# *GANViz*: A Visual Analytics Approach to Understand the Adversarial Game

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

**Abstract**—Generative models bear promising implications to learn data representations in an unsupervised fashion with deep learning. Generative Adversarial Nets (GAN) is one of the most popular frameworks in this arena. Despite the promising results from different types of GANs, in-depth understanding on the adversarial training process of the models remains a challenge to domain experts. The complexity and the potential long-time training process of the models make it hard to evaluate, interpret, and optimize them. In this work, guided by practical needs from domain experts, we design and develop a visual analytics system, *GANViz*, aiming to help experts understand the adversarial process of GANs in-depth. Specifically, *GANViz* evaluates the model performance of two subnetworks of GANs, provides evidence and interpretations of the models' performance, and empowers comparative analysis with the evidence. Through our case studies with two real-world datasets, we demonstrate that *GANViz* can provide useful insight into helping domain experts understand, interpret, evaluate, and potentially improve GAN models.

**Index Terms**—Generative adversarial nets, deep learning, model interpretation, visual analytics.

## 1 Introduction

Generative models bear promising implications to learn data representation in an unsupervised fashion. Generative Adversarial Nets (GAN) [1] is one of the most popular frameworks in this field. Since the GAN idea was first proposed by Goodfellow et al. [1] in 2014, numerous extensions have been proposed (e.g., DCGAN [2], InfoGAN [3]) and they have successfully demonstrated the powerful capability of the GAN framework. The basic idea of GANs is to set up two deep neural nets to compete against each other with opposite target functions. One neural net, called Generator ($G$), generates fake data and uses them to fool the other neural net, called Discriminator ($D$), who learns to discriminate fake data from real data. These two nets are trained against each other, and to the end, $G$ develops the capability of generating data that are amazingly similar to real data, and $D$ learns the data representations in an unsupervised fashion. Researchers can further use the trained $G$ and $D$ for different purposes. For example, well-trained $G$s can generate data in fields where data is not always sufficient [4]. $D$s are often used as pre-trained parameters for later fine-tuned trainings [5].

Despite the promising results from GANs, optimizing, interpreting, and evaluating such models remain challenging to domain experts [6], [7]. First, the adversarial process is complicated and has no theoretical guarantee of convergence so far, which presents challenges for model optimizations. Second, it is a non-trivial task to interpret the details inside GANs during trainings. For example, what features does $D$ learn to differentiate real/fake data? what patterns do the results generated by $G$ have, and how do the patterns evolve over time? Finally, model evaluation is also quite challenging. For example, domain experts are eager to know: is the network architecture an evenly matched game for both $D$ and $G$? Is $G$ learning data distributions only in a local feature space (i.e., the mode collapse issue [6])? However, these questions cannot be directly answered by model statistics (e.g., losses, activation means, parameter gradients) captured from conventional approaches.

One attempt to alleviate these issues comes from visual analytics. There are some visualization tools (e.g., TensorBoard [8]) to understand the training statistics of general neural nets. Also, an increasing amount of model-specific visualizations have been proposed recently, including MLP visualization [9], convolutional network visualization [7], [10], [11], and seq-to-seq model visualization [12], [13].

However, few visual analytics tools have been proposed to help domain experts understand, interpret, and evaluate GANs. The most recent work in this thread is the DGMTracker [14]. While this tool is very helpful to understand the overall training process and diagnose potential training failures for *general* generative models, we are still puzzled with the challenges of GAN models mentioned above (e.g., the detailed analysis/evaluation of $D/G$, the mode collapse issue). Thus, it is worthwhile to study GAN models in-depth for both machine learning and visualization community.

In this work, we focus on a popular GAN model, DCGAN [2], to design and develop a visual analytics system, i.e., *GANViz*. The system helps domain experts understand the adversarial process of GANs in-depth, including evaluating the model performance for both $D$ and $G$, providing evidence and interpretations of the performance, and conducting comparative analysis with the evidence. *GANViz* consists of five components. The first component of metric view provides an overview of model dynamics in the training process. The three components of probability view, distribution view, and *TensorPath* view offer rich evidence and interpretations of the model performance, such as the

- *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@osu.edu, shen.94@osu.edu*
- *L. Gou and H. Yang are with Visa Research, Palo Alto, CA, 94306. E-mail: ligou@visa.com, haoyang@visa.com*

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

decision features of $D$, feature patterns of both real and fake data, neural network structures, and associated decision-critical data in the structures (e.g., neurons, activations, parameters, etc.). Finally, the comparative analysis component allows users to understand how the decision-critical data are evolved and how they work for the data from different sources (e.g., fake/real, true/false predictions). We demonstrate the effectiveness of the system through case studies of two real-world datasets with domain experts. To sum up, the contributions of this work include:

- We design and develop an interactive visual analytics system, *GANViz*, to provide an in-depth understanding of the adversarial process of GAN models.
- We introduce a compact matrix-based visual design, *TensorPath*, to reveal the full-spectrum information inside deep neural networks. The design is independent of specific models, and thus can be applied to many other deep neural network models.
- From case studies with domain experts, we summarize feedback and derive insights with important implications on understanding, evaluating, and improving GANs.

## 2 Related Work

***GAN models and model challenges.*** Deep generative models are emerging as an active area for unsupervised learning. The power of these models lies in learning and extracting features from large scale unlabeled data, and later using them for other tasks, such as classification and clustering. One of the most popular frameworks is Generative Adversarial Nets (GAN) [1], which trains two networks of Generator ($G$) and Discriminator ($D$) as feature extractors.

There are many extensions of the original GAN [1]. Their goal is to improve the quality of feature extractions by modifying the network structure of $G$ and/or $D$. For $G$, the extensions mainly improve $G$ in the latent variable space, e.g., InfoGAN [3]. For $D$, the extensions utilize additional label information, e.g., semi-supervised GAN [15]. Some other methods introduce additional components for both $G$ and $D$. For example, conditional GAN [16] uses class information to separate data in $G$ and $D$. One popular such extension for image data is DCGAN [2], which uses convolutional and deconvolutional layers to encode and generate images. This model is also our focus in this work.

There are several challenges to understand and train GANs [6]. One is the stability and convergence of the models. Current research is debating the theoretical guarantee on the convergence of the GANs game [17]. Some approaches, like MixGAN [18], can only guarantee an approximate equilibrium between $D$ and $G$. Another challenge is the mode collapse issue [19], i.e., the generator can only learn a local distribution of the real data. Although WGAN [20] introduces the earth-mover distance to alleviate the problem, it is still unknown if the learned features really capture the distribution of real data or not. The third challenge is to understand and evaluate GAN models. The current evaluation methods are either through measuring the performance of additional classification tasks or qualitatively observing some generated results in a cherry-picking way. In this work, we adopt a visual analytics approach and attempt to shed some light on the above challenges.

***Visual Analytics for Deep Learning.*** Many visualization or visual analytics tools for neural networks have been proposed. Some of them directly visualize the network structures, parameters, and activations, e.g., [7], [8], [9]. They offer a high level understanding of neural networks, but lack the capability of interactive analysis. Other visualizations for deep learnings are usually model-specific. For example, CNNVis [11] treats a Convolutional Neural Network (CNN) as a directed acyclic graph, and visualizes the multiple facets of neurons from different layers. DeepVis Toolbox [7] visualizes the learned filters and activated features across layers. LSTMVis [13] is designed to understand Long Short-Term Memory networks (LSTMs) with a temporal visualization by revealing the dynamics of hidden states for sentences. The character-level features learned by Recurrent Neural Networks (RNNs) or LSTMs have been visualized by [12].

Although the machine learning community has witnessed the promising results from GANs, effective understanding, interpreting, and evaluating GANs is still a new and challenging problem for visual analytics. A recent successful attempt to this problem is DGMTracker [14], which helps to understand the data flow in model trainings and identify neurons causing training failures for *general* generative models. Our focus, in this work, is to have an in-depth comprehension of GANs, including: evaluating the quality of $G/D$ in the adversarial game, interpreting what features $G/D$ can extract, and how the model uses these features to distinguish real/fake data over the training process.

## 3 Background and Motivation

In this section, we provide a brief introduction on the basic ideas and important concepts of GANs. We also introduce one of the most successful GANs for image generation, i.e., DCGAN, used in this work.

