DeepVID: Deep Visual Interpretation and Diagnosis for Image Classifiers via Knowledge Distillation

Junpeng Wang, Liang Gou, Wei Zhang, Hao Yang, and Han-Wei Shen, *Member, IEEE*

Abstract—Deep Neural Networks (DNNs) have been extensively used in multiple disciplines due to their superior performance. However, in most cases, DNNs are considered as black-boxes and the interpretation of their internal working mechanism is usually challenging. Given that model trust is often built on the understanding of how a model works, the interpretation of DNNs becomes more important, especially in safety-critical applications (e.g., medical diagnosis, autonomous driving). In this paper, we propose *DeepVID*, a ***Deep*** learning approach to ***V***isually ***I***nterpret and ***D***iagnose DNN models, especially image classifiers. In detail, we train a small locally-faithful model to mimic the behavior of an original cumbersome DNN around a particular data instance of interest, and the local model is sufficiently simple such that it can be visually interpreted (e.g., a linear model). *Knowledge distillation* is used to transfer the knowledge from the cumbersome DNN to the small model, and a deep generative model (i.e., variational auto-encoder) is used to generate neighbors around the instance of interest. Those neighbors, which come with small feature variances and semantic meanings, can effectively probe the DNN's behaviors around the interested instance and help the small model to learn those behaviors. Through comprehensive evaluations, as well as case studies conducted together with deep learning experts, we validate the effectiveness of *DeepVID*.

Index Terms—Deep neural networks, model interpretation, knowledge distillation, generative model, visual analytics.

1 Introduction

Classification is a fundamental problem in machine learning, and there are numerous algorithms/models to solve this problem, e.g., naïve Bayes classifiers, Support Vector Machines (SVM) [1], and decision trees [2]. These models have been used extensively in different disciplines and achieved extraordinarily good performance. Recently, Deep Neural Network (DNN) classifiers demonstrated even better performance, and have made a breakthrough in the object recognition challenge of ImageNet via Convolutional Neural Networks (CNN) [3], [4], [5].

However, the outstanding performance of the classification models does not lead to easy model interpretations or model trusts. Recently, people start concerning more about the trust and interpretability issues when deploying those accurate yet complex models, especially in safety-critical applications. For example, for self-driving cars, recognizing a stop sign as a speed-limit sign [6] will lead to a grave consequence. However, most of the classification models, especially DNN models, are considered as black-boxes and interpreting their internal working mechanism is usually challenging, as the data transformation in High Dimensional (HD) space goes beyond humans' interpretation capability.

For image classifiers, model interpretation mostly aims at finding what pixels of an image contribute positively/negatively to which class probabilities [7], [8], and two types of interpretation techniques can be found from the literature. The first type attempts to open the black-boxes and is designed for specific DNN classifiers (e.g., CNN), such as guided back-propagations [9], layer-wise relevance propagations [10], [11], and class activation maps [12]. These techniques are limited to DNN classifiers and cannot be used if the network structure is unknown, which happens very often in real-world applications. The second type of techniques is model-agnostic, which treats the model to be interpreted as a black-box and uses various approaches to probe and observe the black-box's behavior [13], [14]. One technique is to train a small model to locally probe and interpret the behavior of the original cumbersome classifiers at a specific data instance of interest, such as LIME [8]. The local model is trained using the perturbed neighbors around the data instance of interest, and the model is usually small and simple enough to be easily interpretable (e.g., linear regressors). Nevertheless, the existing ways of generating perturbed neighbors have fundamental limitations. First, perturbing a data instance in the input space is not efficient, as the input is usually high dimensional and the combination of perturbed dimensions is massive. Second, the neighbors generated by directly perturbing a data instance in the input space may have no semantic meanings, leading to potential biases when being used to train the small model.

In this paper, we propose *DeepVID* (Fig. 1), which brings together the power of ***Deep*** learning (i.e., knowledge distillation theories and generative models) and visual analytics to ***V***isually ***I***nterpret and ***D***iagnose image classifiers. Focusing on one image of interest, *DeepVID* trains a locally-faithful and explainable classifier (a small model) to mimic the behavior of an original cumbersome classifier (a big model). The training data of the small model are the localized neighbors of the interested image. Different from the existing model-agnostic approaches, *DeepVID* generates semantically meaningful training neighbors using deep generative networks, and the labels of the generated neighbors are obtained directly from the big model. Using the neighbors and their associated labels, the knowledge distillation mechanism [15] is then adopted to train the small model to effectively distill

- *J. Wang and H.-W. Shen are with the Department of Computer Science and Engineering, the Ohio State University, Columbus, OH, 43210. E-mail: {wang.7665, shen.94}@osu.edu*
- *L. Gou, W. Zhang and H. Yang are with Visa Research, Palo Alto, CA, 94306. Email: {ligou, wzhan, haoyang}@visa.com*

Manuscript received April 19, 2005; revised August 26, 2015.

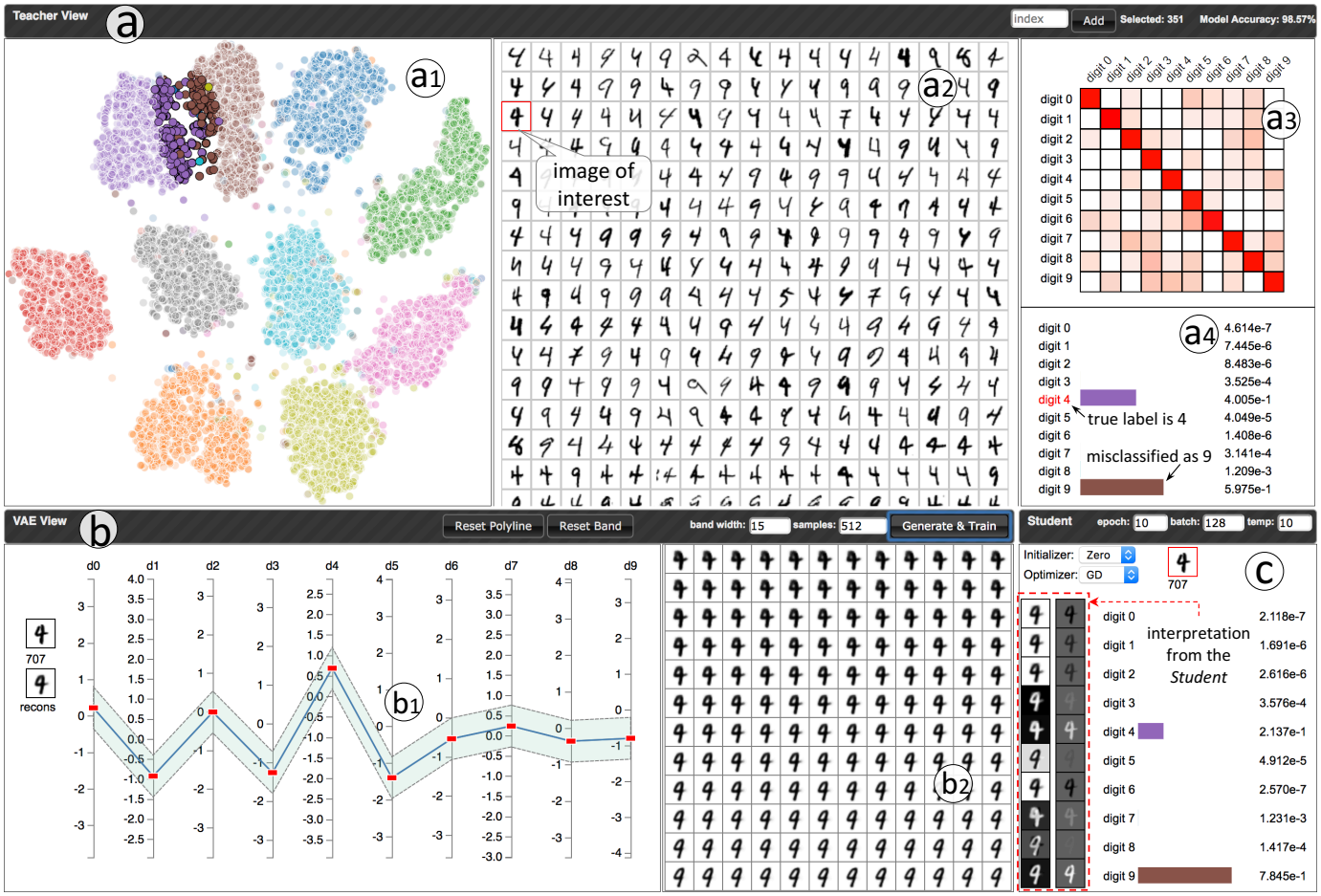


Fig. 1: *DeepVID*. (a) Teacher view: explore test data and assess the *Teacher*'s performance with the t-SNE view (a1); Image-Grid view (a2); Confusion Matrix view (a3); Probability Distribution view (a4). (b) VAE view: explore the latent space with a parallel coordinates plot (b1); and the generated neighbors with the Neighbors view (b2). (c) Student view: visualize the trained *Student* for interpretation.

knowledge from the big model for interpretation. *DeepVID* is designed and implemented as a visual analytics system, and aims to help domain experts conveniently interpret classifiers and diagnose their performance by dissecting why and how they succeed or fail in specific cases. Through comprehensive evaluations, we demonstrate the effectiveness of *DeepVID*. To sum up, our contributions include:

- We propose *DeepVID*, a model-agnostic interpretation framework that uses deep generative models (i.e., variational auto-encoders [16]) to generate local samples with semantic feature variance and adopts knowledge distillation for interpretation.
- We implement *DeepVID* as a visual analytics system, which empowers domain experts to interactively explore the feature space of their data and interpret their classifiers' behaviors.
- We validate the effectiveness of *DeepVID* through both qualitative and quantitative evaluations, as well as case studies conducted together with deep learning experts.

