DeepVisualInsight: Time-Travelling Visualization for Spatio-Temporal Causality of Deep Classification Training

Xianglin Yang[1*], Yun Lin[1†*], Ruofan Liu[1], Zhenfeng He[1], Chao Wang[1], Jin Song Dong[1], Hong Mei[2]

[1]School of Computing, National University of Singapore, Singapore
[2]Key Lab of High-Confidence Software Technology, MoE (Peking University), China
xianglin@u.nus.edu, {dcsliny, dcslirf}@nus.edu.sg, {he.zhenfeng, wang.chao}@u.nus.edu, dcsdjs@nus.edu.sg, hongmei@pku.edu.cn

Abstract

Understanding how the predictions of deep learning models are formed during the training process is crucial to improve model performance and fix model defects, especially when we need to investigate nontrivial training strategies such as active learning, and track the root cause of unexpected training results such as performance degeneration.

In this work, we propose a time-travelling visual solution DeepVisualInsight (DVI), aiming to manifest the spatio-temporal causality while training a deep learning image classifier. The spatio-temporal causality demonstrates how the gradient-descent algorithm and various training data sampling techniques can influence and reshape the layout of learnt input representation and the classification boundaries in consecutive epochs. Such causality allows us to observe and analyze the whole learning process in the visible low dimensional space. Technically, we propose four spatial and temporal properties and design our visualization solution to satisfy them. These properties preserve the most important information when projecting and inverse-projecting input samples between the visible low-dimensional and the invisible high-dimensional space, for causal analyses. Our extensive experiments show that, comparing to baseline approaches, we achieve the best visualization performance regarding the spatial/temporal properties and visualization efficiency. Moreover, our case study shows that our visual solution can well reflect the characteristics of various training scenarios, showing good potential of DVI as a debugging tool for analyzing deep learning training processes.

Introduction

Interpreting model predictions is a well-reconsigned challenge when training and analyzing deep learning models (Zhang et al. 2021). Various explainable AI techniques have been proposed to understand model predictions including input attribution analysis, training data analysis, model abstraction, etc. Generally, existing solutions focus on:

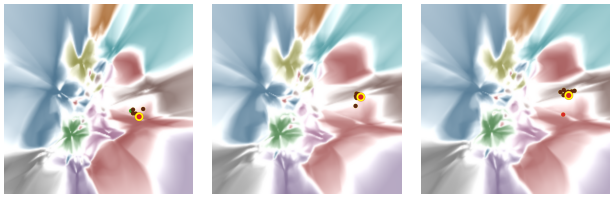
- **Individual prediction analysis**: identifying the most important features of an individual input to explain a model prediction (Sundararajan, Taly, and Yan 2017; Chattopadhyay et al. 2019; Kapishnikov et al. 2019; Simonyan, Vedaldi, and Zisserman 2013; Selvaraju et al. 2017; Chattopadhay et al. 2018);
- **Training data slicing**: identifying the most influential training samples that impact the model (Sagadeeva and Boehm 2021; Bhatt et al. 2021; Koh and Liang 2017);
- **Model abstraction**: abstracting simplified and explainable models (e.g., SVMs and decision trees) to explain the deep learning models (Ribeiro, Singh, and Guestrin 2016; Frosst and Hinton 2017; Zhang et al. 2019).

Despite those techniques are useful for explaining a trained model, few works are proposed to explain *how the model predictions are formed during the training process*. While some works focusing on the progressive training information (e.g., loss and accuracy) can be useful, they fail to abstract the underlying model evolving semantics. The semantic questions can be (but not limited to): (1) how the (re)training process gradually improves the model robustness, and reshapes the classification boundary? (2) how the model gradually makes a trade-off to fit some samples while sacrificing the others? (3) how the model struggles to fit and learn the hard samples?

In this work, we design a time-travelling visualization solution DeepVisualInsight (DVI), focusing on manifesting the spatio-temporal causality of the training progress of deep learning classifiers. DVI projects the learned input representation and their classification landscape into a visible low dimensional space, showing how model predictions are formed during training process, from both spatial and temporal perspectives. Spatially, DVI visualizes (1) the layout of learned input representation and (2) the classification landscape describing the "territory" of each class. Temporally, DVI visualizes (1) how the classification landscape and the training input representation evolve over the training and (2) how the new sampled training inputs reshape the classification boundaries. The spatio-temporal information allows us to observe training anomalies (e.g. noisy dataset) and verify some specific training strategies (e.g. effectiveness of active learning sampling strategies).

Comparing to designing measurements to analyze specific sample or model properties (e.g., Shapley value (An-

*These authors contributed equally.
†Corresponding author.
Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.



(a) Iteration 1
- adv acc 51.3%
- testing acc 92.3%

(b) Iteration 2
- adv acc 67.8%
- testing acc 90.3%

(c) Iteration 3
- adv acc 68.8%
- testing acc 89.9%

Figure 1: Adversarial training process: dynamics of one testing point and its ten neighbor adversarial points (adv acc and test acc stand for adversarial accuracy and test accuracy)

cona, Oztireli, and Gross 2019) and hard sample detection (Wu et al. 2017)), we design DVI to support *open-ended* exploration. That is, DVI faithfully reflects how deep models are learned through the training process, which not only can confirm known model properties, but also support the discovery of unknown phenomena and model defects.

Our approach takes inputs as classifiers trained under different training stages and its training/testing dataset, then learns *visualization models* (i.e., via an autoencoder) to (1) project high-dimensional samples into a visible low-dimensional space, (2) inverse-project low-dimensional points back to high dimensional space (for visualizing classification landscape), and (3) ensure that the visualization models can satisfy a set of spatial and temporal constraints. We propose four *visualization properties* for any time-travelling visualization solutions, to preserve (1) the topological structure between high and low dimensional manifolds, (2) the distance between training sample representations and latent decision boundaries, (3) the semantics of samples after projection and inverse-projection to low/high-dimensional space, and (4) the continuity of visualized landscape of all trained classifiers in chronological orders. In summary, we make the following contributions:

- We propose a time-travelling visualization solution, DeepVisualInsight (or DVI), which aims to visualize the evolving of classification landscape with spatio-temporal causality, to facilitate verifying the model properties and discovering new model behaviors.
- We propose four spatial and temporal properties for any time-travelling visualization techniques, and design a deep learning solution to satisfy them, for reflecting the classification landscape.
- We build our visualization framework DVI to support visualizing various deep classifiers.
- We conduct extensive experiments and case studies, showing (1) the effectiveness of DVI to satisfy the properties and (2) how DVI can help understand the training process and diagnose model behaviors.

