EuroVis Workshop on Visual Analytics (2019)
C. Turkay and T. von Landesberger (Editors)

Deep Learning Inverse Multidimensional Projections

M. Espadoto, F. C. M. Rodrigues[1,2], N. S. T. Hirata, R. Hirata, Jr.[2] † and A. C. Telea[1]

[1]Johann Bernoulli Institute, Faculty of Science and Engineering, University of Groningen, The Netherlands
[2]Institute of Mathematics and Statistics, University of São Paulo, Brazil

Abstract

We present a new method for computing inverse projections from 2D spaces to arbitrary high-dimensional spaces. Given any projection technique, we train a deep neural network to learn a low-to-high dimensional mapping based on a projected training set, and next use this mapping to infer the mapping on arbitrary points. We compare our method with two recent inverse projection techniques on three datasets, and show that our method has similar or higher accuracy, is one to two orders of magnitude faster, and delivers result that match well known ground-truth information about the respective high-dimensional data.
Visual analytics Unsupervised learning Dimensionality reduction and manifold learning

CCS Concepts

• ***Visualization*** → *Visualization application domains;* • ***Machine learning*** → *Learning paradigms;*

1. Introduction

Multidimensional projections take a high-dimensional dataset $D = \{\mathbf{x}_i\}$ consisting of samples $\mathbf{x} \in \mathbb{R}^n$ and generate a (typically two-dimensional) scatterplot $P(D) = \{\mathbf{y}_i\} \subset \mathbb{R}^2$, $P(\mathbf{x}) = \mathbf{y}$. Projections are frequently used in visual analytics to examine the structure of high-dimensional data to find outliers, help data clustering, and assist classifier engineering, with tens of available algorithms [NA18, vP09]. Inverse projections, aim the opposite task: Given a dataset D and its 2D projection $P(D) \subset \mathbb{R}^2$, take *any* point $\mathbf{y} \in \mathbb{R}^2$ (not necessarily part of $P(D)$) and infer a high-dimensional point $\mathbf{x} \in \mathbb{R}^n$ (not necessarily part of D) that the direct projection function P would have projected, if given, to $\mathbf{y}$, *i.e.*, $P(\mathbf{x}) = \mathbf{y}$. We denote an inverse projection by P^{-1}, thus, in the above explanation, $P^{-1}(\mathbf{y}) = \mathbf{x}$.

Computing inverse projections has received recently increased importance in visual analytics and beyond. The iLAMP [ABD*12] and Radial Basis Functions (RBF) [ABMC*15] inverse projections were used to interpolate between high-dimensional observations $\mathbf{x}$ so that these vary smoothly in a given projection, for 3D shape morphing applications. The same methods were adapted to create space-filling dense maps of decision zones of classifiers to help understanding their behavior in machine learning engineering [ERT19, RJT18]. Dense maps are prominently featured in the well-known TensorFlow framework for classifier engineering [SC19]. However, computing inverse projections is hard: Algorithms like iLAMP and RBF are slow, have multiple free parameters, and their quality strongly depends on the dataset D and direct projection technique P being used [ERT19]. Inverse projections in TensorFlow can only handle 2D datasets $D \subset \mathbb{R}^2$.

To address the above, we propose a new way to compute inverse projections: Given a dataset D and projection $P(D)$, computed by any desired, user-chosen, technique P, we *learn* the inverse mapping $P(D) \to D$ using deep learning. Next, we use the learned mapping to project any sample $\mathbf{y} \in \mathbb{R}^2$ to nD. We validate the accuracy, speed, and ease of use of our technique using both quantitative quality metrics and dense maps on a couple of real-world datasets, and compare our results with iLAMP and RBF.

2. Method

We start with a dataset $D \subset \mathbb{R}^n$ and a projection technique P. Both can be freely chosen by users depending *e.g.* on their application of interest and the features that P should manifest, *e.g.*, good cluster segregation, distance preservation, or any other known quality metrics [NA18, vP09, XLX17, CG15]. We hypothesize that the way in which P captures the data structure in D can be used to create an inverse projection P^{-1} by using a small training set $D_s \subset D$ and its respective projection $P(D_s) \subset P(D)$. We next construct P^{-1} by training a neural network on the training set $T_s = (D_s, P(D_s))$, with D_s selected by random sampling of D. We use the remaining data $T_p = (D \setminus D_s, P(D) \setminus P(D_s))$, unseen during training, for validation. The cost function aims to generate samples in D that are as close as possible to the training ones in D_s. Summarizing, our method has three steps: In step 1, we create the projection $P(D_s)$ of the training samples D_s using any desired projection technique P. In step 2, we train a neural network using the training set T_s. In step 3, we validate the trained network using the test set T_p. The trained network is our inverse projection P^{-1}. For any given 2D point $\mathbf{y}$, we can now infer its high-dimensional counterpart by $P^{-1}(\mathbf{y})$.

