algorithms

Article

Quantitative and Qualitative Comparison of Decision-Map Techniques for Explaining Classification Models

Yu Wang, Alister Machado and Alexandru Telea *

Department of Information and Computing Science, Utrecht University, 3584 CS Utrecht, The Netherlands; y.wang6@uu.nl (Y.W.); a.machadodosreis@uu.nl (A.M.)
* Correspondence: a.c.telea@uu.nl

Abstract: Visualization techniques for understanding and explaining machine learning models have gained significant attention. One such technique is the decision map, which creates a 2D depiction of the decision behavior of classifiers trained on high-dimensional data. While several decision map techniques have been proposed recently, such as Decision Boundary Maps (DBMs), Supervised Decision Boundary Maps (SDBMs), and DeepView (DV), there is no framework for comprehensively evaluating and comparing these techniques. In this paper, we propose such a framework by combining quantitative metrics and qualitative assessment. We apply our framework to DBM, SDBM, and DV using a range of both synthetic and real-world classification techniques and datasets. Our results show that none of the evaluated decision-map techniques consistently outperforms the others in all measured aspects. Separately, our analysis exposes several previously unknown properties and limitations of decision-map techniques. To support practitioners, we also propose a workflow for selecting the most appropriate decision-map technique for given datasets, classifiers, and requirements of the application at hand.

Keywords: decision boundary; classification; dimension reduction; inverse projection; visual analytics

Citation: Wang, Y.; Machado, A.; Telea, A. Quantitative and Qualitative Comparison of Decision-Map Techniques for Explaining Classification Models. *Algorithms* **2023**, *16*, 438. https://doi.org/10.3390/a16090438

Academic Editors: Mario Rosario Guarracino, Laura Antonelli and Pietro Hiram Guzzi

Received: 30 July 2023
Revised: 5 September 2023
Accepted: 9 September 2023
Published: 11 September 2023

1. Introduction

Machine learning (ML) has become a dominant force in various fields, leading to many applications in healthcare, finance, and autonomous driving, and advancing scientific research in areas such as geoscience, biology, and physics [1–4]. However, as ML models become more complex and are increasingly used to make critical decisions, their lack of interpretability becomes a significant issue. The 'black box' nature of many such models, particularly deep learning ones, can lead to mistrust and misuse of the models, as users may not understand how the model is making its decisions [5,6]. This lack of transparency can also hinder troubleshooting and the improvement of the models [7,8]. To address this issue, researchers have proposed *interpretable* machine learning and visualization frameworks that aim to provide insights into the inner workings of these models [9–12].

Decision maps are one such visualization framework. Simply put, decision maps are two-dimensional (2D) depictions of the way a classification model splits its high-dimensional feature space into areas that are assigned the same label (so-called decision zones), separated by so-called decision boundaries. Decision boundaries play a pivotal role not only in model interpretation but also in contexts like active learning, decision making, domain adaptation, and adversarial attacks [13–16]. Several techniques exist to visualize decision boundaries and construct such decision maps for arbitrary classifiers [17–22], the most notable being Decision Boundary Maps (DBMs) [21], Supervised Decision Boundary Maps (SDBMs) [22], and DeepView (DV) [20]. These techniques offer different perspectives on the decision-making process of a classifier, allowing users to gain a better understanding of the model's behavior. These techniques have found applications in areas like model steering [23], detecting backdoor attacks [20], and even interpreting geoscience models [24].

Overall, these techniques take away some of the complexity of interpreting the working of a machine learning model while depicting the functioning of the model in more detailed ways than classical aggregate performance metrics. As such, decision-map techniques are particularly attractive for users of machine learning who are not domain experts in the operation of the underlying ML algorithms.

Despite their growing popularity, there has been no in-depth assessment of the quality of decision maps [22]. From a technical perspective, it is unclear how effectively different decision-map techniques capture information present in high-dimensional decision zones and boundaries. For instance, it is unclear whether the smoothness or fragmentation visible in decision boundaries truly depicts the same properties of the actual, high-dimensional boundaries. Similarly, it is unclear whether a sample located close to a decision boundary in the map genuinely reflects its proximity to the high-dimensional point where a classifier changes output. From a practical perspective, this lack of evaluation makes it hard to select the most suitable decision-map technique for a given dataset and classifier.

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

Summarizing the above, our key contributions are as follows:

- We propose a suite of metrics to quantitatively evaluate decision-map techniques.
- We conduct a comprehensive comparison of existing decision-map techniques, both quantitatively (by comparing the aforementioned metrics) and qualitatively (by visually comparing the obtained decision-map images).
- We propose a workflow to guide the selection of the most suitable decision-map technique for a given dataset based on a set of desirable requirements.

2. Related Works

2.1. Preliminaries

We start by introducing a few notations. Let $D = \{\mathbf{x}_i\} \subset \mathbb{R}^n$, $1 \leq i \leq N$ be a dataset of n-dimensional data points (samples) $\mathbf{x}_i = (x_i^1, x_i^2, \ldots, x_i^n)$, with the corresponding labels $y_i \in C$. Let $\mathbf{x}^j = \{x_1^j, x_2^j, \cdots, x_N^j\}$, $1 \leq j \leq n$ be the j-th feature, or dimension, of D. Thus, D can be seen as a table with N rows (samples) and n columns (dimensions), with $\mathbf{y} = \{y_1, y_2, \ldots, y_N\}$ being their label vector. Given a dataset D and its corresponding labels $\mathbf{y}$, a *classifier* constructs a function $f \colon \mathbb{R}^n \to C$ so that $f(\mathbf{x}_i) = y_i$ ideally for all $\mathbf{x}_i \in D_t$, where $D_t \subseteq D$ is the training set. After training, the classifier f is typically further evaluated in the same way on a test set $D_T \subset D$, $D_T \cap D_t = \emptyset$. After training and testing, the classifier f can be used to predict the labels of new samples $\mathbf{x} \in \mathbb{R}^n \setminus D$.

A dimensionality reduction (DR) method, also called a *projection*, is a function $P : \mathbb{R}^n \to \mathbb{R}^q$ that maps an n-dimensional sample $\mathbf{x} \in \mathbb{R}^n$ to a q-dimensional sample $P(\mathbf{x}) \in \mathbb{R}^q$, where $q \ll n$. Typically, one uses $q = 2$ or 3 for visualization purposes. An *inverse projection* $P^{-1} : \mathbb{R}^q \to \mathbb{R}^n$, also called backprojection or unprojection [25], is a function that maps a q-dimensional sample $\mathbf{z} \in \mathbb{R}^q$ to an n-dimensional sample $\mathbf{x} \in \mathbb{R}^n$ so that it approximates the inverse of P, i.e., $P^{-1}(P(\mathbf{x})) \approx \mathbf{x}$. We next denote as $P(D)$ the application of P to all points of a dataset D (and analogously for P^{-1}).

2.2. Overall Workflow of Decision Map

Given a classifier f, a *decision boundary* is a surface in $\mathbb{R}^n$ that separates high-dimensional data points $\mathbf{x} \in \mathbb{R}^n$ into regions, also called *decision zones*. All points in a given zone are assigned the same label $y \in C$ by the model f. Decision zones are separated by *decision boundaries*, which are hypersurfaces embedded in $\mathbb{R}^n$, where the classifier f changes the output. Understanding how the high-dimensional space is partitioned into such decision zones and how data samples in a training or test set are distributed across the zones effectively helps understand how a classifier behaves [20,21]. For example, seeing how labeled samples distribute close to decision boundaries can help categorize misclassification problems, and seeing how unlabeled samples (whose class is to be predicted by f) spread across decision zones can help understand how well f can handle a given data distribution.

Decision-map algorithms are techniques that aim to create 2D representations of the decision zones and boundaries of any classifier f [20–22]. Their general workflow is as follows (see also Figure 1):

1. Train a classifier f on a dataset D_t. This is the classifier whose decision map we next want to visualize.
2. Construct a direct projection P and inverse projection P^{-1} using a dataset D'.
3. Project D' to create a 2D scatter plot $P(D')$.
4. Sample the extent of $P(D')$ on a uniform pixel grid I.
5. Backproject all pixels $\mathbf{y} \in I$ to the data space using P^{-1}.
6. Use f to predict the labels of the backprojected points $P^{-1}(\mathbf{y})$.
7. Color I according to the predicted labels $f(P^{-1}(\mathbf{y}))$.

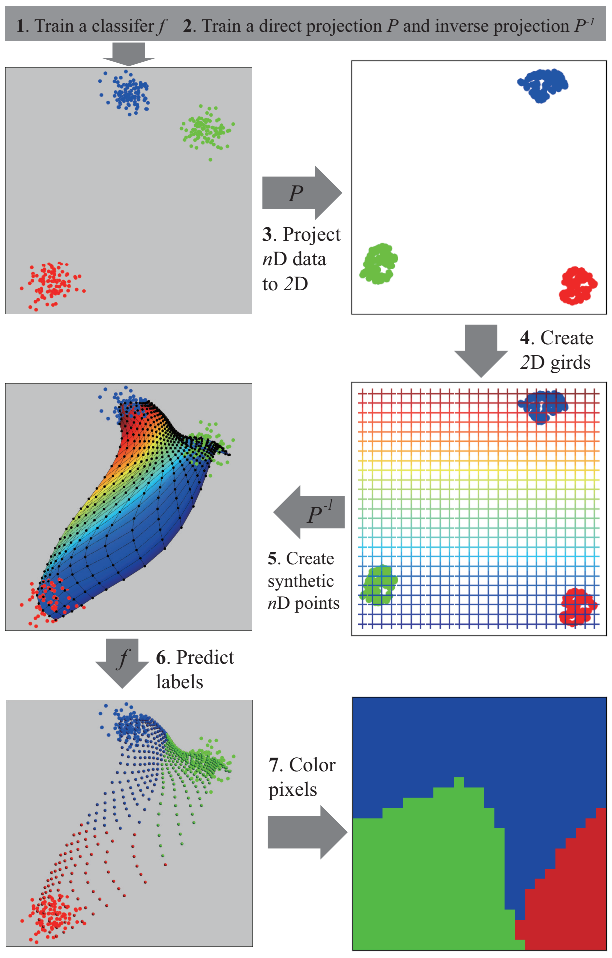
1. Train a classifer f
2. Train a direct projection P and inverse projection P^{-1}
P
3. Project nD data to 2D
4. Create 2D girds
P^{-1}
5. Create synthetic nD points
f
6. Predict labels
7. Color pixels

Figure 1. General workflow of decision-map techniques (see Section 2.2).

In the above, D' can be, at a minimum, a subset of the training set D_t. However, if more samples of the target domain are available, such as in the form of test data D_T or even unlabeled data, these can be added to D'. By doing this, the constructed decision map can better sample the actual data distribution of the investigated phenomenon and thus the classifier's behavior. In this paper, we set $D' = D_t$ following earlier examples of this workflow [22].

We next describe three specific techniques that implement the above workflow and introduce some of their key features.

