true for DBMs: An area in a DBM can show tortuous decision boundaries or a wealth of small-scale islands – see *e.g.* the right image in Fig. 2.5 and related discussion in Sec. 2.4. Such artifacts can be caused by projection errors rather than representing actual classifier problems.

To alleviate these problems, we can compute projection error metrics such as Trustworthiness T, Continuity C, or Neighborhood Hit N (Sec. 2.3.4.1) in a *local* fashion. That is, rather than computing aggregate values for the entire projection (as given by Eqns. 2.7, 2.8, and 2.10 respectively), we evaluate the considered metric for each projected point. Next, we can color the respective projection by the computed errors.

A second question concerns which of the T, C, and N metrics we would like to use. A simple solution is to allow users to select the metric they want to examine. However, we believe that this would make the approach less practical, as one would have to cycle through multiple metrics *and* mentally combine the insights they each provide. Instead, we decided to compute a single error metric

$$\epsilon = ((1 - T) + (1 - C)/2. \tag{6.1}$$

This metric, ranging in $[0, 1]$, allows a simple interpretation: The closer its value is to 1, the worse the projection is (in the area the metric is computed) in both T and C; conversely, a value of ϵ close to zero implies a good projection (locally speaking) in both T and C. A separate advantage of combining T and C in a single metric is that we do not need to distinguish between the two separate components. Indeed, a projection having poor T *or* C values (locally) is not a good projection for performing manual labeling decisions. Finally, we note that we discarded N in our error assessment, since this metric resides on the assumption that point labels are well separated in the high-dimensional space (see Sec. 2.3.4.1). As most of our labels are actually *propagated* (using DeepFA), this assumption will likely hold for most data points. As such, N is not a good indicator of the projection quality itself.

A further question concerns the *visualization* of the aggregate error metric ϵ. A simple solution would be to color the projected

points by its value. However, this would make it hard to see *regions* of points which have high, respectively low, error values, especially in the presence of potential overplotting (see the related discussion on overplotting in Ch. 5). A slightly better solution would be to extrapolate the error values up to a small, fixed, distance from their respective points, using radial basis functions, as demonstrated earlier in (Martins et al., 2014). However, a problem with this technique is that controlling the respective fixed extrapolation distance is tricky in practice. Too small distances yield large empty areas in the projection far away from actual projected points. Too large distances can yield interpolation errors, *i.e.*, points where the computed error exceeds, or drops below, the error values of close projection points.

To avoid such issues, we propose to interpolate projection errors (from the projected points to all image pixels) using the computed inverse projection function. For each such pixel $\mathbf{y} \in \mathbb{R}^2$, we can compute its corresponding data point $\mathbf{x} = P^{-1}(\mathbf{y}), \mathbf{x} \in \mathbb{R}^n$. Next, for both $\mathbf{x}$ and $\mathbf{y}$, we can find their respective k-nearest neighbors in $\mathbb{R}^2$, respectively $\mathbb{R}^n$. With this information, we can directly evaluate the expressions for T and C given by Eqns. 2.7 and 2.8 respectively.

Figure 6.3 shows a visualization of the combined projection error ϵ computed using the inverse projection technique described above. Error values are encoded into brightness (high ϵ are bright; low ϵ are dark, pixels, respectively). As visible, the projection errors are low close to most of the actual scatterplot points, which is expected, since the projection technique used here (t-SNE) is known to have low errors everywhere on the considered dataset (MNIST). As we go further from the projected points, we see how errors increase. As a caveat of our approach, we should note that it is dependent on the quality of the inverse projection technique being used. However, note that, if a projected point has poor T and C values, this will appear as a high-error point in our visualization, even in the presence of an imperfect inverse projection. At the point itself, using P^{-1} is not needed, since we know its data-point counterpart; hence, at that location, we will simply display $\epsilon = ((1 - T) + (1 - C)/2$, which, as said above, will show a high error. Close to that point, the inverse projection will use, in its computation, the wrongly-projected point and, as such, yield a data point which will be incorrectly placed in data space. As such, the results of T and C will show a high error value.

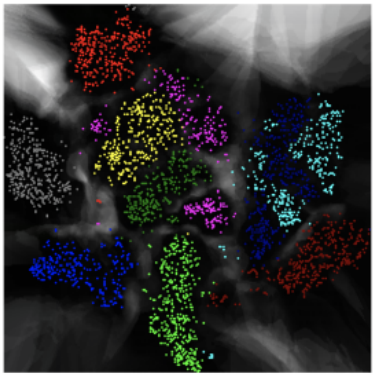


Figure 6.3: Scatterplot image with 2D projected points for the MNIST dataset with points colored by class. Projection errors ϵ are computed using the inverse projection technique at every image pixel. Light regions refer to a high error values; dark regions represent low error values.

6.3.2 *Inverse projection errors*

Besides direct projection errors, also the inverse projection errors can adversely influence the insights users get from the visualization. Indeed, as explained in Sec. 2.4, all DBM methods in existence rely on using inverse projections. As such, if these inverse projections are incorrect (in some area), the DBM will be misleading (in that area).

We quantify *locally* the quality of inverse projections using the *gradient map* technique proposed by (Espadoto et al., 2023). In detail, let $\mathbf{y}$ be a 2D pixel in the considered image, and $\mathbf{y}_r$ and $\mathbf{y}_b$ its right, respectively bottom, neighbors. The gradient map method computes the value

$$G(\mathbf{y}) = \sqrt{\|P^{-1}(\mathbf{y}_r) - P^{-1}(\mathbf{y})\|^2 + |P^{-1}(\mathbf{y}_b) - P^{-1}(\mathbf{y})\|^2}, \quad (6.2)$$

which can be seen as the finite-difference approximation of the norm of the gradient of the function P^{-1} evaluated at $\mathbf{y}$. The interpretation of G is simple: Image points $\mathbf{y}$ with low G values indicate low inverse projection errors. Indeed, small changes in $\mathbf{y}$ correspond to small changes in the data space, which we expect from a well-behaved inverse projection function. Conversely, image points $\mathbf{y}$ with high G values indicate problems (errors) in

the inverse projection function: Small changes in the image correspond to 'jumps' in the data space[2].

Figure 6.4 shows the visual representation of the inverse projection error G for the same dataset and projection as in Fig. 6.3, encoded by luminance. As visible, most image areas have low G values, except some thin 'bands' where G increases a lot. Note also that this image is different from the direct projection errors shown in Fig. 6.3. As such, users should consider both types of errors when assessing if an area in the visualization is suitable to perform manual labeling.

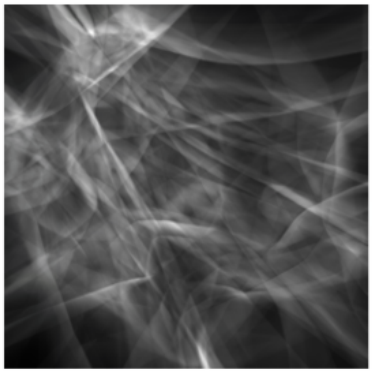


Figure 6.4: Image of inverse projection errors for the same dataset and projection as in Fig. 6.3. Error values are mapped to brightness.

6.3.3 *VA tool for active learning*

We next present the Visual Analytics (VA) tool that we constructed to assist users in interactively labeling samples for improving the training of a given classifier (Fig. 6.2, large gray box). When needed, throughout the explanation of the workflow supported by the tool, we will refer to the steps (1)-(8) of the pipeline in Fig. 6.2.

Pre-labeling: Our VA tool expects to start with a completely labeled dataset. Of course, if ground-truth labels are available for

2 To be more exact, we should say that large values of the *gradient* of G, *i.e.*, discontinuities in the G signal, indicate inverse projection problems. In practice, we have seen that G consists of large low-G areas separated by thin 'bands' of high-G values. As such, for simplicity of exposition, we consider that high G values are a reliable indicator for the mentioned problems.

such a dataset, they can be directly used in this process. When this is not the case, *i.e.*, when only a small fraction of the dataset's points have supervised labels, we use DeepFA to compute pseudolabels for all remaining points (1). In our experiments, we used for this 1% of the supervised samples in the dataset. This pre-labeling process is done only once, before our VA tool is actually started.

After the pre-labeling completes, we user uploads the pre-labeled dataset and the classifier to train and chooses various general settings in a configuration dialog, *e.g.*, techniques to use for the DBM generation. We will explain these settings further on during the presentation of the VA workflow.

Interactive labeling: This is the main operation supported by our VA tool. It starts with generating and displaying a DBM image for the classifier and labeled dataset provided during tool start-up. The DBM, (labeled) samples, projection (and inverse projection) error metrics, and tooltips showing information about specific samples, are all displayed, and offered to user interaction, in a so-called *labeling window*, which we describe next.

Figure 6.5 shows an overview of the labeling window. In the left part of the window, the tool shows the 2D projection of the loaded dataset (according to the projection method selected by the user during tool start-up). Training and test points are shown here in white, respectively black. The DBM for the classifier trained by the labeled set loaded by the tool (that is, at the beginning of the active learning process, before any manual labeling has been executed) is also shown underneath the projection scatterplot. The DBM pixels $\mathbf{y}$ are colored by the class assigned to the corresponding data points $P^{-1}(\mathbf{y})$ computed by the inverse projection technique P^{-1} selected by the user during tool start-up. As a modification to the standard DBM visualization presented in Espadoto et al. (2019b), and in line with further extensions of DBMs Oliveira. et al. (2022); Espadoto et al. (2023), we also consider the *confidence* of the trained model evaluated at each DBM pixel. We allow users to display this confidence; the combined direct-projection error (Eqn. 6.1), the gradient-map showing inverse projection errors (Eqn. 6.2), or a user-selected combination of these measures. Since all these measures share the same scale $[0, 1]$, we combine them by multiplication. The final result is then mapped to image saturation (low saturation indicates a low value; high saturation indicates a high value). Figure 6.5 (left) illustrates this by showing the classifier confidence

mapped to saturation. As visible, DBM areas close to the decision boundaries have a lower classification confidence.

When hovering a pixel $\mathbf{y}$ in the scatterplot, a tooltip image with the data value $P^{-1}(\mathbf{y})$ corresponding to that pixel is shown. This way, users can inspect both actual data points (present in the dataset loaded by the tool) or newly inferred points corresponding to areas in the DBM outside of the projection scatterplot. This is useful for deciding how to manually label data points next. Top right in the interface, we show the accuracy and κ value of the *current* classifier (that is, trained with the actual set of labeled points, including whatever points the user has manually labeled so far). We also show the *evolution* of the classifier accuracy over labeled iterations in a 2D line chart. This allows users to determine whether their manualk labeling actions have increased, or decreased, the classifier accuracy, and take corrective actions (more on this below).

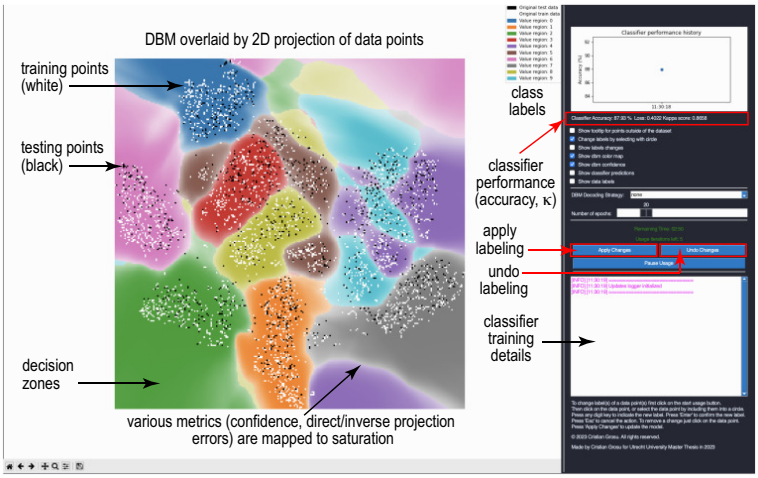
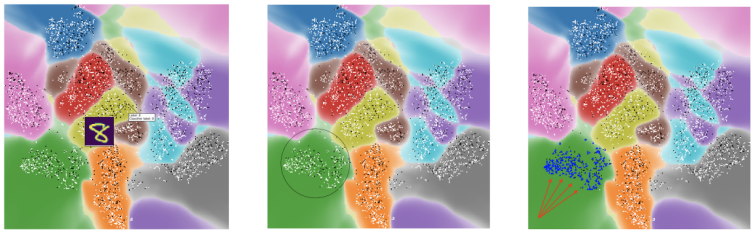


Figure 6.5: Overview of the visualization tool's labeling window.

