Analyzing the Noise Robustness of Deep Neural Networks

Mengchen Liu*, Shixia Liu*, Hang Su†, Kelei Cao*, Jun Zhu†

*School of Software, TNList Lab, State Key Lab for Intell. Tech. Sys., Tsinghua University
†Dept. of Comp. Sci.Tech., TNList Lab, State Key Lab for Intell. Tech. Sys., CBICR Center, Tsinghua University

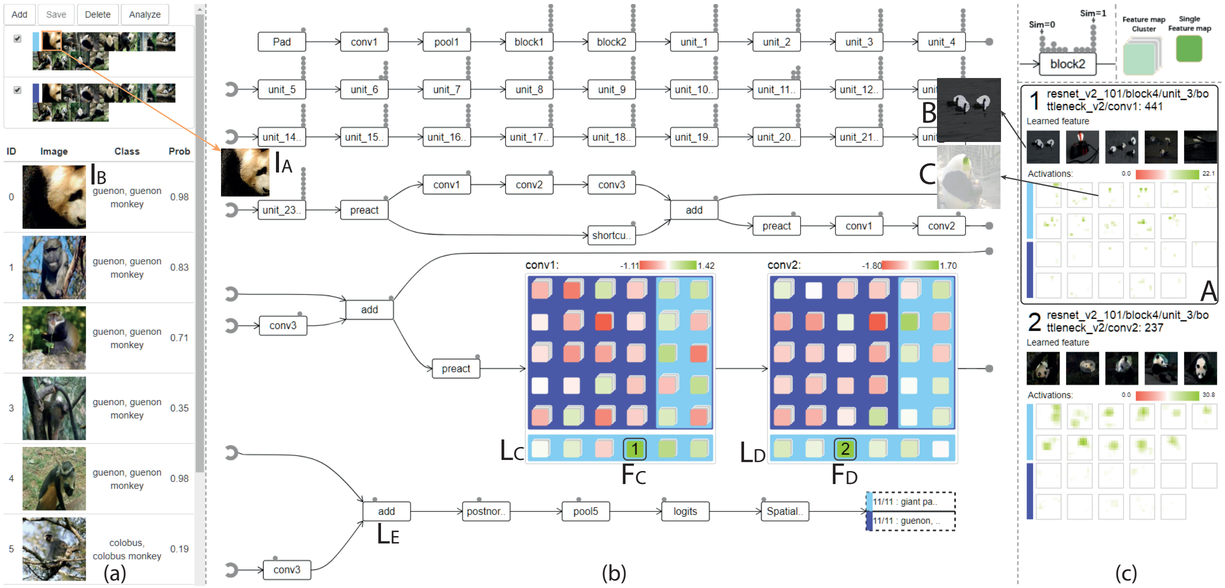

Figure 1: Explaining the misclassification of adversarial panda images. The root cause is that the CNN cannot detect panda's ears in the adversarial examples (F_C), which leads to the failure of detecting a panda's face (F_D). As a result, these adversarial examples are misclassified: (a) input images; (b) datapath visualization at the layer and feature map levels; (c) neuron visualization.

Abstract

Deep neural networks (DNNs) are vulnerable to maliciously generated adversarial examples. These examples are intentionally designed by making imperceptible perturbations and often mislead a DNN into making an incorrect prediction. This phenomenon means that there is significant risk in applying DNNs to safety-critical applications, such as driverless cars. To address this issue, we present a visual analytics approach to explain the primary cause of the wrong predictions introduced by adversarial examples. The key is to analyze the datapaths of the adversarial examples and compare them with those of the normal examples. A datapath is a group of critical neurons and their connections. To this end, we formulate the datapath extraction as a subset selection problem and approximately solve it based on back-propagation. A multi-level visualization consisting of a segmented DAG (layer level), an Euler diagram (feature map level), and a heat map (neuron level), has been designed to help experts investigate datapaths from the high-level layers to the detailed neuron activations. Two case studies are conducted that demonstrate the promise of our approach in support of explaining the working mechanism of adversarial examples.

Keywords: Deep neural networks, robustness, adversarial examples, back propagation, multi-level visualization.

1 Introduction

Deep neural networks (DNNs) have evolved to become state-of-the-art in a torrent of artificial intelligence applications, such as image classification and language translation [26, 29, 59, 60]. However, researchers have recently found that DNNs are generally vulnerable to maliciously generated adversarial examples, which are intentionally designed to mislead a DNN into making incorrect predictions [34, 37, 53, 63]. For example, an attacker can modify an image of a panda (I_A in Fig. 1) slightly, even imperceptibly to human eyes, and the generated adversarial example (I_B in Fig. 1) is able to mislead a state-of-the-art DNN [21] to classify it as something else entirely (e.g., a monkey), because the DNN detects a monkey's face in the top right corner of the adversarial example (Fig. 11A). This phenomenon brings high risk in applying DNNs to safety- and security-critical applications, such as driverless cars, facial recognition ATMs, and Face ID security on mobile phones [1]. Hence, there is a growing need to understand the inner workings of adversarial examples and identify the root cause of the incorrect predictions.

There are two technical challenges to understanding and analyzing adversarial examples, which are derived from the discussions with machine learning experts (Sec. 3) and previous research on adversarial examples [15, 34, 37]. The first challenge is how to extract the datapath for adversarial examples. A datapath includes the critical neurons and their connections that are responsible for the predictions of the examples (Fig. 2 (a)). Disclosing the datapath will help experts design more targeted defense approaches. However, in a DNN, the neurons have complex interactions with each other [6]. Thus, it is technically demanding to disentangle these neurons from the whole network and thus form the datapath. The second challenge is how to effectively illustrate the inner workings of adversarial examples based on the extracted datapaths. A state-of-the-art DNN

*e-mail: {{liumc13,ckl17}@mails, shixia@mail}.tsinghua.edu.cn. S. Liu is the corresponding author.
†e-mail: {suhangss,dcszj}@mail.tsinghua.edu.cn

usually contains hundreds of layers, with millions of neurons in each layer [21]. Thus, an extracted datapath potentially contains millions of neurons and even more connections. Directly visualizing all the neurons and the corresponding connections in a datapath will lead to excessive visual clutter.

To tackle these challenges, we have developed a visual analytics tool, AEVis, to explain the root cause of the wrong predictions introduced by adversarial examples. The key is to effectively extract and understand the datapath of adversarial examples. We formulate the datapath extraction as a subset selection problem, which is NP-complete [12]. To analyze the adversarial examples in large DNNs, we approximate the subset selection problem as an easy-to-solve quadratic optimization by Taylor decomposition [45], and solve the quadratic optimization using back-propagation [8]. Based on the extracted datapaths, we design a multi-level visualization that enables experts to effectively explore and compare datapaths from the high-level layers to the detailed neuron activations. In particular, at the layer-level, we design a segmented directed acyclic graph (DAG) visualization to provide an overview of the datapaths (Fig. 1 (b)). As shown in Fig. 1 (c), the detailed neuron activations are presented as heat maps that are familiar to machine learning experts (neuron level). Between the layer level visualization and neuron level, we have added a feature map level because a layer may contain millions of critical neurons. A DNN, especially a convolutional neural network (CNN), organizes neurons in feature maps, each of which is a set of neurons sharing the same weight and thus detecting the same feature. This inherent property enables the features to be recognized regardless of their position in the input (e.g., an image) and thus improves the generalization of DNNs [17]. At the feature map level, we employ an Euler diagram to illustrate and compare critical feature maps belonging to different datapaths. Two case studies are conducted to demonstrate that our approach can better explain the working mechanism of both white-box and black-box adversarial examples.

The key technical contributions of this paper are:

- **A visual analytics tool** that explains the primary cause of the wrong predictions introduced by adversarial examples.
- **A datapath extraction method** that discloses critical neurons and their connections that are responsible for a prediction.
- **A multi-level datapath visualization** that enables experts to examine and compare datapaths, from the high-level layers to the detailed neuron activations.

In this paper, we focus on analyzing adversarial examples generated for CNNs on the task of image classification, because currently most adversarial example generation approaches focus on attacking CNNs on the image classification task [1] . Besides CNNs, AEVis can be directly used to analyze other types of deep models, such as multilayer perceptrons (MLPs).