**GAN** is a neural network framework to extract features of a target data set and generate new data items that are similar to those in the data set. It works in the following way. Suppose we have a data set $X$ (e.g., a collection of images, sounds, texts, etc.), and want to generate a new data item, $x'$, that is similar to those in $X$. GAN achieves this goal by finding the distribution of $X$, i.e., $P_t(x)$. Generating $x'$ is, then, similar to take a sample from $P_t(x)$. However, finding the exact $P_t(x)$ is challenging, as $X$ can be infinitely large (e.g., images of all human faces) but a model can only take a finite set of inputs. GAN, therefore, resorts to find a distribution $P_g(x)$ that can best approximate $P_t(x)$. In detail, it takes a random sample $z$ from a simple distribution $P_z(z)$ (e.g., a uniform or Gaussian distribution) and maps it to $x'$ through a deep neural network (called Generator, $G$). To make $x'$ similar to data items in $X$, GAN tunes the parameters of $G$ with the help of another deep neural network (called Discriminator, $D$), which learns data features in $X$ and discriminates $x'$ from data items in $X$. During the training, they compete with each other iteratively by optimizing a minimax target function, i.e., Equation 1 [1]. To the end, $G$ learns $P_g(x)$ and can generate data that are almost indistinguishable from real data; and $D$ learns to extract reusable features from real data.

$$\min_G \max_D \mathbb{E}_{x\sim P_t(x)}[logD(x)] + \mathbb{E}_{z\sim P_z(z)}[log(1-D(G(z)))] \quad (1)$$

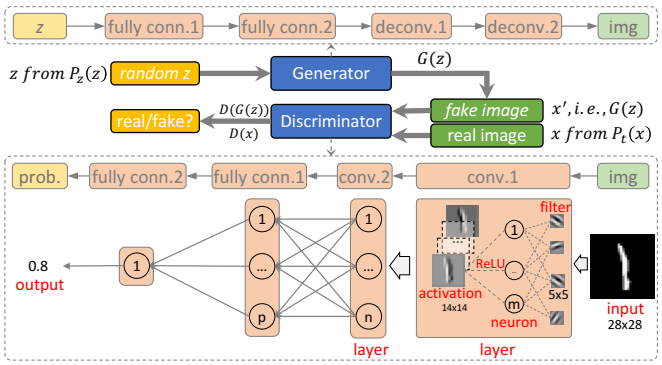



Fig. 1. Simplified architecture of the DCGAN model used in this paper.

**DCGAN** (Deep Convolutional GAN [2], shown in Figure 1) is one of the most successful extensions of the GAN idea on image data. In this model, $G$ is a deep deconvolutional network, which unrolls features layer by layer from random inputs (i.e., $z$ vectors) and generates images. $D$ is a deep convolutional network (a binary classifier similar to CNN [10], [11]), which discriminates the generated images from real images. Further details and source code of the model can be found in [2] and [21]. The following concepts are frequently referred in deep neural networks:

- Neuron (circles in Figure 1): A basic computational unit conducting linear transformation of previous inputs.
- Layer (vertical rectangles in Figure 1): A collection of neurons that perform the same computation, but with different parameters. There are different types of layers (e.g., convolutional layer, fully connected layer). Chaining layers one after another becomes a deep neural network.
- Activation (thumbnails in Figure 1): Activation occurs when the value of a neuron goes over a certain threshold, often decided by an activation function (e.g., ReLU used in DCGAN). In the image domain, the activation of an image indicates what features are activated by the corresponding neurons. For this reason, it is also called feature map.
- Filter and weight: Parameters of neurons to conduct linear transformation of inputs. If the neuron is in a convolutional layer, its parameters are called filters. Parameters of neurons in fully connected layers are often called weights.
- Loss function: A function measures the cost of inaccuracy of predictions. The loss values are often considered as the indicators for the model quality. Specifically, the loss function for GANs is a two-player minimax [1] (Equation 1).
- Confusion matrix: A matrix measures the performance of a classifier by showing the number of test items in four cells: true-positive ($TP$), false-positive ($FP$), true-negative ($TN$), and false-negative ($FN$). In DCGAN, $TP$ is the case of real images predicted as real by $D$ (Similarly, $FN$: real images predicted as fake; $FP$: fake images predicted as real; $TN$: fake images predicted as fake).
- Batch: A collection of input data (e.g., 64 images) used to train/update the neural network at one training iteration.
- Epoch: One epoch means all input data have been iterated. For example, the MNIST has 70,000 images. If the batch size is 64, one epoch will have 1094 (70,000/64) batches.

For the DCGAN model, challenges of general GANs still remain, including the issues of convergence, mode collapse, interpretation and evaluation as we discussed in the related work, though DCGAN has achieved promising results in image generation and classification problems [2]. We believe visual analytics is a promising solution to fill the gap between the success of machine learning models and the deficiency in model understanding, interpretation, and evaluation. Motivated by the practical requirements from domain experts (Section 4.1), we aim to design and develop a visual analytics system to alleviate those challenges.

## 4 Requirements and Approach Overview

### 4.1 Requirement Analysis

We worked with three domain experts to elicit the design requirements to understand GANs. The design and development went through three stages: (1) initial understanding of GANs (how the model works, the current challenges); (2) prototyping with CNN visualization approaches [10] for DCGAN; (3) iterative design and development of *GANViz*.

Along with the iterative design process and our interaction with the experts, three main themes: ***model evaluation***, ***model interpretation*** and ***comparative analysis***, are emerging to aid their understanding and improving GAN models.

**R1: Supporting *model evaluation* during dynamic training process.** It is a challenging task to evaluate GANs because of the nature of unsupervised learning and the dynamic training process. The existing evaluation method of using additional classification tasks only gives a score without in-depth justification, and eyeballing the generated results is subjective and time-consuming. Both methods fail to reveal the model quality over the training process. Therefore, we distilled following requirements under this theme:

- *R1.1: What is the overview of GANs' quality during the training?* For example, how do the loss functions of $D/G$ evolve over time? Besides the loss functions, what other measures can we use to help us understand the model convergence and mode collapse problem?
- *R1.2: How is the performance of $D$ evolved in different stages?* For $D$, it is important to measure and present its capability to distinguish fake and real data over the training.
- *R1.3: How is the performance of $G$ changed in different training stages?* One challenge with $G$ mentioned by domain experts is the mode collapse issue (i.e., $G$ tends to learn local information and generate similar results again and again). Therefore, the tool should enable users to monitor the quality of $G$, to diagnose the mode collapse problem.

**R2: Enabling *model interpretation* for detailed analysis on model quality.** Besides the overall understanding of model quality, it is desirable to interpret how and why the quality is achieved. Furthermore, it is also helpful to interpret the features learned and extracted, especially for $D$ who extracts reusable features that can be applied to other tasks. Towards the two ends, we embraced the following goals:

- *R2.1: Revealing the features and data distribution learned by both $G$ and $D$.* For $G$, it is helpful to see the typical features and the distribution of the generated images in comparison with the real images. For $D$, the domain experts want to know what visual features are used to make the decision of real or fake images.
- *R2.2: Demonstrating the network architecture of $D$ and interpreting the extracted features.* This aims to present hidden

features (e.g., important neurons, learned features) of the $D$'s network to support detailed investigations.

**R3: Providing *comparative analysis* to enrich the understanding of model dynamics.** One need mentioned by the domain experts is to compare how image features are learned and used from two dimensions: (1) the image sources, e.g., positive/negative predictions, real/fake, and different classes, such as *apple* or *orange*; (2) the time dimension: how the learned features evolve during a training. This type of comparative analysis assists users to develop further understandings on the power of $D$ and model evolution.

- *R3.1: Between-Source Comparison: comparing two images from different sources (e.g., $TP$ vs. $FP$ images; apple vs. orange), to examine how $D$ could differentiate them.* For example, comparing the two images representing $TP$ and $FP$, and revealing the differences between their corresponding activations can help users understand why $D$ uses certain features to make the decision, and what neurons are important in extracting these features.
- *R3.2: Temporal Comparison: tracking the same pair of images across the training process.* Users should be able to compare the same two images in different stages, to track how the importances of neurons (in differentiating them) change over time. This would lead users towards meaningful inferences on the formation of $D$'s decision over time.

### 4.2 *GANViz* Framework Overview

With the aforementioned design requirements, we designed and developed an interactive visual analytics system, i.e., *GANViz*, for domain experts to analyze GAN models. This section provides an overview of the system (Figure 2).