The user can start his/her first iteration of manual labeling. Using the tooltip, currently-labeled samples in the scatterplot, DBM, confidence, and direct/inverse projection information, the user decides on a set of unlabeled points to which he/she wants to assign one of the existing labels in the dataset. Of course, this is not a deterministic process – if it were, we would not need the user's help but, rather, automate this labeling, using *e.g.* techniques presented earlier in Chapter 3. Rather, our claim is that, by studying all the above-mentioned information (scatterplot, DBM, confidence, errors), the user can spot patterns in the data struc-

ture which lend themselves well to adding manual labels so as to increase the classifier's performance. We will demonstrate this next in Secs. 6.4 and 6.5.

At each labeling iteration, the user can select any subset of points in the projection scatterplot to assign any of the dataset's labels to them. Multiple such sets can be selected in an iteration and assigned the same, or different, labels. Figure 6.6 illustrates this process. When the user is satisfied with the manual labeling, he/she confirms this to the tool by pressing the *apply labels* button. At this point, and at each iteration, the tool stores (i) the classifier accuracy, (ii) the classification κ score, (iii) the labeled samples, and (iv) a screenshot with the users' view of the tool at that moment. This information is used next to support undoing manual labeling changes and also compute performance statistics on the manual labeling, as discussed next.



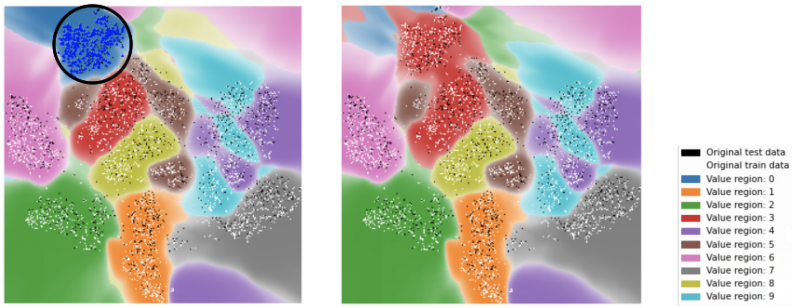
(a) (b) (c)

Figure 6.6: User labeling procedure. (a) The user can see the tooltip image for each point. (b) The user selects a set of points to label by drawing a circle with the mouse. (c) Next, the user can assign a new label to the selected points. In this artificial example, the user has labeled a large set of points located in the green decision zone with the blue label.

Classifier re-training: After an iteration is completed, the classifier is re-trained to use the newly assigned (pseudo) labels – to be more exact, the total set of labeled points, which contains points whose labels the user has not changed manually, and points affected by the manual labeling in the last iteration. The training progress is shown in a text window in the tool so users can spot possible problems. After the re-training completes, the labeling window is updated to show the DBM of the newly-trained classifier. Note that position of points in the projection scatterplot does not change since the user can only change the labels of the data

points, but not add, remove, or otherwise change the values of, these points.

Figure 6.7 shows scatterplots (a) before and (b) after a manual labeling iteration, including the subsequent classifier re-training and DBM recalculation. In this example, for illustration purposes, the user manually selected a large set of points in the blue decision zone (marked by the black circle) and assigned them the label 3 (red). Image (b) shows how the re-trained classifier now has a large red decision zone that includes largely all the points the user has manually labeled as red. However, due to this massive re-labeling, the classifier's performance decreases significantly – accuracy drops from 0.8793 to 0.7927; κ drops from 0.8685 to 0.7689. This is, of course, expected, given that the user has basically forced the disappearance of roughly the entire blue decision zone. In practical use, manual labeling will select significantly fewer samples to label during an iteration.



(a) (b)

Figure 6.7: Example of classifier re-training and DBM re-calculation. (a) Initial state (classifier accuracy: 0.8793, κ: 0.8685). User selects a large set of points in the blue decision zone (marked by the black circle) and decides to manually assign them the label 3 (red). (b) Situation after classifier re-training with the new manually added labels (classifier accuracy: 0.7927; κ: 0.7689).

If the classifier performance decreased (as shown in the tool's interface) as compared to the previous iteration, the user can decide to *undo* the last-performed labeling. The labeling window then changes to show the values (DBM, classifier performance, κ) before this past iteration. The process continues until the user decides to stop it, either because of time constraints or because the desired classifier performance has been reached.

6.3.4 *Implementation details*

We next provide several implementation details concerning the various steps of our proposed workflow.

DeepFA: As explained already, we create pseudo-labels for all the dataset samples to be loaded by our VA tool. For this, we use only 1% of supervised samples and 5 DeepFA iterations. All other technical details concerning DeepFA are the same as the ones we described in Chapter 3.

Direct projection methods: Our tool supports PCA (Jolliffe, 1986), vanilla autoencoders (the encoder part) (Hinton and Salakhutdinov, 2006), t-SNE (van der Maaten and Hinton, 2008), UMAP (McInnes et al., 2018), and SSNP (the encoder part) (Espadoto. et al., 2021). t-SNE and PCA use the Scikit-learn implementation (Pedregosa et al., 2011). UMAP uses the default implementation provided by its authors (McInnes et al., 2018). All parameters are set to their default values, except the perplexity of t-SNE, which we set to 30.

Inverse projection methods: Our tool supports vanilla autoencoders (the decoder part), NNinv (Espadoto et al., 2023), and SSNP (the decoder part).

For NNinv, we use a fully connected neural network with architecture 2-32-64-128-512-α. We use $\alpha = 784$ for MNIST and $\alpha = 5000$ for P.cysts, respectively. Hidden layers use the ReLU activation function, except the last one which uses a sigmoid activation. The first layer uses an L_2 regularization penalty with constant set to 0.0002. Weights are initialized using the HeUniform kernel with bias set to 0.01. We train NNInv for 300 epochs (with early stopping) and mean squared error (MSE) as loss function.

The decoder part of the vanilla autoencoder match the NNInv architecture and use the same activation functions and weight initializer. The encoder architecture is the decoder one, but flipped. We train our autoencoder for 300 epochs with MSE as loss function.

SSNP uses an identical architecture to the vanilla autoencoder with the main difference of adding a data-dependent clustering layer – that is, its output has as many classes as the treated dataset has, *e.g.*, 10 for MNIST. The clustering layer uses the softmax activation function. SSNP uses two loss functions – MSE for data decoding (as the vanilla autoencoder) and sparse categorical cross-entropy loss (for the data clustering). In the

total loss, these two functions contribute with weights of 1, respectively 0.125. For further details, we refer to the original SSNP paper (Espadoto. et al., 2021).

Direct and inverse projection combination: As outlined above, we have a total of five direct projections P (PCA, autoencoders, t-SNE, UMAP, SSNP) and three inverse projections P^{-1} (autoencoders, NNInv, SSNP) to use. This would lead to a total of 15 combinations of (P, P^{-1}). However, we deem some combinations to be less practical and/or useful than others. For instance, it has little sense to use SSNP as P^{-1} with a different P than the one SSNP provides, since SSNP jointly trains for P and P^{-1}. Similarly, when using autoencoders, it makes sense to do that for both P and P^{-1}. Figure 6.8 summarizes all possible combinations of P and P^{-1} that our VA tool supports.

Projection method $\mathcal{P}$	Compatible inverse projection method $\mathcal{P}^{-1}$	Supported in our tool
t-SNE UMAP PCA	NNinv	Yes
Vanilla Autoencoder (encoder part)	Vanilla Autoencoder (decoder part)	Yes
	NNinv	No
SSNP (encoder part)	SSNP (decoder part)	Yes
	NNinv	No

Figure 6.8: Possible combinations of direct and inverse projection methods provided by our VA tool.

DBM: Decision map images are generated as proposed by (Rodrigues et al., 2019) (using NNinv) or by Espadoto. et al. (2021) (using SSNP), at a size of 256 × 256 pixels.

Source code: The proposed VA tool is made publicly available[3][4].

6.4 EVALUATION

We designed an experiment to evaluate the efficiency and effectiveness of our proposed VA tool for manual labeling. In this experiment, we used the VA tool as described above, with two key modifications: (1) a labeling iteration can take a maximum of three minutes; and (2) a maximum of 5 iterations are allowed. The user can apply the manually assigned labels (that is, end the current labeling iteration) at any moment before the mark of three minutes. When this time passes, however, no more manual

3 https://pypi.org/project/decision-boundary-mapper
4 https://github.com/cristi2019255/MasterThesis2023

labeling is allowed. These constraints ensure that we measure (and limit) the manual *effort* users put into labeling. Next, this will allow us to compare the results obtained by different users and, also, reason about the overall efficiency of the proposed procedure.

We next describe the classifier, datasets, and participants involved in our user evaluation.

6.4.1 *Classifier*

The classifier we use in our experiments is a neural network consisting of a flattening layer (with the input size), followed by a dense layer with a softmax activation function (with the number of neurons equal to the number of classes in the considered dataset). The number of epochs for training and re-fitting is set to 20. This is, on purpose, a quite simple classifier architecture. We chose for it so as to offer the possibility for the classifier to generate a significant number of errors, which next can be diminished by the manual labeling proposed by our VA tool.

6.4.2 *Datasets*

Our evaluation is organized as a set of two consecutive experiments. The first one is used as a control mechanism: During this experiment, we select a relatively simple dataset (and classification problem), to calibrate various settings of our tool. Next, we use a significantly more challenging, real-world, dataset to measure the tool's effectiveness and efficiency. The two datasets, and related considerations, are described next.

6.4.2.1 *Toy dataset: MNIST*

We select a small subset of MNIST dataset with 3500 samples and 10 classes to test our entire workflow. As we may have different variables that can influence our results, we intend to explore several possible combinations. For example, the number of training samples can affect the performance of a given classifier – and, thus, indirectly affect the efficiency and/or effectiveness of our proposed VA tool. Indeed, a small labeled training-set likely yields a poor classifier; but, likely too, the VA tool can then quickly help increasing the classifier's performance (because the latter is so poor). Conversely, a large labeled training-set will likely yield a high(er) classifier performance; however, further im-

proving this performance by the VA tool will likely be hard (since it is already quite high).

To test the above, we use 5 subsets of the MNIST dataset containing 20%, 40%, 60%, 80%, and respectively 100% of all samples as input to our VA tool. We next call these subsets $D_{0.2}$, $D_{0.4}$, $D_{0.6}$, $D_{0.8}$, and $D_{1.0}$.

Separately, as explained in Sec. 6.3.4 (Fig. 6.8), our VA tool allows for five combinations of (P, P^{-1}) - namely (t-SNE, NNinv); (UMAP, NNinv); (PCA, NNinv); (autoencoder, autoencoder); and (SSNP, SSNP). For brevity, we next denote by (autoencoder), respectively (SSNP), the combinations (P, P^{-1}) where both P and P^{-1} are provided by an autoencoder's, respectively SSNP's, encoder and decoder parts respectively. We perform our experiments with the five MNIST subsets considering all these five combinations separately. This allows us to select the best combination of (P, P^{-1}) to use next with our tool.

6.4.2.2 *Real-world dataset:* P.cysts

We now use the best combination of (P, P^{-1}), as determined by the toy dataset experiment, to manually label a more challenging dataset. For this, we choose a small subset of the *P.cysts* dataset without impurities with 2,696 samples and 6 classes. This dataset was introduced in detail in Sec. 3.3.1. Given the high dimensionality of *P.cysts* as compared to raw MNIST, we cannot directly feed *P.cysts* to the projection P and expect to obtain good results. Rather, we reduce this dataset to a lower dimensionality ($n = 512$) using a standard autoencoder approach, in line with (Benato et al., 2018). Next, we use this 512-dimensional dataset along the same manual labeling workflow as for the MNIST dataset.

6.4.2.3 *Training, testing, and performance evaluation*

In all our experiments, we split the given input dataset D, that is the five versions of MNIST or the *P.cysts* subset, into D_{train} and D_{test} with a proportion of 80% to 20% respectively.This allows the computation of classification accuracy and κ. It is important to highlight that no true labels are used in this computation. Rather, classification accuracy and κ are computed over pseudo-assigned labels as if they were true labels.