After extensive empirical testing, varying the number of layers, neurons per layer, and activation functions, we set the architecture of

† This study was financed in part by FAPESP (2017/25835-9) and the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001

P^{-1} to four fully-connected hidden layers, with 2048 units each, using ReLU activation functions, followed by an n-element layer, which uses a sigmoid activation to encode the inverse projection, scaled to the interval $[0,1]$ for implementation simplicity – that is, we assume that our high-dimensional data resides in $[0,1]^n$ instead of $\mathbb{R}^n$. We initialize weights with the He uniform-variance scaling initializer [HZRS15], and bias elements by a constant value 0.01, which showed good results during testing. We use the Adam [KB14] optimizer to *train* P^{-1} for up to 300 epochs. We stop training automatically on convergence, defined as the moment when the validation loss stops decreasing. In practice, we need 150 epochs on average for convergence (see Sec. 3.1). As cost function, we use mean squared error, which showed better convergence speed during testing than mean absolute error and log hyperbolic cosine (logcosh). To test quality, we compare the nD inferred samples $P^{-1}(D_p)$ with ground truth D_p using the mean squared error metric.

3. Results

We tested our method on the following materials:

Projections: We use for P t-SNE [vdMH08] and UMAP [MH18], which have high-quality and are well known in the dimensionality reduction community [NA18]. We also tested other methods such as PCA and LAMP [JCC*11], with similar results, omitted here for space constraints.

Inverse projections: We compare our method with two alternatives: iLAMP [ABD*12] and RBF [ABMC*15]). Besides PCA, these are the only inverse projection methods we are aware of. PCA shows poor results as both direct and inverse projections for data of high intrinisic dimensionality, so we omit this from the presentation.

Datasets: We use one synthetic dataset and two well-known real-world benchmark datasets in machine learning. The synthetic dataset (*Blobs*) has 60K observations sampled from a Gaussian distribution with 5 different centers (clusters) and 50 dimensions. The *MNIST* dataset [LCB10] has 70K observations of handwritten digits from 0 to 9, rendered as 28×28-pixel grayscale images, flattened to 784-element vectors. The *Fashion MNIST* dataset [XRV17] has 70K observations of 10 types of pieces of clothing, rendered as 28x28-pixel grayscale images, flattened to 784-element vectors.

We next discuss our method in terms of scalability (Sec. 3.1), quantitative assessment of quality (Sec. 3.2), and qualitative assessment of quality (Sec. 3.3)

3.1. Scalability in training and inference

Scalability implies the effort required to *train* our method and, separately, the effort needed to *infer* $P^{-1}(Y)$ as function of the size $|Y|$ of the dataset Y to inversely project. Table 1 shows the number of training epochs needed to obtain convergence (defined as in Sec. 2) as function of the training set size $|D_s|$, for all three considered datasets and P = t-SNE. The figures for other projections (UMAP, PCA) are very similar. Columns 2..4 indicate averages for multiple runs that select D_s by randomly sampling D (see Sec. 2). Overall, we see that we obtain convergence for roughly 150 epochs for all datasets and training-set sizes, and also that this number of epochs is quite stable for training-set sizes $|D_s|$ larger than 1K samples.

Figure 1 shows the inference speed for all three datasets. Note that speed does not depend on the projection method P, by construction. Also, in this experiment, we consider *any* point $\mathbf{y} \in \mathbb{R}^2$, *i.e.*, not only

Table 1: Training effort until convergence.