2 Related Work

2.1 Visual Analytics for Explainable Deep Learning

A number of visual analytics approaches [7, 27, 28, 33, 40, 41, 49, 57] have been developed to illustrate the working mechanism of DNNs. A comprehensive survey on exploring the space of interpretability interfaces was presented by Olah et al. [38]. Most recent approaches on explainable deep learning can be categorized into two groups: network-centric [29, 54, 57] and example-centric approaches [20, 24].

Network-centric approaches focus on illustrating the network structure of a DNN. Tzeng et al. employed a DAG visualization to illustrate the neurons and their connections. In particular, each neuron is represented by a node and their connections are represented by edges. Their method can illustrate the structure of a small neural network, but suffers from severe visual clutter when visualizing state-of-the-art DNNs. To solve this problem, Liu et al. [29] developed a scalable visual analytics tool, CNNVis, based on clustering techniques. It helps machine learning experts to diagnose a failed training process. Wongsuphasawat et al. [57] developed a tool with a scalable graph visualization (TensorFlow Graph Visualizer) to present the dataflow graph of a DNN. To produce a legible graph visualization, they apply a set of graph transformations that converts the low-level dataflow graph to the high-level structure of a DNN. The aforementioned approaches facilitate experts in better understanding the network structure, but they are less capable of explaining the predictions of individual examples. For example, the TensorFlow Graph Visualizer developed by Wongsuphasawat et al. [57] does not extract and disclose the datapath of a set of examples, which is critical for identifying the root cause of the misclassification produced by adversarial examples.

There are several recent attempts to explain how DNNs make predictions for examples (example-centric approaches). A widely used approach is to feed a set of examples into a DNN, and visualize the internal activations produced by the examples. For example, Hartley et al. [20] developed an interactive node-link visualization to show the activations in a DNN. Although this method is able to illustrate detailed activations on feature maps, it suffers from severe visual clutter when dealing with large CNNs. To solve this problem, Kahng et al. [24] developed ActiVis to interpret large-scale DNNs and their results. They employed a multiple coordinated visualization to facilitate experts in comparing activations among examples and reveal the causes for misclassification. Although ActiVis can show the causes for misclassification to some extent, it cannot be directly used to illustrate the primary causes of the wrong prediction caused by adversarial examples as it heavily relies on expert knowledge to select which layer to examine. In addition, we argue that purely relying on activations in one layer will result in misleading results (Sec. 6.1). To solve this problem, we propose combining activations and gradients for selecting critical neurons at different layers, which are connected to form a datapath for the adversarial examples of interest. In addition, we integrate a DAG visualization with dot plots, which provide guided visual hints to help experts select the layer of interest.

2.2 Adversarial Attacks on Deep Learning

Adversarial attacks are a new research focus of DNNs [1]. Existing efforts mainly focus on the generation of adversarial examples [34, 37] and how to defend against adversarial examples [11, 63].

The key of generating an adversarial example is to find a very small perturbation that can mislead DNNs into misclassification. Recently, researchers have proposed a variety of approaches for finding such perturbations, including the Fast Gradient Sign Method [18], universal adversarial perturbation [34], and DeepFool [35].

The generation of adversarial examples has inspired researchers to develop several methods to defend against adversarial attacks. A natural defense is training a DNN using adversarial examples (adversarial training) [18, 53]. Although adversarial training can defend the training adversarial examples, Moosavi-Dezfooli et al. [34] discovered that new types of adversarial examples can be generated to attack DNNs that have been trained in this way. To tackle this issue, a more effective strategy is to change the network structure to improve the defense against unseen adversarial examples. Adding regularization to the corresponding layer(s) is one of the most commonly used methods for this. The typical regularization methods include input gradient regularization [44] and layer-wise Lipschitz regularization [11]. However, due to the limited understanding of the working mechanism of adversarial examples, the above defense strategies are often very heavy, which usually leads to performance degradation for large, complex datasets such as ImageNet [46]. To solve this problem, we have developed a visual analytics tool to explain the primary cause of the wrong predictions introduced by adversarial examples. Based on a few vulnerable neurons identified by AEVis (Fig. 1A), machine learning experts can design more targeted and effective defense strategies.

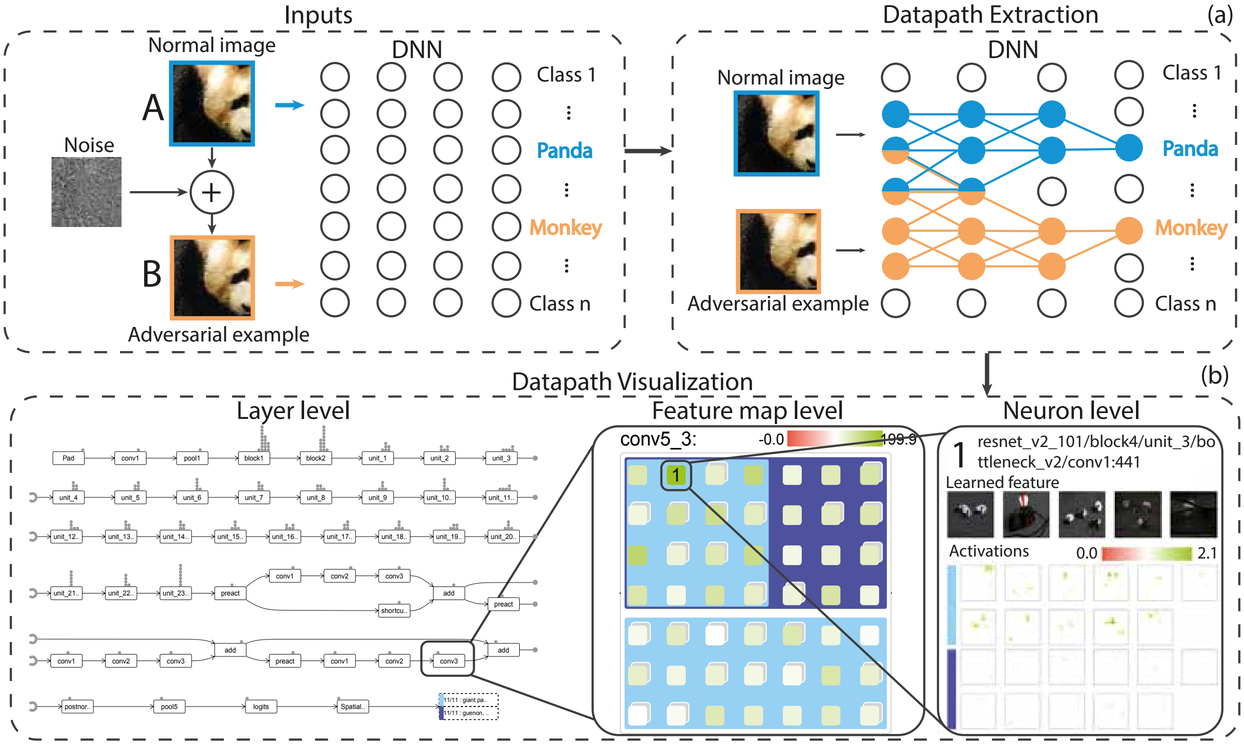

Figure 2: AEVis contains two modules: (a) a datapath extraction module and (b) a datapath visualization module that illustrates datapaths in multiple levels: layer level, feature map level, and neuron level.

3 The Design of AEVis

3.1 Motivation

The development of AEVis is collaborated with the machine learning team that won the first place in the NIPS 2017 non-targeted adversarial attack and targeted adversarial attack competitions, which aim at attacking CNNs [15, 51]. Despite their promising results, the experts found that the research process was inefficient and inconvenient, especially the explanation of the model outputs. In their research process, a central step is explaining misclassification introduced by adversarial examples. Understanding why an error has been made helps the experts detect the weakness of the model and further design a more effective attacking/defending approach. To this end, they desire to understand the roles of the neurons and their connections for prediction. Because there are millions of neurons in a CNN, examining all neurons and their connections is prohibitive. In the prediction of a set of examples, the experts usually **extract** and **examine** the critical neurons and their connections, which are referred to as **datapaths** in their field.