More details of our tool/experiments are at (DVI 2021).

Motivating Example

Figure 1 shows our visualization of an adversarial training process on CIFAR-10 dataset. Each point represents a sample and each color represents a class. The color of a point represents its label, and the color of a region represents the predicted class. For example, a point in red (class cat) located in brown (class dog) territory indicates that it is labelled as cat but classified as dog. Moreover, the color shade indicates the confidence of prediction, unconfident regions (i.e. classification boundaries) are visualized as white regions. Overall, the classification regions and boundaries form the *classification landscape*. Here, *the model fitting process is visualized by the process of (1) classification boundaries being reshaped and (2) those data points being pulled towards the territory of their corresponding colors*.

Figure 1 shows that DVI manifests (1) the boundary reshaping process when the model is adapting new adversarial and training samples, and (2) the process of trade-off being made between adversarial robustness and testing accuracy. For clarity, we show one testing point (large red point with yellow edge) and its ten nearest neighbour adversarial points (in brown) in Figure 1. During adversarial training, (1) the adversarial points are gradually pulled to their color-aligned territory, while (2) the testing point is also gradually "pulled" away from its color-aligned territory to the territory of its adversarial neighbours. Such trade-off is formed gradually. In (DVI 2021), we further show such trade-off exists by visualizing the dynamics of overall data points. DVI tool can further visualize the process as animation. In addition, it supports samples and iteration queries from users to observe the dynamics of interested samples and iterations of interest, gaining deep insights into the model training process.

Related Work

Explainable AI (XAI) via Attribution Techniques Researchers have proposed approaches to track the prediction back to input, i.e. attribution method (Selvaraju et al. 2017; Sundararajan, Taly, and Yan 2017; Chattopadhyay et al. 2019; Simonyan, Vedaldi, and Zisserman 2013; Shrikumar et al. 2016; Kapishnikov et al. 2019). Attribution solutions evaluate the contribution of any input components (e.g. some pixels in the image) to the prediction outcome. (Sundararajan, Taly, and Yan 2017) proposed two axioms that every attribution method should satisfy, and developed integrated gradients(IG). (Chattopadhyay et al. 2019) proposed average causal effect (ACE) to mitigate the bias introduced by IG. To visualize the attribution explanation, (Selvaraju et al. 2017) proposes the Grad-Cam solution to highlight the pixels on input images to explain the predictions.

DVI visualizes the process how the classification landscape is formed. DVI and attribution analysis are complementary. DVI shows an overview of classification landscape and the distribution of the input samples, then use any attribution technique to inspect individual samples.

Model Visualization Typically, model visualization is transformed to a dimension reduction problem. Existing techniques include linear methods (e.g. PCA (Wold,

Notation	Definition	Dimension
$\mathbf{S}/\mathbf{X}/\mathbf{Y}$	Training data inputs, representations, and low-dimensional embeddings	$\mathbb{R}^{N\times d}, \mathbb{R}^{N\times h}, \mathbb{R}^{N\times l}$
$\mathcal{S}/\mathcal{X}/\mathcal{Y}$	Input space, manifold space, low-dimensional embedding space	$\mathcal{S} \subset \mathbb{R}^d, \mathcal{X} \subset \mathbb{R}^h, \mathcal{Y} = \mathbb{R}^l$
$\phi(\cdot)$	Projection function	$\mathbb{R}^h \to \mathbb{R}^l$
$\psi(\cdot)$	Inverse-projection function	$\mathbb{R}^l \to \mathbb{R}^h$
$f(\cdot)$	Feature function	$\mathbb{R}^d \to \mathbb{R}^h$
$g(\cdot)$	Prediction function	$\mathbb{R}^h \to \mathbb{R}^C$
$c(\cdot)$	Classifier, i.e. $g(f(\cdot))$	$\mathbb{R}^d \to \mathbb{R}^C$
$\mathbf{B}$	Boundary points in $\mathbb{R}^h$	$\forall \mathbf{b}_i \in \mathbf{B}, \mathbf{b}_i \in \mathbb{R}^h$

Table 1: Notation table for C-class classification task

Esbensen, and Geladi 1987), etc) and non-linear methods (e.g. t-SNE (Van der Maaten and Hinton 2008), UMAP (McInnes, Healy, and Melville 2018). Non-linear solutions preserve the neighbor relations after projecting data to a low-dimensional space. (Van der Maaten and Hinton 2008) propose t-SNE, which transforms the distance of high-dimensional samples into a conditional probability with Gaussian distribution and that of low-dimensional samples into a conditional probability with Student t-distribution as similarity measurements. (Tang et al. 2016) and (McInnes, Healy, and Melville 2018) propose LargeViz and UMAP to further improves the performance. (Rauber et al. 2016) visualizes the trajectories of samples using t-SNE.

One relevant work is DeepView (Schulz, Hinder, and Hammer 2019), aiming to visualize the decision boundaries of a classifier. DeepView projects high-dimensional sample into low-dimensional space via UMAP, with a customized manifold distance regarding the prediction outcome and the Euclidean distance in the input space. DeepView inverse-projects a low-dimensional point regarding the high-dimensional counterparts of its neighbours. DVI is different from DeepView in two folds. First, DVI is way more efficient and scalable than DeepView (see Section). Second, DVI considers boundary-preserving property and temporal property, which are essential in time-travelling visualization.