2.3. Decision Boundary Maps (DBMs)

Decision Boundary Maps (DBMs) [21] closely follow the above workflow. While the original DBM used t-SNE [26] and, alternatively, UMAP [27] for the projection P, any user-chosen projection method can be used, such as PCA [28], LAMP [29], Least-Square Projection (LSP) [30], or Piecewise Laplacian Projection (PLP) [31]. For the inverse projection P^{-1}, DBM evaluated two techniques, namely iLAMP [32] (the inverse of the LAMP projection technique mentioned above) and NNinv [33]. Compared to iLAMP, which constructs the backprojection by interpolating between the samples of D and $P(D)$ using linear or radial kernels, NNinv deep learns a regressor to output D using $P(D)$ as input. NNinv achieves inverse projections with a lower mean absolute error than P and is also simpler to implement and significantly faster [21,33]. Note that the same deep learning idea has been used to construct direct projections with similar speed, quality, and implementation simplicity advantages [34].

A DBM is simple to implement and allows one to use any direct and inverse projection techniques. Also, a DBM works without requiring label information, which means that constructing decision maps only depends on the distribution of points in D' in the feature space. In our subsequent evaluation, we use DBM with UMAP for the direct projection P and NNinv for P^{-1}, as this combination led to the best results in earlier DBM evaluations [21]. More information about DBMs is available at https://mespadoto.github.io/dbm/ (accessed on 1 September 2023).

2.4. Supervised Decision Boundary Maps (SDBMs)

Although they are flexible, the independent choice of P and P^{-1} in DBMs means that these operations have to be constructed separately (including fine-tuning their hyperparameters), which incurs additional effort. Self-Supervised Neural Network Projection (SSNP) [35] alleviates this by constructing P and P^{-1} jointly. Briefly put, SSNP follows a classical autoencoder architecture but adds a classification loss atop the standard reconstruction loss. To minimize this classification loss, either true labels (supervised mode) or pseudo-labels (semi-supervised mode; labels are obtained by running a clustering algorithm on the feature space) can be used. The encoder part then delivers P, whereas the decoder delivers P^{-1}.

Supervised Decision Boundary Maps (SDBMs) [22] directly apply SSNP to construct decision maps following the workflow in Figure 1. Like DBMs, SDBMs are also simple to implement and have a similar speed. More interestingly, SDBMs seem to produce smoother decision boundaries compared to DBMs, and these boundaries appear to agree better with the ground-truth information on the visualized classifiers. The price to pay for this is the inability to choose a specific direct projection P. This can be suboptimal in cases where one has such a technique that is known to be best for depicting the structure of a given dataset. The code of SDBMs is available at https://github.com/mespadoto/sdbm. (accessed on 1 September 2023).

2.5. DeepView (DV)

DeepView (DV) [20] constructs the direct projection P using UMAP. However, in contrast to classical UMAP, the points' similarities are computed using a Fischer distance, which combines their high-dimensional features with the classification function f. This approach, also called discriminative dimensionality reduction [19], favors grouping points in the projection that are similar both feature-wise and in terms of classification by f. The inverse projection P^{-1} of DV also uses UMAP, with the roles of input and output swapped. The inverse projection is next extrapolated to all 2D points by minimizing a Kullback–Leibler (KL) divergence that captures the probabilities of closeness in both 2D and the data space. Once P^{-1} is available, DV colors the 2D pixels following the same approach used in DBMs and SDBMs (Figure 1, step 7).

Although DV produces decision maps with smooth boundaries, the process is quite expensive, given that the computation of the Fisher distance is squared in the number of samples in D'. Also, the process involves optimizing several hyperparameters to construct an accurate P^{-1}. As with DBMs, the direct and inverse projections P and P^{-1} can be, in principle, freely chosen. However, given the aforementioned optimization complexity, we next use the original proposal in [20] to construct both P and P^{-1}. The code of DV is available at https://github.com/LucaHermes/DeepView. (accessed on 1 September 2023).

2.6. Limitations

By far the most evident issue with current decision-map techniques is the very limited *evaluation* they come with. The DBM was evaluated qualitatively using 28 projection techniques for P (and iLAMP for P^{-1}) [21] to conclude that UMAP and t-SNE are among the best options for P to create smooth decision boundaries. However, this evaluation was purely *qualitative*, i.e., based on visually examining the respective decision maps for smoothness. The SDBM was evaluated on four classifiers and four real-world datasets and compared against the DBM. However, as in the previous case, the comparison was purely qualitative and was based on a visual assessment of the decision map's smoothness. Finally, DV was evaluated on two real-world datasets. Its quality was measured by computing two metrics related to our map accuracy and data consistency (described further in Section 3.1). However, it was not compared to any other decision-map techniques. We expand on all these aspects with our proposed evaluation method described in the following section.

3. Evaluation Method

As mentioned in Section 2.6, the current evaluations of decision-map techniques are limited. Due to inevitable errors in the decision-map creation process (detailed next in Section 3.1), using the classifier's accuracy to gauge a decision map's quality is neither appropriate nor comprehensive. Simple visual inspections of a decision map are equally limited in gauging its ability to correctly capture a classifier's behavior since we do not usually know this ground-truth behavior upfront. We aim to improve on this by proposing an extensive set of both quantitative and qualitative evaluations, each characterizing a different desirable property of decision maps, as described below.

3.1. Global Metrics

Global metrics aim to characterize the quality of a decision map with a single (scalar) value, much like the metrics used for direct projections such as trustworthiness, continuity, or normalized stress [36]. We propose five such metrics, as described below (see also Figure 2).

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

$$ACC_C = \frac{|\{\mathbf{x}_i \in D \mid C(\mathbf{x}_i) = f(\mathbf{x}_i)\}|}{|D|}, \tag{1}$$

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

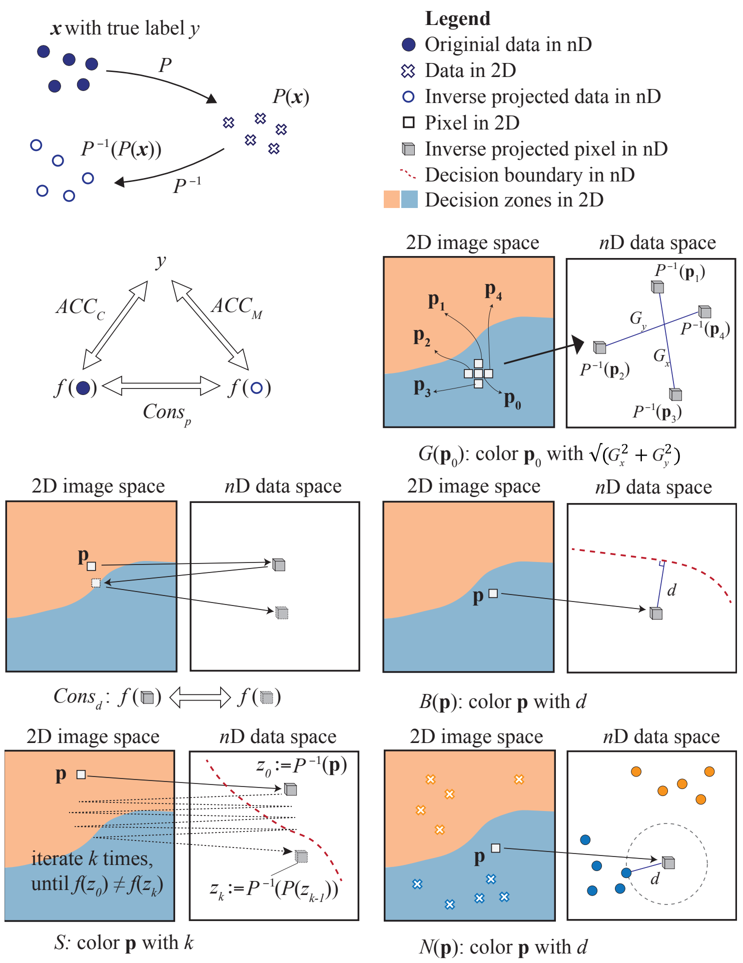

Figure 2. Illustration of the metrics used to evaluate decision maps (see Section 3).

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

$$ACC_M = \frac{|\{\mathbf{x}_i \in D \mid C(\mathbf{x}_i) = f(P^{-1}(P(\mathbf{x}_i)))\}|}{|D|}. \quad (2)$$

Intuitively, this says that data points $\mathbf{x}_i$, for which we have ground-truth labels y_i, are indeed colored correctly (i.e., by y_i) in the computed decision map. As for class accuracy, we compute ACC_M^t and ACC_M^T for the training and test sets, respectively. Note that map accuracy only evaluates the map pixels onto which the data in D projects since for all other pixels, we do not have ground-truth labels. The range of ACC_M is $[0, 1]$, where $ACC_M = 1$ indicates that all data points are drawn on the pixels with the same color as their ground-truth labels. Another way to evaluate map accuracy is to employ an additional 2D classifier, as used in the DV evaluation (see Q_{kNN} in [20]). We do not use this option since we believe it introduces an additional degree of complexity in the selection and training of this additional classifier.

Data consistency, $Cons_d$: In general, both the direct and inverse projections P and P^{-1} unavoidably introduce errors [32,33,37,38]. That is, in general, P^{-1} is not an exact inverse of P, i.e., $P^{-1}(P(\mathbf{x})) \neq \mathbf{x}$, for several data points $\mathbf{x}$. Such errors can be evaluated using the mean square error $MSE = \sum_{\mathbf{x} \in D} \|\mathbf{x} - P^{-1}(P(\mathbf{x}))\|^2 / |D|$ computed over the set D [25]. However, for decision maps, the MSE is not the most relevant metric to consider: a pixel $P(\mathbf{x})$ may be backprojected away from $\mathbf{x}$, i.e., the MSE may be nonzero; still, if the backprojection has the same label as the original point $\mathbf{x}$, then there is no visible error in the map. To account for this, we gauge whether the error introduced by the 'round-trip' direct and inverse projections creates inconsistencies in the decision maps. For this, we define the projection consistency metric

$$Cons_d = \frac{|\{\mathbf{x}_i \in D \mid f(P^{-1}(P(\mathbf{x}_i))) = f(\mathbf{x}_i)\}|}{|D|}. \tag{3}$$

Simply put, $Cons_d$ measures the fraction of so-called consistent projections, i.e., samples, $\mathbf{x}_i$, from a given set D that keep the same classification label after one round trip given by the projection P and inverse projection P^{-1}. This metric is also used in DV's evaluation, denoted as Q_{data} [20]. As for class and map accuracy, we compute $Cons_d^t$ and $Cons_d^T$ for the training and test sets, respectively. Note that in contrast to ACC_C and ACC_M, $Cons_p$ does not use ground-truth labels y_i. That is, consistency only assesses how much the decision map can represent a given classifier and not whether the DBM is correct with respect to the ground-truth labels. The range of $Cons_d$ is $[0, 1]$, where $Cons_d = 1$ indicates perfect consistency.

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

$$Cons_p = \frac{|\{\mathbf{p} \in I \mid f(P^{-1}(P(P^{-1}(\mathbf{p})))) = f(P^{-1}(\mathbf{p}))\}|}{|I|}, \tag{4}$$

where $\mathbf{p}$ is a pixel of the decision-map image I. $Cons_p$ calculates the fraction of consistent pixels in the decision map, that is, pixels whose corresponding data points (obtained by inverse projection P^{-1}) have the same classification label after a round trip of projection P and inverse projection P^{-1}. Note that $Cons_p$ extends the idea of using the round trip to visually evaluate an inverse projection [25] with a quantitative metric used to evaluate a decision map. $Cons_p$ ranges in $[0, 1]$, where $Cons_p = 1$ implies perfect pixel-level consistency in a decision map.