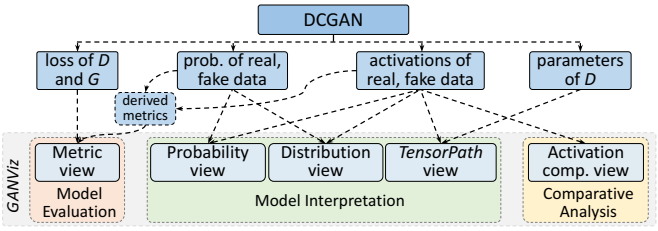



Fig. 2. Framework of our visual analytics system.

***Model Execution and Data Collection.*** The first step is to collect necessary data from the model training for later visualization and analysis. During the training process, we collect four types of data with different frequencies. The **first** type of data is the values of loss functions for both $D$ and $G$. These values are common indicators for the quality of the model, and thus, should be collected frequently to closely monitor the training progress. The **second** type of data is the probability values (likelihood to be real) for each real/fake image. The value is the prediction result of the $D$ network with an image as input. We sampled a subset of real/fake images as the inputs. For real data, 256 images are randomly taken from the training images; for fake data, 256 random $z$ vectors are fed to $G$ to generate 256 fake images. As a result, we collected 512 probability values (256 for real images and 256 for fake images), in total, at each timestamp.

The **third** type of data is the activations (from all layers) for each real/fake image when it goes through the $D$ network. For example, the first layer of activations for an image is the image itself; the second layer of activations is the image after it went through the first convolutional layer; the last layer of activations is the probability value of the image. The **fourth** type of data is the parameters of $D$, i.e., the filters of convolutional layers and weights of fully connected layers. These parameters, especially the convolutional filters, will help users understand how activations are generated and how they lead to the final probability values. Different types of data were collected with different frequencies to reduce the size of data to be collected. As shown in Figure 3, the loss values are collected more frequently than others.

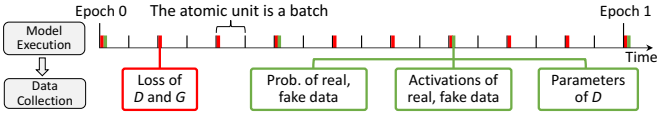



Fig. 3. Collecting data with different frequencies. In this example, losses are collected every two batches (red ticks); probability values, activations, and parameter data are collected every six batches (green ticks).

Besides the directly collected data, we also provide three derived metrics to better evaluate the model quality:

**AUC** (Area Under Curve) reflects the performance of $D$, measured by the area under its Receiver Operating Characteristic (ROC) [22] curve. A ROC curve plots the true positive rate ($TP/(TP+FN)$) against the false positive rate ($FP/(FP+TN)$) of a binary classifier under different decision thresholds (Figure 4). More points on the top-left corner (larger area under the ROC curve) of this plot (i.e., high true positive rate, low false positive rate) indicate a better performance of $D$.

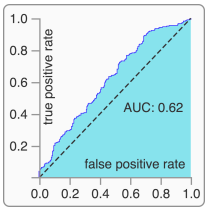



Fig. 4. AUC in cyan.

**Diversity** quantifies the diversity in the generated images, which is used to detect the mode collapse problem in $G$, i.e., the value of this metric should keep high if no mode collapse happens. The metric first derives a similarity value among all generated images, using Structure Similarity (SSIM) [23]. The inverse of this value is then used to indicate the diversity for the set of images. If the number of images is $n$, then,

$$Diversity = 1/\left(\frac{1}{n(n-1)/2}\sum_{i=1}^{n}\sum_{j=i+1}^{n}\left(SSIM(img_i, img_j)\right)\right). \quad (2)$$

**Dist-Diff** (Distribution Difference) measures the difference between the fake data distribution, $P_g(x)$, and real data distribution, $P_t(x)$. The metric is derived through two steps: (1) approximate $P_g(x)$ and $P_t(x)$ from a subset of fake and real images respectively; (2) measure the difference between the two distributions. For the first step, we use PCA to reduce images to lower dimensional space, and employ Gaussian Mixture Model (GMM) [24] to fit them as a distribution. The second step measures the KL divergence [25] between the two GMMs through Monte-Carlo samples (Equation 3).

$$\begin{aligned} &Dist\text{–}Diff = KL\left(P_t(x)||P_g(x)\right), \\ &P_t(x) = GMM(PCA(X)), P_g(x) = GMM(PCA(G(\mathbf{z}))) \end{aligned} \quad (3)$$

***Visual Analytics Modules and Components.*** Aligned with the design requirements, the components of *GANViz* are organized into three modules, including quality monitoring, model interpretation, and comparative analysis. The

three modules are designed upon the top-down exploration flow elicited from domain experts and also organized by Shneiderman's information seeking mantra, i.e., "overview first, zoom and filter, then details-on-demand" [26].

In the three modules, five views (Figure 2) are designed to analyze different data in a top-down style. In the quality monitoring module, the metric view helps users understand the overall quality (R1) by visualizing various model metrics. Then, the probability view, distribution view, and *TensorPath* view enable users to examine and interpret the model at a specific training stage of their interest (R2, also R1.2). Over the journey of exploration, the activation comparison view supports users to collect relevant model information and conduct comparative analysis (R3). The five views are coordinated with each other, and rich interactions (e.g., brushing, filtering, selection, etc.) are provided to link the views for flexible exploration.

## 5 *GANViz* Visual Analytics System

In this section, we explain the design rationales and visual interface for the five components of *GANViz* (Figure 5). The components are: the metric view (Figure 5a) for model quality dynamics (R1); three components of probability view, distribution view, and *TensorPath* view (Figure 5b, 5c, 5d) for model interpretations (R2); and the activation comparison view (Figure 5e) for detailed comparative analysis (R3). The layout of these five views follows the analysis workflow suggested by domain scientists (i.e., from high-level overviews to detailed explorations/comparisons), and the views are linked together with coordinated visualizations.

### 5.1 Metric View

The metric view (Figure 5a, 6) provides an overview on the quality, and quality evolution, of the model. Four types of metrics are analyzed in this view: loss values of $D/G$, AUC, Diversity, and Dist-Diff scores. These metrics varying over the training process are essentially multiple time-series data [27], [28], and organized into two sub-groups: model convergence with loss values (R1.1); model performance for $D$ (AUC, R1.2) and $G$ (Diversity, Dist-Diff, R1.3).

This view is presented by a line chart, which is an intuitive and straightforward representation to show multiple time-series [29], [30]. In the chart, the horizontal axis represents time (epochs/batches) and vertical axis represents values of different metrics. The two sub-groups of metrics are presented in their own views (Figure 5a, 6). Users can easily switch between them by checking the radio box on the top (the inset in Figure 5a). Notice that the second sub-group of metrics do not share the same value range, we normalized them to be in [0, 1] before visualization.

A $focus+context$ zooming is also provided to explore the model dynamics over a lot of epochs/batches. The context view (the bottom half of Figure 5a) shows the whole training in the unit of epoch over the horizontal axis. The focus view (the top half) presents a detailed view of the metrics from the selected epoch and the axis is at the scale of batches. Both views are equipped with a snapped brushing (i.e., the brushed regions will be rounded to one epoch/batch automatically) for easy selection. Once the interested epoch/batch is selected (epoch 1, batch 300 in Figure 5a), other views will be updated accordingly.

### 5.2 Probability View

The probability view (Figure 5b) provides detailed information for the performance of $D$ (R2.1) at a timestamp of selection (e.g., $x$ epoch, $y$ batch). This view presents the predicted probabilities of the sample data (256 real/fake images) along with their distributions in the confusion matrix. It contains two components: (1) a line chart showing the probability; (2) image thumbnails with confusion matrix information.

*Probability Chart.* To clearly demonstrate the separation of probabilities from real/fake samples, we sort those probabilities and present them in a line chart, as shown in Figure 5b. Two curves show the probability of the 512 (256+256) images sampled from the real (in green) and fake (in red) data. The horizontal axis represents the order of the 256 images sorted by their probability value (in a decreasing order) and the vertical axis shows the probability values. The horizontal black dashed line indicates the current decision threshold of $D$: images with probability value less than this threshold are considered as fake, otherwise they are considered as real. Users can drag this line to adjust the current decision threshold. As the decision threshold moves, we can obtain a metric of AUC score [22], which indicates the prediction power of $D$. Users can also check the AUC region from this view by clicking the AUC button on the top. Figure 4 shows the pop-up AUC window for the current timestamp.