2 Related Works

We introduce the relevant works on both model interpretation and visual analytics, in this section, to help our subsequent discussions.

DNN-Structure Aware Interpretation. There are three categories of techniques to interpret DNN classifiers using their network architecture (DNN-structure aware interpretation approaches). The *first* category is the gradient-based explanations (e.g., sensitivity analysis [7], guided back-propagations [9]), which compute the gradient of the network output (class probabilities) in respect of the network input (image pixels), layer-by-layer, and investigate how the output is produced from certain regions of the input. *Second*, the layer-wise relevance propagation approaches propose different policies [10], [11] to decompose the network output and redistribute them (layer-by-layer) back to the input to derive the positive/negative contribution of the input pixels. *Third*, the Class Activation Map (CAM) based approaches (e.g., CAM [12], Grad-CAM [17], Grad-CAM++ [18]) combine the extracted features (i.e., activations of the last convolutional layer) and the class-related weights (i.e., weighs of the last fully connected layer) to relate class probabilities with input pixels. All the above interpretation approaches rely heavily on the architecture of neural networks. However, if the architecture is unknown, which happens in most cases of real-world applications, they are no longer applicable.

Localized Model-Agnostic Interpretation [13], [14]. Another category of techniques interprets a classifier by approximating the classifier's local behavior around a particular data instance of interest with a simple interpretable model (e.g., linear regressions, decision trees). These techniques are model-agnostic, as only the input and output of the original classifier are used to train the simple model and the internal architecture of the classifier is not required. The most notable examples include LIME [8] and meta-predictors [13]. One common limitation of the techniques in this category is the way of generating training data for the small interpretable model. To better localize the training samples around the data instance of interest, they are often generated by

perturbing the instance of interest *in the image space*. For example, LIME decomposes the image of interest into many fractions and considers different combinations of those fractions as local neighbors of the interested instance for training. The approach of meta-predictors blurs a particular region of interest and probes the original classifier's behavior. However, those perturbed neighbors (combinations of image fractions, partially blurred images) are not natural images and lack of semantic meanings, which may bias the training of the interpretable model. Also, these approaches perturb the data in the input space which is not efficient, as the dimensionality of the input is usually very high. Our proposed approach is also model-agnostic. However, we take advantage of the recent advances in deep generative networks to generate localized neighbors with semantic meanings. Our approach is more efficient as we use the generative model's *latent space* to sample neighbors, which is much smaller than the input image space.

Visual Analytics on Deep Learning Models. The recent attempts of applying visual analytics for explainable deep learning can roughly be categorized into two groups: the model-based and the example-based. The model-based solutions focus on revealing the internal architecture of a DNN and the interaction between different components [19], [20], [21]. For example, CNNVis [19] models a CNN as a directional acyclic graph (DAG) and clusters the graph nodes (i.e., neurons) with similar activations to reveal the model's behavior. Conversely, the example-based solutions focus on interpreting the behavior of DNNs with specific data instances as inputs [22], [23]. For example, ActiVis [23] compares the activations from different data instances (i.e., examples) to investigate the potential causes of misclassifications. The combination of both solutions has also been witnessed in recent visual analytics works on interpreting deep learning models [24], [25]. For example, GANViz [24] investigates how the discriminative network of a GAN model can differentiate a pair of real and fake images, by tracking the data pair through the network and comparing them layer by layer. Our visual analytics framework proposed in this work, i.e., *DeepVID*, is an example-based solution. It brings together the power of visual analytics and the state-of-the-art deep learning solutions to interpret a cumbersome classifier's behavior around a particular data instance of interest. Moreover, *DeepVID* is a model-agnostic approach, making it a more general solution for interpreting different types of classification models.

3 Background

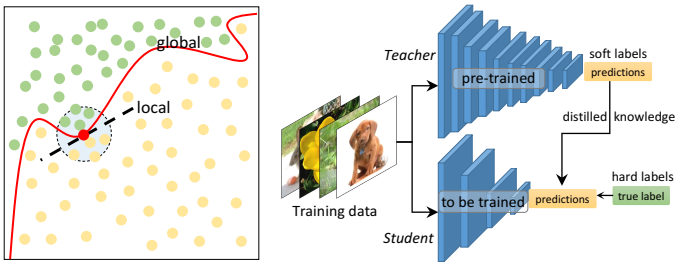


Fig. 2: Left: local interpretation; right: knowledge distillation [15].

Interpretation via Localization. The idea of localization is to approximate the behavior of a classifier at a point of interest using a simple interpretable model [8]. Fig. 2 (left) shows an example: the data points from two different classes are colored in green and orange; a classifier's decision boundary is denoted by the red curve; and the red point represents the sample of interest. Although interpreting the red curve (i.e., the complicated global boundary of the classifier) is difficult, the classifier's local behavior around the sample of interest can be explained with a simple linear model (the black dotted line), which is trained using the samples in the shaded region (neighbors of the sample of interest). Our approach adopts a similar idea of local approximation to interpret and diagnose a complicated model, but improves the localization strategy with a deep generative approach to enhance the efficiency and accuracy.

Knowledge Distillation [15] was proposed to compress cumbersome models into light-weighted models for various purposes (e.g., simplifying the deployment of a model). The idea is to train a small model (*Student*) using knowledge distilled from a pre-trained cumbersome model (*Teacher*). A commonly used method is to match the logits from the two models, such as the example shown in Fig. 2 (right). The *Teacher* is a pre-trained DNN, which takes images as input and outputs their class probabilities. The *Student* is also a DNN but contains much fewer layers, which can be trained using the same image inputs. The *Student*'s training loss contains two parts, which minimize the difference between the predicted labels and the true labels (i.e., *hard labels*), as well as the predictions from the *Teacher* (i.e., *soft labels*). The soft labels (relative probabilities of different classes) provide rich information about how the *Teacher* thinks of the input data. For example, a 'truck' image may have some probabilities to be misclassified as a 'car', but very small chances to be misclassified as a 'dog'.

Mathematically, we can denote an image dataset as $\{X,Y\}$ (X: images; Y: image labels). By feeding an image $x \in X$ into the *Teacher* and *Student*, we get two sets of logits $\boldsymbol{Z_t}$ and $\boldsymbol{Z_s}$. The prediction (i.e., probability distribution) from the *Teacher* and *Student* can be denoted as $\boldsymbol{P_t}=softmax(\boldsymbol{Z_t}/T)$ and $\boldsymbol{P_s}=softmax(\boldsymbol{Z_s}/T)$, where T is the distillation temperature controlling the entropy in $\boldsymbol{P_t}$ and $\boldsymbol{P_s}$ ($T=1$ by default). The `softmax` function is defined as: $softmax = \frac{e^{y_i/T}}{\sum_i e^{y_i/T}}$. The training loss for the *Student* is:

$$l_s = \alpha L_{hard}(\boldsymbol{P_s}, \boldsymbol{y}) + \beta L_{soft}(\boldsymbol{P_s}, \boldsymbol{P_t}), \quad (1)$$

where α, β are two coefficients, $\boldsymbol{y}$ is the true label (hard label) for x (a one-hot vector), and L_{hard}, L_{soft} are measured by cross-entropy. Increasing the value of T will increase the entropy in $\boldsymbol{P_s}$, and thus enhance the *Student* to learn the relative probabilities of different classes from the pre-trained *Teacher*. However, if T is too large, the probability of irrelevant classes will also be over-emphasized.

In this work, instead of compressing models, we adopt the idea of knowledge distillation to distill and transfer knowledge between models for the purpose of interpretation. In the traditional model compression applications, the *Student* is usually required to share similar model natures (e.g., network structures) with the *Teacher*. In our case, however, as the purpose is to locally approximate but not fully mimic a *Teacher*'s behaviors, the *Student* can be much simpler and more explainable to serve our interpretation goal.