To extract datapaths, the experts often treat the most activated neurons as the critical neurons [62]. However, they are not satisfied with the current activation-based approach because it may result in misleading results. For instance, considering an image with highly recognizable secondary objects, which are mixed with the main object in the image. The activations of the neurons that detect the secondary objects are also large, however, the experts are not interested in them because these neurons are often irrelevant to the prediction of the main object. Currently, the experts have to rely on their knowledge to manually ignore these neurons in the analysis process.

After extracting datapaths, the experts examine them to understand their roles for prediction. Currently, they utilize discrepancy maps [64], heat maps [62], and weight visualization [18] to understand the role of the datapaths. Although these methods can help the experts at the neuron level, they commented that there lacked an effective exploration mechanism enabling them to investigate the extracted datapaths from high-level layers to individual neurons.

3.2 Requirement Analysis

To collect the requirements of our tool, we follow the human-centered design process [9, 32], which involves two experts (E_1 and E_2) from the winning team of the NIPS 2017 competition. The design process consists of several iterations. In each iteration, we present the developed prototype to the experts, probe further requirements, and modify our tool accordingly. We have identified the following high-level requirements in this process. Among these requirements, **R2** and **R3** are two initial requirements, while **R1** and **R4** are gradually identified in the development.

R1 - Extracting the datapath for a set of examples of interest. Both experts expressed the need for extracting the datapath of an example, which serves as the basis for analyzing why an adversarial example is misclassified. In a CNN, different neurons learn to detect different features [62]. Thus, the roles of the neurons are different for the prediction of an example. E_1 said that analyzing the datapath can greatly save experts' effort because they are able to focus on the critical neurons instead of examining all neurons. Besides the datapath for individual examples, E_1 emphasized the need for extracting the common datapath for a set of examples of the same class. He commented that the datapath of one example sometimes is not representative for the image class. For example, given an image of a panda's face, the extracted datapath will probably not include the neuron detecting the body of a panda, which is also a very important feature to classify a panda.

R2 - Providing an overview of the datapath. In a large CNN, a datapath often contains millions of neurons and connections. Directly presenting all neurons in a datapath will induce severe visual clutter. Thus, it is necessary to provide experts an overview of a datapath. E_1 commented, "I cannot examine all the neurons in a datapath because there are too many of them. In the examining process, I often start by selecting an important layer based on my knowledge, and examine the neurons in that layer to analyze the learned features and the activations of these neurons. The problem of this method is when dealing with a new architecture, I may not know which layer to start with. Thus, I have to examine a bunch of layers, which is very tedious." Thus, it is necessary to provide the experts an overview of the datapath with visual guidance to facilitates experts in selecting the layer of interest. The requirement of providing an overview of a CNN aligns well with previous research [24, 28, 57].

R3 - Exploring a datapath at the detailed levels. Although the overview of a datapath facilitates experts in finding the layer of interest, it is not enough to diagnose the root cause of the wrong prediction introduced by an adversarial example. The experts said that they wanted to link the overview of a datapath with detailed neuron activations. This linkage helps them identify the most important neurons that lead to the misclassification. Since a layer may contain millions of critical neurons, the experts also desired a medium level between the layer level and neuron level. For CNNs, the experts recommended to group neurons into feature maps. E_2 said that, "Neurons in a feature map learn to detect the same feature. Grouping them is very common in our research." Previous research also indicates that visual analytics for deep learning benefits from multi-level visualization [24, 28].

R4 - Comparing multiple datapaths. An adversarial example is often generated by slightly perturbing the pixel values of a normal image. Accordingly, a normal image and the corresponding adversarial example are nearly the same in the input space. However, their prediction results are different. The experts are interested in how they diverge to different predictions. For example, E_2 commented, "I want to know whether there are some critical 'diverging points' for the prediction difference or it is accumulated gradually layer by layer through the network." To this end, E_2 desired to compare the datapaths of normal examples and adversarial examples. Inspired by E_2, E_1 added that it was interesting to compare the datapath of an adversarial example (e.g., a panda image that is misclassified as a monkey) with that of the images from the predicted class (e.g., normal images containing monkeys). Such comparison helps them understand how these very different images "merge" into the same prediction (e.g., the monkey). The need of visual comparison is consistent with the findings of previous research [2, 16, 30].

3.3 System Overview

Driven by the requirements collected from the experts, we have developed a visual analytics tool, AEVis, to illustrate the root cause of the robustness issues caused by adversarial examples. This tool consists of the following two parts.

- **A datapath extraction module** that extracts the critical neurons and their connections for a set of selected examples (**R1**).
- **A datapath visualization module** that provides an overall understanding of the datapath of interest (**R2**), illustrates datapaths in multiple levels (**R3**), and facilitates experts to visually compare several datapaths (**R4**).

As shown in Fig. 2, AEVis takes a trained CNN and the examples to be analyzed as the input. The examples include both normal examples (Fig. 2A) and adversarial examples (Fig. 2B). Given the examples and the CNN, the datapath extraction module extracts the critical feature maps and their connections that are responsible for the predictions of the examples (Fig. 2 (a)). The extracted datapaths are then fed into the datapath visualization module (Fig. 2 (b)), which supports the navigation and comparison of the datapaths from the high-level layers to the detailed neuron activations.

A typical analysis workflow of AEVis is shown in Fig. 1. An expert explores the image list (Fig. 1 (a)) and selects several groups of images for further understanding and analysis. After clicking on the 'Analyze' button, AEVis first provides an overview of the datapaths at the layer level (Fig. 1 (b)). At the layer level, each rectangle represents a layer group and a dot plot is combined with each layer group to illustrate the similarities between/among the extracted datapaths of the layers in each layer group. By examining the dot plots, the expert is able to detect layers of interest and further examines the feature maps in the layer (L_C and L_D in Fig. 1). In each layer, the Euler-diagram-based design helps the expert focus on the share/unique feature maps of several datapaths. Aided by this design and the color coding (e.g., activations) of feature maps, the expert can select a feature map of interest (F_C and F_D in Fig. 1), and explore the detailed neuron activations as well as the learned features of the neurons in this feature map (Fig. 1 (c)). It facilitates him/her in finding the root cause (e.g., a set of neurons) for the misclassification of the adversarial examples.

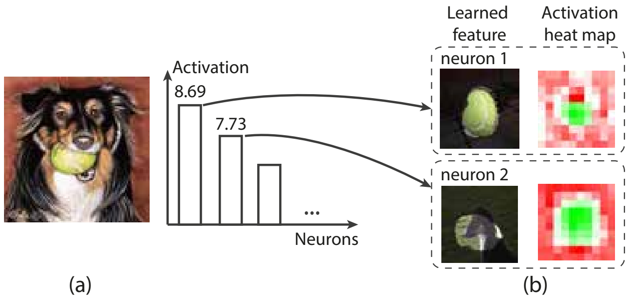

Figure 3: A misleading result of the activation-based datapath extraction approach: (a) the input image; (b) the top 2 critical neurons found by the activation-based approach. The learned feature is computed by the discrepancy maps [64] and the activation heat map show the activations on the corresponding feature map.

4 Datapath Extraction

4.1 Motivation and Problem Formulation

Extracting the datapath that are responsible for the prediction of a group of examples is the basis for analyzing why an adversarial example is misclassified (**R1**). The key challenge is to identify the critical neurons as selecting the corresponding connections to form the datapath is straightforward. Next we discuss how to extract the critical neurons for an individual example and then extend our approach to selecting critical neurons for a set of examples.

A commonly used method is to treat the most activated neurons as the critical neurons [7, 24]. However, this method may result in misleading results, especially when a highly recognizable secondary object is mixed with the main object in an image. For example, Fig. 3 shows the top 2 critical neurons found by such activation-based approach for a sheepdog image (Fig. 3 (a), employed network: ResNet-101 [21], image label: Shetland sheepdog). We can see that the second highly activated neuron learns to detect the head of a dog (Fig. 3 (b)), which is indeed critical for classifying a sheepdog. However, the most critical neuron learns to detect a ball (Fig. 3 (b)), which is not an important feature for classifying a sheepdog. Such a misleading result roots in that the neurons have complex interactions with each other and the activations of the neurons are processed by a highly nonlinear function to generate the final prediction. Thus, highly activated neurons may not be the critical neurons for a prediction. Such critical neurons are the neurons that highly contributed to the final prediction. In other word, by only combining the contributions of the critical neurons, the prediction of an example will not be changed. To this end, we aim at selecting a minimized subset of neurons, which keep the original prediction. Accordingly, we formulate critical neurons extraction as a subset selection problem:

$$N^{opt} = \underset{N_s \subseteq N}{\arg\min} \, (p(x) - p(x; N_s))^2 + \lambda |N_s|^2. \tag{1}$$

The first term is to keep the original prediction and the second term ensures to select a minimized subset of neurons. Specifically, N is the set of the neurons in a CNN, N_s is a subset of neurons N, N^{opt} is the calculated critical neurons, $p(x)$ is the prediction of example x, and $p(x; N_s)$ is the prediction if we only consider the neuron subset N_s. To measure the difference between two predictions, we adopt the widely used L2-norm. $|N_s|$ is the size of N_s and λ is used to balance the two terms. A larger λ indicates a tendency to select a smaller subset of neurons.