Properties of Time-Travelling Visualization

Notation Definition

We denote a C-class classifier as $c(\cdot)$. The input space is denoted as $\mathcal{S} \subset \mathbb{R}^d$. $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, ...\mathbf{s}_N]^T$ is training input set. $f : \mathbb{R}^d \to \mathbb{R}^h$ is a feature function, such that $\mathbf{x} = f(\mathbf{s})$ is a representation vector with h dimensions for an input $\mathbf{s} \in \mathbf{S}$. We denote the manifold space of the representation vectors as $\mathcal{X}$ where $\mathcal{X} \subset \mathbb{R}^h$. The learnt representations for training data is denoted as $\mathbf{X}$ where $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_N]^T$. Let $g : \mathbb{R}^h \to \mathbb{R}^C$ be the prediction function, where $g(\mathbf{x})_i$ represents the logits for i^{th} class. A classifier c consists of f and g, i.e. $c = g \circ f : \mathbb{R}^d \to \mathbb{R}^C$. Taking c and its training inputs, we derive a visualization model $V = \langle \phi, \psi \rangle$:

- A projection function $\phi : \mathbb{R}^h \to \mathbb{R}^l$, which projects manifold space $\mathcal{X}$ to a visible low-dimensional space $\mathcal{Y}$ where $\mathcal{Y} = \mathbb{R}^l$ (l is 2 or 3). Projecting $\mathbf{X}$ on to $\mathcal{Y}$ (i.e. $\mathbf{Y} = \phi(\mathbf{X})$) produces their counterparts $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, ...\mathbf{y}_N]^T$.
- An inverse-projection function $\psi : \mathbb{R}^l \to \mathbb{R}^h$, which inverse-projects visible low-dimensional space $\mathcal{Y}$ back to representation space $\mathcal{X}$.

Neighbour Preserving Property

Definition 1 (k-witness)**.** *Given a training dataset* $\mathbf{S}$ *and a distance metric defined on* $\mathbf{X}$*,* $d : \mathbb{R}^h \times \mathbb{R}^h \to \mathbb{R}_{\geq 0}$*. For a given* $\mathbf{x}_i \in \mathbf{X}$*, we denote the index set of its k-nearest neighbors as* $N_k(\mathbf{x}_i) = \text{argmin}_{\mathcal{J}\subset\{1..N\}\setminus\{i\},|\mathcal{J}|=k} \sum_{j\in\mathcal{J}} d(\mathbf{x}_j, \mathbf{x}_i)$*. We say* $\mathbf{x}_j$ *is k-witnessed by* $\mathbf{x}_i$ *in* $\mathbf{X}$ *if* $j \in N_k(\mathbf{x}_i)$*.*

Given a data sample $\mathbf{s}$, with its representation being $\mathbf{x} \in \mathbf{X}$ and low-dimensional counterpart being $\mathbf{y} \in \mathbf{Y}$, any $\mathbf{x}'$ being k-witnessed by $\mathbf{x}$ should have its counterpart $\mathbf{y}'$ being k-witnessed by $\mathbf{y}$, and vice versa.

Assuming the manifold $\mathcal{X}$ of $\mathbf{X}$ is known, we denote the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ in manifold as $d_{\mathcal{M}}(\mathbf{x}_i, \mathbf{x}_j)$. Similarly, we denote the distance of their counterparts $\mathbf{y}_i$ and $\mathbf{y}_j$ as $d_{\mathcal{E}}(\mathbf{y}_i, \mathbf{y}_j)$ in Euclidean space. Given a witness value k, we define $N_k(\mathbf{x}_i) := \text{argmin}_{\mathcal{J}\subset\{1..N\}\setminus\{i\},|\mathcal{J}|=k} \sum_{j\in\mathcal{J}} d_{\mathcal{M}}(\mathbf{x}_j, \mathbf{x}_i)$ and $N_k(\mathbf{y}_i) := \text{argmin}_{\mathcal{J}\subset\{1..N\}\setminus\{i\},|\mathcal{J}|=k} \sum_{j\in\mathcal{J}} d_{\mathcal{E}}(\mathbf{y}_j, \mathbf{y}_i)$, representing two index sets of neighbours being k-witnessed by $\mathbf{x}_i$ and its counterpart $\mathbf{y}_i$ respectively. The neighbour-preserving property requires to maximize the k spatial neighbour preserving rate

$$nn_{pv}(k) := \frac{1}{N}\sum_{i=1}^{N} \frac{|N_k(\mathbf{x}_i) \cap N_k(\mathbf{y}_i)|}{k} \tag{1}$$

Boundary Distance Preserving Property

Definition 2 (δ-Boundary)**.** *For a small* $\delta \in [0, 1)$*, a prediction function* $g : \mathbb{R}^h \to \mathbb{R}^C$ *and a min-max rescaling function* $r : \mathbb{R}^C \to [0, 1]^C$*, let* $r(g(\mathbf{x}))_{top1}$ *and* $r(g(\mathbf{x}))_{top2}$ *be the largest and second largest value of* $r(g(\mathbf{x}))$ *respectively. We say that a point* x *lies on* δ*-Boundary if* $|r(g(\mathbf{x}))_{top1} - r(g(\mathbf{x}))_{top2}| \leq \delta$*.*

We define classification boundary as a set of points $\mathbf{B} = \{\mathbf{b}|\mathbf{b} \text{ is on } \delta\text{-boundary}\}$. Similar to neighbour preserving property, the boundary distance preserving property requires that any $\mathbf{x}_i \in \mathbf{X}$ should preserve its k nearest boundary neighbours after being projected to $\mathbf{y}_i$ by $\phi(.)$. If we denote $\mathbf{b}$ as a boundary point in $\mathbb{R}^h$, and its counterpart in $\mathbb{R}^l$ as $\mathbf{b}'$. Extending Definition 1, we define $N_k^{(b)}(\mathbf{x}_i) := \text{argmin}_{\mathcal{J}\subset\{1..|\mathbf{B}|\},|\mathcal{J}|=k} \sum_{j\in\mathcal{J}} d_{\mathcal{M}}(\mathbf{b}_j, \mathbf{x}_i)$, $N_k^{(b')}(\mathbf{y}_i) := \text{argmin}_{\mathcal{J}\subset\{1..|\mathbf{B}|\},|\mathcal{J}|=k} \sum_{j\in\mathcal{J}} d_{\mathcal{E}}(\mathbf{b}'_j, \mathbf{y}_i)$ representing two index sets of being k-boundary-witnessed by $\mathbf{x}_i$ and its counterpart $\mathbf{y}_i$. We require the projection function $\phi(\cdot)$ should maximize:

$$boundary_{pv}(k) := \frac{1}{N}\sum_{i=1}^{N} \frac{|N_k^{(b)}(\mathbf{x}_i) \cap N_k^{(b')}(\mathbf{y}_i)|}{k} \tag{2}$$