*Aggregated Image Thumbnails.* To reveal a visual trend of the images, we place the aggregated image thumbnails below the horizontal axis. We aggregate every 16 consecutive images and present them as one thumbnail by averaging the pixel values to save the screen real estate. As a result, only 16 (256/16) thumbnails are shown. The two rows of thumbnails in Figure 5b represent both real (top) and fake (bottom) image samples. Clicking on one thumbnail will pop up a small window to show the 16 original images. The thumbnails from left to right reveal how visual features change with the decrease of their probability value. In Figure 5b (from left to right), we can see how the italic style of digit 1 affects the probability values derived from $D$.

*Confusion Matrix Visualization.* The confusion matrix (Figure 7 left) shows the classification results with current decision threshold indicated by the horizontal black dashed line in Figure 5b. The border style of images indicates whether $D$ has classified them correctly (solid border) or not (dashed border). The bottom left corner of Figure 5b shows the legends for the four categories in the confusion matrix (an enlarged version is shown in Figure 7 left). The number in each legend indicates the number of images falling into that category. A thumbnail with both solid and dashed borders indicates that the thumbnail averages images from two categories. For example, Figure 7 (right) shows the 8th thumbnail in the top row of Figure 5b. The top and right borders of this thumbnail are in dashed style; whereas the bottom and left borders are in solid style. Clicking on this thumbnail allows users to check the original 16 images. From borders of the original images, we know that this thumbnail averages 15 $TP$ images and 1 $FN$ image.

### 5.3 Distribution View

The distribution view (Figure 5c) provides supporting information to understand the performance of $G$ (R2.1, Diversity

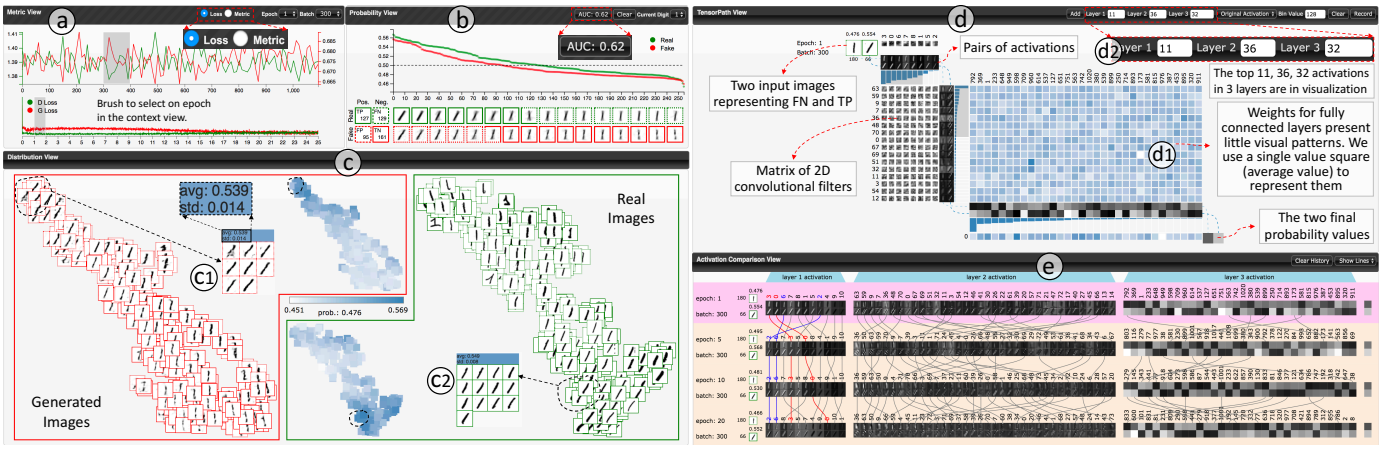



Fig. 5. *GANViz*: (a) the metric view with $focus+context$ support; (b) the probability view shows the probability of real/fake images in a decreasing order; (c) the distribution view reveals the feature distributions of real/fake images; (d) the *TensorPath* view highlights important activations, filters and weights of $D$; (e) the activation comparison view tracks the changes among multiple pairs of images from different sources or timestamps.

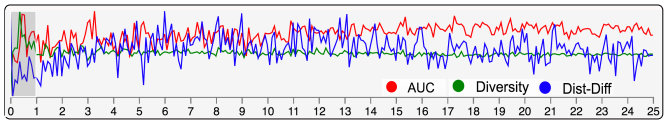



Fig. 6. Metric view: measuring model quality with three metrics.

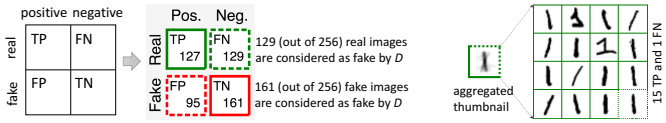



Fig. 7. Left: confusion matrix information with glyphs; right: a thumbnail aggregates 15 $TP$ and 1 $FN$ image (the bottom-right one).

and Dist-Diff). It shows an overview on the appearance of all real/fake images, their feature and probability distributions.

*Image Overview and Feature Distribution.* To provide an overview of the visual features of the sampled (real/fake) images, we visualize those images and cluster them with the t-SNE [31] algorithm to reveal their feature distribution. The 256 fake/real images are shown on the left/right of Figure 5c. Similar to the probability view, the category of each image is encoded with its border style and color. Additionally, users can perform lasso selections in this view. Selected image will be sorted (in probability decreasing order) and displayed in a pop-up window. The header of the window will show the average and standard deviation of probabilities of the selected images (Figure 5-c1). From this view, features of images in different clusters can also be easily observed. For example, the selected images in Figure 5-c1 are generated images with italic styles.

*Probability Distribution.* We show the probability distribution, along with the feature distribution, of the sampled images to reveal what features have tricked $D$. Other design choices, such as encoding the probability values with glyphs placed aside the images, were also considered but introduced more visual cluttering. In the middle of Figure 5c, the probability distribution of fake/real images is presented with a group of colored squares on top/bottom. One square corresponds to one image; the color of the square (from white to blue) represents the corresponding image's probability value (from small to large). The layouts of the squares and images are synchronized, which allows users to find the correspondence between them. Interactions are introduced to link the two parts, as well as the corresponding data in other views. For example, hovering over one image will highlight the image and its probability square. The corresponding sample in other views will also be highlighted.

### 5.4 *TensorPath* View

Two design objectives for the *TensorPath* view are: (1) revealing the details of $D$'s network architecture (layers, activations, parameters, R2.2); (2) showing and comparing the important features that $D$ extracts to make predictions (R3).

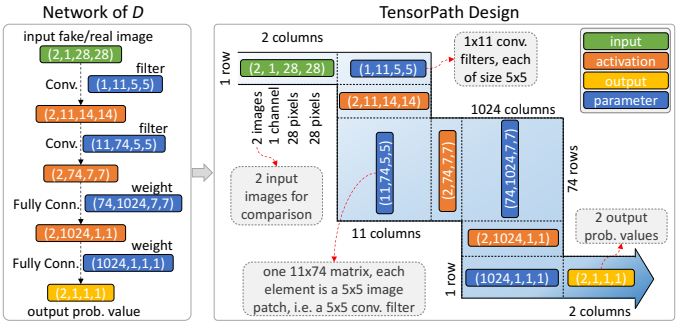



Fig. 8. Left: the neural network architecture of $D$ in the DCGAN model [21]; right: a zigzag layout design of the *TensorPath* view.

The network architecture of $D$ can be considered as a chain of layers (Figure 8 left). There are two categories of data of our interest: (1) the inputs (in green), activations (in orange), and outputs (in yellow); (2) the filters and weights (in blue) learned during the training process. The activations and outputs are generated, as input images (real/fake) pass through the network (the activations in each convolutional layer are the extracted features of that layer). The four numbers in each non-blue rectangle represent the number of images/activations, the number of channels, width and height of each image/activation. The first number is always 2 since we always compare a pair of images.