6.4.3 *Participants*

For both experiments in Secs. 6.4.2.1 and 6.4.2.2, two users are assigned to the task of manual labeling, which are referred next as $U1$ and $U2$. $U1$ was closely involved in developing the VA tool, but has no prior knowledge on the real-world dataset *P.cysts* and its classification challenges. Conversely, $U2$ has detailed knowledge on *P.cysts* but was not involved in implementing the VA tool. Both users have a good understanding of the MNIST dataset. This means that, for the first experiment (MNIST dataset), we can assume that $U1$ has some potential advantage over $U2$. For the second experiment (*P.cysts* dataset), we see no clear advantage of any of the users. Apart from the above, both users have relatively similar profiles in terms of age and experience with machine learning and data visualization.

During both experiments, the participants were not able to exchange any information concerning their way of working and/or intermediate results, to avoid cross-learning or bias effects.

6.5 EXPERIMENTAL RESULTS

We next present the results of our two experiments.

6.5.1 *Toy dataset: MNIST*

6.5.1.1 *Defining the baseline*

It is useful to start by defining a *baseline* for our subsequent experiments. For this, we assess the classifier performance using the *all* true labels; and also the performance obtained using the DeepFA-generated pseudo labels, *i.e.*, those given as input to our tool (Fig. 6.2 (1)). Table 23 shows the DeepFA labeling performance, classifier performance using DeepFA pseudo labels, and classifier performance using all true labels, for the five different subsets of MNIST introduced in Sec. 6.4.2.1. We see that the labeling and classifier performance (when using DeepFA pseudo labels) are around 0.77, largely independent of the MNIST subset being used. Separately, we see that, when using true labels, accuracy and κ increase with the sample count.

We can next see as the true-label performance being the *upper bound* that one can achieve for this dataset; and the DeepFA-based performance as the *lower bound*, respectively. That is, manual labeling cannot, likely, exceed the upper bound performance; but,

for it to be useful, it has to exceed the lower bound performance as much as possible.

	metric	$D_{20\%}$	$D_{40\%}$	$D_{60\%}$	$D_{80\%}$	$D_{100\%}$
labeling performance	acc	0.7757	0.7607	0.7723	0.7728	0.7737
classifier performance	acc	0.7507	0.7780	0.7987	0.7893	0.7540
(pseudo labels)	κ	0.7226	0.7531	0.7761	0.7658	0.7266
classifier performance	acc	0.8347	0.8807	0.8960	0.9000	0.8947
(true labels)	κ	0.8161	0.8673	0.8844	0.8888	0.8829
# of samples		700	1400	2100	2800	3500

Table 23: MNIST dataset. For different number of samples, DeepFA labeling performance, classifier performance using pseudo labels, classifier performance using true labels (upper bound), and number of samples are presented.

6.5.1.2 *Comparison among different techniques and users*

Table 24 presents the results of classification accuracy and κ for pseudo labels generated by manual labeling using our VA tool, for both users, all five subsets of MNIST considered, and all five combinations (P, P^{-1}) mentioned earlier in Fig. 6.8, after all five labeling iterations.

First, we notice only small differences between $U1$ and $U2$ for the same dataset D and combination (P, P^{-1}). As such, we will next focus on the trends of performance *vs* dataset size D and combination (P, P^{-1}) only. In this respect, we notice that all projection techniques using only $D_{20\%}$ samples achieve a similar, poor, result. This means that the small number of samples in $D_{20\%}$ (around 700) was not sufficient for the projection techniques to provide a good visual representation for manual labeling. Separately, we see that for not all (P, P^{-1}) combinations show a performance increase with the size of D. For instance, (PCA, NNinv) showed poorer results in $D_{100\%}$ compared to $D_{20\%}$. In contrast, (t-SNE, NNinv) and (UMAP, NNinv) showed a performance increase with the sample count. This suggests that these last two combinations are more suitable to be used to generate visualizations for manual labeling.

We next explore the performance details aggregated in Table 24 in further detail by showing the actual variation of accuracy and κ over all five labeling iterations.

Classifier performance for both users over the five iterations are shown for accuracy (Figs. 6.9 and 6.11) and κ (Figs. 6.10 and 6.12). In all these figures, blue lines represent the user's perfor-

P, P^{-1}	user	$D_{20\%}$		$D_{40\%}$		$D_{60\%}$		$D_{80\%}$		$D_{100\%}$	
		acc	κ	**acc**	κ	**acc**	κ	**acc**	κ	**acc**	κ
autoencoder	U1	0.8167	0.7961	0.8287	0.8094	0.8353	0.8169	0.8307	0.8117	0.8140	0.7932
	U2	0.8180	0.7976	0.7980	0.7754	0.8160	0.7954	0.8240	0.8043	0.8140	0.7932
SSNP	U1	0.8180	0.7976	0.8167	0.7961	0.8280	0.8087	0.8007	0.7784	0.7860	0.7621
	U2	0.7893	0.7657	0.7840	0.7598	0.7900	0.7664	0.7920	0.7687	0.7973	0.7746
t-SNE, NNinv	U1	0.8207	0.8005	0.8487	0.8317	0.8627	0.8473	0.8827	0.8695	0.8867	0.8740
	U2	0.8400	0.8221	0.8513	0.8348	0.8480	0.8310	0.8613	0.8458	0.8600	0.8443
UMAP, NNinv	U1	0.8167	0.7962	0.8727	0.8584	0.8667	0.8517	0.8887	0.8762	0.8633	0.8481
	U2	0.8333	0.8147	0.8460	0.8288	0.8260	0.8065	0.8473	0.8303	0.8487	0.8318
PCA, NNinv	U1	0.8127	0.7917	0.8120	0.7909	0.8260	0.8065	0.8127	0.7917	0.7620	0.7355
	U2	0.7673	0.7413	0.7720	0.7465	0.7867	0.7628	0.7673	0.7413	0.7333	0.7034

Table 24: MNIST dataset. Results of classification accuracy and κ for classifiers trained with pseudo labels generated by the user interactive labeling using the visualization tool after 5 iterations. Different techniques for direct (P) and inverse projections (P^{-1}) are compared for two different users.

mance over iterations; and green, orange, and red curves represent linear, logarithmic, and exponential trends (for comparison purposes). That is, a blue line close to the green line represents a (roughly) linear increase of classifier performance obtained by the respective user over labeling iterations; a blue line close to the orange curve shows that the user was much more successful in the first labeling iterations than in the later ones; finally, a blue line close to the red curve shows the used has difficulties in the first labeling iterations but achieved better in later ones.

Similar trends can be seen for both users. While (tSNE, NNinv) and (UMAP, NNinv) showed a more logarithmic trend – *i.e.*, user labeling was more effective in the first iterations –, (autoencoder), (SSNP), and (PCA, NNinv) showed an exponential trend – that is, the user had more labeling difficulties in the first iterations. This result may be caused by the visualization generated by the (P, P^{-1}) combination used. For the exponential trend case, the user might have done some significant wrong labeling in the first iterations, possibly because of the unintuitive DBMs generated by the (P, P^{-1}) combination used, but managed to correct it in later ones. Since not *all* (P, P^{-1}) combinations show exponential trends, we cannot say that the key difficulty of users was in learning how to use the VA tool.

6.5.1.3 *Added value of manual labeling*

Let us now analyze the added value of manual labeling compared to the defined baseline (Tab. 23). For this, we show in Table 25 the gain (difference) of classification performance obtained by manual labeling (Tab. 24) *vs* using DeepFA pseudo la-

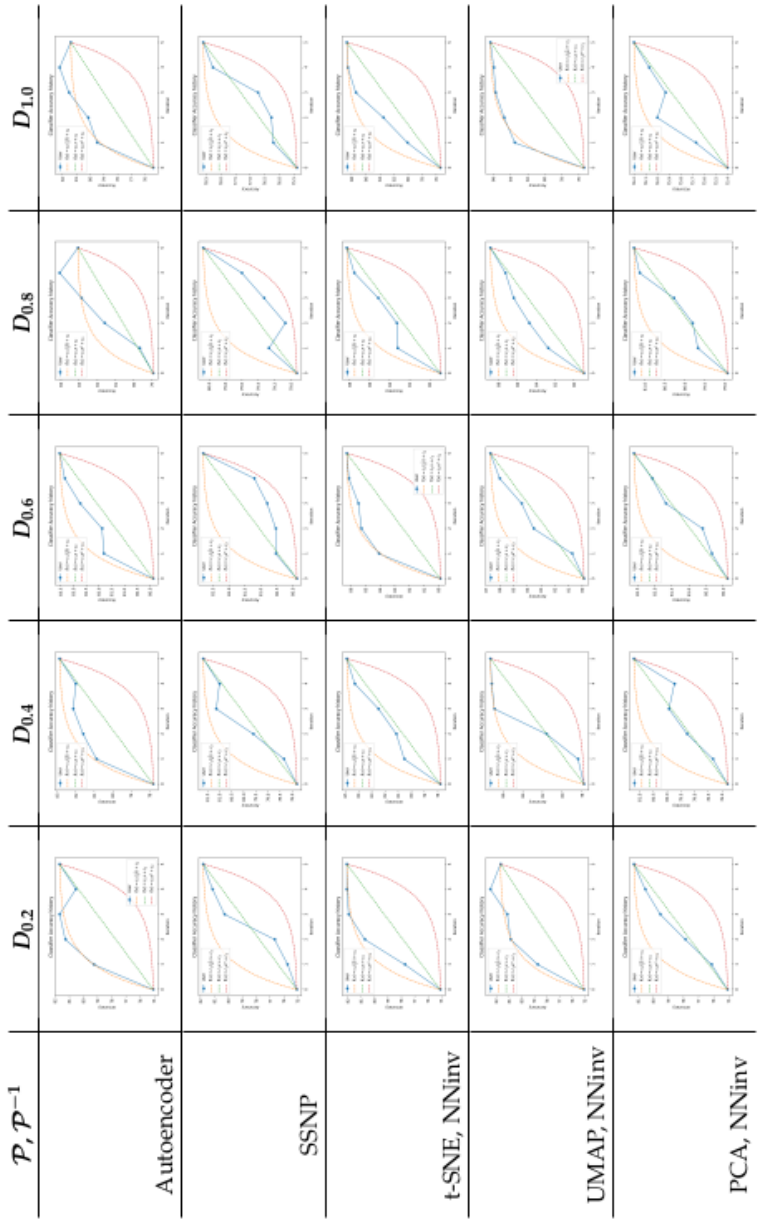


Figure 6.9: Results of $U1$ for **classification accuracy** over five iterations, using different direct and inverse projection techniques (P, P^{-1}) and input dataset fractions (D).