Inverse-Projection Preserving Property

To visualize the classification landscape, the visualization solution needs an inverse projection function $\psi(\cdot)$ to reconstruct high-dimensional representation vectors from low-dimensional vectors in $\mathcal{Y}$. Such a reconstruction needs to satisfy that (1) any low-dimensional vector $\mathbf{y}_i$ projected from a representation vector $\mathbf{x}_i$, should be reconstructed to a $\mathbf{x}'_i$ as close to $\mathbf{x}_i$ as possible; and (2) it can generalize to arbitrary low-dimensional vectors. The first requirement ensures that the projection cause little information loss. Moreover, when representing each class as a distinct color, the second requirement allows us to *color* arbitrary points in a low-dimensional canvas. Given $\mathbf{H} = \{\mathbf{h_i} | \mathbf{h_i} \in \mathcal{X}\}$, it requires that $\psi(\cdot)$ can minimize the reconstruction error:

$$rec_{pv} := \frac{1}{|\mathbf{H}|} \sum_{i=1}^{|\mathbf{H}|} \|\mathbf{h}_i - \psi(\phi(\mathbf{h}_i))\|^2 \tag{3}$$

Temporal Preserving Property

Different from existing *static* visualization as UMAP and t-SNE, our visualized classification landscape requires to preserve the temporal continuity of the classification landscape change of the subject classifier. Assuming that two classifiers c^t and c^{t+1} are classifiers trained in two consecutive epochs, their classification landscapes are supposed to be similar. Thus, their visualization solutions V^t and V^{t+1} should provide similar visualization results.

We consider (1) classifiers $c^t = g^t \circ f^t$ and $c^{t+1} = g^{t+1} \circ f^{t+1}$ taken in chronological order, and (2) a measurement function $eval_{sem}(\cdot)$ to evaluate the semantic similarity of representations $\mathbf{x}^t = f^t(\mathbf{s})$ and $\mathbf{x}^{t+1} = f^{t+1}(\mathbf{s})$ of any input $\mathbf{s} \in \mathbf{S}$. Here, we evaluate the semantic of a input as the index set of its k-nearest-neighbors in manifold space. We denote the semantic similarity as $eval_{sem}(\mathbf{x}^t, \mathbf{x}^{t+1}, k)$. If two epochs have similar semantics, the visualization solutions V^t and V^{t+1} should project $\mathbf{x}^t$ and $\mathbf{x}^{t+1}$ to similar positions in $\mathbb{R}^l$, or have a negative correlation with $d_{\mathcal{E}}(\phi^t(\mathbf{x}^t), \phi^{t+1}(\mathbf{x}^{t+1}))$. We define the correlation as:

$$\begin{aligned} &temporal_{pv}(k) := \\ &corr(eval_{sem}(\mathbf{x}^t, \mathbf{x}^{t+1}, k), d_{\mathcal{E}}(\phi_t(\mathbf{x}^t), \phi^{t+1}(\mathbf{x}^{t+1}))) \end{aligned} \tag{4}$$

Then we require projection function $\phi^t(\cdot)$ and $\phi^{t+1}(\cdot)$ to minimize $temporal_{pv}(k)$:

To the best of our knowledge, none of the existing approaches have addressed all four properties. t-SNE and UMAP only satisfy the neighbour preserving property; DeepView satisfies the neighbour preserving and the inverse-preserving property. We make the first solution regarding all four properties.

Approach

Overview As showed in Figure 2, DVI takes as input a sequence of classifiers trained in chronological order, $\mathbf{C} = \{c^1, c^2, ..., c^T\}$ as subject models, and generates a corresponding sequence of visualization models (i.e. autoencoders) $\mathbf{V} = \{V^1, V^2, ..., V^T\}$ to derive visualized classification landscape. We use superscript to denote the chronological order of all notations. For each visualization model

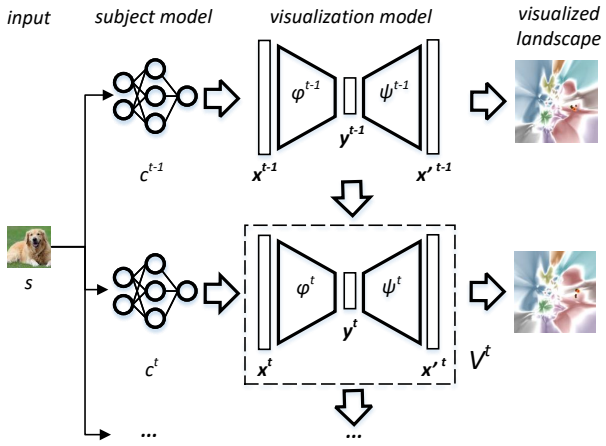


Figure 2: Overview of DeepVisualInsight

$V^t = \langle \phi^t, \psi^t \rangle$, the encoder serves as projection function ϕ^t and decoder as inverse-projection function ψ^t.

A non-white color is assigned to each class, V^t can (1) calculate the coordinate of each input $\mathbf{s} \in \mathbf{S}$ via $\phi^t(f^t(\mathbf{s}))$, and (2) color arbitrary point $\mathbf{y}$ via $g^t(\psi^t(\mathbf{y}))$. If $\mathbf{y}$ lies on δ-boundary (see Definition 2), it is colored in white; otherwise, it is colored in the representing color of class $g^t(\psi^t(\mathbf{y}))_{top1}$.

Each visualization model V^t for c^t is trained regarding the four spatial and temporal properties. We (1) estimate representative δ-boundary points for c^t; (2) construct a topological complex for boundary/training representation vectors and preserve its structure after projection to satisfy (boundary) neighbour-preserving property; (3) minimize the distance between $\mathbf{x}$ and reconstructed $\psi^t(\phi^t(\mathbf{x}))$ to satisfy the inverse-projection preserving property; and (4) build the continuity between (1) ϕ^{t+1} and ϕ^t and (2) ψ^{t+1} and ψ^t ($t \geq 1$) to satisfy the temporal-preserving property.

δ-Boundary Estimation

We estimate δ-boundary by synthesizing boundary samples, regarding the efficiency, authenticity, and diversity.