Variational Auto-Encoder (VAE) is an unsupervised neural network model, which learns a latent representation of the training data and then reconstructs the data from the learned representation. It consists of two sub-networks of an encoder and a decoder (Fig. 3). The encoder compresses the input $x \in X$ into a latent vector z, i.e., $z = encoder(x) \sim q(z|x)$; whereas the decoder reconstructs an image x' from z, i.e., $x' = decoder(z) \sim p(x|z)$. The training of VAE is conducted by: (1) minimizing the difference between x and x'; and (2) limiting the distribution of z to be a unit Gaussian distribution, i.e., $p(z)=\mathcal{N}(0,1)$. The training loss of a VAE is:

$$l(\theta,\phi) = -E_{z\sim q_\theta(z|x)}[\log p_\phi(x|z)] + KL(q_\theta(z|x)||p(z)), \quad (2)$$

where θ and ϕ are the trainable parameters. The power of VAE falls into two folds: (1) capturing the complicated data features from the input space and compressing them into a smaller latent space; (2) generating unseen samples with meaningful semantics in an on-demand fashion. Our method leverages these two strengths to efficiently generate local samples to probe a model's behavior.

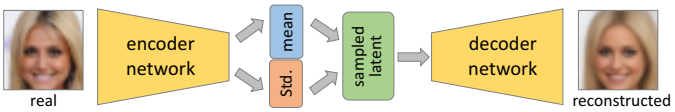


Fig. 3: The architecture of VAE models.

4 Framework Overview

With the power of knowledge distillation and VAE, we propose a new model interpretation framework, i.e., *DeepVID* (Fig. 4).

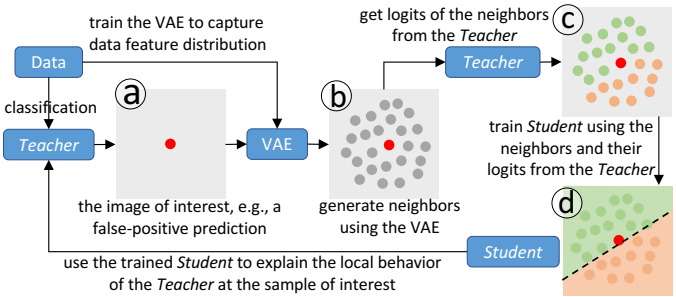


Fig. 4: Overview of the *DeepVID* framework.

With a dataset and a classification task, a complex model (i.e., *Teacher*) is trained and to be interpreted. Also, a VAE model is pre-trained to capture the feature distribution in the dataset. Now, given a data instance of interest identified based on the *Teacher*'s performance (e.g., a false-positive image), shown as the red point in Fig. 4a, we use the pre-trained VAE to generate semantically meaningful neighbors around the instance (gray points in Fig. 4b). Those neighbors are then fed to the *Teacher* to probe its behavior and distill its knowledge (i.e., deriving logits, Fig. 4c). Next, we train a linear *Student* model using those neighbors and their logits from the *Teacher* (i.e., soft labels, note that those neighbors have no hard labels and only soft labels are used to train the *Student*) to mimic the behavior of the *Teacher* (Fig. 4d). Finally, by visualizing the *Student*, we can interactively interpret the *Teacher*'s behavior on the data instance (e.g., why the image instance is misclassified).

Algorithm 1 explains our interpretation framework with more details. From the image of interest, x, we first derive its latent vector, ***lat_x***, using the encoder of the pre-trained VAE (line 1). Instead of sampling the input space (i.e., perturbing image x), which is performed in the previous works [8], [13], to generate neighbors, we sample the latent space (line 7) and use the VAE decoder to reconstruct neighbors from the sampled latent vectors (line 8). The sampling is centered at the latent vector of the interested image and within a user-specified percentage (*Sampling_Percent*) of the entire latent space (***Ranges***). This neighbor generation process is further explained with details in Section 6.2 (several alternative neighbor generation methods, e.g., k Nearest Neighbors search (kNN), have also been discussed and compared in our supplementary material). With those generated neighbors, as well as their soft labels from the *Teacher* (line 9), we train the linear *Student* with temperature T (line 13). Finally, the trained *Student* (i.e., coefficients of the model) will be visualized (line 14) to interpret the *Teacher*'s behavior at x.

Algorithm 1 Pseudocode for our *DeepVID* framework.

Require: *Teacher*: the classifier to be interpreted
Require: *VAE_encoder*, *VAE_decoder*: pre-trained VAE model
Require: x: data instance of interest (an image)
Require: *num_neighbors*: the number of neighbors needed
Require: T: temperature value to train the *Student*
Require: ***Ranges***: the value ranges of all latent dimensions
Require: *Sampling_Percent*: sampling within this percentage

1: ***lat_x*** $=$ *VAE_encoder*(x) // derive the latent vector for x
2: ***band*** $=$ ***Ranges***$\times$*Sampling_Percent* // the sampling band
3: ***interval*** $=$ ***band***$/$*num_neighbors*
4: *neighbors* $=[\,]$ // training data for the *Student*
5: *neighbors_logits* $=[\,]$ // labels for the training data
6: **for** i=0; i<*num_neighbors*; i++ **do**
7: ***lat_nbr***=***lat_x***−***band***$/2+i\times$***interval*** //sampled latent vector
8: *neighbor* $=$ *VAE_decoder*(***lat_nbr***)
9: ***logit*** $=$ *Teacher*(*neighbor*)
10: *neighbors.append*(*neighbor*)
11: *neighbors_logits.append*(***logit***)
12: **end for**
13: *Student* $=$ *Linear*(*neighbors*, *neighbors_logits*, T).*train*()
14: *Visualize*(*Student.weights*, *Student.weights*$\times x$)

5 Visual Analytics Design Requirements

To implement the proposed interpretation framework as a visual analytics system, we worked closely with 4 deep learning experts and elicited the following 3 main themes of design requirements:

- ***R1: Visual understanding of the Teacher's behavior.*** The target system should provide a visual understanding of the test data and enable users to explore the prediction results from the *Teacher*. In detail, this theme needs the system to be able to:
  - R1.1: visualize and explore the feature space of the test data;
  - R1.2: show the *Teacher*'s overall performance over the test data, and the probability scores for a specified data instance;
  - R1.3: help users to easily identify the failure cases, such as hard samples (i.e., false-positives) and uncertain samples.
- ***R2: Flexibly explore the HD space of the generated neighbors.*** The quality of the generated neighbors is critical to train a high-quality *Student*, and thus, the system should enable users to:
  - R2.1: understand the semantics encoded in the latent space of the generator (i.e., the VAE). For example, for MNIST handwriting digits, some latent dimensions encode the boldness, and others encode the italic style of the digits;
  - R2.2: interactively adjust the generated neighbors, such that they are sufficiently close to the instance of interest;
  - R2.3: generate training samples between two instances of interest across class boundaries, e.g., generating samples that smoothly morphing from a user-interested digit 4 to a digit 9.
- ***R3: Interactively train and interpret the Student model.*** Tuning and understanding the performance of a *Student* should be an iterative process, and thus, the system needs to help users:
  - R3.1: understand the performance of the *Student* model and compare it with the *Teacher*'s performance;
  - R3.2: interpret the parameters learned by the *Student* model;
  - R3.3: interactively adjust the training hyper-parameters, such as the distillation temperature, optimizer, epoch number, etc.

6 Visual Analytics System: *DeepVID*

Following the design requirements, we designed and developed *DeepVID* with three modules, visualizing the information of the *Teacher* (R1), the VAE (R2), and the *Student* (R3) respectively.

6.1 Teacher View

The Teacher view (Fig. 1a) demonstrates an overview of all data instances in test, along with the *Teacher*'s prediction performance on specified instances. It contains four juxtaposed sub-views.

The t-SNE view projects all test images into 2D and presents them as a scatterplot (Fig. 1-a1), so that users can overview their clustering result and select groups of similar image instances on-demand (R1.1). Each point represents one image, and the point is colored by the true label of the image. The position of points is from the t-SNE algorithm [26], where the activations from the last hidden layer of the *Teacher* model are used as the t-SNE input. The projection provides an overview of all data samples with similar ones clustered together. Through lasso selections, users can select instances into the Image-Grid view (Fig. 1-a2) to visualize them.

The Confusion Matrix view (Fig. 1-a3) presents the confusion matrix of the *Teacher* derived from the test data (R1.2). The value at the ith row and jth column indicates the number of data instances of class i but predicted as class j. The cells along the diagonal are the correctly classified instances, and other cells are wrongly classified ones. The color from white to red encodes the cell values from small to large with the logarithmic scale. Clicking on any cell will select the images falling in that cell into the Image-Grid view, from where users can have a quick overview of the images (R1.3).