As discussed in Sec. 3.2, extracting the common datapath for a set of examples instead of an individual example can improve the representativeness of the extracted datapath (**R1**). A natural extension of Eq. (1) is minimizing the sum of the difference in the prediction of the example set X:

$$N^{opt} = \underset{N_s \subseteq N}{\arg\min} \sum_{x_k \in X} (p(x_k) - p(x_k; N_s))^2 + \lambda |N_s|^2. \tag{2}$$

Next, we discuss how to effectively solve this problem. For simplicity, we take Eq. (1) (critical neurons for one example) as an example to illustrate the solution.

4.2 Solution to Subset Selection

Directly solving the subset selection problem (Eq. (1)) is time-consuming because: (1) it is an NP-complete problem and (2) the search space is prohibitively large due to the large number of neurons in a CNN. We therefore combine a divide-and-conquer approach with a quadratic optimization to reduce the search space and find a more accurate approximation. In particular, we develop a divide-and-conquer-based **search space reduction** method by splitting the subset selection problem into a series of separate subproblems and grouping the neurons into feature maps. As each subproblem is still NP-complete, we then employ the **quadratic approximation** to more accurately approximate each NP-complete subproblem as an easy-to-solve quadratic optimization by Taylor decomposition [45], and solve it based on back-propagation [8].

4.2.1 Search space reduction

A CNN is traditionally represented as a directed acyclic graph (DAG), where each node represents a neuron and each edge denotes a connection between nodes. A widely-used approach to accelerate DAG-related algorithms is processing the nodes layer by layer [31, 50]. Inspired by these algorithms, we split the original subset selection problem (Eq. (2)) into a set of subproblems. Each selects the critical neurons in one layer: $N^i_{opt} = \arg\min (p(x) - p(x; N^i_s \cup N^{-i}))^2 + \lambda^i |N^i_s|^2$, where $N^i_s \subseteq N^i$, N^i is the set of the neurons in layer i, and N^{-i} is the set of all other neurons in the CNN except the ones in layer i. After solving all the subproblems, we aggregate all the sub-solutions N^i_{opt} into the final critical neuron set $N_{opt} = \bigcup\limits_i N^i_{opt}$.

Although dividing the original problem into a set of subproblems can largely reduce the search space, the search space of each subproblem is still large because a layer may contain more than one million neurons. Thus, we group neurons into a set of feature maps to further reduce the search space. In a CNN, neurons in a feature map share the same weights, and thus learn to detect the same feature. Utilizing this characteristics, we formulate the feature map selection problem as

$$F^i_{opt} = \underset{F^i_s \subseteq F^i}{\arg\min} (p(x) - p(x; F^i_s \cup F^{-i}))^2 + \lambda^i |F^i_s|^2, \tag{3}$$

where F is the set of feature maps in a CNN, F^i is the set of the feature maps in layer i, F^i_s is a subset of F^i, and F^{-i} is the set of all other feature maps in the CNN except the ones in layer i.

4.2.2 Quadratic Approximation

Although we have reduced the search space from millions of dimensions (neurons in a layer) to thousands of dimensions (feature maps in a layer), it is still time-consuming to solve Eq. (3) because it is an NP-complete discrete optimization problem. To tackle this issue, we transform the discrete optimization into a continuous optimization problem. In particular, we reformulate Eq. (3) as: $\mathbf{z}^i_{opt} = \arg\min (p(x) - p(x; \mathbf{z}^i))^2 + \lambda^i (\sum_j z^i_j)^2$, where $\mathbf{z}^i = [z^i_1, ..., z^i_n]$ and $z^i_j \in \{0, 1\}$ is an indicator to represent whether the j-th feature map in layer i is critical. If the feature map is critical, $z^i_j = 1$, otherwise, $z^i_j = 0$. Inspired by spectral clustering [36], we approximate the discrete optimization with a continuous optimization by removing the discreteness condition $z^i_j \in \{0, 1\}$ and allowing z^i_j to take a value in $[0, 1]$:

$$\mathbf{z}^i_{opt} = \underset{z^i_j \in [0,1]}{\arg\min} (p(x) - p(x; \mathbf{z}^i))^2 + \lambda^i (\sum_j z^i_j)^2, \tag{4}$$

Eq. (4) can be solved using gradient-based numerical optimization approaches such as the BFGS (Broyden−Fletcher−Goldfarb−Shanno) algorithm [58]. The gradient $\partial p / \partial z^i_j$ is calculated by back-propagation [8]. However, this method is computationally expensive because the gradient-based optimization is an iterative process where we have to calculate the gradients $\partial p / \partial z^i_j$ at each iteration. According to the calculation process of back-propagation, the deeper a CNN is, the longer it takes to compute the gradients.

To tackle this issue, we approximate Eq. (4) as a quadratic optimization where we calculate the gradients only once. Since $p(x) = p(\mathbf{a}_1, ..., \mathbf{a}_n)$ and $p(x; \mathbf{z}^i) = p(z^i_j \mathbf{a}_1, ..., z^n_j \mathbf{a}_n)$, we rewrite Eq. (4) as a multivariate function. Here $\mathbf{a}_j$ is the activation vector of the j-th feature map produced by example x. We then the Taylor decomposition [45] is to decompose $p(x) - p(x; \mathbf{z}^i)$ into a linear form: $p(x) - p(x; \mathbf{z}^i) \approx \sum\limits_j (1 - z^i_j) \mathbf{a}_j \cdot \left. \frac{\partial p}{\partial \mathbf{a}^i_j} \right|_{\mathbf{a}}$, where $\mathbf{a} = [\mathbf{a}^i_1, ..., \mathbf{a}^i_n]$ and $\mathbf{x} \cdot \mathbf{y}$ represents dot product between vectors $\mathbf{x}$ and $\mathbf{y}$. By substituting the above decomposition into Eq. (4), we obtain a quadratic optimization:

$$\mathbf{z}^i_{opt} = \underset{z^i_j \in [0,1]}{\arg\min} \mathbf{z}^i (Q^i + \lambda^i I)(\mathbf{z}^i)^T - 2q\mathbf{q}^i \cdot \mathbf{z}^i, \tag{5}$$

where $\mathbf{q}^i = [\mathbf{a}^i_1 \cdot \left. \frac{\partial p}{\partial \mathbf{a}^i_1} \right|_{\mathbf{a}}, \cdots, \mathbf{a}^i_n \cdot \left. \frac{\partial p}{\partial \mathbf{a}^i_n} \right|_{\mathbf{a}}]$, q is the sum of $\mathbf{q}^i$, $Q = (\mathbf{q}^i)^T \mathbf{q}^i$ is a $n \times n$ matrix, and I is a $n \times n$ identity matrix. Solving Eq. (5) only needs to evaluate the gradients once. We use the BFGS algorithm to solve Eq. (5). To control the scale of the two terms, the parameter λ^i is set to $0.1/|F^i|^2$, where $|F^i|$ is the number of the feature maps in layer i.

In the same way, we obtain a solution for selecting critical feature maps for a set of examples (Eq. (2)). Compared with Eq. (5), there are two differences: (1) replacing Q_i by $\sum Q^i_k$ and (2) replacing $q\mathbf{q}^i$ by $\sum q_k \mathbf{q}^i_k$, where k is the index of the example in the example set.