Efficiency and Authenticity We propose a novel mixup-based point synthesis method. Given a classifier $c(\cdot)$ and two input images $\mathbf{s}_i, \mathbf{s}_j \in \mathbf{S}$ from two different predicted classes, our rationale lies in that,

1. Their mixed-up inputs (i.e. images) can still largely preserve its inherent distribution of $\mathcal{S}$ (see Figure 3);
2. Assuming continuity of $c(.)$, and the linear interpolation $\mathbf{s}_b = \lambda \cdot \mathbf{s}_i + (1 - \lambda) \cdot \mathbf{s}_j, \lambda \in [0, 1]$, we can find a λ such that $\mathbf{s}_b$ lies on the δ-boundary within $\mathcal{O}(\log_2 \frac{d_{\mathcal{E}}(\mathbf{s}_i, \mathbf{s}_j)}{width(\delta)})$ rounds of binary search, where $width(\delta)$ is the width of δ-boundary on the line segment connecting $\mathbf{s}_i$ and $\mathbf{s}_j$ in Euclidean space.

To synthesize an authentic boundary sample, we set an upper bound for λ. Comparing to adversarial sample generation techniques (e.g., DeepFool (Moosavi-Dezfooli, Fawzi, and Frossard 2016)) with expensive search overhead, our





(a) Image 1 (b) Image 2 (c) Mixed-up

Figure 3: Example of mixed-up image, λ=0.35

mixup-based approach has more guarantee to synthesize a boundary sample within a limited search budget.

Diversity Given C classes in a classifier $c(\cdot)$, we synthesize boundary samples for $\binom{C}{2}$ pairs of classes. Regarding both diversity and efficiency of synthesis, we favour the pairs (1) with less number of boundary samples generated so far and (2) with high successful synthesis rate. Specifically,

$$\begin{aligned}\Pr(p=(C_i,C_j)) = \alpha \cdot \Pr(s(C_i,C_j)) \\ + (1-\alpha)\cdot succ(C_i,C_j)\end{aligned} \tag{5}$$

In Equation 5, we introduce a trade-off parameter $\alpha \in [0,1]$, between $\Pr(s(C_i,C_j))$ (i.e., the relative boundary sample abundance) and $succ(C_i,C_j)$ (i.e., the success rate to synthesize a boundary point). Specifically,

$$\Pr(s(C_i,C_j)) = \frac{max(0,\rho - num(C_i,C_j))}{\sum_{k\neq m} max(0,(\rho - num(C_k,C_m)))} \tag{6}$$

$num((C_i,C_j))$ is the generated boundary points between class C_i and C_j so far, and ρ is the mean number of generated point over all pairs of classes.

We estimate the successful synthesis rate of a pair as:

$$succ(C_i,C_j) = \frac{num_b(C_i,C_j)}{num_{syn}(C_i,C_j)} \tag{7}$$

$num_{syn}(\cdot)$ is the number of trials to synthesize boundary between a pair and $num_b(\cdot)$ is the number of successful trials within a search budget.

(k)-BAVR Complex Construction

Given representation vector set $\mathbf{X}$ and its derived boundary vectors $\mathbf{B}$, we construct a (k)-Boundary-Augmented Vietoris-Rips complex on $\mathbf{U} := \mathbf{X} \cup \mathbf{B}$, to sample a representative subset of edges $(u_i,u_j) \in \mathbf{U}\times\mathbf{U}$ for training an encoder to preserve boundary/non-boundary neighbors.

Definition 3 ((k)-Boundary-Augmented-Vietoris-Rips Complex)**.** *A (k)-Boundary-Augmented-Vietoris-Rips Complex ((k)-BAVR Complex) ($k>0$) is a simplicial complex consisting of 0-simplices and 1-simplices such that (1) each 0-simplex is a point from $\mathbf{U} := \mathbf{X}\cup\mathbf{B}$, and (2) each 1-simplex consists of two points in $\mathbf{U}$ and their connecting edge, satisfying one of the following conditions:*

(a) $\{(\mathbf{x}_i,\mathbf{x}_j) : \forall \mathbf{x}_i \in \mathbf{X}, j \in N_k(\mathbf{x}_i)\}$ *where $N_k(\mathbf{x}_i)$ is the index set of points that are k-witnessed by $\mathbf{x}_i$ in $\mathbf{X}$.*

(b) $\left\{(\mathbf{x}_i,\mathbf{b}_j) : \forall \mathbf{x}_i \in \mathbf{X}, j \in N_k^{(b)}(\mathbf{x}_i)\right\}$*, where $N_k^{(b)}(\mathbf{x}_i)$ is the index set of points being k-boundary-witnessed by $\mathbf{x}_i$.*

(c) $\{(\mathbf{b}_i,\mathbf{b}_j) : \forall \mathbf{b}_i \in \mathbf{B}, j \in N_k(\mathbf{b}_i)\}$*, where $N_k(\mathbf{b}_i)$ is the index set of $\mathbf{b}_i$'s k nearest boundary neighbors.*

Intuitively, (k)-BAVR Complex captures the topological structure of $\mathbf{U} := \mathbf{X}\cup\mathbf{B}$. Based on the complex, we sample a positive pair set $P_{x\times x+} \subset \mathbf{X}\times\mathbf{X}$ where $p=(\mathbf{x}_i,\mathbf{x}_j) \in P_{x\times x+}$ so that $\mathbf{x}_i$ and $\mathbf{x}_j$ form a 1-simplex. Similarly, we obtain $P_{x\times b+} \subset \mathbf{X}\times\mathbf{B}$ and $P_{b\times b+} \subset \mathbf{B}\times\mathbf{B}$. In addition, we randomly choose pairs from $\mathbf{X}\times\mathbf{X}$, $\mathbf{X}\times\mathbf{B}$, and $\mathbf{B}\times\mathbf{B}$ to construct three negative pair sets, i.e., $P_{x\times x-} \subset \mathbf{X}\times\mathbf{X}$, $P_{x\times b-} \subset \mathbf{X}\times\mathbf{B}$, and $P_{b\times b-} \subset \mathbf{B}\times\mathbf{B}$.

Finally, given $P = P_{x\times x+} \cup P_{x\times x-} \cup P_{x\times b+} \cup P_{x\times b-} \cup P_{b\times b+} \cup P_{b\times b-}$, we follow the parametric umap loss function defined in (McInnes, Healy, and Melville 2018) and (Sainburg, McInnes, and Gentner 2020) to train our encoder ϕ.