The design of the *TensorPath* view is illustrated in Figure 8 (right). The design goal is to demonstrate the network structure of D, and the flow of various types of data in a compact way. To efficiently use the space, we design a zigzag path visualization, which we called as *TensorPath*, to illustrate the network structure and data flow in $D$. The inputs, activations, and outputs in neighboring layers are placed to be orthogonal to each other, and connected by filters or weights in the zigzag path. This design is to compactly demonstrate layer structures, easily identify important activations/parameters, and intuitively track data across layers of $D$. Figure 5d shows our implementation of the *TensorPath* view. The enlarged top-left corner is shown in Figure 9. Pairs of activations (Figure 9a) in each layer are sorted by the activation difference of the pairs (mean square error between two activations) in a decreasing order. The blue bars next to the activations encode the difference between them (Figure 9b). They are also used to select and filter out the activations shown in the view.

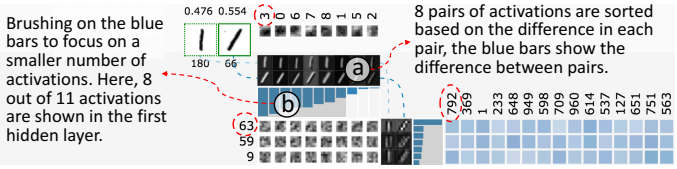



Fig. 9. *TensorPath* view: the enlarged top-left corner of Figure 5d.

The data of interest (input/activation/output/filter/weight) are all presented as 2D images in *TensorPath*. Hovering over an image will show the enlarged version of it. However, those images may have different sizes, e.g., the resolutions of input images and convolutional filters are $28\times28$ and $5\times5$ respectively. To put them consistently in *TensorPath*, we introduce three operations on a 2D image:

- *Scaling* is to scale up/down one image. Scaling up an image is to copy each pixel to its neighboring pixels. For example, to scale an image two times up, each pixel will be duplicated twice horizontally and vertically. Scaling down an image is to reproduce the image but with a fixed stride.
- *Padding* is to append rows and columns of white pixels around an image. In certain cases, the desired image size cannot be achieved by scaling only. For example, if the original image size is $5\times5$ and the desired image size is $12\times12$. In this case, the image can first be scaled up twice, then padded with two rows and two columns of pixels.
- *Aggregation* is to average pixels of an image and use the average value to represent the image. This is a similar idea to heat-map visualization and very efficient to visualize the large amount of parameters (weights) in fully connected layers (Figure 5-d1, the colors from white to blue represent values from small to large).

We provide two interactions to address the scalability issue in *TensorPath*. First, users can use a threshold value to filter out less important activations from the visualization. With our interface (Figure 5-d2), users can flexibly change these values. By default, we use 11 (out of 11), 36 (out of 74) and 32 (out of 1024) for the three hidden layers of $D$ in the DCGAN model used in this work. Second, users can brush on the difference bars to focus on a smaller but more important set of activations, as we have shown in Figure 9b.

### 5.5 Activation Comparison View

The activation comparison view allows users to compare multiple pairs of activations across sources or timestamps (R3.1 and R3.2). This is a typical visual analysis task for high-dimensional data. Among all design alternatives, we use a design that is similar to the Parallel Coordinates Plot (PCP), which can effectively visualize and compare high-dimensional data [32]. As shown in Figure 5e, three horizontal PCPs are used for three layers of activations. Each parallel axis (each row) is a sequence of sorted activation pairs from the *TensorPath*. Across rows of individual PCPs, the pairs of activations with the same indices are linked together with Bézier curves. The number of curves between neighboring rows indicates the number of shared activations in distinguishing different pairs of input images.

This view and the *TensorPath* view work together to support both cross-source and cross-time comparative analysis. During users' explorations in the *TensorPath* view, they can compare a pair of images and record the pair into the activation comparison view (as one row). The new pair will then be compared with other existing pairs (rows) in the view. Clicking on one row in the activation comparison view will trigger the update in the *TensorPath* to show network details for the row of images. The pink row in Figure 5e is in selection now (i.e., the corresponding images of that row are currently presented in the *TensorPath* view).

## 6 Case Studies and Experts' Feedback

### 6.1 Datasets: MNIST and QuickDraw

The MNIST and Google QuickDraw [33] datasets were used in our case studies. Both datasets have 70,000 images (with resolution $28\times28$) in 10 classes (for QuickDraw, we randomly selected 7,000 images from each of the 10 selected classes). Figure 10 shows one sample from each class of these two datasets. As we can see, the two datasets come with different complexities of visual features, i.e., the drawings in QuickDraw are more complex than the digits in MNIST.

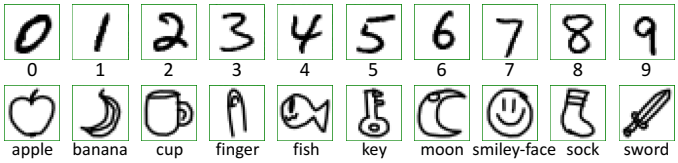



Fig. 10. The two datasets: top: MNIST; bottom: QuickDraw [33].

We trained the DCGAN model with the two datasets separately for 25 epochs. The batch sizes for both trainings were 64, so there were 1094 (70,000/64) batches in each epoch. Loss function values were collected every 10 batches; other data (see Section 4.2) were collected every 100 batches.

### 6.2 Findings from Case Studies with Domain Experts

#### 6.2.1 Model Evaluation

With *GANViz*, we can easily observe the patterns of model training dynamics. From the metric view, we observed a quick drop of losses after several training epochs for both datasets (Figure 5a, 11a). However, we also found there were two different patterns for them: (1) the losses dropped much

faster for MNIST (after the first epoch) than QuickDraw (after 3 epochs); (2) the losses of both $D$ and $G$ oscillated in a relatively smaller range in MNIST than those in QuickDraw.

We also observed some shared trends from the three quality metrics of MNIST and QuickDraw (Figure 6, 11b): (1) the capability of $D$ is improving as the AUC curves (in red) show an increasing trend for both cases (R1.2); (2) the Diversity curves (in green) remain high throughout the 25 epochs, which indicates that mode collapse did not happen in $G$ (R1.3); (3) with the training process moving forward, the Dist-Diff curves (in blue) demonstrate a decreasing trend (after 14 epochs), which means $G$ is progressing towards the right direction to capture the real data distributions (R1.3).

The metric view also reveals a couple of model quality differences between the two datasets. First, $D$ is struggling more in discriminating real/fake digits (from MNIST) than real/fake drawings (from QuickDraw), as the AUC curve oscillates a little more significantly in the MNIST case. This indicates that $G$ can generate better fake digits (compared to fake drawings) to trick $D$. It also verifies our expectation that it is easier for $G$ to learn simple visual features than sophisticated ones. Second, the oscillation of the Dist-Diff curve becomes larger in later epochs of the QuickDraw data. In contrast, the oscillation degree of the Dist-Diff curve in MNIST does not show obvious changes. The lager oscillation degree in QuickDraw indicates that, for $G$, capturing the real data distribution of QuickDraw is more difficult.

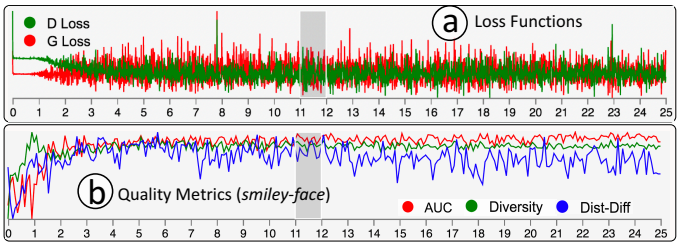



Fig. 11. Loss and metric values of QuickDraw in the metric view.

### 6.2.2 Model Interpretation

*Decision features of D (R2.1).* We used the probability view to explore what decision features $D$ had used, and how $D$ used them over the training process. For MNIST, $D$ learned to use various font styles (italic/bold) to make decisions. For example, from the two rows of thumbnails in Figure 5b, we observed the trend that images with higher probabilities are digit 1s with italic styles; and in Figure 12a, the decision feature of $D$ is the boldness of digits. For QuickDraw (the more intricate data), $D$ used more features to make decisions, including: the size of *apples*, the orientation of *keys*, the contour shape of *smiley-faces*, as shown in Figure 12d, 12e, 12f.