5 Datapath Visualization

An extracted datapath usually contains millions of neurons and even more connections, which prohibits experts from efficiently examining the datapath layer by layer. To help experts effectively explore the extracted datapaths, we design a multi-level datapath visualization, which enables experts to analyze and compare datapaths from high-level layers to detailed neuron activations (**R2**, **R3**, **R4**). Based on our discussions with the experts, we visualize datapaths on three levels: the layer level, the feature map level, and the neuron level. At each level, we (1) calculate the layout of the items (layers, feature maps, neurons) to reveal the relationships among them; and (2) provide visual hints to guide experts in finding the item(s) of interest and zooming in to a more-detailed level (e.g., from the layer level to the feature map level).

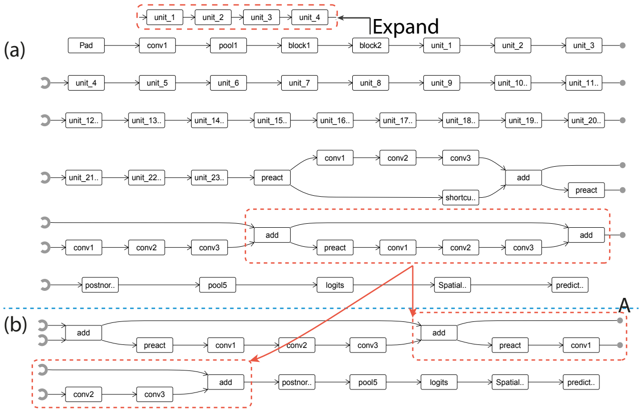

Figure 4: Segmented DAG visualization: (a) showing 49 layers of ResNet101 simultaneously; (b) the segmentation result calculated by only considering the empty space at the end of each segment.

5.1 Layer-level Visualization

The layer-level visualization provides an overview of the extracted datapath, and guides experts in selecting a layer to examine (**R2**).

Layout. At the layer level, we focus on illustrating how layers are connected. Initially, we employ the widely-used TensorFlow Graph Visualizer [57]. In particular, we formulate a CNN as a DAG (directed acyclic graph), where each layer is a node and the connections between layers are edges. Based on this formulation, a node-link diagram with a vertical layout is employed to visualize the layers and their connections. Then the DAG layout algorithm in [50] is employed to calculate the position of each layer. To handle large CNNs, a hierarchy of the layers is built, where each leaf node is a layer and each non-leaf node is a layer group [57].

The experts were overall satisfied with this design, but after they tried the prototype they commented that they often need to zoom in to the lower levels of the layer hierarchy in order to analyze the root cause of a misclassification. It is tedious for them to zoom in for each adversarial example. In addition, when they zoom in to the lower levels, the visualization often suffers from a long skinny strip with a very high aspect ratio (e.g., Fig. 8 (a) in [57]). This phenomenon is worse when they analyze state-of-the-art CNNs, such as ResNet [21] with 50-200 layers.

To solve the above problems, we combine a treecut algorithm and a segmented DAG visualization to save experts' efforts and generate a layout with a better aspect ratio, respectively.

Treecut. To save experts' efforts of zooming in, we use a treecut algorithm [13] to select an initial set of layers (around 50 layers) from the layer hierarchy. In this algorithm, the DOI measures the datapath difference between two sets of images (e.g., adversarial examples and normal examples).

Segmentation. Inspired by the segmented timeline [10], we propose transposing the vertical layout into a horizontal layout, segmenting the initial DAG into several parts, and visualizing the segmented parts from top to bottom. The experts commented that this horizontal design resembles a calendar and thus is familiar to them. As shown in Fig. 4 (a), our segmented DAG visualization effectively illustrates the connections among dozens of layers with good aspect ratios.

The key challenge of segmenting a DAG is to decide where to segment it. We formulate the segmentation problem as a "printing neatly" problem [12], in which we aim to minimize the cost function that sums up the empty space at the end of each line while ensuring that no word is off screen. This optimization method is suitable for a CNN with a chain structure, such as VGG net [47]. However, state-of-the-art CNNs (e.g., ResNet [21], and DenseNet [22]) often contain basic building blocks, whose layers can split and merge (e.g., Fig. 4A). These building blocks are often connected with others as a chain. Directly minimizing the above cost function may cut the basic building blocks apart, which hinders the understanding of the network structure (Fig. 4 (b)). To solve this problem, we add a regularization term to the cost function to penalize a segmentation scheme that cuts a building block into two parts:

$$\min_{e_i \geq 0} \sum_{i=1}^{k-1} e_i + \lambda c_i, \tag{6}$$

where k is the number of segments, e_i is the empty space at the end of segment i, c_i represents whether a building block is cut by segment i, and λ is used to balance the two terms. In AEVis, experts can interactively modify λ. Eq. (6) can be efficiently solved using dynamic programming. As shown in Fig. 4, if we only consider the empty-space objective, building block A will be cut into two parts (Fig. 4 (b)). Adding the regularization term can avoid unnecessary cuts by balancing between the small empty space and the protection of the building blocks (Fig. 4 (a)).

Visual hint. Showing how the layers are connected is not enough in guiding experts to find the layer of interest. Thus, for each node (a layer or layer group) in the segmented DAG, we provide a visual statistical description of the datapath(s) in that layer (group). Because there is limited space for each node and a layer group usually contains dozens of layers, we employ a dot plot as the visual hint, which is a compact visualization and widely used for relatively small datasets [56]. In particular, in a dot plot, each dot represents a high-level statistics of a layer (Fig. 5). The position of a dot on the x-axis denotes the value of the high-level statistics of the datapath(s) in that layer, such as the activation similarity between two datapaths (the datapath for adversarial examples and the one for normal examples). In particular, the activation similarity is calculated as $\sum sim(I_A^i, I_N^i)/N$, where $sim(.,.)$ is the widely-used cosine similarity, I_A^i, I_N^i is a pair of adversarial and normal examples, and N is the number of such pairs in the examples. Other high-level statistics include the averaged activations of an example set on the corresponding datapath in that layer and the topological similarity between two datapaths (measured by Jaccard similarity). Examining the dot plots node by node helps experts detect the "diverging point," where a normal image and its corresponding adversarial example diverge into different predictions (Fig. 10). Experts are able to examine the feature maps in the layer group of interest (Fig. 1) or expand it to examine child layers or layer groups (Fig. 4 (a)).

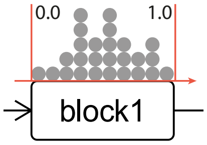

Figure 5: A dot plot as the visual hint for layer selection.

5.2 Feature-map-level Visualization

When an expert detects a layer of interest, he/she then zooms in to the layer to examine the critical feature maps in that layer. To preserve the analysis context, we visualize the feature maps in the selected layer as the focus and other layers are shown as context. (Fig. 1 (b)).

Layout. The belonging relationships between the feature maps and the extracted datapaths in one layer, including the unique feature maps of each datapath and the shared ones between/among them, are very useful for understanding the roles of these feature maps. As finding unique/share elements based on their set (datapath) membership is an important task tackled by the set visualization [4], we then formulate the feature map layout problem as a set visualization problem. Among various set visualization techniques, such as the Euler diagram, the line-based techniques (e.g., LineSets [3]), and the matrix-based techniques (e.g., ConSet [25]), we decide to employ on the Compact Rectangular Euler Diagram [42] (ComED) because:

- It can well depict set relations and thus disclose the unique/share feature maps [4];

- Machine learning experts are familiar with Euler diagrams and they often use them to understand the set relationships;
- The number of feature maps in each datapath is clearly conveyed in the Euler diagram, which is important for the analysis.

As in ComED [42], we split the datapaths in the layer of interest by their intersection relations. This produces a hierarchy, which is visualized by non-overlapping rectangles (Fig. $1L_C$). Then the split parts of datapaths are connected with lines, which can be shown on demand. Compared with ComED, we make two modifications to reduce visual clutter and better facilitate experts' visual comparison:

- Utilize K-Means [8] to cluster the feature maps by their activations to reduce the visual clutter caused by a large number of feature maps (Fig. 6). Experts can interactively modify the parameters of the clustering (e.g., the number of the clusters k) to reduce the parameter sensitivity.
- Use the Treemap layout instead of the graph layout because the graph layout result has an irregular boundary, which is not suitable for juxtaposing multiple layers to compare the datpaths in them (Fig. $1L_C$ and L_D).