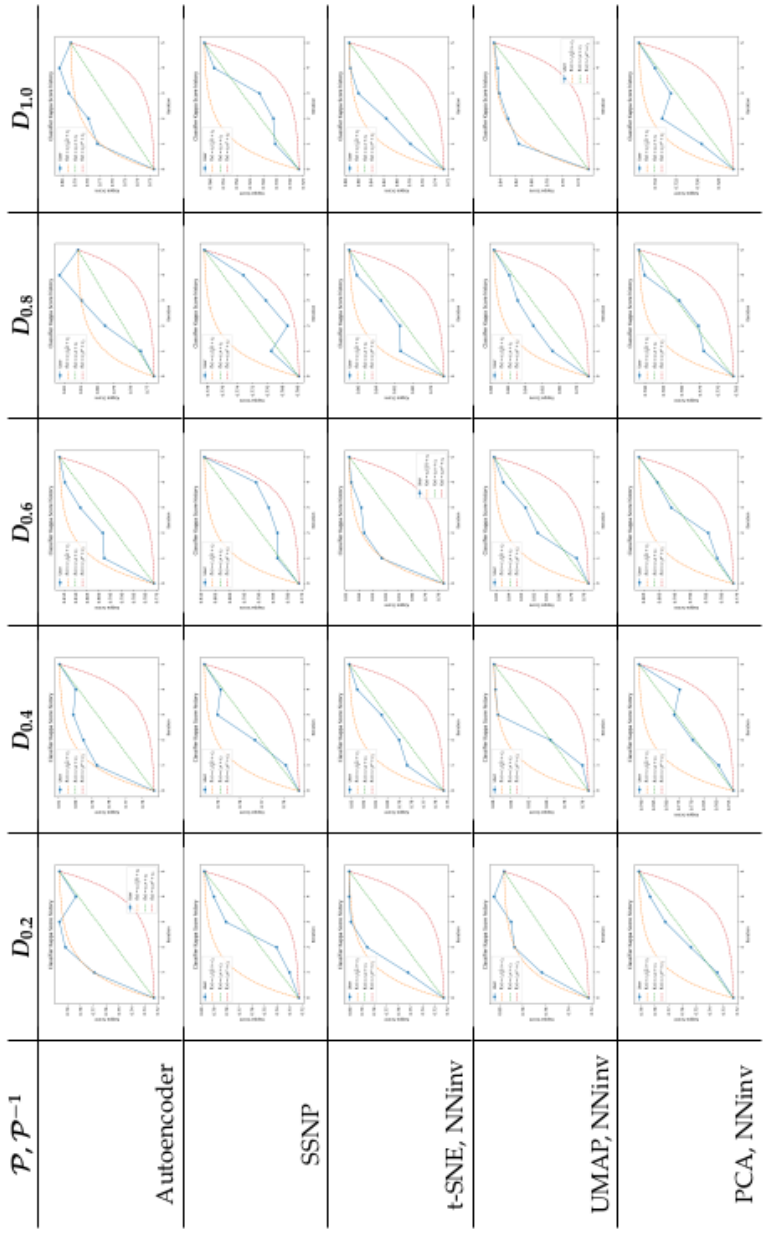


Figure 6.10: Results of $U1$ for **classification** κ over five iterations, using different direct and inverse projection techniques (P, P^{-1}) and input dataset fractions (D).

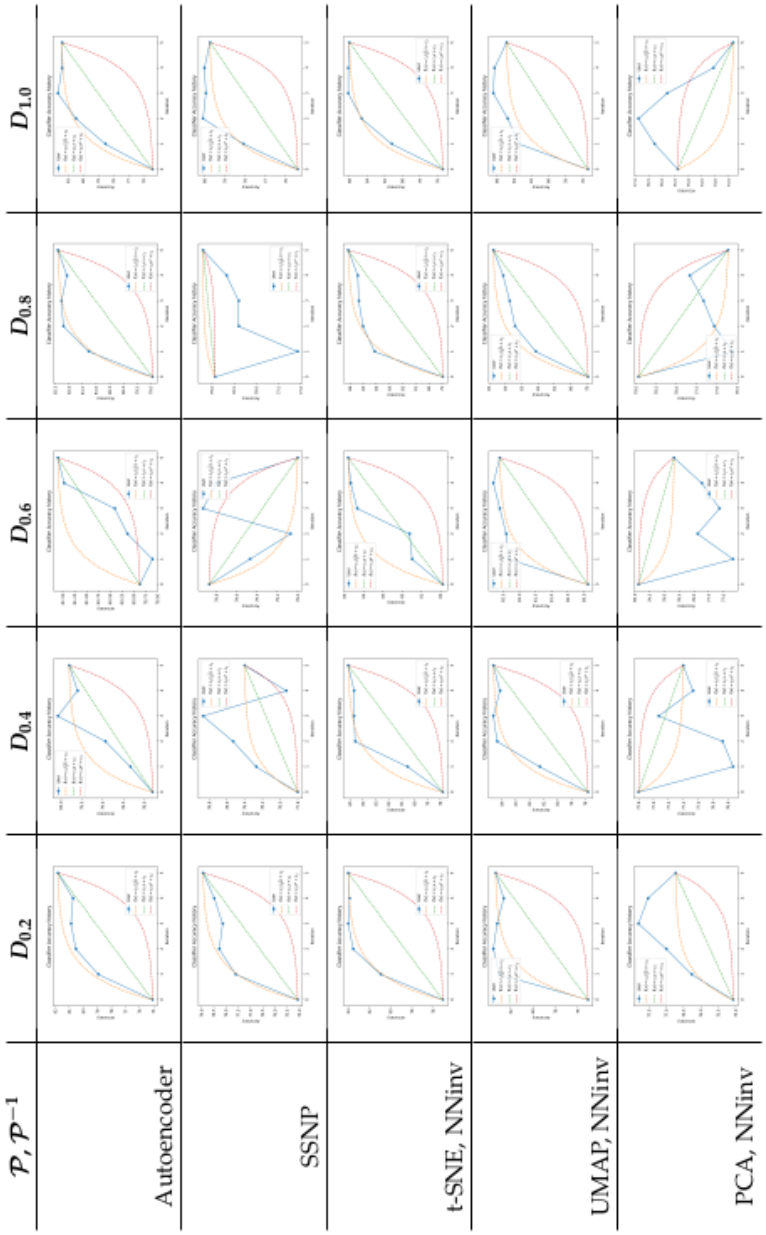


Figure 6.11: Results of $U2$ for **classification accuracy** over five iterations, using different direct and inverse projection techniques (P, P^{-1}) and input dataset fractions (D).

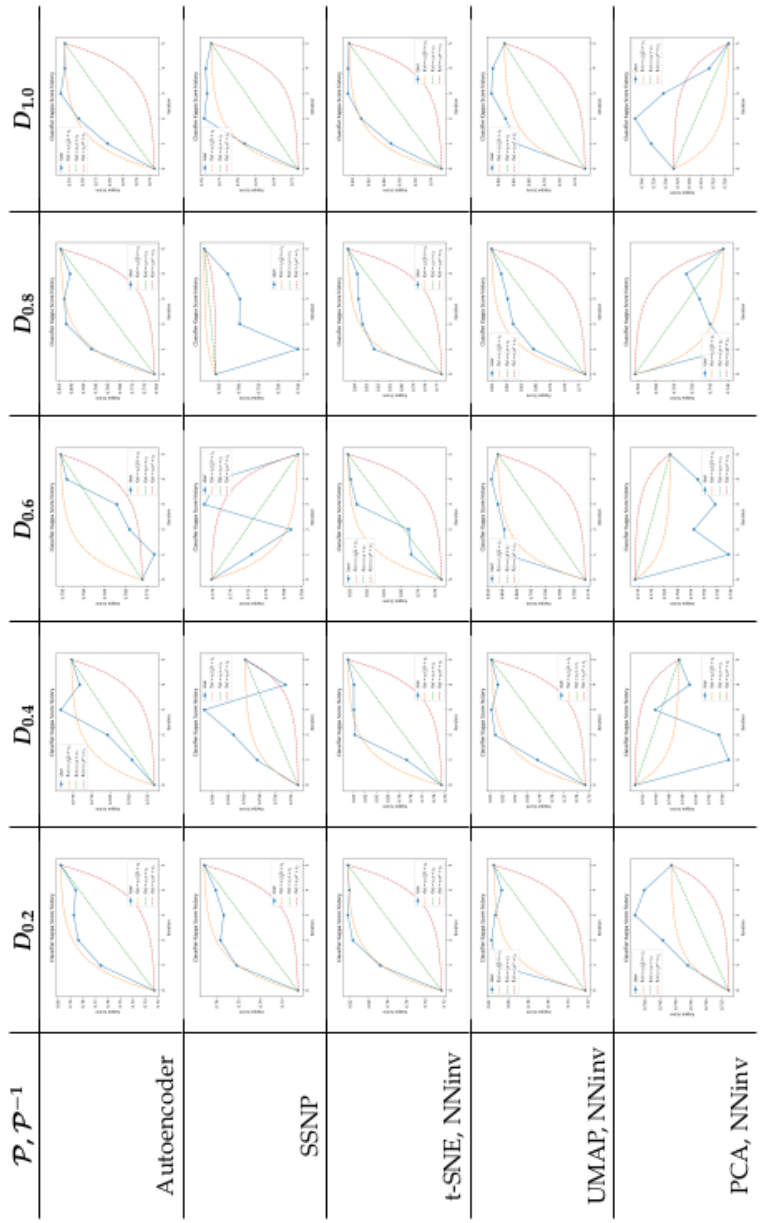


Figure 6.12: Results of $U2$ for **classification** κ over five iterations, using different direct and inverse projection techniques (P, P^{-1}) and input dataset fractions (D).

bels (Tab. 23). Positive values here indicate that manual labeling exceeds the lower bound (DeepFA); negative values indicate the opposite.

P, P^{-1}	user	$D_{20\%}$		$D_{40\%}$		$D_{60\%}$		$D_{80\%}$		$D_{100\%}$	
		acc	κ	**acc**	κ	**acc**	κ	**acc**	κ	**acc**	κ
autoencoder	U1	0.0660	0.0735	0.0507	0.0563	0.0366	0.0408	0.0414	0.0459	0.0600	0.0666
	U2	0.0640	0.0710	0.0440	0.0488	0.0620	0.0688	0.0700	0.0777	0.0600	0.0666
SSNP	U1	0.0673	0.0750	0.0387	0.0430	0.0293	0.0326	0.0144	0.0126	0.0320	0.0355
	U2	0.0386	0.0431	0.0060	0.0067	-0.0087	-0.0097	0.0027	0.0029	0.0433	0.0480
t-SNE, NNinv	U1	0.0700	0.0779	0.0707	0.0786	0.0640	0.0712	0.0934	0.1037	0.1327	0.1474
	U2	0.0893	0.0995	0.0733	0.0817	0.0493	0.0549	0.0720	0.0800	0.1060	0.1177
UMAP, NNinv	U1	0.0600	0.0736	0.0947	0.1053	0.0680	0.0756	0.0994	0.1104	0.1093	0.1215
	U2	0.0826	0.0921	0.0680	0.0757	0.0273	0.0304	0.0580	0.0645	0.0947	0.1052
PCA, NNinv	U1	0.0620	0.0691	0.0340	0.0378	0.0273	0.0304	0.0234	0.0259	0.0080	0.0089
	U2	0.0166	0.0187	-0.0060	-0.0066	-0.0120	-0.0133	-0.0220	-0.0245	-0.0207	-0.0232

Table 25: MNIST datset. Gain in classification performance obtained by manual labeling *vs DeepFA* pseudo labels. Different direct and inverse projection techniques and different amounts of samples in D subsets for two users are considered.

We see that (SSNP) and (PCA, NNInv) yielded the lowest gain for both users. Separately (autoencoder) also yielded lower gain values for $U1$. The highest gain was consistently reached for (tSNE, NNinv) and (UMAP, NNinv) for both users. Among these two combinations, (tSNE, NNinv) got the best results with more than 0.10 increase in accuracy and κ as compared to (UMAP, NNInv).

Summarizing the above, we see that, when using specific (P, P^{-1}) combinations, manual labeling can surpass the performance of DeepFA labeling, even when users are offered a relatively short amount of time to visually explore (and label) the dataset. The (P, P^{-1}) combinations can be roughly grouped in two classes – those which help manual labeling, *i.e.*, (tSNE, NNinv) and (UMAP, NNinv), and the remaining ones. Among the 'good' combinations, (tSNE, NNinv) consistently showed the best-added value for all considered dataset sizes and for both users.

6.5.2 *Evaluation in real-world problem:* P.cysts

We now present the evaluations of manual labeling for the more complex *P.cysts* dataset. For this, we only considered the (tSNE, NNinv) combination, as this offered the best results for the manual labeling of the simpler dataset (MNIST).

6.5.2.1 *Defining the baseline*

As for the MNIST dataset, we first establish a baseline indicating the lower bound performance (reached by DeepFA labeling) and the upper bound performance (reached by using all true labels). Table 26 shows these results, as well as the DeepFA labeling performance. Note that, as opposed to the MNIST experiments, we now use the entire set of $|D| = 2696$ samples. This is due to the larger difficulty of the *P.cysts* dataset, but also due to the earlier observation telling that too few samples can create problems for the manual labeling process (Sec. 6.5.1.2). We see that the difference between the upper bound and lower bound is 0.0225 and 0.0334 for accuracy and κ, respectively. That is, DeepFA already yields a quite good result, leaving only a small interval where manual labeling could present improvements.

	metric	D
labeling performance	acc	0.8560
classifier performance	acc	0.8564
(pseudo labels)	κ	0.8043
classifier performance	acc	0.8789
(true labels)	κ	0.8377
# of samples		2696

Table 26: *P.cysts* dataset. Performance of DeepFA labeling, classification using the DeepFA pseudolabels (lower bound), and classification using all true labels (upper bound). Right column shows percentage of samples in the dataset used in the experiments.