Class stability, $\bar{S}$: One application of decision maps concerns improving a classifier, for instance, by adding extra labeled training samples by backprojecting selected decision-map pixels. In this case, *multiple* round trips between the 2D map space and data space occur. Since P^{-1} is not an exact inverse of P, several such round trips can increasingly accumulate errors. We measure this as follows. Given a pixel in the decision map, we apply P^{-1} to obtain a data point and then apply the classifier to it. Next, we project this point to the 2D space and repeat the backprojection and labeling process until the class label changes or a maximum number of iterations (set to $k_{max} = 10$ in our experiments) is reached. Let S be an image recording this maximum iteration count (normalized by k_{max}), which keeps the class label constant at every map pixel. We then define the class stability $\bar{S}$ as the average of S over all pixels, with values in $[0, 1]$, where a value of $\bar{S}$ close to 1 indicates a stable decision map with consistent class assignments through multiple P and P^{-1} round trips. Conversely, a value of $\bar{S}$ close to 0 suggests an unstable decision map, sensitive to distortions introduced by (inverse) projection, potentially misrepresenting the classifier's decisions. As explained in Section 3.2, we also directly visualize S to obtain local insights into the class stability over the map.

Average gradient, $\bar{G}$: Since P^{-1} is a function of two variables (the x- and y-coordinates of a pixel), one can measure its gradient magnitude G at every pixel $\mathbf{p}$ through the central differences [25] as follows:

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

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

$$G(\mathbf{p}) = \sqrt{\|G_x(\mathbf{p})\|^2 + \|G_y(\mathbf{p})\|^2}. \tag{5}$$

Large gradient values indicate pixels where the decision map has a high likelihood of being incorrect, since neighboring pixels correspond to faraway data points; thus, data points can be classified differently. Separately, discontinuous changes in this map indicate areas where P^{-1} is not smooth, where we expect to see errors in the decision map.

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

3.2. Local Metrics

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

To gain more insights into these local phenomena, one can use so-called *local metrics*. Introduced to study direct projections [39], these metrics evaluate the quality of a map at every 2D spatial position and typically display the result as a color-coded visualization. We propose three local metrics to assess the quality of decision maps, as described below (see also Figure 2).

Gradient map: We display the gradient map G, computed as explained in Section 3.1, to help understand the smoothness of the inverse projection and, as outlined earlier, check for the presence of high-gradient regions that are prone to creating errors in the map.

Distance to boundary: Due to the nonlinearity of the mappings P and P^{-1}, if a pixel is visually close to a decision boundary *in the map*, this does not necessarily mean that its corresponding data point is close to the classifier's decision boundary *in the data space*. Yet, a key aim of decision maps is precisely to indicate points that are close to decision boundaries, since these are prone to misclassifications upon slight changes in the data or model parameters. Rodrigues et al. [21] addressed this by computing the actual distance in the data space between the backprojection of each pixel and its closest decision boundary. However, the methods they proposed for doing this, based on an iterative search in the data space of the precise location of the decision boundaries, are both inexact and computationally very expensive.

We mitigate this by computing the aforementioned distance to the boundary by resorting to an adversarial example generation technique [16,40]. Specifically, we employ DeepFool [41]. An adversarial example of a given classifier is a synthetic data point $\tilde{\mathbf{x}} = \mathbf{x} + \Delta\mathbf{x}$, such that $f(\tilde{\mathbf{x}}) \neq f(\mathbf{x})$, where $\|\Delta\mathbf{x}\|$ is as small as possible. Here, $\Delta\mathbf{x}$ indicates a small perturbation or change in the sample $\mathbf{x}$. In practice, it is complicated to generate the smallest perturbation possible that generates an adversarial example so DeepFool approximates it instead for every map pixel $\mathbf{p}$ as

$$B(\mathbf{p}) \approx \min_{\|\Delta \mathbf{x}\|} \text{ s.t. } f(P^{-1}(\mathbf{p}) + \Delta \mathbf{x}) \neq f(P^{-1}(\mathbf{p})) \tag{6}$$

In other words, $B(\mathbf{p})$ indicated how close $P^{-1}(\mathbf{p})$ is to a change in the class predicted by f, i.e., to the closest decision boundary to the backprojection of $\mathbf{p}$.

To compute the above, a differentiable classifier is needed. For this, we use a logistic regression classifier implemented in PyTorch.

Distance to data: Besides knowing how close a decision-map pixel $\mathbf{p}$ is to its closest decision boundary, it is also useful to know how close such a pixel is (via backprojection) to the nearest data point in the training set D_t used to construct f. We compute this as

$$N(\mathbf{p}) = \min_{\mathbf{x} \in D_t} \|P^{-1}(\mathbf{p}) - \mathbf{x}\|, \tag{7}$$

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

Class stability map: As outlined in Section 3.1, the class stability map S can act as a local metric. As explained, pixels with high S values will have the same class label, even after multiple round-trip P and P^{-1} iterations. When a pixel is close to a decision boundary, S is likely to be lower since the chance that a point there 'jumps' to the other side of a decision boundary due to these round trips increases. However, if the pixel is far from a decision boundary, its S value should be higher. If this is not the case, then the decision map may be unreliable in such areas. As such, visualizing how S matches the distances to the depicted boundaries in a decision map indicates how confident we are about the quality of that map.

3.3. Datasets

We evaluate the three decision-map techniques (DBM, SDBM, and DV) using both a synthetic dataset and real-world datasets. The real-world datasets are chosen following the same criteria as in [22]. They are chosen because they are openly accessible, representative of different types of data (e.g., time series, image, text), and have different numbers of classes and dimensions. The datasets used are listed below and are summarized in Table 1.

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

Human Activity Recognition (HAR) [42]: This is a dataset with time-series data from smartphone sensors. The goal is to classify the type of physical activity (e.g., walking, climbing stairs) performed by the user. This dataset has 10,299 samples, 561 dimensions, and six classes.

MNIST [43]: This dataset is a collection of handwritten digits that is commonly used for training various image classification systems. The dataset contains 60,000 training images and 10,000 testing images. Each image is a 28×28 grayscale image associated with a label from 0 to 9. The dataset is downsized to 10 k samples for all our experiments.

FashionMNIST [44]: This is a dataset containing images of Zalando items, spanning 10 fashion categories. It is the same size as MNIST and is downsized to 10 k samples for our experiments.

Reuters Newswire Dataset [45]: This dataset contains 8432 samples of news report documents, from which 5000 features were extracted using the standard TF-IDF [46] text processing method. From the full dataset, we only use the six most frequent classes.

Table 1. Datasets used in our decision-map evaluations and their properties.

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

3.4. Classifiers

We evaluate decision maps using four classifiers—logistic regression [47], random forest (200 estimators) [48], neural network (with three hidden layers, each with 200 units), and support vector machine (SVM, with an RBF kernel) [49]—which are all extensively used in machine learning. They represent different families of algorithms: logistic regression is a linear classification model; random forest is an ensemble method; neural network represents deep learning; and SVM is a maximum margin classifier. Importantly, these classifiers are frequently studied in existing decision-map research, thus allowing for meaningful comparisons. The four machine learning classifiers are implemented using *scikit-learn* [50].

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

4. Comparison Results

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

4.1. Global Metrics of Real-World Datasets

Figure 3 shows the global metrics for all combinations of decision-map techniques, classifiers, and datasets. Note that DV failed to run with SVM on the real-world datasets and thus is not included in the figure. From a dataset perspective, the results varied significantly based on the specific dataset being considered. On HAR, the SDBM and DV achieved comparable (high metric) results, whereas the DBM achieved slightly lower scores. On MNIST, the DBM achieved the best results in all aspects, even though the DBM's training did not use label information. On FashionMNIST, the SDBM achieved the highest scores, whereas DV achieved comparable results in the data-level metrics (ACC_C, ACC_M, $Cons_d$) but much lower results in the pixel-level metrics ($Cons_d$, $\bar{S}$). Finally, on Reuters, DV achieved the best results. From a classifier perspective, the results were more stable. The accuracy of the classifiers themselves varied only slightly. The most noticeable difference was that random forest always achieved lower scores (in all global metrics, except for ACC_C and $1-\bar{G}$) compared to the other classifiers in all the considered metrics, particularly on the MNIST dataset. For $1-\bar{G}$, almost all the results were close to 1. This was due to a large maximum value of DV, which influenced the normalization (see the definition of $\bar{G}$ in Section 3.1). We explore this in more detail in Section 4.3.

The datasets in Figure 3 are sorted (from top to bottom) by increasing dimensionality (see also Table 1). The higher the dimensions, the more difficult for the models to learn. So, it is not surprising that the DBM, which is a fully unsupervised method, experienced trouble when aiming to depict classifiers for higher-dimensional datasets like Reuters (5000 dimensions). In contrast, on this dataset, the SDBM achieved higher quality metrics, whereas DV achieved the highest values. This suggests that DV is the decision-map method of choice—from the perspective of the global quality metrics—for high-dimensional datasets. However, as we will demonstrate, there are other factors that influence the three decision-map methods in different ways.

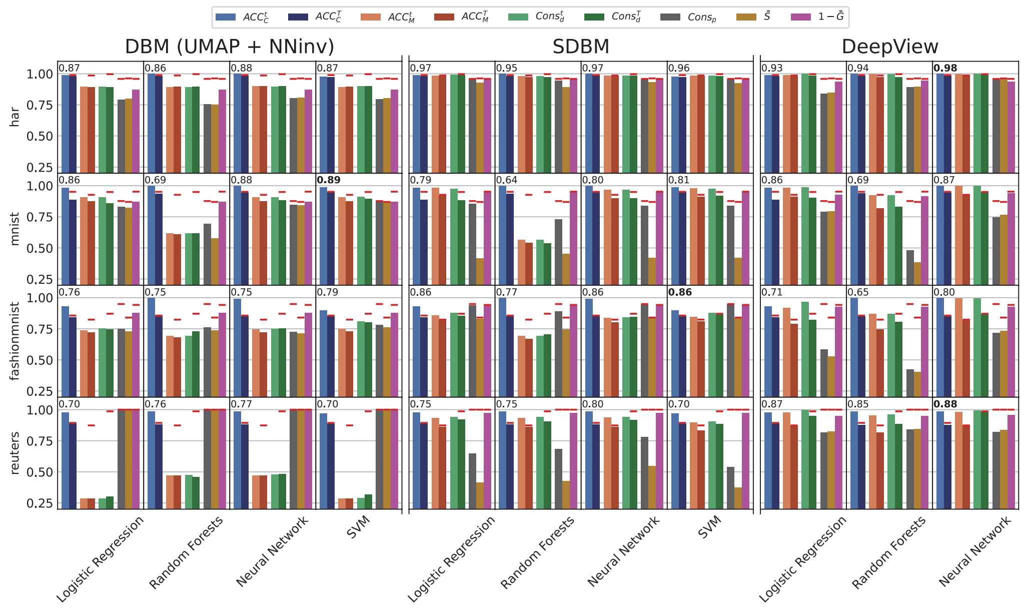

Figure 3. Aggregated global metrics for each decision map, classifier, and dataset combination. ACC_C, ACC_M, $Cons_d$, $Cons_p$, $\bar{S}$, and $1 - \bar{G}$ are the global metrics defined in Section 3.1. Red dashes show the highest values across each dataset (row). The top-left numbers in each plot show the average values of ${ACC_C}^T$, ${ACC_M}^T$, ${Cons_d}^T$, $Cons_p$, and $\bar{S}$. Bold values indicate the highest value among the dataset (row).