Inverse-Projection Preserving

The loss function of the encoder ϕ and the decoder ψ is:

$$\mathcal{L}_{rec} := \frac{1}{Nh}\sum_{i=1}^{N}\sum_{m=1}^{h}(1+grad_i^m)^{\beta}||\mathbf{x}_i^m - \psi(\phi(\mathbf{x}_i^m))||^2 \tag{8}$$

$$grad_i := abs(\frac{\partial g(\mathbf{x}_i)_{top1}}{\partial \mathbf{x}_i}) + abs(\frac{\partial g(\mathbf{x}_i)_{top2}}{\partial \mathbf{x}_i}) \tag{9}$$

where h is the number of dimensions, $g(\mathbf{x}_i)_{top1}$ is the largest value in $g(\mathbf{x}_i)$ and $g(\mathbf{x}_i)_{top2}$ is the second largest value in $g(\mathbf{x}_i)$. The rationale lies in that we need to preserve the most critical information of representation vector $\mathbf{x}$ after projecting and inverse-projecting back to the original space. Such information lies in the top-1 dimension of $g(\cdot)$ (for predicting its class) and $g(\mathbf{x}_i)_{top2}$ (for measuring the boundary).

Temporal Continuity

We preserve the temporal continuity with transfer learning and a temporal loss function. Given $V^{t-1}(t\geq 2)$, V^t is initialized with V^{t-1}'s weights. We bound the change of V^t from V^{t-1} by defining a temporal loss regarding the *temporal neighbour preserving rate*.

$$\mathcal{L}_t := \frac{1}{N}\sum_{i=1}^{N} eval_{sem}(\mathbf{x}_i^{t-1},\mathbf{x}_i^t,k)\cdot\|\mathbf{W}_{t-1}-\mathbf{W}_t\|^2 \tag{10}$$

We define input similarity semantics as its shared k-witnessed neighbors between consecutive epochs, Let $N_k(\mathbf{x}_i^{t-1})$ be the index set of all points being k-witnessed by $\mathbf{x}_i^{t-1}$ in epoch $t-1$, and $N_k(\mathbf{x}_i^t)$ in epoch t,

$$eval_{sem}(\mathbf{x}_i^{t-1},\mathbf{x}_i^t,k) := \frac{|N_k(\mathbf{x}_i^{t-1}) \cap N_k(\mathbf{x}_i^t)|}{k} \tag{11}$$

In Equation 10, $\mathbf{W}_t$ is the weights of the $\phi(\cdot)$ and $\psi(\cdot)$, while $\mathbf{W}_{t-1}$ is the weights of $\phi(\cdot)$ and $\psi(\cdot)$ learned in previous epoch. The final loss function to train $\phi(\cdot)$ and $\psi(\cdot)$ is the weighted sum of all the loss functions, i.e.,

$$\mathcal{L}_{total} = \lambda_1\cdot\mathcal{L}_{umap} + \lambda_2\cdot\mathcal{L}_{rec} + \lambda_3\cdot\mathbb{1}(t\geq 2)\cdot\mathcal{L}_t \tag{12}$$

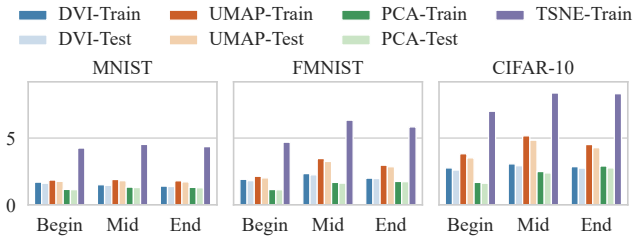


Figure 4: k Neighbour Preserving (k=15)

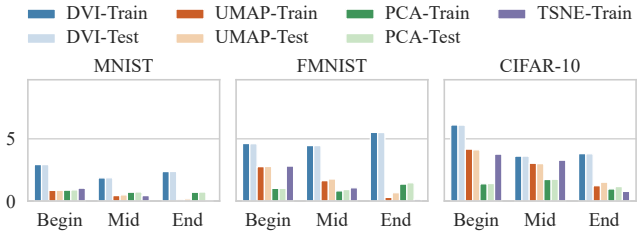


Figure 5: k Boundary Neighbour Preserving (k=15)

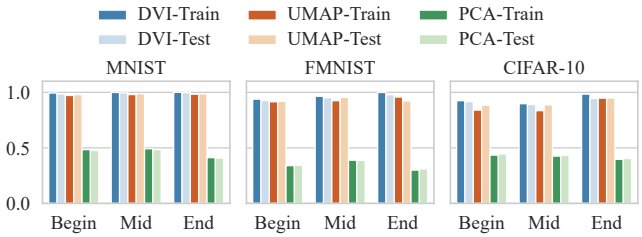


Figure 6: PPR between DVI, UMAP, and PCA

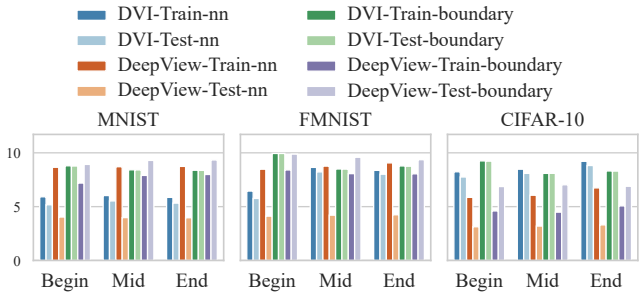


Figure 7: k-(NN/Boundary) Neighbour Preserving between DVI and DeepView (k=15)

Evaluation

Property Measurement. We measure the spatial and temporal properties, i.e., $nn_{pv}(k)$, $boundary_{pv}(k)$, rec_{pv}, and $temporal_{pv}(k)$ (see Section) as follows.

- **Preserving Neighbour and Boundary Distance**: We use $nn_{pv}(k)$ and $boundary_{pv}(k)$, and let $k = 10, 15, 20$.
- **Preserving Inverse-Projection**: We evaluate the prediction preserving rate, i.e., $PPR := \frac{|\mathcal{Q}|}{N}$ where $\mathcal{Q} := \{\mathbf{x} | \arg\max_c g_c(\mathbf{x}) = \arg\max_c g_c(\psi(\phi(\mathbf{x}))), \mathbf{x} \in \mathbf{X}\}$.
- **Preserving Temporal Continuity**: For $temporal_{pv}(k)$, we use Pearson correlation and set k as 10, 15, and 20.