6.5.2.2 *Added value of manual labeling*

Table 27 presents the classification accuracy and κ values obtained after manual labeling by both earlier users ($U1$ and $U2$). Obviously, they start from the same baseline value, *i.e.*, those offered by the DeepFA pseudolabeling. We see that both users managed to use the VA tool to increase classification performance over the five available iterations – specifically, $U1$increased accuracy by 0.0112 and κ by 0.0168; while $U2$ increased accuracy by 0.0173 and κ by 0.0249. This proves that our VA tool offers added value even for more complex, real-world, datasets.

Figure 6.13 presents the aggregated results in Tab. 27 in a visual form. Here, we see similar trends like the one we found for MNIST (Figs. 6.9-6.12) – that is, exponential, respectively logarithmic, improvement of performance over the iterations. However, these trends are now not related to the use of different (P, P^{-1})

	accuracy		κ	
iteration	**U1**	**U2**	**U1**	**U2**
0	0.8564		0.8043	
1	0.8538	0.8521	0.8019	0.7987
2	0.8581	0.8728	0.8096	0.8279
3	0.8590	0.8763	0.8107	0.8322
4	0.8616	0.8763	0.8131	0.8321
5	0.8676	0.8737	0.8211	0.8292

Table 27: *P.cysts* dataset. Results of classification accuracy and κ for classifiers trained with pseudo labels generated by the user interactive labeling using the visualization tool after 5 iterations by two different users.

combinations, but to the two different users. For ease of comparison, the last row of the figure shows the accuracy and κ trends for the two users ($U1$: blue; $U2$: orange) superimposed. The difference in these two trends is consistent with our knowledge about the users: Indeed, $U2$ was familiar with the *P.cysts* dataset, which explains the quick gains obtained in the first iterations and, likely, also the fact that, at the end, $U2$ obtains a slightly higher end performance than $U1$. However, we believe that the results for $U1$ are more interesting: Even if this user had zero prior knowledge of (and exposure to) this dataset, $U1$ managed to achieve an almost as high performance as $U1$ with the same effort (time).

6.6 DISCUSSION

We next discuss the overall advantages and limitations of our proposed VA-based approach for manual labeling.

Genericity: Our approach is generic in the sense that it can be applied to any dataset (and classifier), in a black box manner. That is, no details of the internal operation of the classifier are required by our VA tool. However, if the data dimensionality is too large – roughly, over a few thousand dimensions – directly using the tool's projection technique (t-SNE) to construct the DBM can be problematic. This can be inspected by the user by monitoring the projection errors present in the visualization (Sec. 6.3.1). When these are too high, the dataset can be reduced to a smaller number of features using *e.g.* autoencoders prior to its use in the tool.

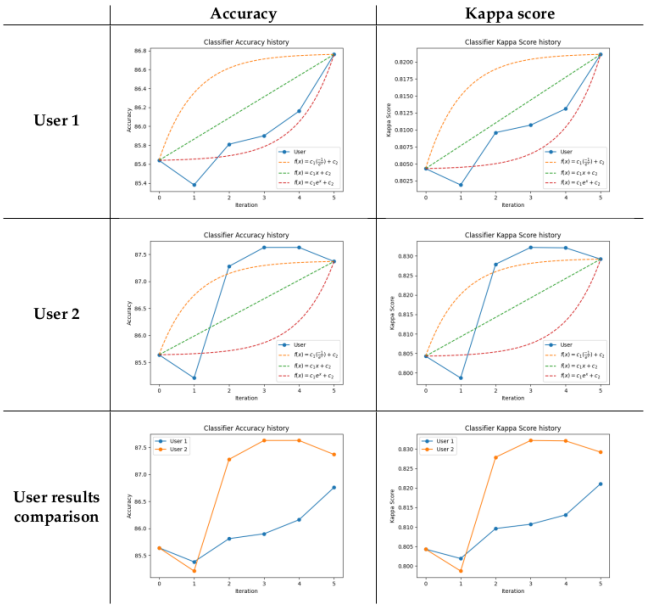


Figure 6.13: *P.cysts* dataset. Classification results over five iterations, considering two different users.

Effectiveness: While demonstrated only up to a limited extent – that is, for two datasets and two users only – we have shown evidence that our VA tool can help constructing classifiers with a higher performance than what can be fully automatically obtained, *i.e.*, by using an automatic pseudolabeling algorithm such as DeepFA. This is, we believe, the strongest contribution of our work as it underlies our original hypothesis, namely that a *combination* of automatic algorithms and human insight is the optimal way for ML engineering problems. That being said, several limitations exist to this end. First, the increase of effectiveness – the so-called gain that manual labeling offers *vs* automatic labeling – is quite small (a few percent points), at least for the problems we have studied. Secondly, it is not clear how generalizable this gain is, *i.e.*, if it could be consistently observed for a wide spectrum of users, classifiers, and datasets.

Workflow: So far, our VA tool does not provide a specific way to 'instruct' users on how they can best perform manual labeling, apart from the general interpretation of the visualizations described in Sec. 6.3 – that is, the direct projection errors, inverse projection errors, confidence-annotated DBMs, and sample

tooltips. Users learn how to use the tool via a trial-and-error procedure, *i.e.*, selecting a few samples to label and monitoring the change in DBMs and classifier performance that the new labels create. Key to the effectiveness of this procedure is a *fast* execution of the label-retrain-visualize loop, so that one can perform multiple such iterations quickly, including undo operations when negative effects are observed. In our current implementation, this loop takes a few up to ten seconds, depending on the used hardware (a mid-range PC).

We see two avenues for improvement in this respect. First, a more scalable computation of the (re)training and calculation of the DBM is needed if one aims to use our VA tool for larger datasets and/or more complex classifiers. Secondly, and more importantly, we believe that *explanations* should be added to the provided visualizations to make them more effective. For instance, we have (anecdotically) noticed that users do not know how to exactly interpret the direct and inverse error maps, apart from ignoring samples that fall in high-error areas. Moreover, we know that non-linear (inverse) projections can introduce deformations in DBMs which are, next, potentially misleading users Wang et al. (2023). Additional techniques such as interactive tools that show users *e.g.* which are the true nearest-neighbors, in data space, of a (set of) point(s) in the projection; and depicting the actual distance to the closest decision boundary of every DBM point Rodrigues et al. (2019), could increase the efficiency and effectiveness of manual labeling.

A small, but important, final improvement point concerns the treatment of true labels. Currently, our VA tool does not distinguish between such labels and those pre-assigned by the DeepFA pseudo labeling step executed prior to the start-up of the VA workflow. As such, users may, during manual labeling, inadvertently change such true labels, therefore unnecessarily decreasing the performance of the training. A simple fix could prevent this from taking place.

6.7 CONCLUSION

In this chapter, we have shown how automatic and user-driven pseudo labeling methods can be combined in a single workflow that helps constructing high-performance classifiers. Our proposed workflow starts by executing DeepFA, the automatic pseudo labeling method we introduced in Chapter 3, from a (very) small set of true labels to the entire training set; next, users can examine this training set, together with the classifica-

tion model that this set leads to (depicted as decision boundary maps), and additionally label data points to improve classification performance.

We have provided two sets of experiments, on both a relatively simple dataset (MNIST) and a more complex one (*P.cysts*), both to be classified by a neural network architecture. We have shown that, when using a specific combination of direct and inverse projection techniques to create the decision maps (t-SNE and NNInv, respectively), users can achieve an average increase in classification accuracy and κ of 8.21% and 0.00913 (MNIST), and 1.43% and 0.0209 (*P.cysts*) with a relatively small effort – namely, a maximum of 15 minutes of tool operation. We have also shown evidence that supports the hypothesis that this gain is not dependent on prior knowledge of the dataset and/or classification problem at hand, and thus can be reached by a wide spectrum of users having general knowledge in machine learning and visual analytics. To our knowledge, this is the first study that explores the use of decision boundary maps in an active learning looping scenario with the aim of improving classifier performance.

Many avenues are now opened for future work. First and foremost, testing our VA approach with more users, datasets, and classifiers is needed to confirm (or refine) our findings and thus get a better idea of the added value it provides. Secondly, improvements in the depiction of the classifier behavior (via annotated decision maps), but also for the depiction of how the classifier actually uses the provided manual labels during training, can reduce the iterative effort needed to manually construct a good set of pseudo labels.

7 CONCLUSIONS

We close this thesis by revisiting our proposed contributions and also outlining directions for future work.

In Chapter 1 of this thesis, we introduced our general research question:

How can we exploit the synergy between ML and DR techniques to improve each other?

We further refined this general question into two more specific ones that address relationship between machine learning (ML) and dimensionality reduction (DR), as follows:

RQ1: How to use multidimensional projections to build better models for machine learning?

RQ2: How does projection quality relate to data separation and classification performance?

Throughout the Chapters 3 to 6, we explored this questions. We next summarize the results obtained during this exploration, as well as how these answer our initial questions.

We addressed RQ1 in Chapter 3 by proposing a pseudo-labeling approach, called DeepFA, that explores the ability of a multidimensional projection to generate a reduced (two-dimensional) feature space with enough information to improve feature learning and classifier performance over iterations. Our findings confirm that the t-SNE projection technique – well known in information visualization for its high ability to capture *data structure* present in high dimensions – can generate 2D feature spaces which are also effective for feature learning and classifier engineering. In other words, we have shown that 2D projections can be used instead of the original feature space, which typically has hundreds of dimensions or even more, for designing feature and classifier learning models through pseudo-labeling.

In addition to that, and in line with (Benato et al., 2021c), our findings show that using such 2D projection spaces can lead to better results (in terms of classifier performance) than using the original high-dimensional feature space in the context of feature learning strategies. Differently than (Benato et al., 2021c), we extrapolate from a few supervised examples to use a more robust

feature learning strategy as a pre-trained convolutional neural architecture instead of an unsupervised autoencoder. This offers a better initial feature space as input to the multidimensional projection algorithm. For each chosen step in our pipeline, we presented an extensive set of experiments that demonstrate the power of our approach.

Research question RQ2 is explored in Chapter 4. In this chapter, we explore pseudo-labeling and projection techniques to link data separation (DS), visual separation (VS), and classifier performance (CP). We assessed DS by evaluating the data structure of learned features when using a linear classifier. High classification values using such a classifier tell when data is *linearly* separable in high dimensions and, thus, whether a dataset has a good DS. From a high-dimensional feature space with good DS, we showed that projections can produce a 2D feature space having a high VS. Finally, we used such a 2D projection space (with good VS) to pseudo label samples and showed that these can train a classifier which exhibits a high CP.

In our work in Chapter 4, we define 'good VS' as having a projection whose 2D scatterplot appears to be separated into distinct groups, each one having labels of mainly a single class. After evaluating various projection techniques, our findings show that VS indeed can be related to DS and CP for a group of such techniques. Knowing which techniques exhibit this property, *i.e.*, are part of the aforementioned group, is valuable for supporting classifier engineering and, arguably, for other infovis applications where VS is important. In addition, we demonstrate *how* to model a classifier by certifying that this correlation exists. That is, by improving DS and then VS, one can obtain a classifier with high CP. To our knowledge, this is the first time that (a) the correlation between DS, VS, and CP is demonstrated, and (b) one has indicated which *specific* projection techniques preserve a strong DS-VS-CP correlation (or not).

We further address RQ2 in Chapter 5 by reverting the roles of VS and CP analyzed in Chapter 4. That is, we extend our earlier observation that high VS (assessed visually by users) implies a high CP to hypothesize that a high CP (measured automatically) implies a high VS. As such, we propose to assess the VS of 2D projections by a metric that evaluates the CP of a graph-based semi-supervised classifier used to propagate labels in the 2D projection. We concern ourselves with cases where the dataset being projected is labeled. In more detail, we claim that, when a classifier achieves high CP in the labeling task, then the 2D projection has a good VS. Intuitively, this is based on the observation that,

when a classifier propagates labels with low errors, then labeling (in the 2D projection space) is an easy task; this next implies that the 2D projection is well separated into distinct groups of same-label samples.