4.2. Interpreting Local Metrics on a Synthetic Dataset

We start by explaining the local metrics proposed in Section 3.1 using the simple synthetic blobs dataset, which is, as explained earlier, straightforward to classify. As such, we only studied its decision maps for the simple logistic regression classifier (we obtained very similar results for the other three considered classifiers). The blobs dataset is split into 1000 training and 500 testing samples. All three map methods—DBM, SDBM, and DV—achieved 100% on the test set for all data-wise global metrics. We then used the blobs dataset to establish and validate our *expectations* for a 'good' decision map based on the local metrics introduced in Section 3.2 as follows:

1. Distance to the nearest data N: We expected this distance to be small for pixels close to the actual projections of data points and larger for pixels far away from these points. In other words, we expected the 2D distance (to the projections of the data points) to mimic the nD distance (to the actual data points).
2. Distance to the decision boundary B: Similar to the above, we expected that the points close to the decision boundaries (in 2D) would also be close to the decision boundaries (in nD), and vice versa.
3. Class stability map S: Ideally, we would have liked most pixels to have high stability values, especially those close to the decision boundaries (where we were most interested in studying in a decision map).
4. Gradient map G: We expected a *smooth* gradient map without any discontinuities or peaks. Ideally, we also would have liked low gradients close to the decision boundaries (for the same reason mentioned above for class stability).

With these expectations in mind, we analyzed the results shown in Figure 4 for the three decision-map methods and the above-mentioned four local metrics. The first row in Figure 4 shows the decision maps and projected data points for the three methods.

At a glance, all maps appeared to be similar. However, a deeper analysis of the local metrics revealed more details. The results for N (second row) and B (third row) met our expectations across all methods. Some discontinuities in DV, however, were noticeable—the N and S images appear to contain some 'cuts' that perturbed their overall smoothness. S (fourth row) primarily manifested as expected, with all low-value pixels situated around the decision boundaries. A notable difference arose with the SDBM, which exhibited larger regions of unstable pixels, indicating its vulnerability to projection errors. Regarding G, the DBM and SDBM were smooth, whereas DV exhibited peaks, thus not satisfying our first expectation regarding G. Even more interestingly, the local maxima regarding G for DV took the form of *lines* roughly connecting the clusters of the projected points. As for our second expectation, we observed a surprising result: the DBM and SDBM exhibited relatively higher values close to the decision boundaries, whereas DV exhibited lower values in these areas. Consequently, no method simultaneously satisfied both expectations of G.

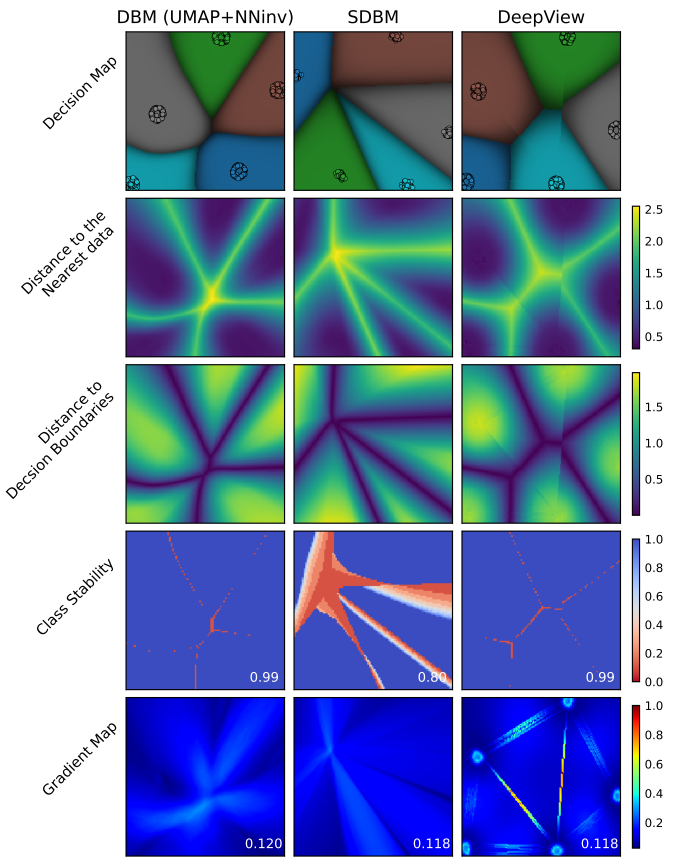

Figure 4. Decision maps and local metrics of the synthetic blob dataset (see Section 4.2). Row 1: Decision map with data points projected onto it (marked in black). The brightness of pixels encodes the confidence of f's prediction (dark = low, bright = high confidence). Row 2: Distance of each map pixel to the nearest data point N, depicted using a blue (low distance) to yellow (high distance) colormap. Row 3: Distance of each map pixel to the closest decision boundary B, using the same colormap as in row 2. Row 4: Class stability map S, depicted using a red–white–blue colormap. Blue values indicate stable pixels, whereas red values indicate pixels for which fewer direct and inverse projection round trips changed the classifier's decision depicted at that pixel. Row 5: Gradient map G depicted using a rainbow colormap. Blue and red pixels, respectively, indicate low and high values of the derivatives of the inverse projection function.

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

4.3. Analyzing Local Metrics on Real-World Datasets

To refine our preliminary insights concerning the differences between the three studied decision-map techniques, we now use our four proposed local metrics to study their behavior on the four real-world datasets shown in Table 1. We start by presenting the actual decision maps (Section 4.3.1). The next sections (Sections 4.3.2–4.3.5) interpret these maps using our four local metrics.

4.3.1. Decision Maps

Figure 5 shows the decision maps computed using training and testing data, respectively. The decision zones are color-coded by category, whereas the brightness of the pixels indicates the confidence of f's prediction (as used earlier for all decision-map techniques). The projected samples are also color-coded according to their class but are made slightly brighter to distinguish them from their surrounding zones. Misclassified samples, i.e., samples for which $f(\mathbf{x}^i) \neq y^i$, are marked with a white outline. Theoretically, the points near decision boundaries represent the samples for which the classifiers are most uncertain. This can be observed at the boundaries between the pink and the yellow zones in the HAR dataset, where some misclassified points are highlighted with white outlines. Notably, there are even some misclassified light-blue points at the boundaries of the pink and yellow zones. Knowing the nature of this dataset—classification of human activities—this observation aligns well with the understanding that static activities (corresponding to the class labels yellow and pink) are more challenging to distinguish compared to dynamic activities, such as various walking activities.

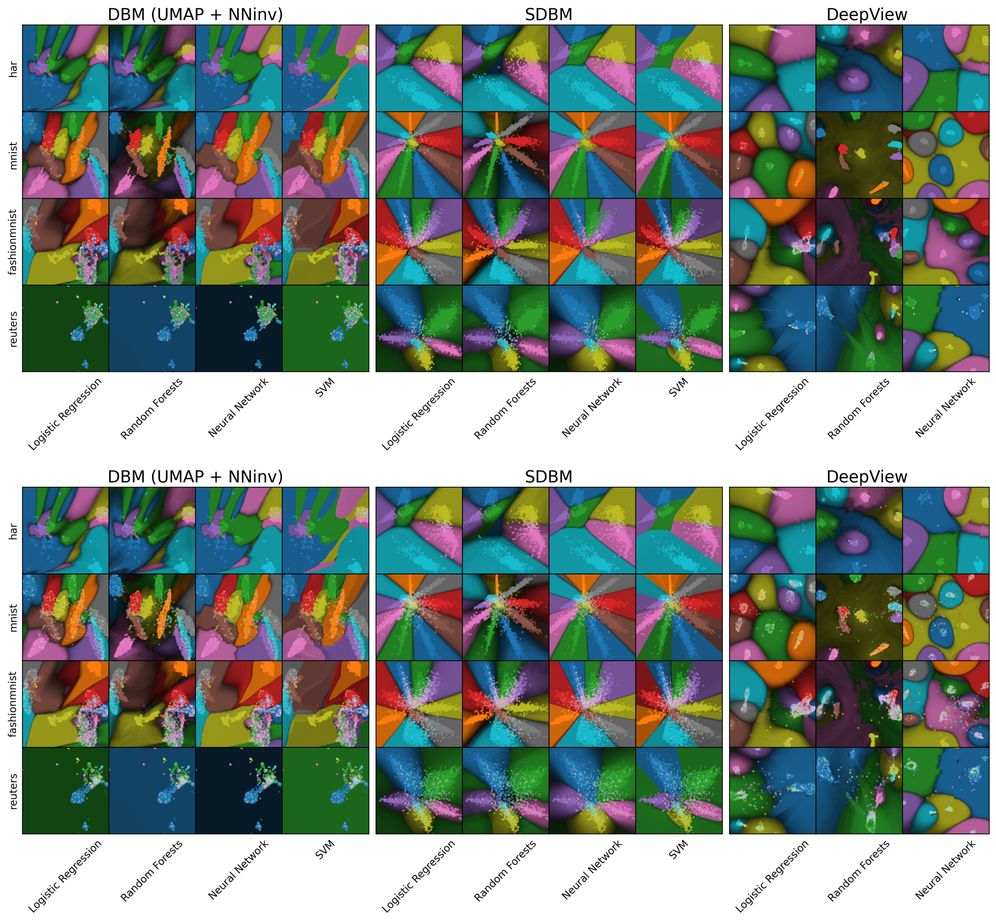

Figure 5. Decision maps using training data (**top**) and test data (**bottom**).

We can see that each decision-map technique has its own 'signature': the DBM is more random; the SDBM shows radial patterns; and DV presents blob-like patterns. Also, we can see that prediction confidences are always lower near the decision boundaries and higher within the decision zones, which is expected. In more detail, the DBM shows some 'island' decision zones that have no data points. Without more information, it is hard to tell whether these islands actually exist in the high-dimensional data or are artifacts of the decision map. In contrast, DV shows some discontinuities, indicated by 'breaks' or 'jumps' in the decision boundaries. Regarding the smoothness of the decision boundaries, the SDBM displays the smoothest ones; the DBM has more complex but still smooth boundaries; and DV exhibits the least smoothness. The shapes of the decision zones seem to be strongly influenced by the underlying projection method used for P: the SDBM consistently shows radial, elongated, star-like structures (a property of autoencoders used for dimensionality reduction); DV presents well-separated, blob-like, structures (likely due to its use of discriminative dimensionality reduction); and the DBM exhibits more variable structures (due to its UMAP projection). It is worth noting that the DBM failed on the Reuters dataset, as also reflected in the low global metrics score in Figure 3.