Dataset and Subject Model. We choose three datasets, i.e., MNIST, Fashion-MNIST, and CIFAR-10. We use ResNet18 (He et al. 2016) as the subject classifier, and the output of global average pooling layer as the feature vectors (i.e., 512 dimensions).

Baseline. We select PCA, t-SNE, UMAP, and DeepView as baselines. We compare DVI with PCA, t-SNE, UMAP on the whole datasets, with DeepView on subsets of datasets due to its limitation for scalability. The subset training/testing sizes are set to 1000/200 (an empirical size suitable for DeepView) and the experiments are repeated 10 times to mitigate the bias.

Runtime Configuration. We design our autoencoder as follows. Given the dimension of the feature vector is h, we let the encoder and decoder to have shape $(h, \frac{h}{2}, \frac{h}{2}, \frac{h}{2}, \frac{h}{2}, 2)$; and $(2, \frac{h}{2}, \frac{h}{2}, \frac{h}{2}, \frac{h}{2}, h)$ respectively. Learning rate is initialized with 0.01 and decay every 8 epochs by factor of 10. The threshold δ to decide boundary point is set to be 0.1. We generate $0.1 * N$ boundary points, shared by all the solutions. The upper bound for λ in boundary point generation is set to 0.4, α in Equation 5 to 0.8, β in Equation 8 to 1.0, and the trade-off hyper-parameters in total loss (Equation 12) to 1.0, 1.0, 0.3 respectively.

Results (Spatial Property). Figure 4, 5, 6 and Table 2 show the performances of DVI and PCA, UMAP, and t-SNE on spatial properties on three datasets. Given the space limit, we show the results with $k = 15$, which shares similar performance when $k \in \{10, 20\}$ (see (DVI 2021)). We report the results of three representative epochs, i.e., the 1st epoch (representing the beginning epoch), the $\lfloor \frac{1+n}{2} \rfloor$th epoch (representing the middle epoch), and the nth epoch (representing the final epoch). We observe as follows:

- **PCA vs DVI**: PCA is a highly efficient solution (see Table 2). But its linear transformation has limitation, thus outperformed by DVI and UMAP (see Figure 4, 5, 6).
- **t-SNE vs DVI**: On training dataset, t-SNE significantly outperforms all other approaches regarding the preserved neighbours after projection. However, it cannot (1) generalize the projection to any unseen samples and (2) inverse-project a 2-dimensional point back to feature vector space. Moreover, t-SNE fails to preserve boundary neighbours as DVI and UMAP (see Figure 5).
- **UMAP vs DVI**: UMAP has comparable performance with DVI on the neighbour-preserving projection and prediction-preserving inverse-projection (as showed in Figure 4 and Figure 6). However, even trained with boundary samples, UMAP is largely outperformed by DVI regarding the boundary-neighbour preserving projection. Noteworthy, UMAP takes a much larger runtime overhead than DVI when inverse-projecting the low-dimensional points to the feature space ($\sim$16.8s for UMAP vs $\sim$0.002s for DVI, see Table 2).
- **DeepView vs DVI**: Regardless of the limited scalability of DeepView, DeepView is outperformed by DVI regarding:

1. DeepView is more likely to overfit the training dataset,

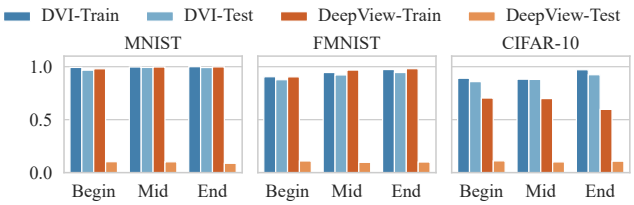


Figure 8: PPR between DVI and UMAP

Solution	Overhead	CIFAR-10	MNIST	FMNIST
DVI	Offline	792.784	914.921	896.296
	Online	0.016	0.010	0.010
UMAP	Offline	50.170	58.311	58.748
	Online	1819.598	2187.888	2150.703
t-SNE	Offline	207.757	286.068	282.725
	Online	/	/	/
PCA	Offline	0.803	0.958	0.951
	Online	0.035	0.036	0.035
DVI	Offline	19.801	17.150	18.896
	Online	0.004	0.004	0.004
DeepView	Offline	1305.229	506.839	506.394
	Online	563.471	204.473	204.436

Table 2: Visualization Overhead (in seconds)

thus its preserved neighbours on the test set is much less than that on the training set (see Figure 7).

2. DeepView can hardly preserve the prediction results after projection and inverse-projection (see Figure 8).

Results (Temporal). We compares the $temporal_{pv}$ value on (1) UMAP trained with transfer learning (denoted as UMAP-T); (2) DVI trained with transfer learning but without temporal loss (denoted as DVI-T); and (3) DVI (denoted as DVI), as in Table 3. Overall, DVI surpasses UMAP-T and DVI-T regarding the temporal continuity.

Runtime Efficiency. Table 2 shows the runtime efficiency of all the solutions. The offline overhead is the time spent on training the visualization model; the online overhead is the time spent on visualizing a new sample. Overall, DVI takes more time to train the encoder and decoder, while it is efficient to visualize the runtime new data. In contrast, UMAP is efficient to train but takes considerable time to inverse-project the low-dimensional points back to representation vector space. Moreover, DVI outperforms DeepView in both the offline and online efficiency.

Case Study

Noise (Hard) Sample Detection We flip the label of 10% of the CIFAR-10 samples to train a classifier. Figure 9 shows how clean/noisy sample embeddings are learned during training. Orange dots represent clean samples and black ones represent noisy samples. Comparing to the clean samples smoothly pulled into their color-aligned territory in the first few epochs, noisy samples show "reluctance" to be pulled (i.e., learned). Those "hard" samples stay in their "original" territory in early-mid epochs, but are forcefully pulled into their "expected" territory in late epochs.