	Average # epochs for each dataset D			
training set size $\lvert D_s \rvert$	Blobs	Fashion-MNIST	MNIST	Avg.
500	268.0	214.0	213.5	192.5
1000	190.5	129.0	147.5	149.0
2000	153.0	112.0	111.0	112.5
5000	103.0	120.5	138.0	127.5
7000	127.0	118.5	151.0	144.0
10000	82.0	124.5	142.5	146.5
average $\lvert D_s \rvert$ per D	153.9	136.4	150.6	145.3

points in the test-set D_s, since we don't need ground truth information to assess speed, and since in actual use one would not have such ground truth available. We see that both RBF and iLAMP have a superlinear behavior, while iNN (our method) is almost linear. More importantly, iNN is roughly one magnitude order faster than RBF and nearly two orders of magnitude faster than iLAMP for 40K samples or more. This speed-up is crucial for applications that need to inversely project hundreds of thousands of samples (or more), like in the construction of dense maps (see [ERT19, RJT18] and Sec. 3.3 next). In such cases, iNN allows constructing such maps in seconds, whereas iLAMP and RBF require (tens of) minutes, which makes human-in-the-loop usage of such dense maps impossible in visual analytics scenarios – which are one of the key reasons why dense maps are built in the first place.

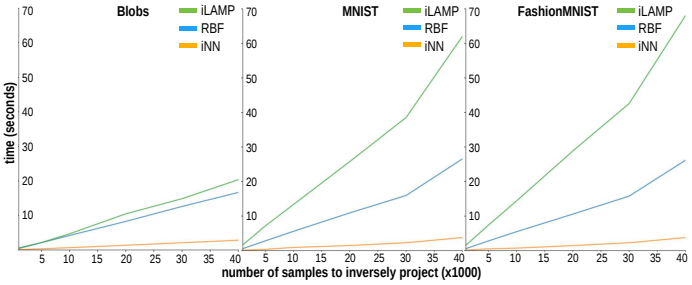

Figure 1: Inverse projection speed as function of number of samples.

3.2. Quantitative Assessment of Quality

Besides being fast, we want an inverse projection to be *accurate*. That is, given some ground truth pair $(\mathbf{x} \in \mathbb{R}^n, \mathbf{y} = P(\mathbf{x}) \in \mathbb{R}^2)$, *unseen* by training, we want that $P^{-1}(\mathbf{y})$ be as close as possible to $\mathbf{x}$. This follows the same idea as, on the one hand, normalized stress metrics used to gauge the quality of projections in the literature [SVPM14, vP09], and on the other hand classical validation of inference models in machine learning. We measure quality in our case by computing the average inverse-projection mean square error $MSE = \|\mathbf{x} - P^{-1}(P(\mathbf{x}))\|^2 / |D_p|$ over the test set D_p. The closer MSE is to zero, the better P^{-1} is. Figure 2 shows MSE for our three datasets, two projections (t-SNE and UMAP), three tested inverse projections (iLAMP, RBF, and iNN). We also consider several training-set sizes $|D_s|$ to show how MSE depends on the training amount. For *Blobs*, a relatively easy-to-project synthetic data, all methods have basically zero error, except RBF. *MNIST* and *FashionMNIST* show similar behavior: Our method (iNN) achieves consistently lowest error. The second-best method is iLAMP. Errors are larger for these real-world complex datasets than for the synthetic *Blob*, which is expected.

3.3. Qualitative Assessment of Quality

We now show why having a fast and accurate inverse projection is important for a concrete application – understanding the decision

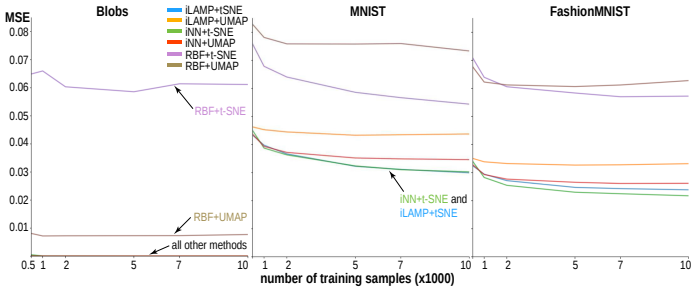

Figure 2: Mean square error of inverse projection (lower=better).