Our findings show that our proposed metric can better gauge VS in projections than projection-quality metrics commonly used in the DR literature. We further justify our proposal by performing a user evaluation that shows that our metric correlates well with VS as perceived by over 100 subjects. To our knowledge, this is the first time that VS of 2D projections is measured by a pseudo labeling task. Our proposal is simple to implement, works for any projection technique, and scales computationally very well, being faster than existing projection-quality metrics. We can link this contribution with the ones presented in Chapter 3 as follows. Whenever we measure κ at the end of a DeepFA pseudo labeling iteration, *not only* the produced feature space is evaluated, but also the 2D projected space. This means that if we obtain a high κ for the 2D feature space – which is precisely our proposed metric to evaluate VS – then we guarantee that a 2D projected space with good VS is achieved for the next DeepFA iteration.

Finally, in Chapter 6, we turn back to RQ1. In contrast to our work in Chapter 3, which has shown how to use 2D projections to *automatically* improve the construction of classification models (by means of automatic pseudo labeling), we now consider assisting the human *user* to perform such improvements. For this, we incorporate an active learning looping to improve pseudo labels produced by the DeepFA method (Chapter 3). For this, we combine several visualization techniques, including direct projection errors, inverse projection errors, decision boundary maps (DBMs), and data tooltips to present to the user both the training dataset and the behavior of the classifier trained on this dataset. This allows users to detect areas – more specifically, unlabeled samples in the training set – which, if correctly labeled by manual intervention, could improve the classifier performance. We combine our techniques to produce a visual analytics (VA) tool for manual labeling, training, and testing of classification models.

Our results show that, when starting from an automatic set of pseudo labels computed by the DeepFA strategy in Chapter 3, and using the assistance of the aforementioned VA tool, users can achieve an increase in classification performance (as opposed to the DeepFA baseline) after only a few iterations totalling 15 minutes of manual work. Our findings indirectly support our hypothesis that multidimensional projections capture well *data structure* present in high dimensions in a way that the resulting 2D feature

space offers enough information for further human abstraction to support manual labeling. Unlike Chapter 3, the information provided by the multidimensional projection is now exploited by a user (human, interactively) *instead* of an algorithm (machine, automatically). At a higher level, the work in this chapter answers RQ1 by showing that *both* humans and machines can use multidimensional projections to build better classifiers; and that, when *cooperating* (by using such projections), humans and machines obtain results which surpass what can be obtained when using only manual, or only automatic, methods.

As it should be evident by now, dimensionality reduction techniques and pseudo-labeling play an essential role in all our contributions. Dimensionality reduction successfully captures data structure from high dimensions in a way that machines and humans can distill relevant information to design and improve machine learning models. Pseudo labeling can retain, and next create, substantial knowledge in the produced labels during the entire data flow from the learned space, to the reduced 2D projection space, and next to the way users abstract this information during their iterative manual exploration and labeling of data via 2D projection spaces.

7.1 DIRECTIONS FOR FUTURE WORK

Based on our research, we foresee the following potential directions for future work.

Feature and classifier learning: In our work, we have presented (and validated by evaluations) an approach that results in labels for unsupervised samples and, in the end, delivers a high-quality trained classifier. However, as mentioned at the beginning of this chapter, our approach has an indirect effect of added value: Over the iterations of our DeepFA method, we improve the feature space (in terms of data separation), which is our main goal, needed for the final construction of high-quality classifiers. As a side effect of this, given the observed correlation of data separation with visual separation, we also improve the visual separation exhibited by 2D scatterplots created by projections from the feature space. This means that we could use a very similar iterative pipeline to design novel ways to create high-quality projections (in terms of visual separation) of high-dimensional data.

Relationship between VS, DS, and CP: We have examined this relationship under the pseudo labeling task, that is, using a classifier to predict labels for unsupervised samples and considering this prediction accurate in the other steps. In other words, we explore the classifier's ability to predict labels (measured as classifier performance) to link the VS, DS, and CP concepts. However, an open question remains: Can we link VS, DS, and CP using other ML tasks? Given the high level definition of 'separation' (either in the data or visual space), we believe this to be the case. As such, one can next explore clustering or metric learning strategies to find novel ways in which VS, DS, and CP can be linked to each other – or, in more technical terms, novel ways in which one can measure how well these aspects are correlated with each other.

Projection quality: Our work in Chapter 5 has shown how VS in projections can be simply and efficiently measured by means of the computation of a classifier performance metric. However, the respective metric – Cohen's κ – requires the presence of labels for the projected dataset. It is interesting to consider how we could extend this approach to assess the VS of unlabeled data. This requires, in the first place, a different definition of VS, likely considering only the relative (visual) separation of closely-packed groups of points in a 2D projection. To measure this, one could explore using an unsupervised version of the same classifier as we used so far in supervised mode. A separate direction involves exploring the same type of classifier to evaluate projection quality in terms of *reducing* the high-dimensional space while preserving data patterns, *e.g.*, by combining optimum path forests computed in both high and low-dimensional spaces. This would open new avenues to quantifying the much-discussed, but still insufficiently captured by current metrics, concept of preservation of data structure by projections.

Active learning and decision boundary maps: In Chapter 6, we have shown how visualization techniques (decision boundary maps and error maps) can be employed to guide users to manually label datasets so as to lead to the construction of higher-performance classifiers than when using purely automatic labeling. Still, this limits the impact of the human analytic power to the investigation of the 2D projection space only. We believe that users, assisted by suitable visualization techniques, can be effective in driving the feature and classifier learning looping so as to produce even higher-quality pseudo labels and, ultimately, clas-

sifiers. For example, users can be empowered, by visual analytics tools, to manually change the classifier boundaries in the 2D projections during classifier retraining. This way, they could see how their actions can improve the feature space during the online learning of the classifier. Ultimately, and at a higher level, the key message we are sending for future work is that the gap between the operation of machine learning tools (label propagation, classifier training) and human insight can, and should, be reduced by means of visual analytics techniques. In the end, humans and machines should seamlessly work with and alongside each other, on all parts of the machine learning pipeline, dynamically complementing and improving each others' actions and results, so as to lead to a higher-quality, and more explainable, final result.

BIBLIOGRAPHY

M. M. Abbas, M. Aupetit, M. Sedlmair, and H. Bensmail. ClustMe: A Visual Quality Measure for Ranking Monochrome Scatterplots based on Cluster Patterns. *Computer Graphics Forum*, 2019.

D. K. Agrafiotis. Stochastic proximity embedding. *Journal of Computational Chemistry*, 24(10):1215–1221, 2003.

G. Albuquerque, M. Eisemann, and M. Magnor. Perception-based visual quality measures. In *Proc. IEEE VAST*, pages 13–20, 2011.

T. A. Almeida, J. M. G. Hidalgo, and A. Yamakami. Contributions to the study of sms spam filtering: new collection and results. In *Proc. ACM symposium on Document engineering*, pages 259–262, 2011.

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

W. Amorim, G. Rosa, Rogério, J. Castanho, F. Dotto, O. Rodrigues, A. Marana, and J. Papa. Semi-supervised learning with connectivity-driven convolutional neural networks. *Pattern Recognit. Lett.*, 128:16 – 22, 2019.

N. Andrienko, G. Andrienko, G. Fuchs, A. Slingsby, C. Turkay, and S. Wrobel. *Visual Analytics for Data Scientists*. Springer, 2020.

R. G. Andrzejak, K. Lehnertz, F. Mormann, C. Rieke, P. David, and C. E. Elger. Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Physical Review E*, 64(6):061907, 2001.

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. IWAAL*, pages 216–223, 2012.

E. Arazo, D. Ortego, P. Albert, N. E. O'Connor, and K. McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In *Proc. IJCNN*, pages 1–8. IEEE, 2020.

R. Becker, W. Cleveland, and M. Shyu. The visual design and control of trellis display. *Journal of Computational and Graphical Statistics*, 5(2):123–155, 1996.

M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Proc. NIPS*, volume 14, pages 585–591, 2001.

B. C. Benato, A. C. Telea, and A. X. Falcão. Semi-supervised learning with interactive label propagation guided by feature space projections. In *Proc. SIBGRAPI*, pages 392–399, 2018.

B. C. Benato, A. X. Falcão, and A. C. Telea. Measuring the quality of projections of high-dimensional labeled data. *Computers & Graphics*, 116:287–297, 2023a.

B. C. Benato, C. Grosu, A. X. Falcao, and A. C. Telea. Human-in-the-loop: Using classifier decision boundary maps to improve pseudo labels. In *submitted*, 2024.

B. C. Benato, J. F. Gomes, A. C. Telea, and A. X. Falcão. Semi-supervised deep learning based on label propagation in a 2D embedded space. In *Proc. CIARP*, pages 371–381. Springer, 2021a.

B. C. Benato, A. C. Telea, and A. X. Falcao. Iterative pseudo-labeling with deep feature annotation and confidence-based sampling. In *Proc. SIBGRAPI*, pages 192–198. IEEE, 2021b.

B. C. Benato, A. X. Falcao, and A. C. Telea. Linking data separation, visual separation, and classifier performance using pseudo-labeling by contrastive learning. In *Proc. VISAPP*. SciTePress, 2023b.

B. C. Benato, A. X. Falcão, and A. C. Telea. Linking data separation, visual separation, and classifier performance using pseudo-labeling by contrastive learning. In *Proc. VISAPP*, 2023c.

B. C. Benato, A. X. Falcão, and A. C. Telea. Code repository. https://github.com/barbarabenato/measuring_quality_of_projections, 2023d.

B. C. Benato, J. F. Gomes, A. C. Telea, and A. X. Falcão. Semi-automatic data annotation guided by feature space projection. *Pattern Recognit.*, 109:107612, 2021c. ISSN 0031-3203.

B. C. Benato, A. C. Telea, and A. X. Falcão. Deep feature annotation by iterative meta-pseudo-labeling on 2d projections. *Pattern Recognition*, 141:109649, 2023e.

Benato, B.C. Deepfa: "deep feature annotation", 2022. https://github.com/barbarabenato/DeepFA.

J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13(null):281–305, feb 2012. ISSN 1532-4435.

J. Bernard, M. Hutter, M. Zeppelzauer, D. Fellner, and M. Sedlmair. Comparing visual-interactive labeling with active learning: An experimental study. *IEEE Trans. Vis. Comput. Graph.*, 24 (1):298–308, Jan 2018. ISSN 1077-2626.

J. Bernard, M. Zeppelzauer, M. Lehmann, M. Müller, and M. Sedlmair. Towards user-centered active learning algorithms. *Computer Graphics Forum*, 37(3):121–132, 2018.

J. Bertin. *La graphique et le traitement graphique de l'information*. Flammarion, 1977.

J. Bertin. *Semiology of Graphics*. University of Wisconsin Press, Madison, WI, 1983.

M. Brand. Charting a manifold. In *Proc. NIPS*, pages 985–992, 2002.

C. Bredius, Z. Tian, and A. Telea. Visual exploration of neural network projection stability. In *Proc. Machine Learning Methods in Visualisation for Big Data*. Eurographics, 2022.

B. Broeksema, T. Baudel, and A. Telea. Visual analysis of multi-dimensional categorical datasets. *Computer Graphics Forum*, 32 (8):158–169, 2013.

C. J. C. Burges. Dimension reduction: A guided tour. *Foundations and Trends in Machine Learning*, 2(4):275–365, 2010.

T. Caliński and J. Harabasz. A dendrite method for cluster analysis. *Commun. Stat.*, 3(1):1–27, 1974.

P. Cascante-Bonilla, F. Tan, Y. Qi, and V. Ordonez. Curriculum labeling: Revisiting pseudo-labeling for semi-supervised learning. *arXiv preprint arXiv:2001.06001*, 2020.

W. Castelein, Z. Tian, T. Mchedlidze, and A. Telea. Viewpoint-based quality for analyzing and exploring 3d multidimensional projections. In *Proc. IVAPP*, 2023.

T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

Y. Chen, M. Crawford, and J. Ghosh. Improved nonlinear manifold learning for land cover classification via intelligent landmark selection. In *Proc. IEEE IGARSS*, pages 545–548, 2006.