More interestingly, we found that $D$ may flip its decision features in different training stages, in both datasets. For example, Figure 12b, 12c show that $D$ used the italic style of digit 0 as its decision feature, and it flipped its decision between the two timestamps. For QuickDraw, we saw that *smiley-faces* with circular boundary were considered to have higher probabilities to be real in epoch 1 batch 400 (Figure 12f); however, in epoch 3 batch 300 (Figure 12g), images with those features received lower probabilities. A complete set of examples (of decision features) for all 10 classes of images can be found from our supplementary material.

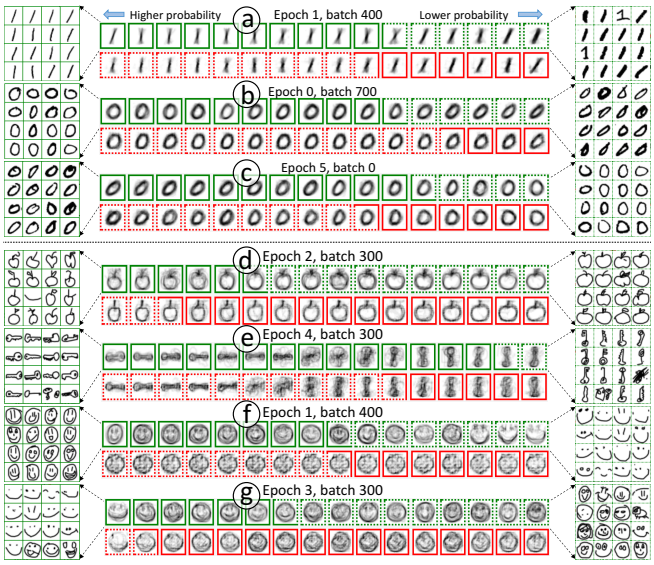



Fig. 12. (a) Digit 1 with bold style as the decision feature; (b, c) digit 0 with the flipping of decision feature (italic style) in two timestamps; (d, e, f) the decision features of $D$ in different classes of drawings in QuickDraw; (f, g) the flipping of decision feature in *smiley-faces*.

Based on our explorations, we notice that the decision feature of $D$ is not predictable, and $D$ may flip its decisions back and forth several times during a training. Even with the same model, same dataset, $D$ may have different decision features or decisions, in the same timestamp of two separate trainings. This interesting observation calls for more theoretical explorations from the domain experts.

*Visual explanation for the performance of G (R2.1).* We used the distribution view (Figure 5c) to verify and explain the overall performance of $G$ discussed in the metric view (i.e., the Diversity and Dist-Diff curves). This view provides an overview on the quality of images generated by $G$, as well as the visual feature distributions of fake and real images.

We first observed that $G$ always attempted to learn the decision-critical features over the training, even though $G$ could not capture the full distribution of real data in the very early stages. In Figure 5-c1, although most of the generated images were still blurry ($G$ was not very sophisticated) at this early stage (epoch 1, batch 300), some digits with italic styles had successfully cheated $D$, and received high probability values to be real (the average probability value of the selected images is 0.539). Similar phenomena were also found in the QuickDraw data. For example, in Figure 13, $G$ learned to generate horizontal $keys$ to trick $D$, and the image clusters with horizontal orientation in the top-right corner received higher probability values.

Moving to later stages, we observed that $G$ could generate diverse images with good qualities, which explained the high Diversity and low Dist-Diff values shown in the metric view, and proved that the mode collapse issue did not happen. For example, in epoch 24, batch 100 of the MNIST training (Figure 14), $G$ learned that the real data have many different features (e.g., italic/vertical, bold/slim) and could generate images that are very similar to the real data. As a result, $D$ got really confused and could not tell

what features lead to real data. The even distribution of white/blue squares in the middle of Figure 14 shows this.

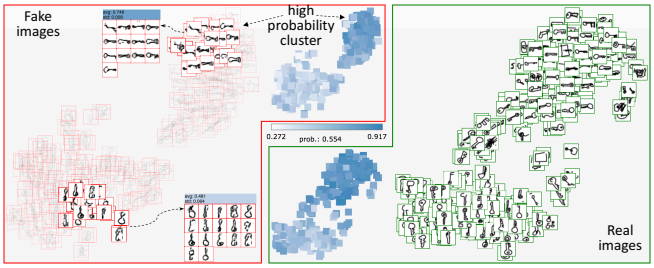



Fig. 13. Distribution view for the drawings *key* from QuickDraw.

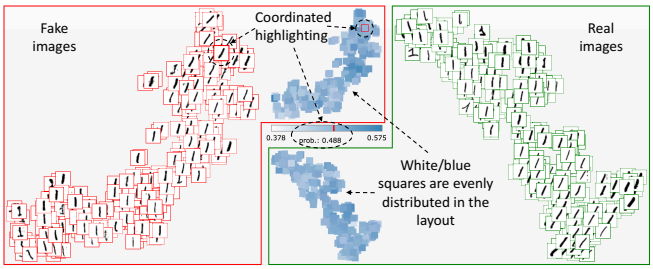



Fig. 14. Distribution view in epoch 24, batch 100. The probability distributions (color squares) are more even, compared with Figure 5c.

*Understanding the process of extracting decision-critical features by $D$ (R2.2).* The power of generative models comes from their capability of extracting reusable features. From previous explorations, we knew that both $D$ and $G$ learned the decision-critical features (e.g. italic, bold styles in MNIST digits; size, orientation, contour shape in QuickDraw drawings) over the training process. Next, we used *GANViz* to understand and interpret how those features were extracted through the network. Specifically, we focused on $D$, which usually serves as feature extractors for other tasks.

Towards this goal, we used the *TensorPath* view to open the hood of $D$ and observe how features were extracted through the layers of $D$'s network. From Figure 5b, we already knew that the decision feature of $D$ in this timestamp is the italic styles of digit 1. We then selected two images of digit 1 from false-negative (i.e., without italic style) and true-positive (with italic style) to conduct detailed investigations in the *TensorPath*. Figure 5d shows how this pair of images flow through $D$'s architecture with important details, e.g., filters, activations. For example, it is obvious that the most important filters in discriminating the pair of images are: 3, 63, and 792 in the three hidden layers respectively (Figure 9).

*Semantic interpretation of the extracted decision-critical features by $D$ (R2.2).* After getting an overview on how the pair of images flowed through the network, we further investigated the details of filters/activations to see how the decision-critical features were extracted through the filters. For example, Figure 15f shows the enlarged view of the two most important filters in differentiating the two previously selected digit 1s, in a later training stage (epoch 5, batch 300, we chose to show a later stage as the filters were formed more completely). Their corresponding activations from the *TensorPath* view are also shown. From the figure, we can see that filter 2 and 6 are two directional filters. The effect of a directional filter is similar to cast a light from different directions to the content of the original image, and the filter decides the source and direction of the light. The left side of filter 2 is white, and the color changes to black towards the right side. This indicates the light starts from left and is cast towards the right. Under this filter, the shape of digit 1s (both vertical and italic) can be extracted. The binary activations illustrate the difference of the extracted features more clearly. These activations are binary images derived from the original activations by thresholding (i.e., pixels with values greater than 128 are set to 255, otherwise they are set to 0). Filter 6 can be interpreted in the same way, as it is also a directional filter (from bottom-right to top-left). In Figure 15e, the most important filters of vertical/italic digit 0s in the same timestamp present another example. Under the effect of filter 7 (a directional filter from top-left to bottom-right), two sides of the italic digit 0 are activated. However, only a small portion of the vertical digit 0 is activated under this filter. As a result, filter 7 plays a very important role in differentiating the two digit 0s.

When it comes to the QuickDraw data, we observed that filters with more functions were involved in differentiating two drawings. For example, in Figure 16a, a pair of *smiley-face* images with different visual features are selected into *TensorPath*. In the enlarged version of this view (Figure 16c), one can figure out that, in addition to the two directional filters, i.e., 8 and 5, which capture the horizontal and vertical features, filter 2 (that captures the contour of one image) is also important to differentiate the two images. The differences in the three pairs of activations in Figure 16c help $D$ generate different probability values for these two drawings.

### 6.2.3 Comparative Analysis

*Understanding the evolution of decision-critical features (R3.2).* *GANViz* also enables users to understand how the decision-critical features evolve over time. We observed that for both MNIST and QuickDraw, the important filters to differentiate two images became stable after several epochs, even $D$ may still flip its decision features. For example, the activation comparison view in Figure 5e records four rows of activations (of the selected digits 1s) in four different timestamps. Figure 15a is an enlarged version of the first hidden layer. From this figure, we observed that filter 3 and 0 were the most important in epoch 1, but they became less and less important over time. In contrast, filter 2 and 6 became more important since epoch 5, and they kept being the most important two filters in later training stages.