zones of classifiers. In supervised machine learning, a classifier C is trained on a *labeled* training-set D_s, which typically sparsely samples some continuous universe U ($U \subset \mathbb{R}^n$ in our case), and partitions U in so-called *decision zones* Z_i, so that any $\mathbf{x} \in Z_i$ gets label l_i at inference time. The classifier accuracy is evaluated using a labeled test-set D_p. Visualizing the decision zones Z_i helps understanding the behavior of C, as well as areas in U where training may have been suboptimal and thus may need extra effort. Recently, a method to construct dense decision maps using inverse projections was proposed [RJT18]. Briefly put, every pixel $\mathbf{y}$ in an image is colored by the label assigned to $P^{-1}(\mathbf{y})$ by a classifier C trained on the same data used to construct the projection P from which P^{-1} was derived. The quality of such decision maps depends on the projection P used, with t-SNE and UMAP performing better than several other tested methods [ERT19]. However, while acknowledging its importance, the effect on quality of the *inverse* projection P^{-1} used was not tested. We alleviate the above limitation in [ERT19] and also show why our inverse projection achieves better results for dense maps visualizing classifier decision zones. For this, we construct decision maps for projections $P \in \{\text{tSNE}, \text{UMAP}\}$, datasets $D \in \{\text{Blobs}, \text{MNIST}, \text{FashionMNIST}\}$, inverse projections $P^{-1} \in \{\text{iLAMP}, \text{RBFp}, \text{RBFc}, \text{iNN}\}$, and classifiers $C \in \{\text{LR}, \text{CNN}\}$. Here, RBFp and RBFc are two versions of the RBF inverse projection, using fixed control points, respectively control points defined as centers of clusters obtained from the input data D (for details, we refer to the original paper [ABMC*15]). *LR* is a simple logistic regression classifier, used since we know it produces piecewise-linear decision boundaries and hyperpolyhedral decision zones; and CNN is a convolutional neural network, which we know it works well for image data like (Fashion)MNIST. All decision maps are images of 500^2 pixels, so $|D_p| = 250000$ points (Fig. 3). Importantly, all maps were constructed completely from *unseen* data – that is, we do not use any of the data points or their projections present in the training set D_s. We discuss our results next.

Blobs dataset: As expected, for this simple dataset, both t-SNE and UMAP separate well the 5 clusters present in the data. The LR trained on this dataset achieved 100% accuracy. All inverse projections P^{-1} appear as compact zones that surround the corresponding projection scatterplots. For the LR classifier, we *know* that the decision boundaries should be piecewise linear. UMAP yields more concentrated clusters, so the corresponding dense maps resemble very much Voronoi diagrams of the respective cluster configurations – which is indeed expected, and a positive sign of the correctness of the dense maps. For the t-SNE projection, iLAMP and iNN are closest to such linear boundaries, while RBFp and RBFc create more jagged boundaries. This is a first hint that iLAMP and iNN are better inverse projections.

MNIST dataset: The CNN classifier used obtained a 99.6% training-set accuracy. As the projection (and underlying dataset) is more complex, the inverse projections are more challenged. Recent studies have empirically shown that decision zones of such neural networks, used for natural-image dataset classification, are *connected*, with relatively *smooth* boundaries [FMDFS18]. Hence, we *expect* our dense maps to show this. In Fig. 3, we first observe that both iLAMP and iNN are closest to the above properties, while RBFc generates highly noisy, sprayed-points-like, disconnected, and complex-shaped decision zones (see dashed-line annotations in figure). These generate the false impression that the classifier has difficulties for such samples, which is not true, given the observed accuracy. RBFp also generates noisy/disconnected zones, albeit less than RBFc, but more than iLAMP and iNN. Both RBFp and RBFc also generate visible 'false islands', *i.e.*, significant-size areas in the decision maps that have a label which does not match any significant number of points having the same label in the scatterplots (see continuous-line annotations in figure). These convey the false impression that the classifier creates certain decision zones in areas where actually nothing like this happens. While both above phenomena exist also for iNN, this is to far smaller extents.

FashionMNIST dataset: The CNN classifier used obtained a 98.7% training-set accuracy. We can make the same observations made for MNIST's decision zones, even to stronger extents. RBFc and RBFp generate highly fragmented, jagged, and disconnected decision zones, with RBFp being better than RBFc. iLAMP and iNN generate smoother, more connected, and quite similar zones. This is quite interesting, since the two methods are completely different. However, iLAMP generates noisier zones and more jagged boundaries (see annotations in figure). Given, again, the mentioned insights on how such zones/boundaries should be [FMDFS18], we find iNN being better than iLAMP.