Separately, we examined the issue of overfitting. By displaying both the training and test data, we can check whether the decision maps can visually indicate overfitting. This overfitting was reflected by the noisier scatters plotted on incorrect background colors in the test sets (Figure 5 bottom). For the test data, we observed overfitting in all three methods across all four datasets, except for the simplest one, the HAR dataset. Among the three methods, DV appeared more prone to overfitting, a fact also evident from the noticeable difference between $ACC_M^t - ACC_M^T$ and $Cons_d^t - Cons_d^T$ in Figure 3.

4.3.2. Smoothness

The gradient map G, as defined in Equation (5), captures the smoothness of decision maps. Figure 6 shows this map for the three studied decision-map techniques. We can see that each technique created its own unique type of gradient. The DV technique shows high gradient values close to the projected points, which indicates a high degree of data compression during the projection P in these areas. Additionally, the DV technique shows some high gradient 'lines' connecting the clusters of the projected points, indicating discontinuities. The remaining areas have uniformly low gradient values. In contrast, the SDBM shows low gradient values almost everywhere, with slightly higher values in the gap areas between point clusters. Both the above patterns for DV and the SDBM are similar to what we observed on the blob dataset in Figure 4. Finally, the DBM's gradient map presents a more complex pattern, unlike what we observed on the easily classifiable blob dataset (Figure 4). As stated in Section 4.2, we expected that the gradient maps would not show peaks and would have low values in areas close to the decision boundaries. However, *no* method simultaneously exhibited these two properties. Notably, the SDBM's relatively high gradient values in the gap areas between point clusters were still low in absolute terms, making the SDBM the method that best satisfied the two mentioned gradient-map properties. The SDBM's high smoothness can be attributed to the joint training of P and P^{-1}. DV was clearly the least smooth method, as indicated by the number of discontinuities reflected in the 'peaks' in its gradient map. The DBM fell in between, exhibiting a complex pattern that displayed less smoothness but still maintained continuity.

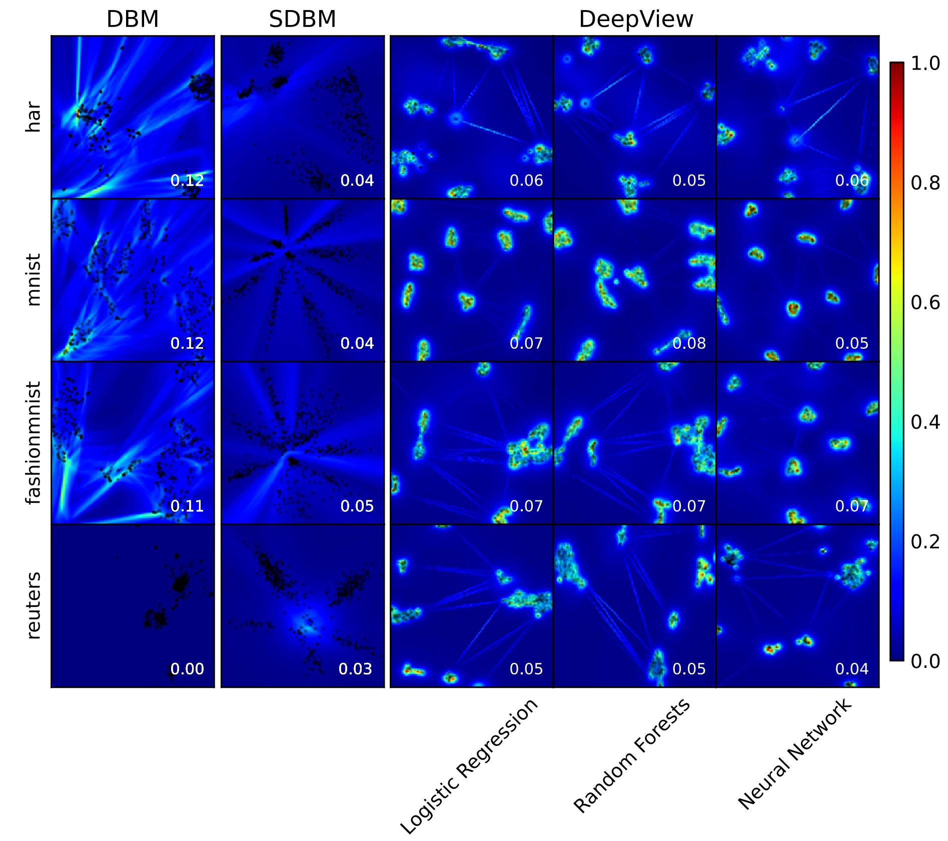

Figure 6. Gradient maps G of the studied decision-map methods. Gradient values are scaled to [0, 1] within each dataset. The number in the bottom-right corner of each plot is the average $\bar{G}$ of each map.

4.3.3. Class Stability Map

As explained earlier in Section 3.1, the global average value for class stability $\bar{S}$ indicates whether a decision map is robust to (inverse) projection errors. The class stability map S reveals *where* the decision map is susceptible to such errors. Figure 7 shows the S maps for the three studied decision-map techniques. As explained in Section 4.2, pixels with low S values were expected to appear primarily near the decision boundaries. This assumption held true, especially for simpler datasets, such as the HAR dataset.

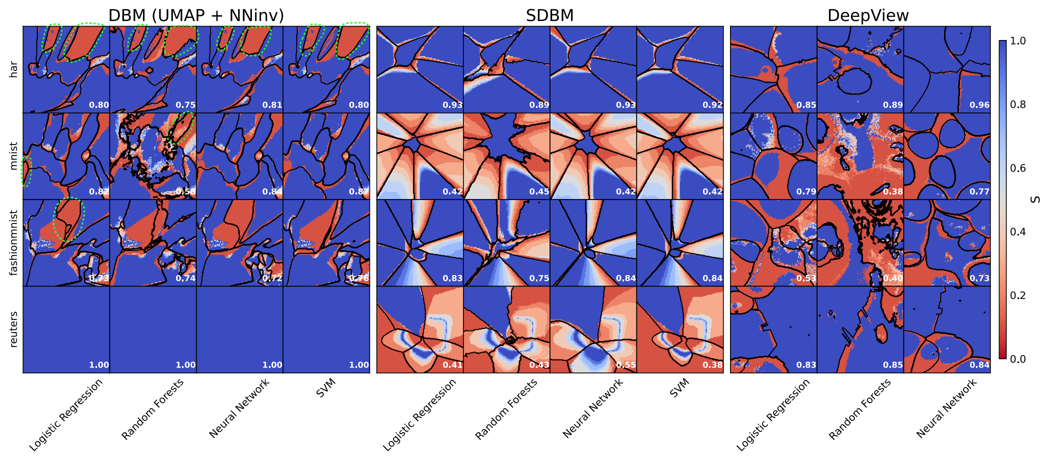

Figure 7. Class stability map S. The number in the bottom-right corner of each plot is the $\bar{S}$ of each map.

However, some anomalies can be seen for each decision-map technique: the DBM exhibited large zones with low S values. Referring to the decision map (Figure 5), most of these zones were *no data zones* (NDZs), in which no data points were projected—see the zones circled with dotted green lines in Figure 7 (leftmost image). Another interesting observation is that the most unstable pixels tended to change their labels *immediately* after the first round trip in the DBM. This observation was also made for DV. Unlike these two methods, the SDBM continued to be affected by round-trip errors. This can be observed by the gradual changes in the S values for the SDBM in Figure 7. After several rounds, most of the pixels in the SDBM changed their labels, except for those with the most robust labels, which almost never changed. DV exhibited another salient pattern: given the blob-like patterns in this decision map, a particular class often formed the 'base' of the blob-like shapes in some cases, e.g., the MNIST dataset using the random forest and neural network classifiers. The pixels in these areas were more prone to round-trip errors. In summary, the SDBM performed the worst regarding S, as also indicated by the results for the global metrics (Figure 3) on the synthetic blob dataset (Figure 4). The DBM and DV exhibited similar patterns for S, with the DBM slightly outperforming DV on the MNIST and FashionMNIST datasets. However, *all* three methods were prone to instabilities, and many such instabilities existed along their decision boundaries, which, as explained earlier in Section 4.2, is an undesirable aspect of trustworthy decision maps.

4.3.4. Distance to Decision Boundary

A key application of decision maps is to show how close a point on the map is to the actual decision boundary of the studied classifier. For this, the 2D distances in the map should depict the corresponding nD distances as closely as possible. Figure 8 (left) shows, for each map pixel, its distance to its closest decision boundary in the high-dimensional space, computed using Equation (6). In this figure, we can see that our expectations align with the observations—pixels close to the decision boundaries in the map are dark (meaning that they mapped points close to the actual decision boundaries in nD), whereas pixels deep in the decision zones are bright (meaning that they mapped points far away from the nD decision boundaries). Yet, we can also see variations in these distances across the decision-map methods. The SDBM and DV display patterns closer to the desired outcome compared to the DBM. DV shows a wider area with low distance values, whereas the SDBM shows a more concentrated area. This follows the patterns observed in the gradient maps (Figure 6), where DV expanded the gaps between point clusters, whereas the SDBM compressed them. The DBM exhibits a complex pattern, where certain zones display low distance values, even for pixels located in the center of the decision zones. These zones are particularly noticeable in the upper areas of the HAR and FashionMNIST datasets, as well as the leftmost area of the MNIST dataset. Interestingly, these zones (circled with red dotted lines in Figure 8, left) are also NDZs, which coincide with the zones of low S (see Figure 7). This correlation indicates reduced confidence in the inverse projection in those zones. The importance of confidence in extrapolation is underscored by these findings and is discussed further in Section 4.3.5.

4.3.5. Distance to the Nearest Training Data

The distance to the nearest data point N (Equation (7)) indicates how far the inverse projection for a given pixel is from the actual data distribution. If an inverse projection is significantly far from this data distribution, t likely corresponds to a point where the classifier will have generalization difficulties. Ideally, a decision map should not contain points that are far away from the actual training or testing points, and the pixels close to these points in the map should actually also be close to the respective data points.

Figure 8 (right) shows the distance N for the three decision-map methods. Data points are marked as white dots. For DV, the pixels near the data points exhibit low distance values, as expected. However, this pattern is less evident for the DBM and SDBM. In the SDBM representation of the FashionMNIST dataset, a zone on the right-hand side displays high distance values, even though a data point cluster is projected there. For the DBM, an entire region in the HAR dataset map also shows very high distance values. Unlike the SDBM's case, this is an NDZ, which also has low S values and low B values in the entire zone (see Figrues 7 and 8 (left), respectively). This high-value region, which is considerably distant from the data distribution, indicates that the inverse projections in this zone are unlikely to align with user expectations based on the class labels. This might be due to the square shape of the 2D map. Depending on the distribution of the data, the inverse projection is unlikely to uniformly populate the square region. In this case, the inverse projection has to extrapolate certain pixels that correspond to locations further away from the training data. This scenario underscores the value of the distance N in offering insights into potential issues.