Visual hint. Although the Euler-diagram-based design reveals the shared/unique feature maps very well, it does not significantly help to efficiently select an individual feature map cluster of interest for further examination. To this end, for each cluster, we illustrate its average importance (z^i_j in Eq. (4), Sec. 4) or average activation, because experts usually start their analysis from the most critical or most highly activated feature maps. The averaged importance/activation of a feature map cluster is encoded by the color of the feature map cluster (Fig. 6). In addition to importance/activation, we also encode the size of a feature map cluster because it is a basic information for a cluster. We employ a series of stacked rectangles to represent a feature map cluster (Fig. 6), and use the number of stacked rectangles to encode the size of a cluster. To avoid the visual clutter caused by a large number of stacked rectangles, the number of rectangles N_R is proportional to the log of the cluster size N_C: $N_R = log_{10}(N_C) + 1$. For the sake of consistency, a cluster with only one feature map in it is shown as a single rectangle.

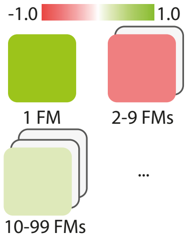

Figure 6: The visual hints for a feature map cluster. FM is short for feature map.

5.3 Neuron-level Visualization

When an expert finds a feature map of interest, AEVis allows him/her to examine the neurons in that feature map. As there are hundreds or even thousands of neurons in a feature map and dozens of images are often analyzed simultaneously, we cannot show all the activations in place due to the limited space. Thus, we add a neuron panel (Fig. 1 (c)) to show the neurons. To preserve the visual link [48] of the selected feature map and the corresponding neurons, we add the same label to the feature map and the neurons (Fig. 7A).

Layout. In a feature map, the neurons are naturally organized on a grid. The position of a neuron is determined by the position of the image patch that influences the activation of this neuron [17].

Visual hint. Following previous research [7, 29], we employ the learned features of neurons and the activations of the neurons to help expert understand the roles of the neurons for the prediction.

The activation of a neuron is represented by its color (red: negative activation; green: positive activation). Combining the color coding and the grid-based layout of neurons, experts can detect which part of the image highly activates the neurons in the feature map. For example, in Fig. 7, we can find the neurons on the top right corner are highly activated, which corresponds to the panda head in the image.

To visualize the learned features of neurons, we first try the method used in previous papers on explainable deep learning [7, 29]. This method selects the image patches that highly activate a neuron to represent its learned feature. However, we discovered that when handling very deep CNNs (e.g., ResNet101 [21]), this method cannot illustrate the exact region in each image patch that highly activates a feature map neuron, especially for the neurons in top layers. This is because the activations of neurons in top layers of a very deep CNN are influenced by a large image patch. For example, the neurons in more than at layer 10 and deeper are influenced by all the pixels in an image in ResNet101. Thus, we employ the discrepancy map [64] to highlight which region in the image actually highly activates a neuron and treat this region as the learned feature of this neuron. To calculate the discrepancy map for a neuron, Zhou et al. [64] occluded a very small patch of an image (8 pixels× 8 pixels), and calculated the new activation of the neuron produced by the occluded image. If the activation changes a great deal, the small patch is marked as important. This process iterates many times, and for each iteration, a different patch is checked. After this process, all the important small patches of the image remain unchanged and other pixels are deemphasized by lowering their lightness. For example, in Fig. 7, by examining the discrepancy maps, we found that the neurons in the selected feature map learned to detect a panda head. Without detecting the important region, we may draw the wrong conclusion that the neurons learned to detect a whole panda.

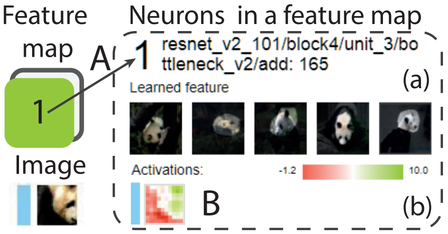

Figure 7: Neuron-level visualization: (a) the learned feature is shown as discrepancy maps; (b) activations are shown as a heat map.

6 Evaluation

As part of our evaluation, we first performed a qualitative evaluation to demonstrate the effectiveness of the datapath extraction algorithm. Then, two case studies were conducted to illustrate how AEVis helps the experts E_1 and E_2 analyze both white-box and black-box adversarial examples.

6.1 Qualitative Analysis of Datapath Extraction

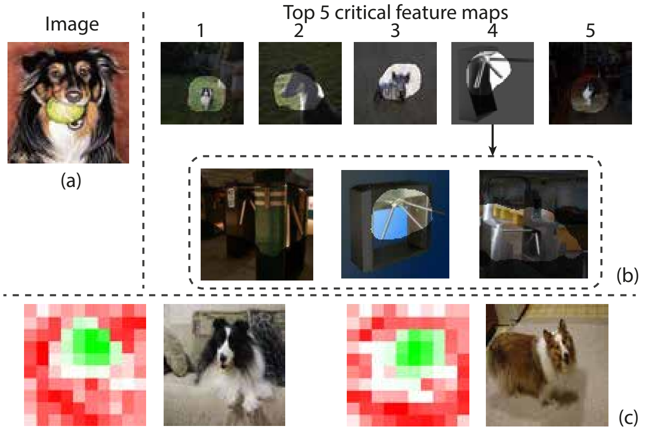

Figure 8: Critical feature maps extracted by our approach for a sheepdog image: (a) the input image; (b) top 5 the critical feature maps; (c) activation heat maps of the neurons in feature map 4 on two examples.

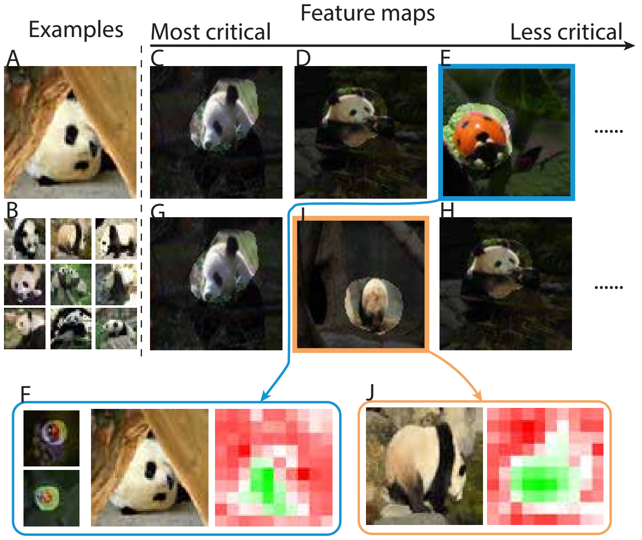

Figure 9: Comparison between the extracted datapath for one example and a set of examples.

Datapath extraction for a single example. Fig. 8 shows the top 5 critical feature maps extracted by our approach for the same image in Fig. 3. Feature maps 1,2,3 and 5 (Fig. 8 (b)) learn to detect the features of dog ears, head, etc. These features are indeed important to classify a Shetland sheepdog. Compared with the activation-based approach, our approach is able to ignore the irrelevant feature map that learns to detect a ball (Fig.3 (b)). Besides these easy-to-understand feature maps, the learned feature of feature map 4 (Fig 8 (b)) is difficult to understand at first glance. To understand what this feature map actually learns, we calculate a set of discrepancy maps (Fig 8 (b)). A common property of these discrepancy maps is that they have white stripes with black strokes. Thus, we conclude that this feature map learns to detect such a feature. However, it is still unclear why this feature map is considered critical in classifying a sheepdog. To answer this question, we loaded more sheepdog images. We found that a distinctive feature of a sheepdog was a white strip between its two dark-colored eyes. This feature was ignored by us at first because it is not obvious in the original example (Fig. 8 (a)). To verify our assumption, we further visualized the activations on the feature map (Fig. 8 (c)) and found that this feature map was indeed highly activated by the white strip in the images of sheepdogs. Our experts were impressed with this finding, because the datapath extraction algorithm not only finds a commonsensical feature map, but also finds unexpected feature map(s).

Comparison between the extracted datapaths for one example and a set of examples. As shown in Fig. 9, we compared the extracted datapath for an image with only one panda face (Fig. 9A) with the one for a set of images that contain both panda faces and whole pandas (Fig. 9B).