4. Discussion and Conclusion

We have presented a new method for computing inverse projections from 2D to high-dimensional data spaces by learning the behavior of a direct projection method. Our method is generic (can handle any direct projection method and type of high-dimensional dataset), automatic (does not require any user parameters), one to two orders of magnitude faster than existing inverse projection methods, and simple to implement using existing out-of-the-box deep learning toolkits [C*15]. We compared our method on three datasets, two state-of-the-art projections (UMAP and t-SNE), against three inverse projection methods (iLAMP, RBFc, and RBFp). We found our method to deliver higher accuracy, and decision zones that match equally well or better to known properties of such zones for both simple (linear regression) and more complex (convolutional neural network) classifiers.

Our method can be extended in several directions. First, the design space of its underlying neural network can be better explored to reach higher accuracy and/or less training effort. Secondly, different quality metrics can be used to deliver inverse projections which are specifically suited for specialized tasks such as assessing confusion zones of classifiers. Finally, we can apply our inverse projection to support more applications beyond decision map exploration in machine learning.

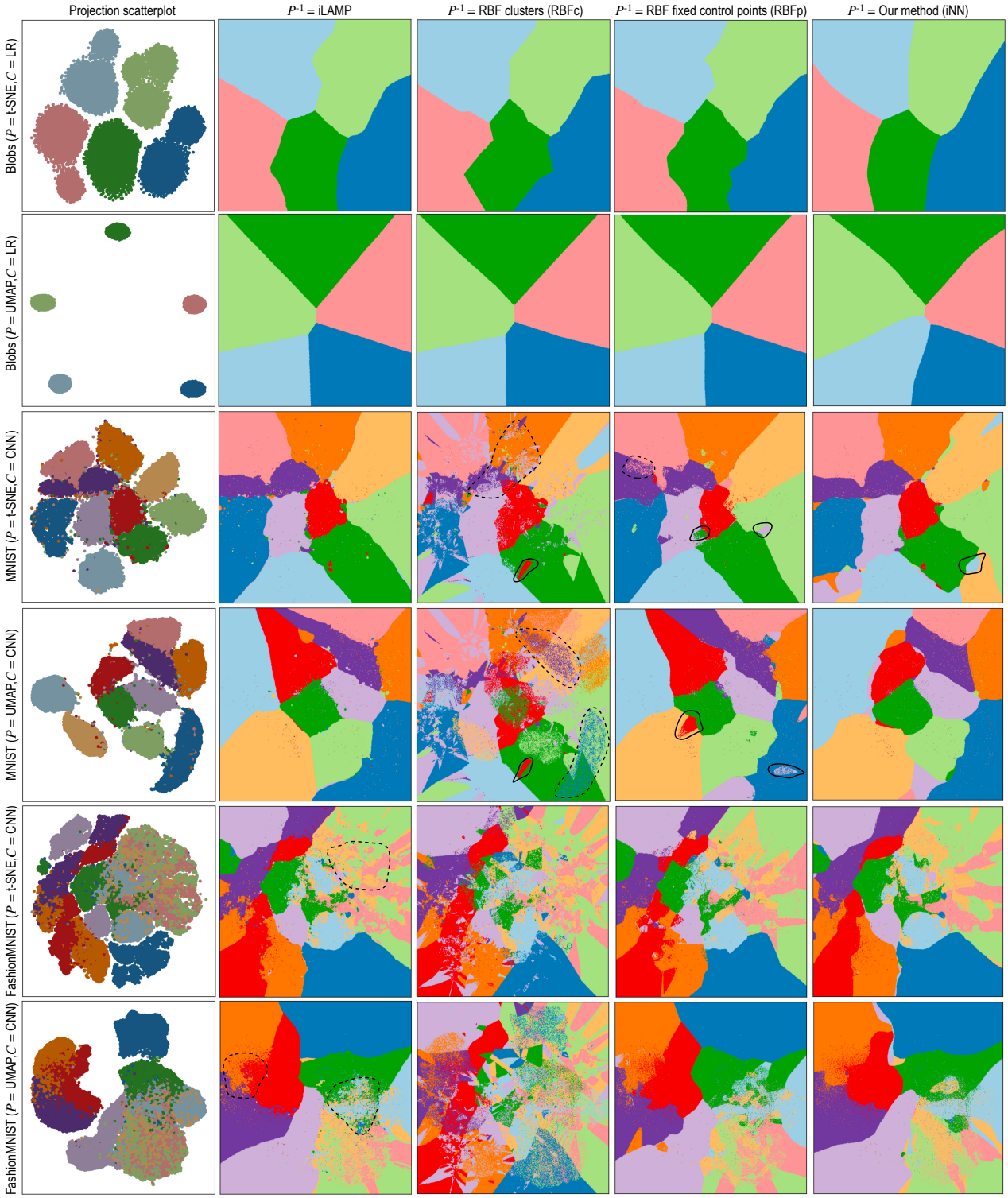

Figure 3: Dense maps constructed for combinations of classifiers C, projections P, inverse projections P^{-1}, and datasets. See Sec. 3.3.

References

[ABD*12] AMORIM E., BRAZIL E. V., DANIELS J., JOIA P., NONATO L. G., SOUSA M. C.: iLAMP: Exploring high-dimensional spacing through backward multidimensional projection. In *Proc. IEEE VAST* (2012), pp. 53–62. 1, 2

[ABMC*15] AMORIM E., BRAZIL E. V., MENA-CHALCO J., VELHO L., NONATO L. G., SAMAVATI F., SOUSA M. C.: Facing the high-dimensions: Inverse projection with radial basis functions. *Computers & Graphics 48* (2015), 35–47. 1, 2, 3

[C*15] CHOLLET F., ET AL.: Keras. https://keras.io, 2015. 3