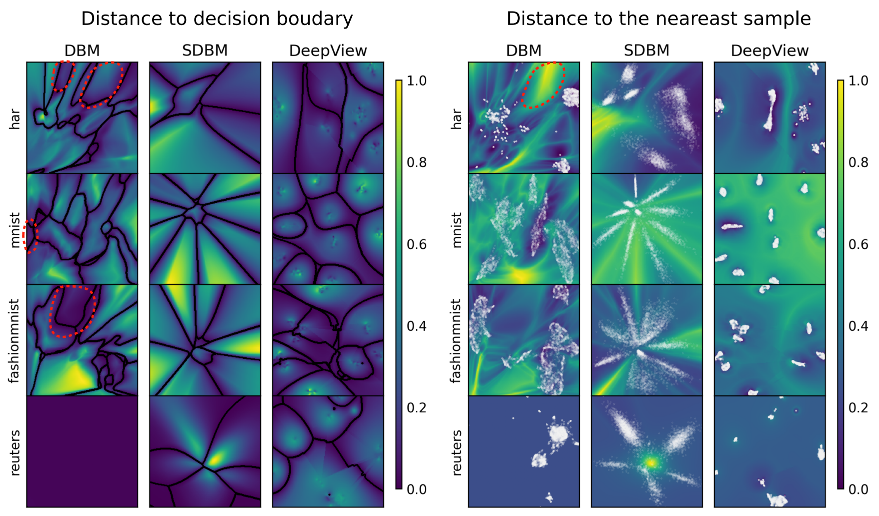

Figure 8. (**Left**) Distance to decision boundary B. (**Right**) Distance to the nearest training data N. Projections of the training data are shown as white dots.

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

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

4.4. Computational Efficiency

We measured the training and inverse projection time of the three studied decision-map techniques on an Intel Core i7-12700 CPU machine with 32 GB of RAM and an NVIDIA GeForce RTX 3070 GPU with 8GB of RAM. For this, we used the synthetic blob dataset with varying dimensions (10–500) and numbers of samples (250–5000). In particular, we evaluated DV, which requires a pre-trained classifier, with logistic regression, random forest, neural network, and SVM classifiers. It is important to note that DV's training time varied based on the classifier used (as detailed below).

Figure 9 shows the training times for the compared methods (that is, the time needed to construct the functions P and P^{-1}). When the number of samples N and dimensions n were both small ($N < 1000$, $n < 50$), the training time of DV (with logistic regression) was comparable to that of the SDBM and DBM. For larger n and/or N, compared to the real-world datasets (Table 1), the DBM and SDBM showed comparable training times. DV, however, exhibited substantially higher training times. The SDBM remained the fastest method, with average training times of less than 10 s across all tested scenarios. The DBM's training time was affected more by the number of samples N and far less by the number of dimensions n. DV's training time, when using SVM, increased drastically for larger n and/or N—training DV with SVM for $n = 500$ and $N = 5000$ took over 2.7 h. This steep increase made SVM-based DV experiments unfeasible on our real-world datasets.

Figure 10 shows the time required to create the decision maps (given trained direct and inverse projections P and P^{-1}) for various numbers of dimensions n, samples N, and grid resolution I. As for training, DV took the longest and was heavily influenced by all these parameters, even failing to handle resolutions over $I = 150$ pixels squared. Conversely, both the DBM and SDBM achieved relatively high and consistent performance, roughly linear in the number of pixels in the map and the number of samples N. In summary, we conclude that the DBM and SDBM are practical methods for creating decision maps for real-world datasets, whereas DV is not.

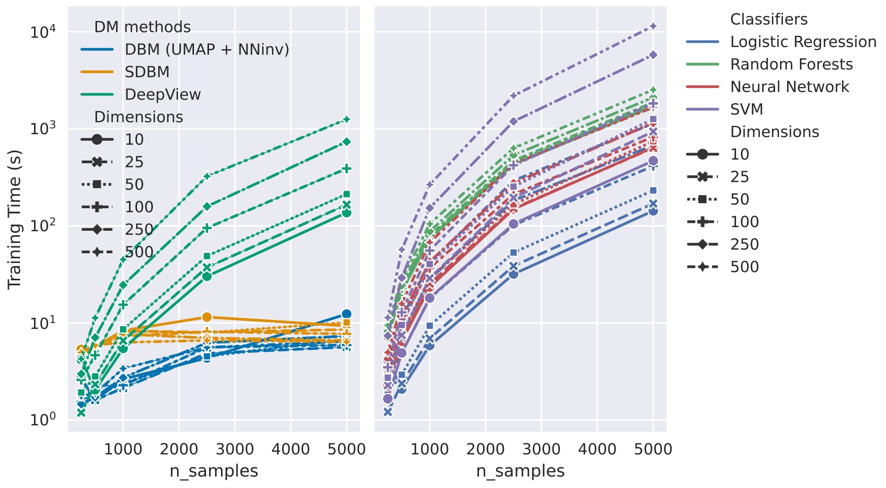

Figure 9. Times taken to train P and P^{-1} pairs. (**Left**) Training times of the DBM, the SDBM, and DV with logistic regression. (**Right**) Training times of DV with 4 different classifiers.

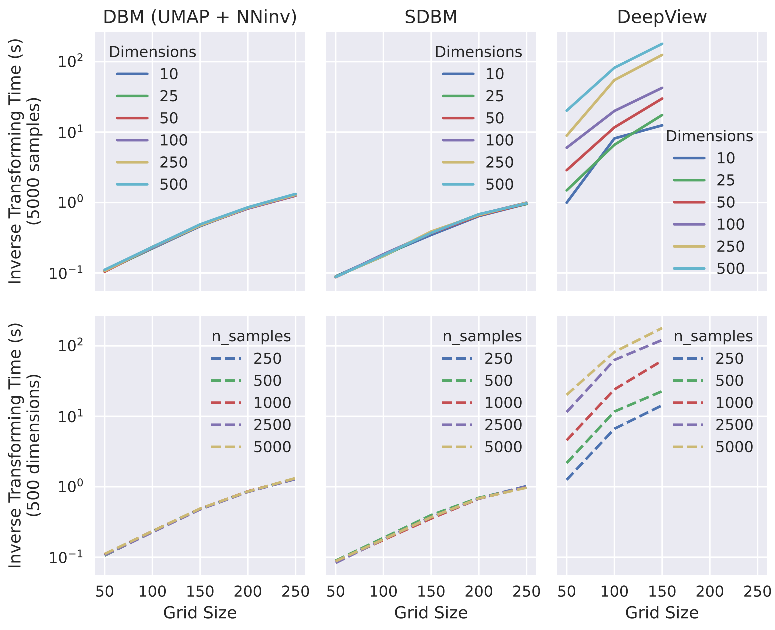

Figure 10. Times taken to create decision maps for the DBM, the SDBM, and DV.

5. Discussion

5.1. Decision Maps for Deep Learning Variations

Previous evaluations covered the way the three studied decision-map methods—DBM, SDBM, and DeepView—perform over a wide range of classification models. However, in the respective study, we used a single model based on a deep learning architecture (see Section 3.4). It can be argued that with the increasing prominence of deep learning, it is especially important to gather insights into how decision-map methods perform on this particular family of algorithms.

We discuss this next by separately studying the decision maps created for four such deep learning architectures. First, we considered the original and relatively simple neural network architecture, already used in previous evaluations, for comparison purposes. This architecture contains three fully connected layers, each with 200 units. Note that, although small, it is an architecture that has been used in previous deep learning applications to compute direct projections [34], inverse projections [25], and decision maps [22]. We then used two larger variants of this architecture, with four layers of 1024 units and four layers of 2048 units, respectively. Finally, we used TabNet [51], a very recent architecture designed for tabular data, which combines the principles of decision trees and neural networks and uses attention mechanisms [52] to prioritize important features during decision making. We did not include more specialized architectures, such as convolutional neural networks (CNN) [53], recurrent neural networks (RNN) [54], long short-term memory (LSTM) networks [55], or deep belief networks (DBMs) [56], in this comparison. Although these architectures are quite powerful, they are usually designed to be used with one particular type of data, e.g., images or time-dependent signals.

Figure 11 shows the performance metrics for the three decision-map techniques trained to classify four datasets using the above-mentioned four deep learning architectures. We can see that the four deep learning techniques—that is, within one set of four colored bars in the respective plots—were quite similar. The large differences that occurred were dependent on the dataset or metric, similar to what we saw for the architectures evaluated earlier (see Figure 3). The only outlier in this respect was TabNet when run on the Reuters dataset to compute the SDBM map and, to a much lesser extent, when run on the FashionMNIST dataset to compute the DBM map. We examine these two situations separately below.

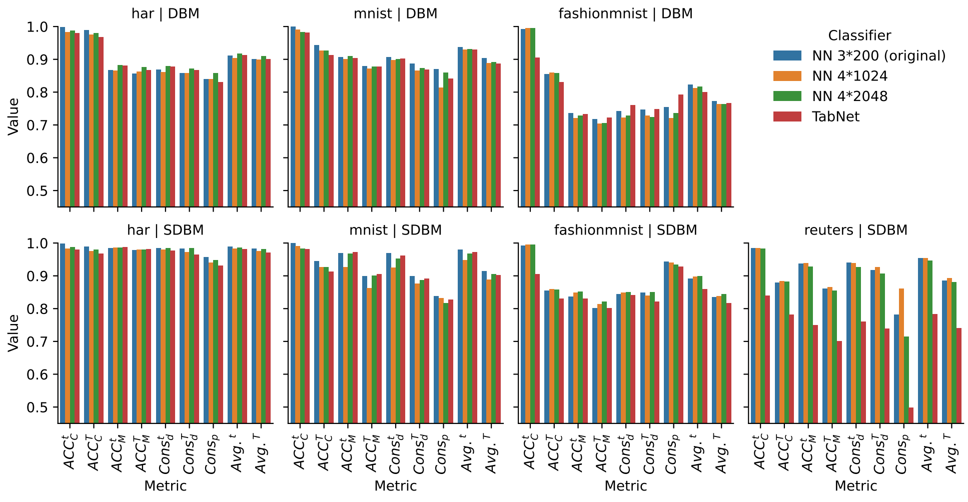

Figure 11. Label-related metrics for different neural networks. NN $m * n$ denotes a neural network with m hidden layers, each with n units.

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

In the former case, TabNet achieved lower scores in all metrics. If we look at the actual decision maps (Figure 12), we can see that these were, for all four architectures, overall quite similar for the same dataset. Moreover, these maps look quite different from those of the other classifiers depicted in Figures 4–6, apart from the neural network classifier for the two sets of images. In other words, the variability of the decision maps was much smaller over one type of architecture (neural networks) than across different architectures, which was expected. However, Figure 12 also shows some subtle differences between the maps in the middle of the depicted data clusters, that is, close to the locations where the decision boundaries appear. This indicates that the four studied deep learning classifiers, indeed, behaved slightly differently in the most uncertain areas. Also, we can see that the more complex the model, the more complex its decision boundaries. For example, in the case of SDBM–HAR, the decision boundary along the green zone is relatively simple for the NN $3 * 200$ model. For the NN $4 * 1024$ model, the dark blue intersects between the green zone and the other zones (pink and yellow). Further, for the most complex model NN $4 * 2048$, the pink zone thrusts into the middle of the green zone and the light-blue zone. Finally, with TabNet, the decision boundary of the green zone becomes rugged, and the whole decision boundaries become more complicated. This can be explained by the fact that the more complicated the classifier, the greater the chance it will overfit the given training data. Although this aspect is known in machine learning, the fact that we can show it directly using decision maps has, to the best of our knowledge, not been done previously.