The Image-Grid view (Fig. 1-a2) shows the selected images (either through lasso selections in the t-SNE view or by clicking cells in the Confusion Matrix view) without overlap. To effectively use the limited screen space, images with resolutions larger than 32×32 are scaled down to 32×32. This view is also scrollable if all images cannot be presented in the limited screen space. Hovering over any image in this view will highlight the corresponding point in the t-SNE view to help users build connections between views. Clicking on any image will update the Probability Distribution view (Fig. 1-a4) to show the *Teacher*'s prediction on it.

The Probability Distribution view demonstrates the *Teacher*'s prediction for the selected image as a bar chart (Fig. 1-a4, R1.2). The true label of the selected image is highlighted in red texts.

6.2 VAE View

The VAE view empowers users to explore the feature space (i.e., the latent space from the VAE) of the data, and presents an overview of the generated neighbors to reveal the feature trend in them.

Parallel Coordinates Plot (PCP) view. The latent vector for the image of interest is a HD vector (10D in Fig. 1-b1). We present it with a PCP, which can effectively visualize HD data and allow users to interactively explore different dimensions of the corresponding HD space (i.e., the dragging interaction explained later). Each axis of the PCP shows the available range of the corresponding latent dimension and the red rectangle on the axis marks the value of the current latent vector on that dimension. The polyline connecting the red rectangles across all axes represents the latent vector of the currently selected image. Users can drag the red rectangle along each axis to see how the corresponding latent dimension affects the visual appearance of the reconstructed image (i.e., the image with the label "recons" on the left of the PCP), in comparison with the original image, i.e., image 707 in Fig. 1-b1 (R2.1).

The light blue band in the PCP shows the available perturbation range of the 10 latent dimensions. We generate n neighbors of the selected image by: (1) perturbing its 10D latent vector in the band to generate n 10D latent vectors; and (2) reconstructing from those latent vectors (using the VAE decoder) to generate n neighbors. In detail, the band is generated with a user-specified percentage value, which is 15% by default (it can be changed through the interface on the header of the VAE view). For each latent axis, a 15% range centered at the latent value of that axis is generated. Connecting the ranges from all axes forms the perturbation band. Next, n polylines are generated evenly within this band. Currently, we generate them by uniformly sampling 10D vectors within the band (see line 7 of Algorithm 1). This is not the only sampling method, but the most straightforward choice that meets our need. One can definitely resort to more sophisticated methods if necessary (e.g., exhausting all possible combinations of the values from the 10 axes).

If users have obtained a good understanding on the semantics of different latent dimensions (after interacting with the PCP view), they can manually brush each axis to specify the perturbation band, which will control and optimize the generated neighbors (R2.2).

The ***Neighbors view*** demonstrates m (m<=n, m=144 in Fig. 1-b2) evenly sampled images (out of the total n generated neighbors), and those images are sorted from top-left to bottom-right following the increasing order of the corresponding latent vectors (i.e., the ***lat_nbr*** in line 7 of Algorithm 1). From the visual trend of those images, one can easily identify their feature distribution (R2.1).

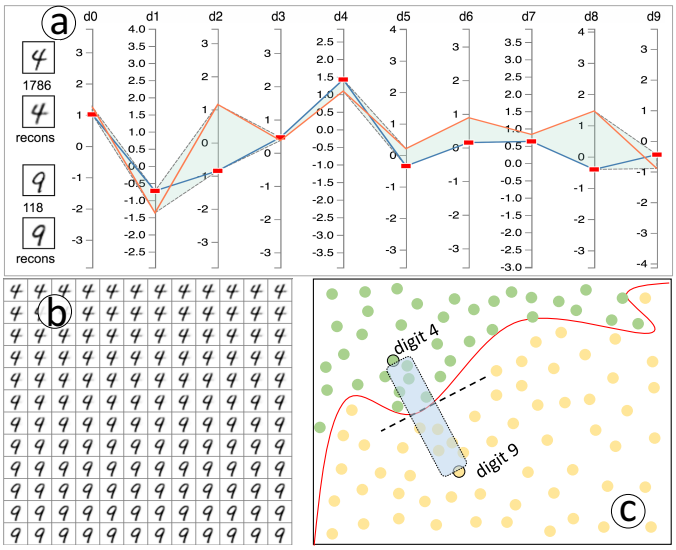


Fig. 5: Morphing from digit 4 (image 1786) to digit 9 (image 118).

Morphing between two images. One can also select two images of interest to investigate how the *Teacher* differentiated them (R2.3). In this case, *DeepVID* generates image samples between the two selected images. In Fig. 5a, image 1786 and 118 (a digit 4 and 9) are selected, and their latent vectors are presented as polylines in blue and orange respectively in the PCP. We generate the perturbation band by connecting the covered ranges (of the two polylines) across all axes, and the perturbed latent vectors are generated by performing a linear interpolation between the two latent vectors. For example, the Neighbors view in Fig. 5b shows 144 generated neighbors, which present how image 1786 is smoothly morphing to image 118. Since the two images belong to different classes, the generated samples are images that across a certain point of the boundary between those two classes (the shaded region in Fig. 5c). The *Student* trained on these samples will be able to explain how the two images are differentiated by the *Teacher*.

6.3 Student View and *DeepVID*'s Interpretation

The Student view helps to train the *Student* with customized hyper-parameters (R3.3). It also visualizes the trained *Student* (R3.2)

and its predictions (R3.1). The top part of this view shows the selected image and the hyper-parameters. For example, in Fig. 1c, the batch size is 128; the temperature is 10; the *Student*'s weights are initialized as zeros; and the optimizer is Gradient-Decent (GD).

The *Student* is a linear model, which takes an image (a set of pixels) as input and outputs a probability value. Mathematically, it can be expressed as $f=softmax(\sum_i w_i a_i + b)$, where a_i are the feature dimensions (pixels), w_i are the weights (coefficients), b is the bias, and f is the produced probability. To interpret f, two types of information should be visualized: (1) the learned weight for each feature/pixel, i.e., w_i; (2) the weighted feature, i.e., $w_i a_i$. Taking the MNIST data as an example, the input and output dimensions of the *Student* are 784 (pixels) and 10 (probabilities) respectively. Therefore, the weights are in the shape of 784×10, which can be considered as ten 28×28 weight matrices. Each matrix works like a template that can be multiplied with the 28×28 input image (pixel-wise) to weight different pixels. Fig. 6 shows the detailed architecture of our *Student*, where the *Teacher* is the LeNet [27] (a CNN). We use different line colors (blue, red, and green) to differentiate different weight matrices of the *Student*.

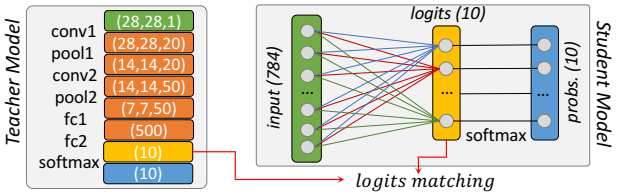


Fig. 6: An example architecture of knowledge distillation.

The Student view uses images to visualize the trained *Student*'s weights and the weighted features. As highlighted in the red box of Fig. 1c, the 10 weight matrices are normalized individually and visualized as the first column of images. The second column of images show the weighted features (i.e., multiplying individual matrices with the image of interest). Images in this column are normalized together to highlight how different pixels contribute to the 10 classes (values from small to large are mapped to colors from black to white). In Fig. 1c, the selected image has high probabilities to be digit 4 and 9 due to the extracted white pixels in the second column of images. Hovering over any image in this view will pop up an enlarged version of that image to present more details.

The Student view also presents the *Student*'s predicted probabilities, which should be as close to the probabilities from the *Teacher* as possible, i.e., the bar chart in Fig. 1c should match with the bar chart in Fig. 1-a4. When the *Student*'s prediction is not similar to the *Teacher*'s prediction, the explanations should not be trusted. In that case, one should try to adjust the hyper-parameters (e.g., modifying the temperature value or the sampling bandwidth) and re-train the *Student* to more accurately mimic the *Teacher*'s behavior. When training the *Student*, *DeepVID* will also print the *Student*'s training loss in the back-end (measured by cross-entropy), i.e., how well the *Student*'s predictions on the generated neighbors agree with the *Teacher*'s predictions, to monitor the training progress.

7 Case Studies with Domain Experts

We worked with four domain experts to explore the power of *DeepVID* using the commonly used image datasets and CNN classifiers. All experts have more than 10 years experience with machine learning and 4∼6 years experience with deep learning.