Also, *GANViz* can reveal how filters are formed and what features are extracted over time. Figure 16b shows the activation comparison view of two *smiley-faces* in three different epochs (epoch 0, 1 and 2, we present three early epochs here, as filters change more significantly at the beginning). From the activation comparison view, we realized that filter 8 was becoming more and more important in discriminating the two images. In the enlarged view of the three epochs, we observed that: (1) the horizontal filter 8 was not formed completely in epoch 0, and few features were extracted by it (Figure 16e, left); (2) the filter was roughly formed in epoch 1 (Figure16d, left) and could extract some disjointed pixels in this epoch; (3) in epoch 2 (Figure 16c, left), the filter could extract continuous regions from images and we can roughly

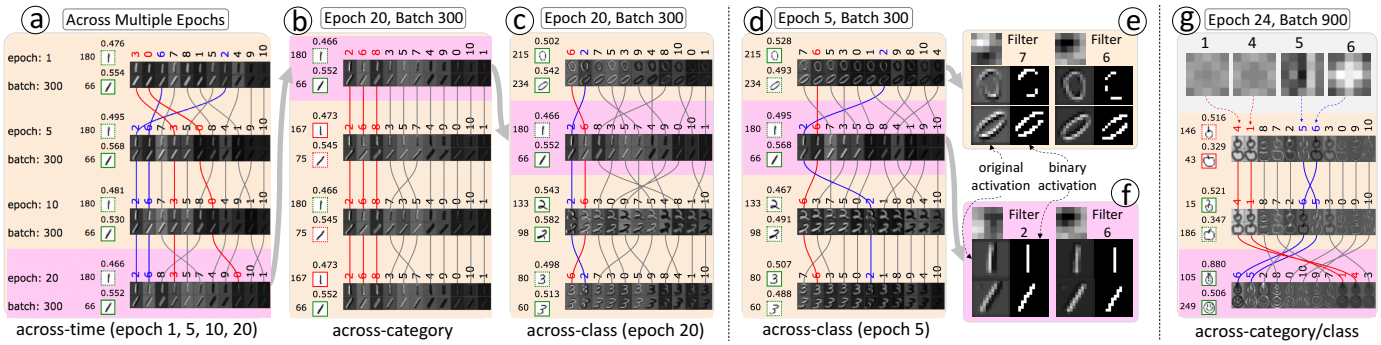



Fig. 15. (a) Tracking a pair of digit 1s across four timestamps; (b) comparing vertical/italic digit 1s across categories; (c, d) comparing vertical/italic digits across classes in epoch 20, epoch 5; (e, f) enlarged important neurons/activations of digit 0s, 1s from *TensorPath*; (g) comparing across categories (top two rows) and across classes (bottom two rows) in QuickDraw, along with three circular filters: 1, 4, 6; and one directional filter: 5.

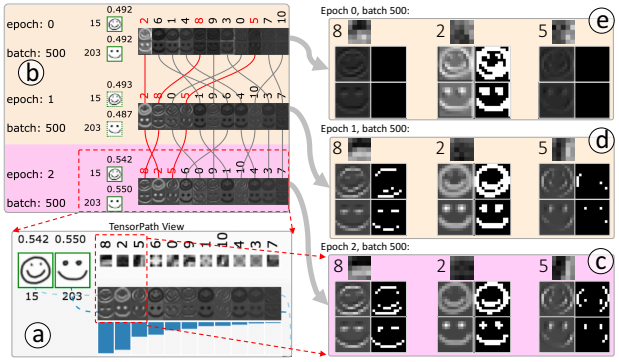



Fig. 16. Across-time comparison: (a) the *TensorPath* view shows filter 8, 2, 5 are the most important; (b) trace the three neurons in epoch 0, 1, 2; (c, d, e) enlarged filters and activations in the three epochs.

recognize the *smiley-face* from the activated pixels. The same learning pattern can also be found in filter 5. Filter 2 extracts the contour of a *smiley-face* image. Over the three epochs, the extracted contour became more and more precise.

*Comparing decision-critical features over different decision categories (R3.1).* For images of the same class, we found that the decision-critical features were fairly consistent for images categorized into different categories (e.g., $FN$, $TP$). For MNIST, as shown in Figure 15b, although the four pairs of images (one vertical, one italic) represent different pairs of categories (category information is encoded in the images' border style and color), we can see that the four orders of filters in differentiating the four pairs are mostly the same. For example, filter 2, 6 and 8 (filter 0, 10 and 1) are always the most (least) important. The similar orders indicate that $D$ treats images from different sources equally. Before seeing the visualization results, the experts were uncertain about this phenomenon. They responded that the comparative analysis enhanced their understanding about $D$.

The decision-critical features were also observed to be consistent in QuickDraw. From the top two rows of Figure 15g, we noticed that $D$ used similar decision features when differentiating fake (the top row) and real (the second row) *apples*, as the orders of activation pairs were mostly the same. Big *apples*, either from fake or real data, were considered as fake by $D$; small *apples* were considered as real. Circular filter 1 and 4 played the most important roles.

*Comparing the decision-critical features over different image classes (R3.1).* At a specific timestamp, the domain experts were wondering if the decision-critical features are also consistent in differentiating images from different classes (e.g., digit 0 vs. 1). For simple images (from MNIST), we found that the decision-critical features were more stable in later training stages. Figure 15c shows the comparison of vertical/italic digit 0, 1, 2 and 3 in epoch 20. Filter 2, 6 are always very important in differentiating vertical and italic digits, which indicates the learned decision-critical features are quite stable. However, in a relatively earlier stage, such as epoch 5 (Figure 15d), the importance of filter 2 is different in different classes, though filter 6 is always very important.

For complicated images (from QuickDraw), however, we found the decision-critical features varied even in later training stages. The last two rows in Figure 15g show the comparison of pairs of *apples* and *smiley-faces* (across-class), in epoch 24. Filter 6, 5 replaced filter 1, 4 and became the most important filters in differentiating small (oval) and large (circular) *smiley-faces*, and we did not observe consistent decision-critical features across classes in QuickDraw.

### 6.3 Feedback from Domain Experts

We conducted case study workshops with three domain experts ($E1$, $E2$, $E3$), who are senior researchers with 3+ years experience in deep learning, and 5∼10 years experience in machine learning. We used a guided exploration and think-aloud approach as the study protocol. We first went over the high-level goals and introduced the components of *GANViz*, and then guided them to walk through the case study scenarios with both datasets, as explained in Section 6.2. The walk-through was in an interactive way to think aloud the insights, pros and cons of the design. The study ended up with an exit interview to collect feedback and suggestions.

Overall, all experts appreciated the rich details behind the scene of GANs' training revealed by *GANViz*. $E2$ stated that the components offered him new insights at different levels, especially, the probability distribution of $D$ and the generated results of $G$ over the training. He also commented that the top-down analysis flow bears the intrinsic logics to understand GAN models. The experts all agreed that *"only looking at loss function is quite limited and sometimes even misleading"*. $E1$ *"liked the distribution view most"* and observed some generated drawings with broken strokes in the distribution view even in the later training stages, and

she suspected the training should carry on. They all liked the AUC curve for the performance of $D$, which is usually overlooked in many evaluations. $E3$ mentioned that the two datasets together revealed the strength and weakness of GAN models. Both $E1$ and $E2$ were impressed by the comparative analysis, which showed how the decision features were evolved, and commented "*it is enlightening to know the top filters were playing different roles in D in the two datasets*".

More interestingly, their explorations triggered some discussions on model improvements. The probability view of $D$ inspired $E2$ to think about the way of separating the training data based on current predicted scores for next training iteration, as he observed that some real images with poor quality (e.g., bad doodling in QuickDraw) actually distracted the training of $D$. Both $E1$ and $E3$ noticed that $D$ made incorrect predictions for some creative drawings in the real data (e.g., *smiley-faces* with a stuck-out tongue), and they discussed about extending GANs with the capability of generating and discriminating creative data. $E1$ even threw out a paradigm-shift question: "*is it reasonable to have the one-to-one mapping from the data point in z sample space to the generation space in GANs?*" Further investigations on this could nurture a novel GAN framework.