In summary, our findings highlight the consistent performance of the decision-map methods across different datasets and classifiers. The absence of a clear best method for computing decision maps underscores the importance of a comprehensive selection workflow for decision maps. We propose such a workflow for choosing the decision-map method in the next section (see Section 5.2).

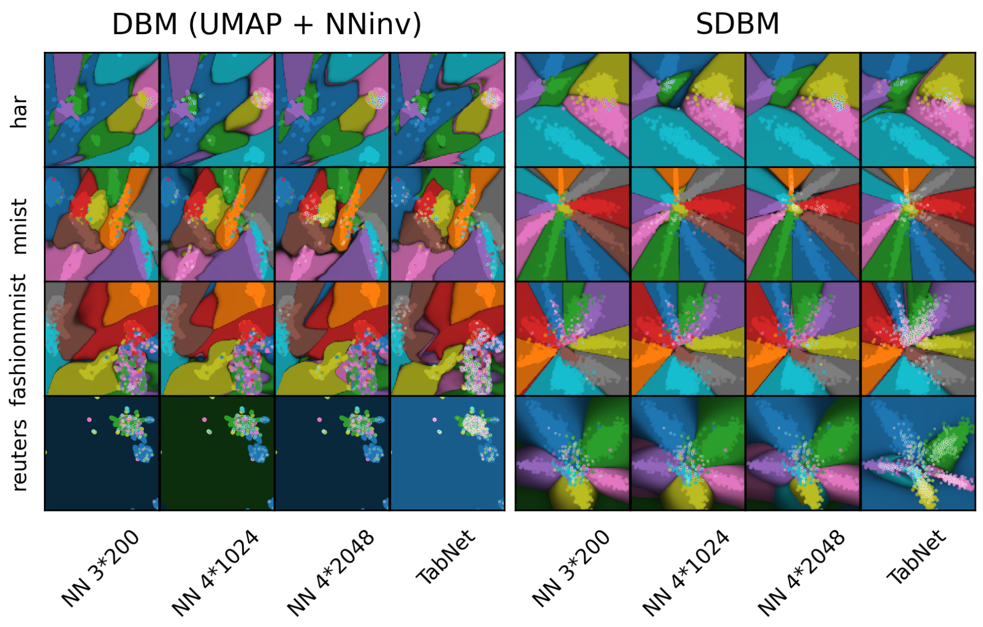

Figure 12. Decision maps for different neural networks (see Section 5.1).

5.2. *Workflow to Guide the Selection of a Decision-Map Technique*

After combining all the results of our evaluation, we can see that there was no clear winner among the studied decision-map techniques. When we considered the global quality metrics (Section 3.1), we found that each method reached its maximal quality on different datasets. Surprisingly, the unsupervised method (DBM) sometimes scored higher than the supervised ones (SDBM and DV) on the MNIST dataset. However, the DBM's failure on the Reuters dataset indicates that more challenging datasets may still require some level of supervision.

The results of the local metrics provide more insights into our conclusion about there being no winner. For the synthetic dataset, all methods exhibited quite similar patterns, except for DV's poor smoothness and the SDBM's slightly worse class stability. For the real-world datasets, the patterns were less distinct. Yet, some trends can be discerned. The smoothness of DV was consistently low, regardless of the simplicity or complexity of the datasets. Additionally, DV and the SDBM consistently showed more representative distances (B and N) compared to the DBM.

Concerning speed, the SDBM and SBM were clear winners—they definitely surpassed DV in both training and inference (map construction) times by up to three orders of magnitude, which makes the latter unsuitable for creating decision maps in real-world application scenarios, especially when using more time-consuming classifiers such as SVM.

Compared to the SDBM and DV, the DBM has a separate advantage. As outlined in Section 2, the DBM allows one to use any chosen direct projection function P to create decision maps. This can be important in cases where one knows that a given P is optimal, either for quantitative reasons or because it creates projections that are easier to interpret by users.

In conclusion, since there was no clear winner, the choice of a decision-map method should be guided by the specific requirements and constraints of individual use cases. We encapsulate this by proposing a workflow for choosing a method to create decision maps, as illustrated in Figure 13. Given a specific dataset and a set of potential classifier candidates, the workflow can assist users in making a choice by following the steps below:

1. If the user has already chosen a projection function P, they should select the DBM, as this is the only method that can accommodate a predefined P, and proceed to step 4.
2. If the user does not have a specific P, the next key aspect to consider is computational efficiency. If speed is important and the data to be visualized are large, DV should be excluded from consideration, and the workflow proceeds to step 4.
3. If the user does not have a specific P and computational efficiency is not a concern, one should consider if smoothness is important. If yes, DV should be excluded, and one can proceed to step 4.
4. If more than a single classifier–decision map combination remains to be chosen from, global quality metrics can be used to select the optimal one. The key metrics to use here are ACC_C, ACC_M, and $Cons_d$ (defined in Section 3.1), which can be computed for any combination of direct projection, inverse projection, and classifier. Note that $Cons_p$ and $\bar{S}$ are not available when the selected projection P cannot infer new data, such as in the case of non-parametric, non-out-of-sample projections like t-SNE.
5. Finally, visualizations of local metrics can be used to gain more trust in and/or a better understanding of the behavior of the chosen decision map, as described in the scenarios in Section 4.

5.3. What Decision Maps Really Are

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

Figure 14 shows these 'backprojected' decision maps for the DBM, the SDBM, and DV, with colors mapping the six classes. Strikingly, in all three cases, the backprojected shapes are actually *surfaces* embedded in 3D, which connect the six data-point clusters. The SDBM and DBM create smoother surfaces, whereas DV creates a tighter, locally less smooth surface that connects the six clusters (this fully matches the earlier observations concerning the line-like gradient maxima of the DV method). The fact that decision maps are actually sampling a *surface* from the high-dimensional data space is, to the best of our knowledge, an insight that has not been described in any earlier works on decision maps. Note that this is not a trivial finding—the fact that the backprojection of a 2D decision map is itself a surface may not be immediately apparent based on the construction of current decision-map techniques. Also note that the actual decision boundaries in the data space (which are themselves surfaces) are not visualized by the decision maps as such—rather, what the decision maps do show are the *intersections* of these decision boundaries with the aforementioned surface. As such, the way this surface is constructed by a given decision-map method will strongly influence which parts of the actual decision surfaces will be shown in the final decision map.

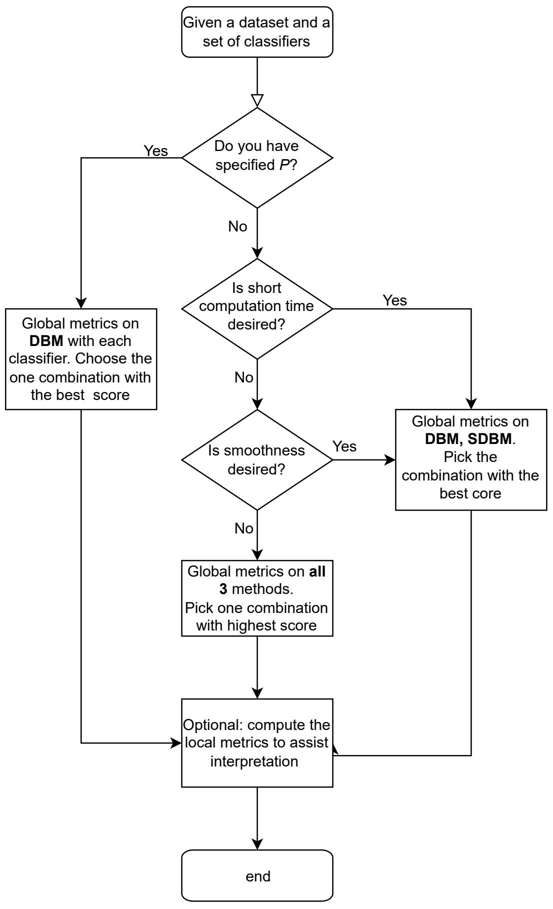
Given a dataset and a set of classifiers
Do you have specified P?
Yes
No
Global metrics on DBM with each classifier. Choose the one combination with the best score
Is short computation time desired?
Yes
No
Global metrics on DBM, SDBM. Pick the combination with the best core
Is smoothness desired?
Yes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

Figure 13. Workflow for choosing the most suitable decision map in a given user application context.

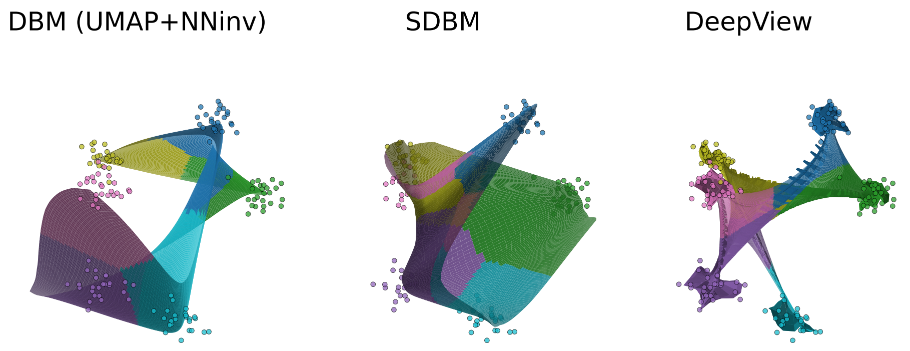

Figure 14. Decision maps for a synthetic blob dataset with 3 dimensions and 6 classes (see Section 5.3).

5.4. Limitations

We single out two limitations of our current work.

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

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

6. Conclusions

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

Our results showed that there is no decision-map method that consistently scores better than its competitors in all aspects deemed relevant for quality. Furthermore, we highlighted that all the studied decision-map methods have inherent limitations in various quality aspects and that these limitations can fluctuate significantly depending on the studied dataset and/or classifier being explored. To aid users in choosing a suitable decision-map method for practical applications, we proposed a workflow that considers all the studied quality aspects and proceeds with an elimination, followed by the optimization of these aspects.

Separately, we showed that all the studied decision maps have an inherent, previously unknown limitation—they can only visualize a surface from the entire high-dimensional space. The way this surface is constructed depends on the actual decision-map technique. As a consequence, the decision-map visualization reflects both the actual decision boundaries in the data and the way these intersect with the surface constructed implicitly by the decision-map technique.

There are several directions for future work. First, our evaluation can be extended by considering more datasets and classifiers and, when these are available, more decision-map techniques. Separately, our finding that decision maps actually visualize a single surface from the high-dimensional space can turn this inherent limitation of decision-map techniques into a strength. We can imagine ways to parameterize this implicit surface under user control to let it 'slice' through the actual high-dimensional decision boundaries in an interactive way, thereby offering the user the possibility to examine these decision boundaries in a more controlled and global way.

Author Contributions: Conceptualization, A.T. and Y.W.; methodology, A.T., Y.W., and A.M.; software, Y.W. and A.M.; validation, A.T., Y.W. and A.M.; resources, Y.W. and A.M.; data curation, Y.W. and A.M.; writing—original draft preparation, A.T., Y.W. and A.M.; writing—review and editing, A.T., Y.W. and A.M.; supervision, A.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by the China Scholarship Council (grant number 202206400020).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data and source code to reproduce the results are available at https://github.com/wuyuyu1024/metrics_for_decsion_maps (accessed on 1 September 2023).