7.1 Datasets and Network Structure

Three image datasets: MNIST, QuickDraw [28], and CelebA [29] were used in our experiments. Fig. 7 shows the typical images from each class of the three datasets. The MNIST dataset contains 65000 (train:55000; test:10000) handwriting digits in 10 classes (digit 0-9), and each image instance is a gray scale image with the resolution of 28×28. The QuickDraw dataset also has 10 image classes, which we extracted from the Google Quick-Draw game [28]. Each class in this dataset has 11000 image instances. As a result, there are 110000 gray scale images in total (train:100000, test:10000), and each image is also of size 28×28. To demonstrate the scalability of our approach, we have also worked with the CelebA dataset, which contains 202599 (train:192599; test:10000) human face images (in the resolution of 178×218) with annotations of 40 binary attributes per image. Those RGB images are first cropped into 148×148 and then rescaled to 64×64 for our usage. We randomly selected 3 binary attributes ("Male", "Blond Hair", "Glass") to separate the 202599 images into eight (2^3) classes. For example, the first class is the images that have false values on all the three binary attributes.

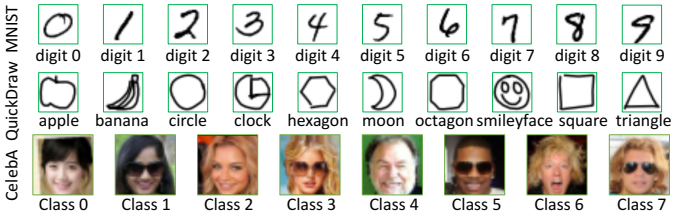


Fig. 7: The experimental data: MNIST, QuickDraw, and CelebA.

Two CNN classifiers, LeNet [27] and VGG16 [3], were often used by our domain experts, and we used *DeepVID* to interpret their behavior on the above datasets. As shown in Table 1, the MNIST and QuickDraw datasets were used to train two LeNets and achieved accuracy 98.57% and 89.73% respectively. The VGG16 model is more complicated (deeper) than the LeNet. It was used to train the CelebA dataset and achieved accuracy 93.74% after 100 training epochs. For MNIST and QuickDraw, the feature space is relatively small. A VAE with two fully connected layers [30] and a 10D latent space is able to capture the feature space (Fig. 8, left). For CelebA, however, the feature space is much larger (64×64×3 dimensions). After several attempts, we finally adopted a four convolutional layers DFC-VAE [31] to capture the complicated feature space (Fig. 8, right). DFC-VAE (Deep Feature Consistent VAE [32]) is a variant of the regular VAE. It improves the reconstruction performance by feeding both the input and reconstructed images into a pre-trained CNN model, and minimizing the difference between activations from all the intermediate layers. Moreover, for CelebA, we used 100D latent vectors to preserve more information for better reconstructions.

TABLE 1. Details of different *Teacher* and VAE models.

Dataset	*Teacher*	Epochs	Accuracy	Generative Model	Latent	Epochs
MNIST	LeNet	20	98.57%	regular VAE	10D	150
QuickDraw	LeNet	20	89.73%	regular VAE	10D	150
CelebA	VGG16	100	93.74%	DFC-VAE	100D	50

7.2 Interpreting the *Teachers* with Domain Experts

Overview Teacher's Behavior. The experts first explored with the MNIST dataset to understand the performance of the LeNet (*Teacher*). From the t-SNE view (Fig. 1-a1), which lays out all data instances using their last hidden layer representation from the *Teacher*, the experts knew details on the *Teacher*'s performance

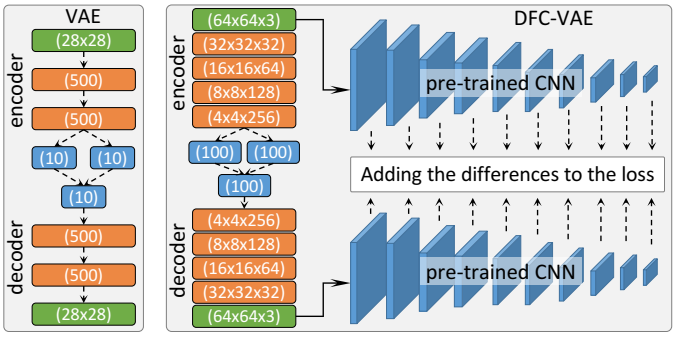


Fig. 8: The structures of our VAE and DFC-VAE models.

beyond a single accuracy value (R1.1, R1.2). For example, from the clear separation of the 10 clusters, the experts became aware of the good performance of the *Teacher*. By hovering over different data points, they realized what digits different clusters represented for. They commented that digit 4 and 9 might be more difficult to be differentiated, as the purple and brown cluster (clusters for digit 4 and 9) are much closer than others. The Confusion Matrix view (Fig. 1-a3) echoes the good performance (R1.2), as most of the data instances fall into cells along the diagonal (i.e., true-positives).

Explore Test Data and Identify Instance of Interest. Next, the experts selected data instances along the boundary of the purple and brown cluster in the t-SNE view to analyze the uncertain samples along the decision boundary between digit 4 and 9. The juxtaposed Image-Grid view (Fig. 1-a2) visualizes the selected instances as images (R1.1). From there, the experts randomly clicked different images to explore their probability predicted by the *Teacher* (R1.2). For example, Fig. 1-a4 shows the probability of a misclassified digit whose true label is digit 4 (highlighted in red) but misclassified as digit 9 with 59.75%. The experts were very interested in why the LeNet misclassified such a normal-looking digit 4 into digit 9 (R1.3). They then selected it for further interpretation.

Visualize the Latent Space and the Generated Neighbors. Fig. 1b shows the VAE view for the selected digit 4 (i.e., image 707 shown on the left of this view). The latent vector of this selected digit is visualized in the PCP (Fig. 1-b1), and the 144 evenly sampled neighbors (out of the total 512 neighbors) generated within the 15% sampling band are shown in the Neighbors view (Fig. 1-b2). From the visual appearance of those neighbors, the experts saw the smooth transition from digit 4 (top-left) to 9 (bottom-right). Fig. 9 shows our post-analysis on the probability of all the 512 generated neighbors predicted by the *Teacher* (the horizontal axis aligns the 512 neighbors, whereas the vertical axis shows their probabilities). From left to right, we can see that the probability of digit 4 is decreasing while the probability of digit 9 is increasing, indicating the VAE did a good job in generating neighbors from one side of the *Teacher*'s decision boundary (between 4 and 9) to the other side. Those neighbors' probabilities for other classes are very low.

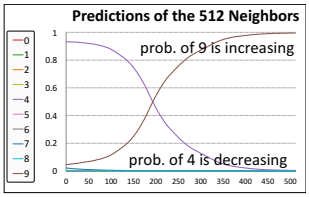


Fig. 9: Probabilities of the 512 generated neighbors.

Interpreting Misclassifications. Fig. 1c shows the trained *Student* using the 512 generated neighbors. Initially, the experts set the temperature as 2 when training the *Student*. As the predicted probability from the *Student*, i.e., 97.24%/2.76% to be digit 4/9 (Fig. 10a), is not very similar to the *Teacher*'s prediction, i.e., 40.05%/59.75% to be digit 4/9 (Fig. 1-a4), they tried to adjust the temperature value. After some explorations, they found that the *Student* could achieve similar prediction performance when the temperature was 10 (Fig. 10b, R3.1, R3.3). From the weighted features (the second column of images in Fig. 10b), the experts understood that the misclassification is because the highlighting of pixels in the top of digit 9 (comparing Fig. 10-b1 and 10-b3) and the suppression of pixels in the little tail of digit 4 (comparing Fig. 10-b2 and 10-b4, R3.2). The experts also trained the *Student* with different optimizers and initializers (R3.3), and the results are shown in Fig. 10c-e. It can be seen that the *Student* achieved similar performance with the *Teacher* in all the three configurations (see the probabilities in Fig. 10c-e). Although the trained weight matrices are different (due to different parameter update mechanisms), the weighted features (the second column of images) are similar.

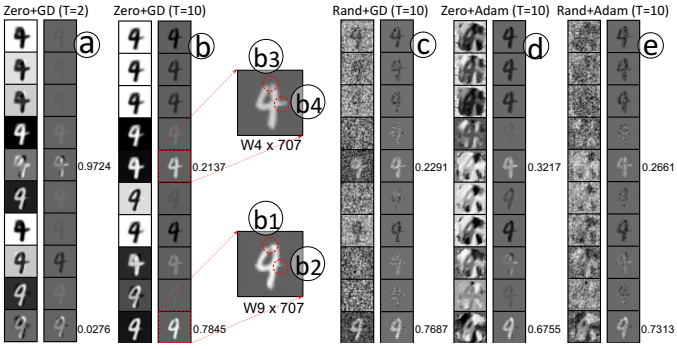


Fig. 10: The effects of different hyper-parameters.