The experts also mentioned some frustrations and desirable features. One frustration is to understand the heatmap like weights of the fully connected layers in the *TensorPath* view, because of the "*overwhelming information without sufficient interpretations*". One metric suggested by $E2$ is looking at the variation of the generated images by $G$ over the training process to understand how stable $G$ is. $E3$ suggested that the Diversity metric should also consider the complexity level of real images. $E3$ also mentioned that the feature of problem diagnosing is desirable to help users quickly locate patterns with potential pitfalls, such as detecting the mode collapse issue in certain epochs.

## 7 Discussion, Limitations, and Future Work

*Model quality measurement.* Measuring the quality of a GAN model is still an open question in the machine learning community [6]. The loss functions cannot reflect sufficient details about the model quality, as we have demonstrated in our case studies. Proposing the three new metrics to measure GANs' quality is an initial attempt. The metrics did reveal certain aspects of model quality, however, their effectiveness in different scenarios requires further validations, which is out of the scope of this paper. Also, those metrics can be easily replaced by others to plug into our system.

*Flipping of decision-critical features.* One interesting observation we had (and the domain experts also thought "*quite phenomenal*") is that the model flipped the decision-critical features from time to time, and such flipping occurred more frequently for simple images than complicated ones. One possible explanation is that: as $G$ gets stronger, it is hard for $D$ to select critical features to make decisions. Therefore, measuring the decision flipping could potentially be a useful metric to monitor the training process and model quality. This requires more explorations with theoretical supports, and we consider it as an important extension of our work.

*Representative image selection.* We selected images with representative features into *TensorPath* and compared their details in $D$'s network to examine how $D$ differentiated them. However, this selection is inevitably subjective. Different people may select different images of digit 1 to represent images with/without italic styles. We consider this as one limitation of our work, and a remedy we have planned for the future is to derive some summary statistics or priority scores for each image to facilitate the selection.

In the future, we would like to generalize *GANViz* from two perspectives: (1) analyzing more complex image data; (2) extending to other types of GANs. For the first part, we target on images with higher resolutions and more complex features. As we can easily scale/pad/aggregate images, we do not expect big challenges in adapting *GANViz* to them, though the decision-critical features may become more complicated and not easily interpretable. For the second part, we plan to explore GANs for other types of data (e.g., texts, sounds). We believe most parts of our framework are still reusable, such as the metrics we proposed to measure GANs' quality and the idea of comparing activations. The challenge appears in the visualization of individual data instances, i.e., how to present a snippet of sounds or texts. We observe that those data can easily be vectorized (e.g., texts are sequences of characters), and thus we should be able to transfer/reorganize them into 2D spaces/shapes and visualize them as image thumbnails in *GANViz*. Moreover, some views may need to undergo a new design process. For example, for text data, the distribution view may use a word cloud to demonstrate the distribution of real/fake texts.

## 8 Conclusion

In this work, we designed and developed *GANViz*, a visual analytics system that helps to evaluate, interpret, and analyze GANs. *GANViz* has five visualization components: the metric view, probability view, and distribution view allow users to conduct model evaluation and interpretation without introducing neural network structures; the *TensorPath* view and activation comparison view enable users to dive into details of network architectures, and perform detailed comparative analysis. All five views are linked together to facilitate users' explorations and visual reasonings. Important findings from real-world datasets and positive feedback from domain experts verified the effectiveness of *GANViz*.

## Acknowledgments

This work was supported in part by NSF grants IIS-1250752, IIS-1065025, and US Department of Energy grants DE-SC0007444, DE-DC0012495, program manager Lucy Nowell.

## References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

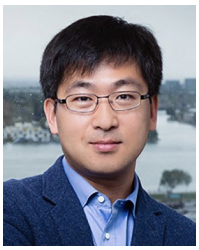
[2] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[3] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2016, pp. 2172–2180.

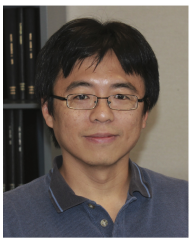
[4] C. Ledig, L. Theis, F. Huszr, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 105–114.
[5] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *Journal of Machine Learning Research*, vol. 11, no. Feb, pp. 625–660, 2010.
[6] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.
[7] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," in *Deep Learning Workshop, International Conf. on Machine Learning*, 2015.
[8] "Tensorboard," https://www.tensorflow.org/get_started/summaries_and_tensorboard, accessed: 2017-08-22.
[9] F. Y. Tzeng and K. L. Ma, "Opening the black box - data driven visualization of neural networks," in *VIS 05. IEEE Visualization, 2005.*, Oct 2005, pp. 383–390.
[10] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.
[11] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu, "Towards better analysis of deep convolutional neural networks," *IEEE Trans. on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 91–100, 2017.
[12] A. Karpathy, J. Johnson, and L. Fei-Fei, "Visualizing and Understanding Recurrent Networks," *arXiv:1506.02078 [cs]*, Jun. 2015. [Online]. Available: http://arxiv.org/abs/1506.02078
[13] H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush, "Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 667–676, Jan 2018.
[14] M. Liu, J. Shi, K. Cao, J. Zhu, and S. Liu, "Analyzing the training processes of deep generative models," *IEEE transactions on visualization and computer graphics*, vol. 24, no. 1, pp. 77–87, 2018.
[15] A. Odena, "Semi-Supervised Learning with Generative Adversarial Networks," *arXiv:1606.01583 [cs, stat]*, Jun. 2016, arXiv: 1606.01583. [Online]. Available: http://arxiv.org/abs/1606.01583
[16] A. Odena, C. Olah, and J. Shlens, "Conditional Image Synthesis With Auxiliary Classifier GANs," *arXiv:1610.09585 [cs, stat]*, Oct. 2016. [Online]. Available: http://arxiv.org/abs/1610.09585
[17] I. Goodfellow, "NIPS 2016 Tutorial: Generative Adversarial Networks," *arXiv:1701.00160 [cs]*, Dec. 2016, arXiv: 1701.00160. [Online]. Available: http://arxiv.org/abs/1701.00160
[18] S. Arora, R. Ge, Y. Liang, T. Ma, and Y. Zhang, "Generalization and Equilibrium in Generative Adversarial Nets (GANs)," Mar. 2017. [Online]. Available: http://arxiv.org/abs/1703.00573
[19] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," *arXiv preprint arXiv:1611.02163*, 2016.
[20] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," Jan. 2017. [Online]. Available: http://arxiv.org/abs/1701.07875
[21] "Dcgan, tensorflow code," https://github.com/carpedm20/DCGAN-tensorflow, accessed: 2017-08-08.
[22] J. Fogarty, R. S. Baker, and S. E. Hudson, "Case studies in the use of roc curve analysis for sensor-based estimates in human computer interaction," in *Proceedings of Graphics Interface*, 2005, pp. 129–136.
[23] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.
[24] J. Bilmes, "A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models," Tech. Rep., 1998.
[25] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.
[26] B. Shneiderman, "The eyes have it: A task by data type taxonomy for information visualizations," in *Visual Languages, 1996. Proceedings., IEEE Symposium on*. IEEE, 1996, pp. 336–343.
[27] M. Krstajic, E. Bertini, and D. Keim, "Cloudlines: Compact display of event episodes in multiple time-series," *IEEE Trans. on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2432–2439, 2011.
[28] C. Perin, F. Vernier, and J.-D. Fekete, "Interactive horizon graphs: Improving the compact visualization of multiple time series," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2013, pp. 3217–3226.
[29] H. Hochheiser and B. Shneiderman, "Dynamic query tools for time series data sets: timebox widgets for interactive exploration," *Information Visualization*, vol. 3, no. 1, pp. 1–18, 2004.
[30] W. Javed, B. McDonnel, and N. Elmqvist, "Graphical perception of multiple time series," *IEEE transactions on visualization and computer graphics*, vol. 16, no. 6, pp. 927–934, 2010.
[31] 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.
[32] J. Wang, X. Liu, H.-W. Shen, and G. Lin, "Multi-resolution climate ensemble parameter analysis with nested parallel coordinates plots," *IEEE TVCG*, vol. 23, no. 1, pp. 81–90, 2017.
[33] "Google quick draw game," https://quickdraw.withgoogle.com/, accessed: 2017-08-08.

**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 Sr. Researcher of 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, and M.S. in Information Science from Renmin Univ., China.



**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.