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Javaid, M.; Haleem, A.; Pratap Singh, R.; Suman, R.; Rab, S. Significance of Machine Learning in Healthcare: Features, Pillars and Applications. *Int. J. Intell. Netw.* **2022**, *3*, 58–73. [CrossRef]
2. LeCun, Y.; Bengio, Y.; Hinton, G. Deep Learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
3. Mathur, P. *Machine Learning Applications Using Python: Cases Studies from Healthcare, Retail, and Finance*; Apress: New York, NY, USA, 2018.
4. Bergen, K.J.; Johnson, P.A.; de Hoop, M.V.; Beroza, G.C. Machine Learning for Data-Driven Discovery in Solid Earth Geoscience. *Science* **2019**, *363*, eaau0323. [CrossRef]
5. Gilpin, L.H.; Bau, D.; Yuan, B.Z.; Bajwa, A.; Specter, M.; Kagal, L. Explaining Explanations: An Overview of Interpretability of Machine Learning. In Proceedings of the 2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA), Turin, Italy, 1–3 October 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 80–89.
6. Rudin, C. Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. *Nat. Mach. Intell.* **2019**, *1*, 206–215. [CrossRef]
7. Doshi-Velez, F.; Kim, B. Towards a Rigorous Science of Interpretable Machine Learning. *arXiv* **2017**, arXiv:1702.08608. Available online: http://xxx.lanl.gov/abs/1702.08608 (accessed on 1 September 2023).
8. Iooss, B.; Kenett, R.; Secchi, P. Different Views of Interpretability. In *Interpretability for Industry 4.0: Statistical and Machine Learning Approaches*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 1–20.
9. Ribeiro, M.; Singh, S.; Guestrin, C. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *arXiv* **2016**, arXiv:1602.04938.
10. Yuan, J.; Chen, C.; Yang, W.; Liu, M.; Xia, J.; Liu, S. A Survey of Visual Analytics Techniques for Machine Learning. *Comp. Visual Media* **2021**, *7*, 3–36. [CrossRef]
11. Molnar, C. *Interpretable Machine Learning*; Lean Publishing: Victoria, BC, Canada, 2020.
12. Kaur, H.; Nori, H.; Jenkins, S.; Caruana, R.; Wallach, H.; Wortman Vaughan, J. Interpreting Interpretability: Understanding Data Scientists' Use of Interpretability Tools for Machine Learning. In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, Honolulu, HI, USA, 25–30 April 2020; pp. 1–14.
13. Monarch, R. *Human-in-the-Loop Machine Learning: Active Learning and Annotation for Human-Centered AI*; Manning Publ.: New York, NY, USA, 2021.
14. Ma, L.; Li, N.; Yu, G.; Geng, X.; Huang, M.; Wang, X. How to Simplify Search: Classification-Wise Pareto Evolution for One-Shot Neural Architecture Search. 2021. Available online: http://xxx.lanl.gov/abs/2109.07582 (accessed on 1 June 2023).
15. Lee, S.; Kim, D.; Kim, N.; Jeong, S.G. Drop to Adapt: Learning Discriminative Features for Unsupervised Domain Adaptation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 91–100.

16. Tsipras, D.; Santurkar, S.; Engstrom, L.; Turner, A.; Madry, A. On the Connection between Adversarial Robustness and Saliency Map Interpretability. *arXiv* **2019**, arXiv:1905.04172. Available online: http://xxx.lanl.gov/abs/1905.04172 (accessed on 1 June 2023).
17. Hamel, L. Visualization of Support Vector Machines with Unsupervised Learning. In Proceedings of the 2006 IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology, Toronto, ON, Canada, 28–29 September 2006; pp. 1–8.
18. Migut, M.A.; Worring, M.; Veenman, C.J. Visualizing Multi-Dimensional Decision Boundaries in 2D. *Data Min. Knowl. Discov.* **2015**, *29*, 273–295. [CrossRef]
19. Schulz, A.; Gisbrecht, A.; Hammer, B. Using Discriminative Dimensionality Reduction to Visualize Classifiers. *Neural Process. Lett.* **2015**, *42*, 27–54. [CrossRef]
20. Schulz, A.; Hinder, F.; Hammer, B. DeepView: Visualizing Classification Boundaries of Deep Neural Networks as Scatter Plots Using Discriminative Dimensionality Reduction. In Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence. International Joint Conferences on Artificial Intelligence Organization, Yokohama, Japan, 11–17 July 2020; pp. 2305–2311.
21. Rodrigues, F.C.M.; Espadoto, M.; Hirata, R.; Telea, A.C. Constructing and Visualizing High-Quality Classifier Decision Boundary Maps. *Information* **2019**, *10*, 280. [CrossRef]
22. Oliveira, A.A.; Espadoto, M.; Hirata, R., Jr.; Telea, A.C. SDBM: Supervised Decision Boundary Maps for Machine Learning Classifiers. In Proceedings of the VISIGRAPP (3: IVAPP), Online Streaming, 6–8 February 2022; pp. 77–87.
23. Rodrigues, F.C.M. Visual Analytics for Machine Learning. Ph.D. Thesis, University of Groningen, Groningen, The Netherlands, 2020.
24. Zhou, T.; Cai, Y.W.; An, M.G.; Zhou, F.; Zhi, C.L.; Sun, X.C.; Tamer, M. Visual Interpretation of Machine Learning: Genetical Classification of Apatite from Various Ore Sources. *Minerals* **2023**, *13*, 491. [CrossRef]
25. Espadoto, M.; Appleby, G.; Suh, A.; Cashman, D.; Li, M.; Scheidegger, C.E.; Anderson, E.W.; Chang, R.; Telea, A.C. UnProjection: Leveraging Inverse-Projections for Visual Analytics of High-Dimensional Data. *IEEE Trans. Visual. Comput. Graphics* **2021**, *29*, 1559–1572. [CrossRef]
26. Van der Maaten, L.; Hinton, G. Visualizing Data Using T-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
27. McInnes, L.; Healy, J.; Melville, J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv* **2018**, arXiv:1802.03426.
28. Jolliffe, I.T.; Cadima, J. Principal component analysis: A review and recent developments. *Phil. Trans. Royal Soc. A* **2016**, *374*, 20150202. [CrossRef]
29. Joia, P.; Coimbra, D.; Cuminato, J.A.; Paulovich, F.V.; Nonato, L.G. Local Affine Multidimensional Projection. *IEEE TVCG* **2011**, *17*, 2563–2571. [CrossRef]
30. Paulovich, F.V.; Nonato, L.G.; Minghim, R.; Levkowitz, H. Least Square Projection: A Fast High-Precision Multidimensional Projection Technique and Its Application to Document Mapping. *IEEE TVCG* **2008**, *14*, 564–575. [CrossRef]
31. Paulovich, F.V.; Eler, D.M.; Poco, J.; Botha, a.C.P.; Minghim, R.; Nonato, L.G. Piecewise Laplacian-Based Projection for Interactive Data Exploration and Organization. *Comput. Graph. Forum* **2011**, *30*, 1091–1100. [CrossRef]
32. dos Santos Amorim, E.P.; Brazil, E.V.; Daniels, J.; Joia, P.; Nonato, L.G.; Sousa, M.C. iLAMP: Exploring High-Dimensional Spacing through Backward Multidimensional Projection. In Proceedings of the 2012 IEEE Conference on Visual Analytics Science and Technology (VAST), Seattle, WA, USA, 14–19 October 2012; pp. 53–62.
33. Espadoto, M.; Rodrigues, F.C.M.; Hirata, N.S.T.; Hirata, R. Deep Learning Inverse Multidimensional Projections. In Proceedings of the Proc. EuroVA, Porto, Portugal, 3 June 2019.
34. Espadoto, M.; Hirata, N.S.T.; Telea, A.C. Deep Learning Multidimensional Projections. *Inf. Vis.* **2020**, *19*, 247–269. [CrossRef]
35. Espadoto, M.; Hirata, N.; Telea, A. Self-Supervised Dimensionality Reduction with Neural Networks and Pseudo-labeling. In Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications–IVAPP, Online Streaming, 8–10 February 2021; SciTePress: Setúbal, Portugal, 2021; pp. 27–37.
36. Venna, J.; Kaski, S. Visualizing Gene Interaction Graphs with Local Multidimensional Scaling. In Proceedings of the Proc. ESANN, Bruges, Belgium, 26–28 April 2006; pp. 557–562.
37. Espadoto, M.; Martins, R.; Kerren, A.; Hirata, N.; Telea, A. Toward a Quantitative Survey of Dimension Reduction Techniques. *IEEE TVCG* **2019**, *27*, 2153–2173. [CrossRef] [PubMed]
38. Nonato, L.; Aupetit, M. Multidimensional Projection for Visual Analytics: Linking Techniques with Distortions, Tasks, and Layout Enrichment. *IEEE TVCG* **2018**, *25*, 2650–2673. [CrossRef]
39. Aupetit, M. Visualizing Distortions and Recovering Topology in Continuous Projection Techniques. *Neurocomputing* **2007**, *10*, 1304–1330. [CrossRef]
40. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and Harnessing Adversarial Examples. 2015. Available online: http://xxx.lanl.gov/abs/1412.6572 (accessed on 1 June 2023).
41. Moosavi-Dezfooli, S.M.; Fawzi, A.; Frossard, P. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. 2016. Available online: http://xxx.lanl.gov/abs/1511.04599 (accessed on 1 June 2023).

42. Anguita, D.; Ghio, A.; Oneto, L.; Parra, X.; Reyes-Ortiz, J.L. Human Activity Recognition on Smartphones Using a Multiclass Hardware-Friendly Support Vector Machine. In Proceedings of the International Workshop on Ambient Assisted Living, Vitoria-Gasteiz, Spain, 3–5 December 2012; pp. 216–223.
43. LeCun, Y.; Cortes, C.; Burges, CJ. MNIST Handwritten Digit Database. 2010. Available online: http://yann.lecun.com/exdb/mnist/ (accessed on 1 June 2023).
44. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv* **2017**, arXiv:1708.07747.
45. Thoma, M. The Reuters Dataset. 2017. Available online: https://martin-thoma.com/nlp-reuters (accessed on 1 June 2023).
46. Salton, G.; McGill, M.J. *Introduction to Modern Information Retrieval*; McGraw-Hill Computer Science Series; McGraw-Hill: New York, NY, USA, 1986.
47. Cox, D.R. Two Further Applications of a Model for Binary Regression. *Biometrika* **1958**, *45*, 562–565. [CrossRef]
48. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
49. Cortes, C.; Vapnik, V. Support-Vector Networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
50. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V. Scikit-Learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
51. Arik, S.Ö.; Pfister, T. Tabnet: Attentive Interpretable Tabular Learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtually, 2–9 February 2021; Volume 35, pp. 6679–6687.
52. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 3058.
53. LeCun, Y.; Boser, B.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.; Jackel, L.D. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Comput.* **1989**, *1*, 541–551. [CrossRef]
54. Elman, J.L. Finding Structure in Time. *Cogn. Sci.* **1990**, *14*, 179–211. [CrossRef]
55. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
56. Salakhutdinov, R.; Murray, I. On the quantitative analysis of deep belief networks. In Proceedings of the ICML–International Conference on Machine Learning, Madison, WI, USA, 24–27 July 1998.