[CG15] CUNNINGHAM J., GHAHRAMANI Z.: Linear dimensionality reduction: Survey, insights, and generalizations. *JMLR 16* (2015), 2859–2900. 1

[ERT19] ESPADOTO M., RODRIGUES F., TELEA A.: Visual analytics of multidimensional projections for constructing classifier decision boundary maps. In *Proc. IVAPP* (2019), SciTePress. 1, 2, 3

[FMDFS18] FAWZI A., MOOSAVI-DEZFOOLI S.-M., FROSSARD P., SOATTO S.: Empirical study of the topology and geometry of deep networks. In *Proc. IEEE CVPR* (2018), pp. 3762–3770. 3

[HZRS15] HE K., ZHANG X., REN S., SUN J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. IEEE ICCV* (2015), pp. 1026–1034. 2

[JCC*11] JOIA P., COIMBRA D., CUMINATO J. A., PAULOVICH F. V., NONATO L. G.: Local affine multidimensional projection. *IEEE TVCG 7*, 12 (2011), 2563–2571. 2

[KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). 2

[LCB10] LECUN Y., CORTES C., BURGES C.: MNIST handwritten digit database. *AT&T Labs [Online]. Available: http://yann.lecun.com/exdb/mnist 2* (2010). 2

[MH18] MCINNES L., HEALY J.: UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (2018). 2

[NA18] NONATO L., AUPETIT M.: Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and layout enrichment. *IEEE TVCG* (2018). 1, 2

[RJT18] RODRIGUES F., JR. R. H., TELEA A.: Image-based visualization of classifier decision boundaries. In *Proc. SIBGRAPI* (2018). 1, 2, 3

[SC19] SMILKOV D., CARTER S.: TensorFlow Playground, 2019. playground.tensorflow.org. 1

[SVPM14] SORZANO C., VARGAS J., PASCUAL-MONTANO A.: A survey of dimensionality reduction techniques, 2014. arXiv:1403.2877 [stat.ML]. 2

[vdMH08] VAN DER MAATEN L., HINTON G. E.: Visualizing data using t-SNE. *JMLR 9* (2008), 2579–2605. 2

[vP09] VAN DER MAATEN L., POSTMA E.: *Dimensionality Reduction: A Comparative Review*. Tech. rep., Tilburg University, Netherlands, 2009. Tech. report TiCC TR 2009-005. 1, 2

[XLX17] XIE H., LI J., XUE H.: A survey of dimensionality reduction techniques based on random projection, 2017. arXiv:1706.04371 [cs.LG]. 1

[XRV17] XIAO H., RASUL K., VOLLGRAF R.: Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms, 2017. arXiv 1708.07747 [cs.LG]. 2