Solution	CIFAR-10		MNIST		FMNIST	
	train	test	train	test	train	test
UMAP-T	-0.453	-0.448	-0.581	-0.578	-0.622	-0.613
DVI-T	-0.442	-0.460	-0.463	-0.466	-0.291	-0.286
DVI	**-0.463**	**-0.498**	**-0.609**	**-0.611**	**-0.626**	**-0.632**

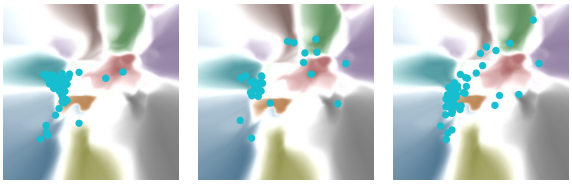
Table 3: Temporal Results, i.e., $temporal_{pv}$ value (k=15)



(a) E10, 60.42% (b) E50, 80.83% (c) E100, 83.59% (d) E150, 87.78% (e) E200, 88.44%

Figure 9: Visualized Training Process With Noise Data (epoch number and training accuracy)

DVI allow users to search their interested samples and tracking their movements, which can serve as a potential model debugging facility.



(a) Random (b) Coreset (c) LL4AL

Figure 10: Visualized active learning (sampling) strategies

Active Learning Strategies Comparison Active learning algorithms sample the most informative unlabelled samples to retrain the classifier. Various algorithms samples data regarding their diversity and uncertainty. Figure 10 compares the new sampled data by different active learning algorithms on the same classification landscape. We select Core-set (Sener and Savarese 2017) and LL4AL (Yoo and Kweon 2019) as diversity and uncertainty based methods. Comparing to random (dots concentrated in color-aligned territory), core-set selects samples evenly distributed in whole landscape and LL4AL selects samples closer to decision boundaries, confirming the effectiveness of DVI visualization.

Conclusion

We propose DVI to visualize how classification predictions are formed. DVI can serve the purpose of education, anomaly diagnose, and sampling strategy comparison for model training processes. In this work, we formally define four properties that any visualization tools should satisfy for spatio-temporal causality analyses. We develop DVI to satisfy them, which visualizes the layout of input samples and classification boundaries. In our future work, we will investigate interactive solutions for DVI such as (Lin et al. 2017) and (Wang et al. 2019) to take users' feedback to help them explore the training process.

Acknowledgements

This work was supported by the Minister of Education, Singapore (No. MOET32020-0004, No. T2EP20120-0019 and No. T1-251RES1901), the National Research Foundation Singapore via its National Satellite of Excellence in Trustworthy Software Systems office (NSOE-TSS2019-05).

References

Ancona, M.; Oztireli, C.; and Gross, M. 2019. Explaining deep neural networks with a polynomial time algorithm for shapley value approximation. In *International Conference on Machine Learning*, 272–281. PMLR.

Bhatt, U.; Chien, I.; Zafar, M. B.; and Weller, A. 2021. DIVINE: Diverse Influential Training Points for Data Visualization and Model Refinement. *arXiv preprint arXiv:2107.05978*.

Chattopadhay, A.; Sarkar, A.; Howlader, P.; and Balasubramanian, V. N. 2018. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 839–847. IEEE.

Chattopadhyay, A.; Manupriya, P.; Sarkar, A.; and Balasubramanian, V. N. 2019. Neural network attributions: A causal perspective. In *International Conference on Machine Learning*, 981–990. PMLR.

DVI. 2021. DVI (Anonymous). https://sites.google.com/view/deepvisualinsight/. Accessed: 2021-03-17.

Frosst, N.; and Hinton, G. 2017. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Kapishnikov, A.; Bolukbasi, T.; Viégas, F.; and Terry, M. 2019. Xrai: Better attributions through regions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4948–4957.

Koh, P. W.; and Liang, P. 2017. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, 1885–1894. PMLR.

Lin, Y.; Sun, J.; Xue, Y.; Liu, Y.; and Dong, J. 2017. Feedback-based debugging. In *International Conference on Software Engineering*, 393–403. IEEE.

McInnes, L.; Healy, J.; and Melville, J. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Moosavi-Dezfooli, S.-M.; Fawzi, A.; and Frossard, P. 2016. DeepFool: a simple and accurate method to fool deep neural networks. arXiv:1511.04599.

Rauber, P. E.; Fadel, S. G.; Falcao, A. X.; and Telea, A. C. 2016. Visualizing the hidden activity of artificial neural networks. *IEEE transactions on visualization and computer graphics*, 23(1): 101–110.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. arXiv:1602.04938.

Sagadeeva, S.; and Boehm, M. 2021. SliceLine: Fast, Linear-Algebra-based Slice Finding for ML Model Debugging. In *Proceedings of the 2021 International Conference on Management of Data*, 2290–2299.

Sainburg, T.; McInnes, L.; and Gentner, T. 2020. Parametric UMAP: learning embeddings with deep neural networks for representation and semi-supervised learning. arXiv. *Preprint] arXiv2009*, 12981.

Schulz, A.; Hinder, F.; and Hammer, B. 2019. Deepview: Visualizing classification boundaries of deep neural networks as scatter plots using discriminative dimensionality reduction. *arXiv preprint arXiv:1909.09154*.

Selvaraju, R. R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; and Batra, D. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, 618–626.

Sener, O.; and Savarese, S. 2017. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*.

Shrikumar, A.; Greenside, P.; Shcherbina, A.; and Kundaje, A. 2016. Not just a black box: Learning important features through propagating activation differences. *arXiv preprint arXiv:1605.01713*.

Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Sundararajan, M.; Taly, A.; and Yan, Q. 2017. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, 3319–3328. PMLR.

Tang, J.; Liu, J.; Zhang, M.; and Mei, Q. 2016. Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th international conference on world wide web*, 287–297.

Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11).

Wang, H.; Lin, Y.; Yang, Z.; Sun, J.; Liu, Y.; Dong, J. S.; Zheng, Q.; and Liu, T. 2019. Explaining regressions via alignment slicing and mending. *IEEE Transactions on Software Engineering*.

Wold, S.; Esbensen, K.; and Geladi, P. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3): 37–52.

Wu, C.-Y.; Manmatha, R.; Smola, A. J.; and Krahenbuhl, P. 2017. Sampling matters in deep embedding learning. In *Proceedings of the IEEE International Conference on Computer Vision*, 2840–2848.

Yoo, D.; and Kweon, I. S. 2019. Learning loss for active learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 93–102.

Zhang, Q.; Yang, Y.; Ma, H.; and Wu, Y. N. 2019. Interpreting cnns via decision trees. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6261–6270.

Zhang, Y.; Tiňo, P.; Leonardis, A.; and Tang, K. 2021. A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*.