F. Chollet et al. Keras. https://keras.io, 2015.

M. E. Chowdhury, T. Rahman, A. Khandakar, R. Mazhar, M. A. Kadir, Z. B. Mahbub, K. R. Islam, M. S. Khan, A. Iqbal, N. Al Emadi, et al. Can ai help in screening viral and covid-19 pneumonia? *IEEE Access*, 8:132665–132676, 2020.

P. M. Ciarelli and E. Oliveira. Agglomeration and elimination of terms for dimensionality reduction. In *Proc. IEEE ISDA*, pages 547–552, 2009.

R. R. Coifman and S. Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30, 2006.

D. Coimbra, R. Martins, T. A. T. Neves, A. C. Telea, and F. V. Paulovich. Explaining three-dimensional dimensionality reduction plots. *J. of Information Visualization*, 2015. DOI:10.1177/1473871615600010.

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.

K. R. Coombes, G. Brock, Z. B. Abrams, and L. V. Abruzzo. Polychrome: Creating and assessing qualitative palettes with many colors. *Journal of Statistical Software*, 90, 2019.

J. Cunningham and Z. Ghahramani. Linear dimensionality reduction: Survey, insights, and generalizations. *J. Mach. Learn. Res.*, 16:2859–2900, 2015.

N. Das, S. Chaba, R. Wu, S. Gandhi, D. H. Chau, and X. Chu. GOGGLES: Automatic image labeling with affinity coding. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, page 1717–1732. Association for Computing Machinery, 2020. ISBN 9781450367356.

S. Dasgupta. Experiments with random projection. In *Proc. UAI*, page 143–151, 2000.

T. Davidson, D. Warmsley, M. Macy, and I. Weber. Automated hate speech detection and the problem of offensive language. In *Proc. AAAI ICWSM*, volume 11, pages 512–515, 2017.

V. De Silva and J. B. Tenenbaum. Sparse multidimensional scaling using landmark points. Technical report, tec. rep., Stanford University, 2004.

D. L. Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. In *Proc. of the National Academy of Sciences*, number 10, pages 5591–5596, 2003.

N. Elmqvist, P. Dragicevic, and J. Fekete. Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. *IEEE TVCG*, 14(6):1539–1148, 2008.

D. Engel, L. Hattenberger, and B. Hamann. A survey of dimension reduction methods for high-dimensional data analysis and visualization. In *Proc. IRTG Workshop*, volume 27, pages 135–149. Schloss Dagstuhl, 2012.

M. Espadoto, R. Martins, A. Kerren, N. Hirata, and A. Telea. Toward a quantitative survey of dimension reduction techniques. *IEEE TVC*, 27(3):2153–2173, 2019a.

M. Espadoto, F. C. M. Rodrigues, N. S. T. Hirata, R. H. Jr, and A. Telea. Deep learning inverse multidimensional projections. In *Proc. EuroVA*. Eurographics, 2019b.

M. Espadoto, N. Hirata, and A. Telea. Deep learning multidimensional projections. *Information Visualization*, 9(3):247–269, 2020.

M. Espadoto., N. S. T. Hirata., and A. C. Telea. Self-supervised dimensionality reduction with neural networks and pseudo-labeling. In *Proc. IVAPP*, pages 27–37, 2021.

M. Espadoto, G. Appleby, A. Suh, D. Cashman, M. Li, C. Scheidegger, E. W. Anderson, R. Chang, and A. C. Telea. UnProjection: Leveraging inverse-projections for visual analytics of high-dimensional data. *IEEE Transactions on Visualization and Computer Graphics*, 29(2):1559–1572, 2023.

C. Faloutsos and K. I. Lin. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proc. ACM SIGMOD*, pages 163–174, 1995.

J. L. Fleiss and J. Cohen. The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educ. Psychol. Meas.*, 33(3):613–619, 1973.

J. A. M. Flores, L. Linsen, and A. Telea. Skeleton-based scagnostics. *IEEE TVCG*, 24(1):542–552, 2017.

J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, 23(9):881–890, 1974.

Y. Gal, R. Islam, and Z. Ghahramani. Deep bayesian active learning with image data. In *Proc. ICML*, pages 1183–1192, 2017.

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.

P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

J. B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

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*, page 545?552, 2004.

I. Guyon, A. Saffari, G. Dror, and G. Cawley. Agnostic learning vs. prior knowledge challenge. In *Proc. IEEE IJCNN*, pages 829–834, 2007.

N. Halko, P. Martinsson, and J. Tropp. Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions. *arXiv:0909.4061*, 2009.

D. Hand, H. Mannila, and P. S. P. *Principles of Data Mining*. MIT Press, 2001.

C. D. Hansen and C. R. Johnson. *The Visualization Handbook*. Elsevier, 2005.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE CVPR*, pages 770–778, 2016.

K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proc. IEEE CVPR*, pages 9729–9738, 2020.

X. He and P. Niyogi. Locality preserving projections. In *Proc. NIPS*, volume 16, pages 153–160, 2003.

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 2006.

A. Hyvarinen. Fast ica for noisy data using gaussian moments. In *Proc. IEEE ISCAS*, volume 5, pages 57–61, 1999.

A. Inselberg and B. Dimsdale. Parallel coordinates: A tool for visualizing multidimensional geometry. In *Proc. IEEE Visualization*, pages 361–378, Los Alamitos, CA, 1990. IEEE Press.

A. Iscen, G. Tolias, Y. Avrithis, and O. Chum. Label propagation for deep semi-supervised learning. In *Proc. ICCV*, pages 5070–5079, 2019.

T. Iwata, N. Houlsby, and Z. Ghahramani. Active learning for interactive visualization. In C. M. Carvalho and P. Ravikumar, editors, *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31, pages 342–350, 2013.

L. Jing and Y. Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE PAMI*, pages 1–1, 2020.

P. Joia, D. Coimbra, J. A. Cuminato, F. V. Paulovich, and L. G. Nonato. Local affine multidimensional projection. In *Proc. IEEE TVCG*, pages 2563–2571, 2011.

I. T. Jolliffe. *Principal Component Analysis (2nd edition)*. Springer, 1986.

M. C. Jones and R. Sibson. What is projection pursuit? *Journal of the Royal Statistical Society*, 150(1):1–37, 1987.

P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. Supervised contrastive learning. *Proc. NeurIPS*, 33:18661–18673, 2020.

Y. Kim, M. Espadoto, S. Trager, J. Roerdink, and A. Telea. SDR-NNP: Sharpened dimensionality reduction with neural networks. In *Proc. IVAPP*, 2022a.

Y. Kim, A. C. Telea, S. C. Trager, and J. B. Roerdink. Visual cluster separation using high-dimensional sharpened dimensionality reduction. *Inf. Vis.*, 21(3):197–219, 2022b.

D. Kotzias, M. Denil, N. De Freitas, and P. Smyth. From group to individual labels using deep features. In *Proc. ACM SIGKDD*, pages 597–606, 2015.

A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. *tech. rep.*, 2009.

J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

N. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In *Proc. NIPS*, volume 16, pages 329–336, 2003.

Y. LeCun and C. Cortes. MNIST handwritten digit database, 2010. http://yann.lecun.com/exdb/mnist.

D. H. Lee. Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks. In *Proc. ICML-WREPL*, 2013.

D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Proc. NIPS*, volume 13, pages 556–562, 2000.

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.

Y. Li and X. Chao. Semi-supervised few-shot learning approach for plant diseases recognition. *Plant Methods*, 17:1–10, 2021.

J. Lim, D. Ross, R. s. Lin, and M. H. Yang. Incremental learning for visual tracking. In *Proc. NIPS*, volume 17, pages 793–800, 2004.

T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *Proc. ECCV*, pages 740–755, 2014.

P. Liu, H. Zhang, and K. B. Eom. Active deep learning for classification of hyperspectral images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(2):712–724, 2017.

F. P. S. Luus, N. Khan, and I. Akhalwaya. Active learning with tensorboard projector. *CoRR*, abs/1901.00675, 2019. URL http://arxiv.org/abs/1901.00675.

G. F. M. Hopkins, E. Reeber and J. Suermondt. Spambase dataset, 1999.

A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proc. NAACL-HLT*, pages 142–150, 2011.

A. Machado, A. Telea, and M. Behrisch. ShaRP: Shape-regularized multidimensional projections. In *Proc. EuroVA*, 2023.

D. Marghescu. Evaluating the effectiveness of projection techniques in visual data mining. In *Proc. IASTED*, pages 186–193, 2006.

R. Martins, D. Coimbra, R. Minghim, and A. Telea. Visual analysis of dimensionality reduction quality for parameterized projections. *Computers and Graphics*, 41:26–42, 2014.

M. McCann and A. Johnston. Secom dataset, 2008.

L. McInnes, J. Healy, and J. Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *ArXiv e-prints*, February 2018.

L. McInnes, J. Healy, N. Saul, and L. Grossberger. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018.

R. Minghim, F. V. Paulovich, and A. de Andrade Lopes. Content-based text mapping using multi-dimensional projections for exploration of document collections. In *Visualization and Data Analysis 2006*, volume 6060, pages 259–270. SPIE, 2006.

T. Miyato, S. i. Maeda, M. Koyama, and S. Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE PAMI*, 41(8):1979–1993, 2018.

S. Moro, P. Cortez, and P. Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62: 22–31, 2014.

R. Motta, R. Minghim, A. de Andrade Lopes, and M. C. F. Oliveira. Graph-based measures to assist user assessment of multidimensional projections. *Neurocomputing*, 150:583–598, 2015.

T. Munzner. *Visualization Analysis and Design: Principles, Techniques, and Practice*. CRC Press, 2014.

S. A. Nene, S. K. Nayar, H. Murase, et al. Columbia object image library (coil-20). *tech. rep.*, 1996.

Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *Proc. NIPS*, 2011.

L. Nonato and M. Aupetit. Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and layout enrichment. *IEEE Trans. Vis. Comput. Graph*, 2018.

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.

D. Osaku, C. F. Cuba, C. T. Suzuki, J. F. Gomes, and A. X. Falcão. Automated diagnosis of intestinal parasites: a new hybrid approach and its benefits. *Comput. Biol. Med.*, 123:103917, 2020.

E. Packer, P. Bak, M. Nikkilä, V. Polishchuk, and H. J. Ship. Visual analytics for spatial clustering: Using a heuristic approach for guided exploration. *IEEE Trans. Vis. Comput. Graph*, 19(12): 2179–2188, 2013.

J. P. Papa and A. X. Falcão. *A Learning Algorithm for the Optimum-Path Forest Classifier*, pages 195–204. Springer, 2009.

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*, pages 564–575, 2008.

F. V. Paulovich and R. Minghim. Text map explorer: a tool to create and explore document maps. In *Tenth International Conference on Information Visualisation (IV'06)*, pages 245–251. IEEE, 2006.

F. V. Paulovich, D. M. Eler, J. Poco, C. P. Botha, R. Minghim, and L. G. Nonato. Piece wise laplacian-based projection for interactive data exploration and organization. *Computer Graphics Forum*, 30:1091–1100, 2011.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

A. Z. Peixinho, B. C. Benato, L. G. Nonato, and A. X. Falcão. Delaunay triangulation data augmentation guided by visual analytics for deep learning. In *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 384–391, Oct 2018.

H. Pham, Z. Dai, Q. Xie, and Q. V. Le. Meta pseudo labels. In *Proc. CVPR*, pages 11557–11568, June 2021.

T. Rahman, A. Khandakar, Y. Qiblawey, A. Tahir, S. Kiranyaz, S. B. A. Kashem, M. T. Islam, S. Al Maadeed, S. M. Zughaier, M. S. Khan, et al. Exploring the effect of image enhancement techniques on covid-19 detection using chest x-ray images. *Comput. Biol. Med.*, 132:104319, 2021.

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 Conference on Human Factors in Computing Systems (CHI)*, pages 318–322, New York, 1994. ACM Press.

P. Rauber, R. da Silva, S. Feringa, M. Celebi, A. Falcao, and A. Telea. Interactive image feature selection aided by dimensionality reduction. In *Proc. EuroVA*, 2015.

P. E. Rauber, A. X. Falcão, and A. C. Telea. Visualizing time-dependent data using dynamic t-SNE. In *Proc. EuroVis – short papers*, pages 73–77, 2016.