Additionally, the experts explored different misclassified images by directly selecting them from the Confusion Matrix view (clicking non-diagonal matrix cells). Fig. 11 shows the case that the *Teacher* misclassified a bold digit 1 (image 9071) into digit 8. As shown in Fig. 11c, the *Teacher* thinks the image has 41.78% to be digit 1 and 57.73% to be digit 8. Fig. 11a shows the generated neighbors around image 9071, and Fig. 11b shows the interpretation from the trained *Student* with the given hyper-parameters. The experts observed that the 10 trained weight matrices (the first column of images in Fig. 11b) work like 10 masks, which can enhance or suppress certain pixels when being multiplied with the image of interest. For example, the weight W_8 enhances most pixels of the bold digit 1, but suppresses pixels in the two holes of digit 8. When multiplying it to image 9071, the bold digit 1 looks more like a digit 8, as the enlarged view shown in the bottom-right of Fig. 11.

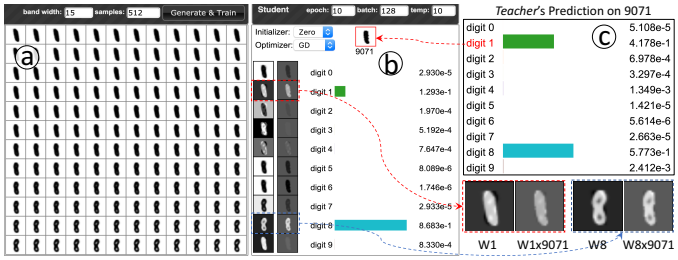


Fig. 11: Explain why digit 1 (image 9071) is classified as digit 8.

Explore the Data Manifold and Control Neighbor Generation. In certain cases, the reconstructed image may not be similar enough to the image of interest (depending on the VAE quality). In those cases, users might want to explore the feature space to manually tune the sampling band and generate qualified neighbors. For example, in Fig. 12a, a true-positive image of digit 7 (*Teacher*'s probability 99.83%) is selected for analysis. The reconstructed

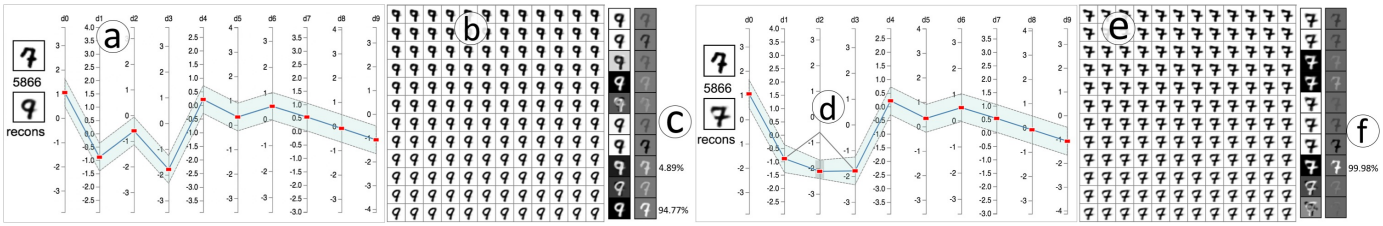


Fig. 12: *DeepVID* allows users to flexibly brush the PCP axes to change the perturbation/sampling band, and generate better neighbors.

image, however, is more similar to a digit 9, so as to the generated neighbors around it in Fig. 12b. When feeding those neighbors to the *Teacher* (to derive logits), the *Teacher* recognized most of them as digit 9. As a result, the trained *Student* incorrectly predicted the image of interest as a digit 9 with probability 94.77% (Fig. 12c). The interpretation from the *Student* cannot be trusted, as the behavior of the *Student* does not match with the *Teacher*.

To help the *Student* better mimic the *Teacher*'s behavior, the experts used *DeepVID* to explore the feature space of the data and generate better neighbors (R2.1, R2.2). As shown in Fig. 12d, the experts dragged the red rectangle on each latent dimension to see its effects to the reconstructed image. From the exploration, they found that adjusting "d2" made the reconstructed image more similar to the image of interest (image 5866). As a result, they manually brushed a region on "d2" to change the sampling band, and the reconstructed neighbors from the new band are shown in Fig. 12e. By training the *Student* with those new neighbors, the *Student* could produce almost the same prediction with the *Teacher* (Fig. 12f), and the *Student*'s interpretation is more meaningful.

Differentiate Two Images of Interest. As explained in Fig. 5, *DeepVID* allows users to select two images and interpret how the classifier differentiated them (R2.3). We worked with the experts to exploit this function using the QuickDraw dataset. Fig. 13 shows how the classifier (LeNet) differentiated a triangle from a square (image 4167 and 7386). First, the samples that morphing from the triangle to the square are generated (Fig. 13a), and they are then fed to the *Student* to train the 10 weight matrices (the first column of images in Fig. 13b). The triangle and square are the 9th and 8th class in QuickDraw. So, W_9 and W_8 are more import (Fig. 13c).

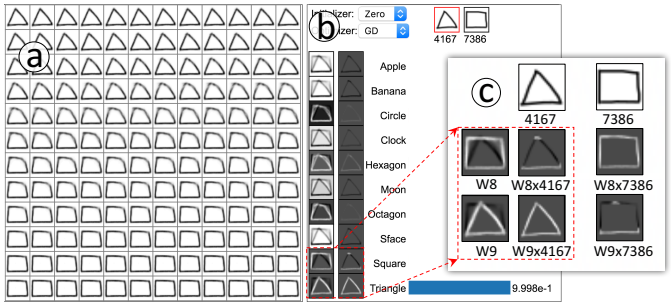


Fig. 13: How a triangle and a square are differentiated.

We found that W_9 enhanced the three edges of the triangle and suppressed the top three edges of the square; whereas W_8 enhanced the four edges of the square and suppressed the top two edges of the triangle. When multiplying W_8 with the triangle image ($W_8 \times 4167$), only the bottom edge was extracted and the probability of being a square was very low. However, when multiplying W_9 with the same image ($W_9 \times 4167$), all three edges of the triangle were extracted and the probability of being a triangle was very high. On the other hand, when multiplying W_8 with the square image ($W_8 \times 7386$), all four edges of the square were extracted and the probability of being a square was very high. However, when multiplying W_9 with the square image ($W_9 \times 7386$), only the bottom edge was extracted, and consequently, the probability of being a square was very low. The experts observed the weight matrices were like masks that only allow pixels in certain regions of the input image to pass through.

Explore Larger Datasets. We have also explored the more complicated CelebA dataset with the experts. Fig. 14 shows an example of interpreting how image 8170 is classified as "Female, Blond Hair, No Glass" (class 2). The image in comparison (image 1682) is from class 0 ("Female, No Blond Hair, No Glass"), and the feature variance of the two images is the existence of the blond hair or not. The Neighbors view on the left shows the images generated between the two images and we can see the smooth transition from the face with black hair to the face with blond hair. The enlarged second weight matrix (learned by the *Student*) works like a mask, which allows the blond hair from image 8170 to pass through it and blocks the rest pixels. This interpretation made the experts feel confident in trusting the classifier as the hair features were in use.

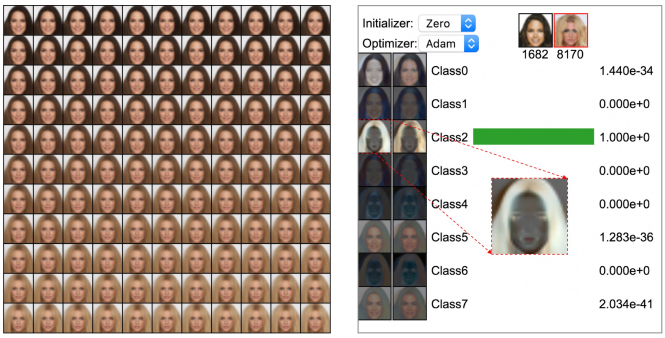


Fig. 14: *DeepVID* identifies the blond hair feature in image 8170.

When exploring the CelebA dataset, the experts also found one limitation in terms of scaling *DeepVID* to larger datasets with higher-dimensional feature spaces. For example, when differentiating a face image with glasses and a face image without glasses, if those images also have feature variances in other regions (e.g., one with hat and one without), those variances could disturb the training of the *Student* from focusing only on the class-sensitive features (see our supplementary material). We discussed this limitation with the experts and brain-stormed potential remedies of using advanced generative models to better control the neighbor generation process.

7.3 Domain Experts' Feedback

We conducted open-ended interviews with the four deep learning experts (E1∼E4). All experts gave their positive feedback on both the interpretation framework and visual analytics system of *DeepVID*, towards helping them interpret and diagnose their models. Their comments and suggestions are summarized as follows.