We found that the most critical feature maps extracted for an image with only one panda face contain the feature maps that detect a panda face (Fig. 9C and D) and important features on a panda face, such as eyes and ears (Fig. 9E). The neuron in the blue rectangle (Fig. 9E) was unexpected at first glance. After examining more discrepancy maps for that feature map and the activation heat map (Fig. 9F), we discovered that the feature map learned to detect black dots. Such a feature is a recognizable attribute of a panda face.

Just like the datapath of a panda's face, the datapath extracted for a set of panda images includes the feature maps for detecting a panda face (Fig. 9G and H). In addition to these feature maps, the most critical feature maps include a feature map for detecting the black-and-white body of a panda (Fig. 9I), which is also an important feature for classifying a panda. We visualized the activations on this feature map to verify that the feature map indeed learns to detect a panda body (Fig. 9J).

This finding echoes requirement **R1** in Sec. 3.2, i.e., that the extracted critical feature maps for one example may not be representative for a given set of images with the same class label.

6.2 Case Study

There are two main types of adversarial attacks: white-box attack and a black-box attack [1]. The white-box attack means that the attacker has a full knowledge of the target model, including the parameters, architecture, training method, and even the training data. While the Black-box attack assumes that the attacker knows nothing about the target model, which can be used to evaluate the transferability of adversarial examples. Next, we demonstrate how AEVis helps experts E_1 and E_2 analyze both white-box and black-box adversarial examples. In both case studies, we utilized the dataset that is from the NIPS 2017 non-targeted adversarial attack and targeted adversarial attack competitions [51], because the experts are familiar with this dataset. It contains 1,000 images (299 pixel × 299 pixel) We employed the white-box non-targeted attacking method developed by the winning machine learning group [14, 15] to generate one adversarial example for each image in the dataset.

6.2.1 Analyzing White-Box Adversarial Examples

Model. One of the state-of-the-art CNNs for image classification, ResNet101 [21], is employed as the target model. It contains 101 layers. We used the pre-trained model from TensorFlow [19]. It achieves a high accuracy (96.3%) on the normal examples, and a very low accuracy (0.9%) on the generated adversarial examples.

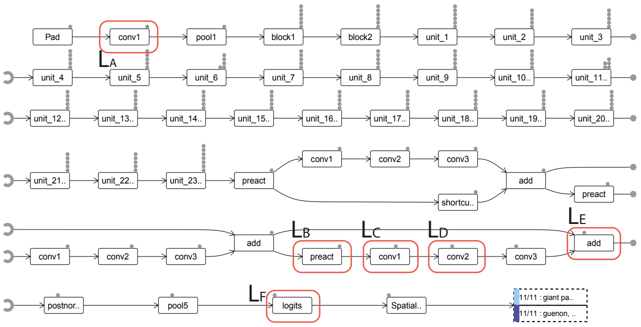

Figure 10: The datapath overview illustrating the activation differences between the datapaths of the adversarial examples and normal examples.

Analysis process. To start the analysis, we calculated an adversarial score for each image [39]. A high score means the image is most probably to be an adversarial example. The expert E_1 then focused on the most uncertain images with medium adversarial scores. After examining these uncertain adversarial examples, E_1 selected one for further investigation (I_B in Fig. 1). It contains a panda head but is misclassified as a guenon monkey. To understand the root cause of this misclassification, he wanted to compare its datapath with the one of the corresponding normal example (I_A in Fig. 1). To improve the representativeness of the extracted datapath for the normal example (Sec. 6.1, Fig. 9), he added 10 more normal panda images as well as the corresponding adversarial examples, to the images of interest (**R1**). Each added adversarial example is misclassified as a guenon monkey. Accordingly, E_1 split these images into two sets. The first set (adversarial group) contained 11 adversarial examples. The second set of images (normal group) contained 11 normal examples. Then he extracted and visualized the datapaths for these two sets of images.

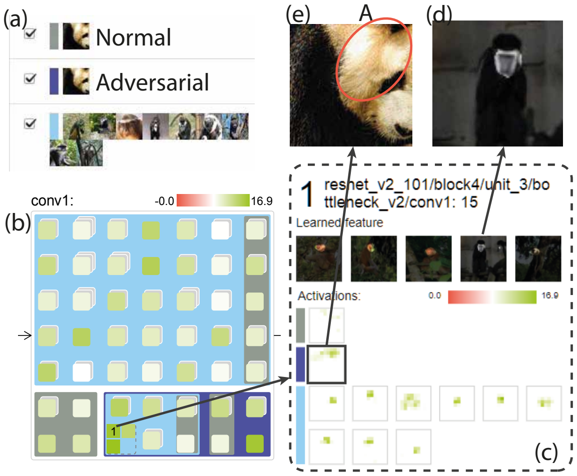

Figure 11: Explanation of why the adversarial panda image is misclassified as a monkey. The CNN erroneously detects a monkey's face: (a) input images; (b) feature maps; (c) neurons; (d) learned feature; and (e) adversarial example.

The datapath overview is shown in Fig. 10. The dot plots encoded the activation similarity between these two datapaths. The expert E$_1$ found that at the bottom layer of the network (Fig. 10L_A), the activation similarity is almost 1.0 (the dot is on the right). This similarity remained to be 1.0 until layer L_B in Fig. 10. After layer L_B, layer L_C appeared as a "diverging point", where the activation similarity largely decreased (**R4**). In the following layers, the activation similarity continued decreasing to 0 (L_D, L_E, L_F), which resulted in the misclassification.

To analyze which feature map is critical for the divergence, the expert E$_1$ expanded the diverging point L_C (Fig. 1). In addition, he set the color coding of each feature map as the activation difference between two sets of examples (activation difference = activations of normal images − activations of adversarial examples). A large activation difference indicates that the corresponding feature map detects its learned feature in the normal images but did not detect such a feature in the adversarial examples. Because the feature map F_C in Fig. 1 shows the largest activation difference, he then checked its neurons. By examining the learned feature of the neurons (Fig. 1A), he discovered that the neurons learned to detect a black patch that resembles a panda's ear (Fig. 1B and C). Such a feature is critical for detecting a panda's face, which does not appear in the adversarial example. To further investigate the influence of this feature map, E$_1$ continued to expand the next layer (layer L_D). He found that there was a large activation difference on the feature map for detecting a panda's face (F_D in Fig. 1). This indicates that the corresponding feature map, F_D, fails to detect a panda's face, which is a direct influence of the large difference on F_C. By the same analysis, E$_1$ found that in the layer that detected the highest-level features (L_E), there was also a large activation difference on the feature map for detecting a panda's face. As the target CNN cannot detect a panda's face in highest-level features, the CNN failed to classify the adversarial example as a panda image.

The above analysis explains why the CNN failed to classify the adversarial example as a panda, but cannot explain why the adversarial example is classified as a monkey. Thus, E$_1$ compared the adversarial example with the corresponding example and a set of normal monkey images (Fig. 11 (a)). Inspired by the above analysis, E$_1$ directly expanded layer L_C (diverging point), and examined its feature maps (Fig. 11 (b)). After checking the activations of the adversarial example on the "monkey's datapath", the expert found that the activations were the largest on the feature map for detecting the face of a monkey (Fig. 11 (c) and (d)). This behavior of the CNN was unexpected because there was no monkey face in the adversarial example at first glance. Thus, E$_1$ examined the detailed neuron activations on this feature map and discovered that the high activations appeared in the top right corner of the neurons, which corresponded to the top right corner of the image (Fig. 11 (e)). E$_1$ did not understand why the CNN detected a monkey face at that part of the image. By carefully examining the adversarial example and the learned features of the neurons, he finally figured out the reason: there is a dark strip with a bright strip on each side (Fig. 11A). It is a recognizable attribute for the face of a guenon monkey, which made the CNN mispredict the adversarial example as a monkey.