P. E. Rauber, S. G. Fadel, A. X. Falcão, and A. Telea. Visualizing the hidden activity of artificial neural networks. *IEEE Trans. Vis. Comput. Graph*, 23(1), 2017a.

P. Rauber, A. Falcão, and A. Telea. Projections as visual aids for classification system design. *Inf Vis*, 17(4):282–305, 2017b.

P. Ren, Y. Xiao, X. Chang, P. Y. Huang, Z. Li, B. B. Gupta, X. Chen, and X. Wang. A survey of deep active learning. *ACM Comput. Surv.*, 54(9), 2021.

F. C. M. Rodrigues. *Visual Analytics for Machine Learning: Computing and Leveraging Decision Boundary Maps*. PhD thesis, University of Groningen, 2020.

F. C. M. Rodrigues, M. Espadoto, R. H. Jr, and A. Telea. Constructing and visualizing high-quality classifier decision boundary maps. *Information*, 10(9):280–297, 2019.

M. F. C. Rodrigues, R. Hirata, and A. Telea. Image-based visualization of classifier decision boundaries. In *Proc. SIBGRAPI*, pages 353–360, 2018.

S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet large scale visual recognition challenge. *IJCV*, 115(3):211–252, December 2015.

F. S. Samaria and A. C. Harter. Parameterisation of a stochastic model for human face identification. In *Proc. IEEE applications of computer vision*, pages 138–142, 1994.

M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proc. CVPR*, pages 4510–4520, 2018.

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 *2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 53–62, 2012.

B. Schölkopf, A. Smola, and K. R. Müller. Kernel principal component analysis. In *Proc. ICANN*, pages 583–588, 1997.

M. Sedlmair, T. Munzer, and M. Tory. Empirical guidance on scatterplot and dimension reduction technique choices. *IEEE TVCG*, 19(12):2634–2643, 2013.

M. Sedlmair and M. Aupetit. Data-driven evaluation of visual quality measures. *Computer Graphics Forum*, 34:201–210, 2015.

M. Sedlmair, A. Tatu, T. Munzner, and M. Tory. A taxonomy of visual cluster separation factors. *Computer Graphics Forum*, 31: 1335–1344, 2012.

B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

L. Sharan, R. Rosenholtz, and E. Adelson. Material perception: What can you see in a brief glance? *Journal of Vision*, 9(8):784–784, 2009.

W. Shi, Y. Gong, C. Ding, Z. M. Tao, and N. Zheng. Transductive semi-supervised deep learning using min-max features. In *Proc. ECCV*, pages 299–315, 2018.

B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proc. ACM VL*, pages 336–343, 1996.

M. Sikora et al. Application of rule induction algorithms for analysis of data collected by seismic hazard monitoring systems in coal mines. *Archives of Mining Sciences*, 55(1):91–114, 2010.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2014. URL arxiv.org/abs/1409.1556.

V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: From transductive to semi-supervised learning. In *Proc. ICML*, pages 824–831, 2005.

M. Sips, B. Neubert, J. P. Lewis, and P. Hanrahan. Selecting good views of high-dimensional data using class consistency. *Computer Graphics Forum*, 28:831–838, 2009.

C. Sorzano, J. Vargas, and A. Pascual-Montano. A survey of dimensionality reduction techniques, 2014. arXiv:1403.2877.

C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proc. ICCV*, pages 843–852, 2017.

F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

C. Suzuki, J. Gomes, A. Falcão, S. Shimizu, and J.Papa. Automated diagnosis of human intestinal parasites using optical microscopy images. In *Proc. Symp. Biomedical Imaging*, pages 460–463, April 2013.

A. Tatu, G. Albuquerque, M. Eisemann, J. Schneidewind, H. Theisel, M. Magnork, and D. Keim. Combining automated analysis and visualization techniques for effective exploration of high-dimensional data. In *Proc. IEEE VAST*, pages 59–66, 2009.

A. Tatu, P. Bak, E. Bertini, D. Keim, and J. Schneidewind. Visual quality metrics and human perception: an initial study on 2d projections of large multidimensional data. In *Proc. AVI*, pages 49–56, 2010a.

A. Tatu, P. Bak, E. Bertini, D. Keim, and J. Schneidewind. Visual quality metrics and human perception: an initial study on 2d projections of large multidimensional data. In *Proc. AVI*, pages 49–56, 2010b.

Y. Teh and S. Roweis. Automatic alignment of local representations. In *Proc. NIPS*, volume 15, pages 865–872, 2002.

A. Telea. Combining extended table lens and treemap techniques for visualizing tabular data. In *Proc. EuroVis*, pages 51–58, 2006.

A. Telea, A. Machado, and Y. Wang. Seeing is learning in high dimensions – the synergy between dimensionality reduction and machine learning. In *SN Computer Science (Proc. VISIGRAPP 2023)*. Springer, 2024. to appear.

A. C. Telea. *Data Visualization: Principles and Practice, Second Edition*. A. K. Peters, Ltd., Natick, MA, USA, 2nd edition, 2014. ISBN 1466585269, 9781466585263.

J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

Z. Tian, X. Zhai, G. van Steenpaal, L. Yu, E. Dimara, M. Espadoto, and A. Telea. Quantitative and qualitative comparison of 2D and 3D projection techniques for high-dimensional data. *Information*, 12, 2021.

B. Tinsley. Evolution of the stars and gas in galaxies. *Fundamentals of Cosmic Physics*, 5:287–388, 1980.

M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society*, 61(3):611–622, 1999.

W. S. Torgerson. Theory and methods of scaling. *Wiley*, 1958.

E. R. Tufte. *The Visual Display of Quantitative Information, 2^{nd} edition*. Graphics Press, Cheshire, CT, 2001.

J. W. Tukey and P. A. Tukey. Computer graphics and exploratory data analysis: An introduction. *The Collected Works of John W. Tukey: Graphics: 1965-1985*, 5:419, 1988.

L. van der Maaten. Accelerating t-SNE using tree-based algorithms. *J. Mach. Learn. Res.*, 15(1):3221–3245, 2014.

L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *J. Mach. Learn. Res.*, 9:2579–2605, 2008.

L. J. P. van der Maaten, E. O. Postma, and H. J. van den Herik. Dimensionality reduction: A comparative review. *Technical Report TiCC TR 2009-005*, 2009.

J. E. Vargas-Muñoz, P. Zhou, A. X. Falcão, and D. Tuia. Interactive coconut tree annotation using feature space projections. In *Proc. IGARSS*, pages 5718–5721, 2019.

J. Venna and S. Kaski. Visualizing gene interaction graphs with local multidimensional scaling. In *Proc. ESANN*, volume 6, pages 557–562, 2006.

D. Wang and Y. Shang. A new active labeling method for deep learning. In *2014 International joint conference on neural networks (IJCNN)*, pages 112–119. IEEE, 2014.

K. Wang, D. Zhang, Y. Li, R. Zhang, and L. Lin. Cost-effective active learning for deep image classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(12):2591–2600, Dec 2017.

Y. Wang, A. Machado, and A. Telea. Quantitative and qualitative comparison of decision map techniques for explaining classification models. *Algorithms*, 16(9), 2023.

Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Comput. Surv.*, 53(3):1–34, 2020.

K. Weinberger, B. Packer, and L. Saul. Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization. In *AISTATS*, pages 381–388, 2005.

L. Wilkinson and G. Wills. Scagnostics distributions. *Journal of Computational and Graphical Statistics*, 17(2):473–491, 2008.

L. Wilkinson, A. Anand, and R. Grossman. Graph-theoretic scagnostics. In *Proc. IEEE InfoVis*, pages 21–21, 2005a.

L. Wilkinson, A. Anand, and R. L. Grossman. Graph-theoretic scagnostics. In *Proc. IEEE InfoVis*, volume 5, page 21, 2005b.

H. Wu and S. Prasad. Semi-supervised deep learning using pseudo labels for hyperspectral image classification. *IEEE TIP*, 27(3):1259–1270, 2018.

R. Wu, N. Das, S. Chaba, S. Gandhi, D. H. Chau, and X. Chu. A cluster-then-label approach for few-shot learning with application to automatic image data labeling. *J. Data and Information Quality*, 14(3), may 2022. ISSN 1936-1955.

H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv:1708.07747*, 2017.

H. Xie, J. Li, and H. Xue. A survey of dimensionality reduction techniques based on random projection, 2017. arXiv:1706.04371.

F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao. LSUN: construction of a large-scale image dataset using deep learning with humans in the loop. *CoRR*, abs/1506.03365, 2015. URL http://arxiv.org/abs/1506.03365.

Z. Yu, L. Chen, Z. Cheng, and J. Luo. Transmatch: A transfer-learning scheme for semi-supervised few-shot learning. In *CVPR*, June 2020.

M. D. Zeiler and R. Fergus. *Visualizing and Understanding Convolutional Networks*, pages 818–833. Springer International Publishing, Cham, 2014.

X. Zhai, L. Yu, X. Chen, and A. Telea. Skeleton-and-trackball interactive rotation specification for 3D scenes. In *Communication in Computer and Information Sciences*, volume 1474, pages 26–52. Springer, 2022.

X. Zhai, A. Oliver, A. Kolesnikov, and L. Beyer. S4l: Self-supervised semi-supervised learning. In *Proc. ICCV*, pages 1476–1485, 2019.

T. Zhang, J. Yang, D. Zhao, and X. Ge. Linear local tangent space alignment and application to face recognition. *Neurocomput*, pages 1547–1553, 2007.

Z. Zhang and J. Wang. Mlle: Modified locally linear embedding using multiple weights. In *Proc. NIPS*, volume 19, pages 1593–1600, 2006.

Z. Zhang and H. Zha. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM journal on scientific computing*, 26(1):313–338, 2004.

A. Zhmoginov, M. Sandler, and M. Vladymyrov. Hypertransformer: Model generation for supervised and semi-supervised few-shot learning. In *International Conference on Machine Learning*, pages 27075–27098. PMLR, 2022.

S. Zhou, Q. Chen, and X. Wang. Active deep learning method for semi-supervised sentiment classification. *Neurocomputing*, 120: 536 – 546, 2013.

Z. Zhou, X. Zu, Y. Wang, B. F. Lelieveldt, and Q. Tao. Deep recursive embedding for high-dimensional data. *IEEE TVCG*, pages 1237–1248, 2022. ISSN 1941-0506.

H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2): 265–286, 2006.

Q. Zou, L. Ni, T. Zhang, and Q. Wang. Deep learning based feature selection for remote sensing scene classification. *IEEE Geoscience and Remote Sensing Letters*, 12(11):2321–2325, Nov 2015.

ACKNOWLEDGMENTS

First, I would like to thank my supervisors: prof. Alex Telea, for all thorough and didactic guidance, supervision, and patience, beyond your always positive insights, feedback, and cachaça preference; and Prof. Alexandre Falcão, for supporting my academic and professional maturity during so many years, constantly raising interesting and differing ideas. You both certainly form a complementary match. It is a privilege having you as supervisors. I can not measure how much I have learned from you.

To members of the assessment committee, thank you for accepting the task of reviewing this manuscript and for your presence at the defense. I sincerely acknowledge your effort in those.

To the UNICAMP and UU staff, for institutional, material, legal, and human help during my joint PhD.

To my LIDS friends, for being a special part of my professional and personal life with me. Thank you for discussions, presentations, hints, feedbacks, coffees and teas. Specially, those I have received most support during those years: Léo, Belém, Samuel, Saulo, Matheus, and Gui. There are much of you with(in) me, guys!

To my UU colleagues, professors, and friends in the Netherlands, thank you for sharing your knowledge and culture, hosting me out of my home (university), and all the nice moments together. I especially thank Alister, Daga, and Heloysa for this amazing year and Carlijne and Cristian for our work together.

Last but certainly not least, I thank my family for endless love, support, care, and example in life. Thank you for teaching me how to be, integrate, and mind about ourselves and other people. To my parents, Julio and Hélen. To my grandmother, Lurdinha, and especially to my grandparents in lovely memory, Mauro, Bela, and Eustásio. To my aunts, Patrici, Helena, Valéria, and uncle João Paulo – and their partners. To my cousins, Gustavo, Bento, and Mateo. To my little brother, Mateus.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, and by grants #2019/10705-8 and #2022/12668-5, São Paulo Research Foundation (FAPESP).

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 6, 2024 (`classicthesis`).