First, we observed that the experts' exercise with *DeepVID* deepened their understanding and confidence on the approach of local approximation for model interpretation. E1 explained that the

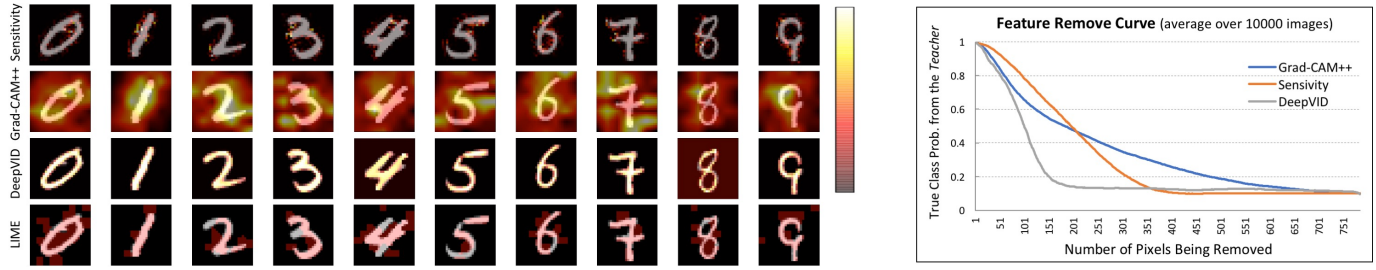


Fig. 15: Left: qualitative comparisons by overlaying the pixel-importance map (derived from different approaches) onto the image in analysis. Right: quantitative comparisons using feature-remove curves (averaged over curves for all the 10000 images from the test data).

case studies "*really helped me understand how and why the local approximation works*", after exploring the visual patterns of the generated neighbors around the interested data point, as well as the corresponding prediction results from both the DNNs and the linear explainer. E3 commented that "*this (DeepVID) is a strong case showing what a weak leaner (the linear explainer) can do (fitting local patterns) and cannot do (capture the overall data patterns)*".

All experts agreed that the visual explanations provided by the *Student* are "*insightful and actionable*". E1 expressed that "*the weighted features are very straightforward to show how the meaningful pixels contributed to the predictions*", and "*they are even more important than the trained weights (coefficients)*". E4 commented that "*this (DeepVID) helps me re-think the assumption about feature distribution, and we do need more data on some rare cases*". E3 was inspired to "*adopt a boosting method that classifies easy instances first and then works on the hard ones*", after observing that some contributing features of the misclassified images were also hard to be captured by the original models.

Notably, the VAE view attracted the most attention from the experts and raised some thoughtful discussions. One observation resonated among all experts is that the neighbor generation process is "*critical to the interpretation performance*", and they commented that "*sampling from the latent space (instead of the input space) is a nice trick*". The experts "*enjoyed*" exploring the VAE latent space and observing the reconstructed images to make sense of the semantics encoded in the latent space. E3 even developed his own sampling strategies with the latent space in an iterative fashion and commented that "*a more efficient way to generate samples is to fix some features, which did not contribute to the prediction in my previous exploration, and then adjust other dimensions*". E1 also suggested other efficient sampling strategies by "*identifying some correlated latent dimensions first to shrink the sampling space*".

The experts also offered insightful suggestions. E3 discussed the feasibility of a hybrid approach that combines both model-agnostic and model-specific techniques. He commented that "*DeepVID cannot offer insight into improving the neural network structures*", and it may be helpful to have "*a layer-by-layer local approximation to show interesting findings on layer designs, such as different choices of filter sizes, activation functions, etc.*" E2 suggested to provide more flexible exploration to understand the feature space of the generated samples, and help users make sense of the semantics encoded in the latent space with some representative samples.

8 Evaluation and Comparison

Qualitative Evaluation. We compared the interpretation generated from *DeepVID* (i.e., the second column of images from the Student view) with three existing solutions: sensitivity analysis [7], Grad-CAM++ [18], and LIME [8]. When interpreting the behavior of an image classifier, most approaches generate a map that colors individual pixels based on their importance to the final predictions (of the corresponding class). By overlapping the map with the image of interest, we can visually quantify if the map really captures the class-sensitive features. Following this, we randomly selected 10 images from the 10 classes of MNIST, and generated their pixel-importance maps (Fig. 15, left). The foreground and background of the original MNIST images are shown with gray and black color respectively, whereas the pixel-importance maps are overlaid on those images. For sensitivity analysis, Grad-CAM++, and *DeepVID*, the color map [black-red-white] reflects pixels with the increasing importance. LIME only outputs binary results, i.e., positive to the class or not, and the positive pixels are colored in red. From the comparison, we can see that *DeepVID* can more accurately extract the gray pixels (the foreground strokes) of different digits.

Quantitative Evaluation. To eliminate the potential cherry-picking comparisons and have a more rigorous evaluation, we conducted the quantitative evaluation using feature-remove curves. This comparison is conducted by sequentially removing the most important pixel from an image based on its order in the pixel-importance maps derived from different approaches, and feeding the modified image into the *Teacher* to see its predicted probability. The horizontal axis of the feature remove curve is the number of pixels being removed, whereas the vertical axis is the *Teacher*'s probability (for the true class). We generate such a curve for each test image in the MNIST dataset and average all those curves (10000 curves) derived from individual methods, as shown in Fig. 15 (right). A sharper drop at the beginning of the curve means more important features are captured and removed, indicating a more accurate interpretation. The curve trend in the later stages (e.g., after 200 pixels being removed) is not important, as the image has been modified significantly. From the comparison, the curve of our method drops the fastest (before 200 pixels being removed). The sharper drop also reflects the consensus between the *Student* and *Teacher* on class-sensitive pixels. LIME is not compared here, as it only outputs binary results (i.e., no order for the pixels).

9 Discussion, Limitations, and Future Work

Latency on Training the Student. *DeepVID* is implemented using Python and Javascript, and the Flask library [33] is used to enable the communication between them. The online training of the *Student* is conducted on the Python side using TensorFlow. As the *Student* is simple, its training latency is usually manageable. For MNIST and QuickDraw, the latency of generating 512 neighbors and training the *Student* for 10 epochs is less than 0.5 second on a modern laptop. For CelebA, the same experiment costs around 19 seconds on average. Dynamically revealing the training progress (e.g., a progress bar) could improve users' experience, especially when generating more neighbors or working with larger datasets.

Hyper-Parameters Tuning. When training the *Student* with *DeepVID*, users need to tune a few hyper-parameters: the perturbation bandwidth, the number of generated neighbors, the number of training epochs, the batch size, the distillation temperature, the type of weight initializer and model optimizer. The default settings for them are: 15%, 512, 10, 128, 10, the random initializer, and the GD optimizer, respectively. These default values work well in most cases of our explorations, except that the temperature value needs to be adjusted frequently. To alleviate the issue of manually adjusting these parameters, some hyper-parameter tunning approaches can be adopted in the future. For example, a naïve grid search algorithm could be very helpful, considering the rather small searching space of the hyper-parameters in *DeepVID*. Apart from these parameters, we have also explored the effects of different regularizers with the guidance from the domain experts. The corresponding results are discussed with details in our supplementary material.

Neighbors Generation. There are two categories of neighbor generation approaches in general. The first category generates neighbors through perturbations in the original data space, like LIME and *k*NN (searching neighbors in the original data space). This group of approaches can be easily conducted, but the number of feature dimensions is usually large and the generated neighbors may not be semantically meaningful. The second category of approaches encodes the original data into a new space (e.g., latent or frequency space) and generates neighbors through perturbations in that space. *DeepVID* is in this category, where it uses a VAE to build the mapping between the original and the latent space. Usually, the latent space is much smaller and the latent dimensions are semantically meaningful. Thus, the neighbor generation process is more controllable. However, getting a high-quality data generator is the bottleneck for this category of approaches. With the clear pros-and-cons of these two categories, a hybrid solution seems to be promising. For example, one can perform *k*NN using the latent vectors of all images (rather than the raw pixels). The latent vectors of the identified *k*NNs can also be used to guide the perturbation process in the latent space to generate new neighbors.

Large Latent Space. In certain cases, the dimensionality of the latent space could be very high (e.g., it is 100 when working with the CelebA dataset), which will bring challenges when perturbing and exploring the latent space. Working with domain experts, we have thought of two potential directions to address this problem. The first one is to resort to more advanced generative models (e.g., progressive GAN [34]) to derive latent dimensions with more stable and controllable semantics and group the dimensions based on their semantics. The second direction is to enhance the PCP interface, so that it can progressively learn users' preferences when exploring the latent space and automatically adjust the positions of the control points on different PCP axes. One such successful attempt has been proposed in color enhancement of images [35].

One ***limitation*** of our approach is that it requires a high-quality pre-trained VAE model on the target dataset to provide good interpretations. For example, *DeepVID* could not interpret classifiers for the CIFAR10 dataset [36], as CIFAR10 does not have enough image instances to train a good VAE and capture its complicated feature space. Detailed results on this dataset have been included in our supplementary material. Another limitation that we planned to address in the future is the scalability of *DeepVID*. In some sophisticated real-world applications, the number of input images, latent dimensions, and output classes could be far more than what we have demonstrated. In those cases, some *DeepVID* views will have to be redesigned to satisfy the demanding need.

More Advanced Interpretations. Apart from interpreting the behavior of image classifiers, there are emerging requirements on interpreting and diagnosing more advanced models. We would like to extend *DeepVID* towards two interesting directions in the future. One is to adapt *DeepVID* to interpret adversarial attacks [37]. Generating adversarial features to probe a model's behavior could reveal how vulnerable the model is, and thus further improve it from there. The other direction is to apply *DeepVID* for model comparisons. Domain experts usually have multiple candidate models for an application. With *DeepVID*, they can train smaller explainable models to better interpret and compare those candidates, and thus, making proper decisions for their application.

Other Classifiers and Other Data Types. We focus on interpreting DNN classifiers in this work. However, as *DeepVID* does not rely on the classifiers' internal structure (only the input and output are needed for knowledge distillation), other non-DNN classifiers (e.g., SVM, random forest) could be plugged into it for interpretation. Apart from extending *DeepVID* to non-DNN classifiers, we also plan to explore *DeepVID* with other types of data in the future. For example, we plan to work with text data to investigate which word in a particular sentence influences the sentiment of the sentence. This is very similar to examine which pixel of an image influences the classification result of the image.

10 Conclusion

In this work, we propose *DeepVID*, a visual analytics system that takes advantages of two deep learning solutions to perform visual interpretation for image classifiers. The first solution of knowledge distillation guided us to train a simple interpretable model using the knowledge distilled from a cumbersome classifier to interpret the classifier itself. The second deep learning solution of generative models helped us generate semantically meaningful neighbors around the data instance of interest, which can effectively probe the behaviors of the cumbersome model and train the small explainer. By virtue of the interactive power of *DeepVID*, we conducted thorough studies with domain experts, compared it with existing model interpretation solutions, and validated its effectiveness.

Acknowledgments

This work was supported in part by UT-Battelle LLC 4000159557, Los Alamos National Laboratory Contract 471415, and NSF grant SBE-1738502.

References

[1] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[2] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE transactions on systems, man, and cybernetics*, vol. 21, no. 3, pp. 660–674, 1991.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[6] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1625–1634.

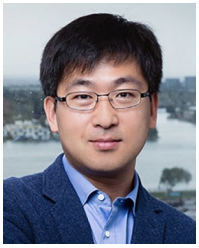
[7] G. Montavon, W. Samek, and K.-R. Müller, "Methods for interpreting and understanding deep neural networks," *Digital Signal Processing*, 2017.

[8] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?: Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2016, pp. 1135–1144.
[9] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv:1412.6806*, 2014.
[10] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PloS one*, vol. 10, no. 7, p. e0130140, 2015.
[11] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller, "Explaining nonlinear classification decisions with deep taylor decomposition," *Pattern Recognition*, vol. 65, pp. 211–222, 2017.
[12] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2921–2929.
[13] R. C. Fong and A. Vedaldi, "Interpretable explanations of black boxes by meaningful perturbation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3429–3437.
[14] J. Zhang, Y. Wang, P. Molino, L. Li, and D. S. Ebert, "Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models," *IEEE Trans. on Vis. and Comp. Graphics*, vol. 25, no. 1, pp. 364–373, Jan 2019.
[15] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.
[16] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.
[17] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization." in *ICCV*, 2017, pp. 618–626.
[18] A. Chattopadhyay, A. Sarkar, P. Howlader, and V. N. Balasubramanian, "Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks," *arXiv preprint arXiv:1710.11063*, 2017.
[19] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu, "Towards better analysis of deep convolutional neural networks," *IEEE Trans. on Vis. and Comp. Graphics*, vol. 23, no. 1, pp. 91–100, 2017.
[20] M. Kahng, N. Thorat, D. H. P. Chau, F. B. Vigas, and M. Wattenberg, "Gan lab: Understanding complex deep generative models using interactive visual experimentation," *IEEE Trans. on Vis. and Comp. Graphics*, 2018.
[21] K. Wongsuphasawat, D. Smilkov, J. Wexler, J. Wilson, D. Man, D. Fritz, D. Krishnan, F. B. Vigas, and M. Wattenberg, "Visualizing dataflow graphs of deep learning models in tensorflow," *IEEE Trans. on Vis. and Comp. Graphics*, vol. 24, no. 1, pp. 1–12, Jan 2018.
[22] A. W. Harley, "An interactive node-link visualization of convolutional neural networks," in *ISVC*, 2015, pp. 867–877.
[23] M. Kahng, P. Y. Andrews, A. Kalro, and D. H. . Chau, "Activis: Visual exploration of industry-scale deep neural network models," *IEEE Trans. on Vis. and Comp. Graphics*, vol. 24, no. 1, pp. 88–97, Jan 2018.
[24] J. Wang, L. Gou, H. Yang, and H.-W. Shen, "Ganviz: A visual analytics approach to understand the adversarial game," *IEEE Trans. on Vis. and Comp. graphics*, vol. 24, no. 6, pp. 1905–1917, 2018.
[25] J. Wang, L. Gou, H. Shen, and H. Yang, "Dqnviz: A visual analytics approach to understand deep q-networks," *IEEE Trans. on Vis. and Comp. Graphics*, vol. 25, no. 1, pp. 288–298, Jan 2019.
[26] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.
[27] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[28] "Google quick draw game," https://quickdraw.withgoogle.com/, accessed: 2017-08-08.
[29] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.
[30] "Github vae code," https://github.com/fastforwardlabs/vae-tf/tree/master, accessed: 2018-07-21.
[31] "Dfc-vae," https://github.com/yzwxx/vae-celebA, accessed: 2018-08-20.
[32] X. Hou, L. Shen, K. Sun, and G. Qiu, "Deep feature consistent variational autoencoder," in *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*. IEEE, 2017, pp. 1133–1141.
[33] "Flask microframework," http://flask.pocoo.org/docs/1.0/.
[34] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *CoRR*, vol. abs/1710.10196, 2017. [Online]. Available: http://arxiv.org/abs/1710.10196
[35] Y. Koyama, D. Sakamoto, and T. Igarashi, "Selph: Progressive learning and support of manual photo color enhancement," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2016, pp. 2520–2532.
[36] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.
[37] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *CoRR*, vol. abs/1312.6199, 2013.

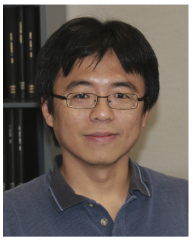
Junpeng Wang is a Ph.D. student in the Department of Computer Science and Engineering at the Ohio State University. He received his B.E. degree in software engineering from Nankai University in 2011, and M.S. degree in computer science from Virginia Tech in 2015. His research interests are mainly in the visualization and analysis of ensemble simulation data, high-dimensional data, and data generated from machine learning models.

Liang Gou is a principal research scientist in Data Analytics team at Visa Research. His research interests lie in the fields of visual analytics, deep learning and human-computer interaction. Liang received his Ph.D. at the College of Information Sciences and Technology from Pennsylvania State University.

Wei Zhang is a principal research scientist and research manager in Data Analytics team at Visa Research and interested in big data modeling and advanced machine learning technologies for payment industry. Prior to joining Visa Research, Wei worked as a Research Scientist in Facebook, R&D manager in Nuance Communications and also worked in IBM research over 10 years. Wei received his Bachelor and Master degrees from Department of Computer Science, Tsinghua University.



Hao Yang is the VP of Data Analytics at Visa Research and he is leading a team for advanced machine learning research to tackle challenging problems in the payment industry and develop the world's best commerce intelligence engine. Hao received his B.S. and M.S. degrees from University of Science and Technology of China (USTC) and Chinese Academy of Sciences (CAS) respectively, and his Ph.D. degree in Computer Science from University of California, Los Angeles (UCLA). He has published over forty papers in international conferences and journals, including a Best Paper Award from the International Conference on Parallel Processing (ICPP) in 2008.



Han-Wei Shen is a full professor at the Ohio State University. He received his B.S. degree from Department of Computer Science and Information Engineering at National Taiwan University in 1988, the M.S. degree in computer science from the State University of New York at Stony Brook in 1992, and the Ph.D. degree in computer science from the University of Utah in 1998. From 1996 to 1999, he was a research scientist at NASA Ames Research Center in Mountain View California. His primary research interests are scientific visualization and computer graphics. He is a winner of the National Science Foundations CAREER award and U.S. Department of Energys Early Career Principal Investigator Award. He also won the Outstanding Teaching award twice in the Department of Computer Science and Engineering at the Ohio State University.