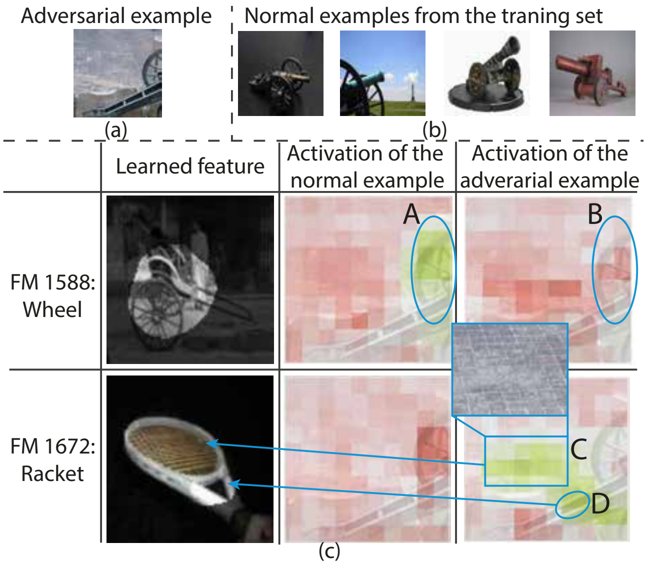
Adversarial example
Normal examples from the traning set
(a)
(b)
Learned feature
Activation of the normal example
Activation of the adverarial example
FM 1588: Wheel
FM 1672: Racket
A
B
C
D
(c)

Figure 12: Explanation of why the adversarial cannon image is misclassified as a racket: (a) the adversarial example image; (b) normal examples from the training set; (c) the learned features and activations of neurons in the layer L$_E$ in Fig. 10.

Another case is shown in Fig. 12, where a cannon image is misclassified as a racket (probability: 0.997). The reasons behind the misclassification are two-fold. First, the CNN recognizes a large wheel in the normal image (Fig. 12A) but cannot recognize a large wheel in the adversarial image (Fig. 12B), while a large wheel is a recognizable attribute of a cannon (Fig. 12 (b)). Second, E$_1$ found that the CNN recognizes the head of a racket (Fig. 12C), which is connected to the throat of a racket (Fig. 12D). These two reasons lead to the misclassification.

E$_1$ commented, "There may be different subtle causes that lead to the misclassification of different adversarial examples. The value of AEVis is that it helps me find such causes quickly and effectively. I can easily integrate my knowledge into the analysis process by leveraging the provided interactive visualizations. I also see a great opportunity to use AEVis in analyzing more adversarial examples and summarize the major causes of the misclassification. This probably benefits future research on robust deep learning."

6.2.2 Analyzing Black-box Adversarial Examples

Model. We used the VGG-16 [47] network to analyze black-box attacks because our employed attacking method has the knowledge of a set of state-of-the-art CNNs, such as ResNet [21] and the Inception network [52]. Thus, we adopted a traditional CNN (VGG) to analyze black-box attacks. We utilized the pre-trained VGG-16 network from TensorFlow [19]. It achieved a 84.8% accuracy on the normal examples, and a 22.4% accuracy on the adversarial examples.

Analysis process. The expert E$_2$ continued to explore the analyzed adversarial example, and tested it on the VGG network. He found that it was misclassified as a beagle (a type of dog, Fig. 13 (a)). To check how the prediction was made, E_2 extracted and compared the datapaths for normal beagle images and the adversarial panda image. He found that the there was no obvious “merging point” in VGG-16, because these images do not truly “merge” due to the high prediction score of beagle images ($0.75-1.0$) and the relatively low prediction score of the adversarial example (0.46). Thus, E_2 relied on his knowledge to select the layer that detected the highest-level features (Fig. 13 (b)). By setting the color coding of feature maps as the activation of the adversarial example, E_2 found a large activation appeared on a feature map in shared feature maps between the beagle’s and adversarial panda’s datapath (Fig. 13A). It is a potential cause that leads to the misclassification. Thus, he examined the learned feature of the neurons in this feature map (Fig. 13 (c)), and found that they learned to detect the black nose of a beagle (a black patch, Fig. 13B). This feature is a recognizable attribute for a beagle. To understand why the adversarial example caused a large activation on this feature map, he further examined the activation heat map, and found that the neurons in the top right corner are highly activated (Fig. 13C). It indicates that there is such a feature in the corresponding part of the image. In particular, that part of the images is the black ear of the panda, which is also a black patch (Fig. 13C). This feature misled the VGG-16 network to detect a black nose in the adversarial panda image, which then led to the misclassification.

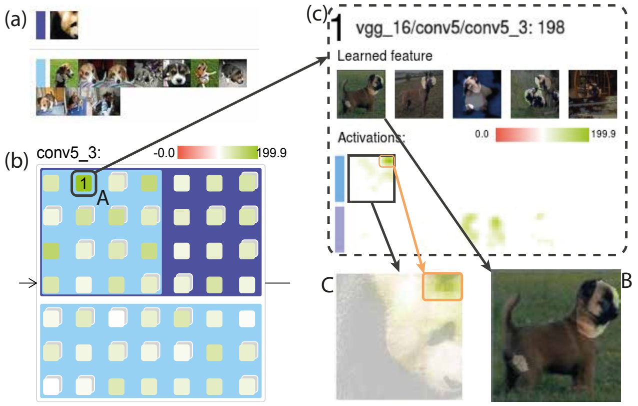

Figure 13: The explanation of why the adversarial panda image was misclassified as a beagle (dog) by the VGG-16 network: (a) input images; (b) feature maps; and (c) neurons.

7 Discussion

AEVis can better disclose the inner workings of adversarial examples and help discover the vulnerable neurons that lead to incorrect predictions. However, it also comes with several limitations, which may shed light on future research directions.

Scalability. We have demonstrated that AEVis is able to analyze a state-of-the-art CNN (ResNet101), which has 101 layers and is much deeper than traditional CNNs (e.g., VGG-Net). More recently, researchers have developed many deeper CNNs with thousands of layers [21]. When handling such deep DNNs, if an expert zooms in to low levels of the layer hierarchy, the layers of interest cannot fit in one screen, even with the help of our segmented DAG. To alleviate this issue, we can employ a mini-map to help the expert track the current viewpoint, which has been proven effective in TensorFlow Graph Visualizer [57]. The dot plot is another factor that hinders AEVis from analyzing CNNs with thousands of layers. This is because a layer group may contain hundreds of layers, and the height of the resulting dot plot will be too large, which is a waste of screen space. To solve this problem, we can use non-linear dot plots [43] to improve the scalability of AEVis.

Currently, we utilize a Euler-diagram-based design to illustrate the overlapping relationship between/among datapaths. Such a design is suitable for comparing several datapaths [4]. Although researchers found that approximately four objects can be tracked in a visual comparison [23, 55, 61], experts may have special needs of comparing a lot of datapaths. To fulfill these needs, we can leverage more scalable set visualization techniques, such as PowerSet [5].

Generalization. AEVis aims at analyzing the adversarial examples of CNNs because most research on adversarial attacks focuses on generating adversarial images for CNNs.

In addition to attacking CNNs, there are several initial attempts to attack other types of DNNs [1], such as multilayer perceptron (MLP), recurrent neural networks (RNNs), autoencoders (AEs), and deep generative models (DGMs). Among these models, AEVis can be directly used to analyze MLPs by treating each neuron as a feature map that contains one neuron. For other types of DNNs, we need to develop suitable datapath extraction and visualization methods. For example, Ming et al. [33] demonstrated that some neurons in an RNN were critical for predicting the sentiment of a sentence, such as the neurons for detecting positive/negative words. Such neurons and their connections form a datapath for an RNN. Thus, by extracting and visualizing datapaths, AEVis can be extended to analyze the root cause of adversarial examples for these types of DNNs. For example, to visualize the datapath of RNNs, we can first unfold the architecture of an RNN to a DAG [17], and then employ a DAG layout algorithm to calculate the position of each unfolded layer.

In addition to images, researchers try to generate adversarial examples for other types of data [1], such as adversarial documents and adversarial videos. To generalize AEVis to other types of data, we need to change the visual hint for neurons (discrepancy map and activation heat map) according to the target data type. For example, when analyzing adversarial documents, we can use a word cloud to represent the “learned feature” of a neuron in an RNN [33]. In the word cloud, we select the keywords that activate the neuron.

8 Conclusion

We have presented a robustness-motivated visual analytics approach that helps experts understand the inner workings of adversarial examples and diagnoses the root cause of incorrect predictions introduced by the adversarial examples. The major feature of this approach is that it centers on the concept of datapaths to tightly combine datapath extraction and visualization. Two case studies were conducted with two machine learning experts to demonstrate the effectiveness and usefulness of our approach in analyzing both white-box and black-box adversarial examples.

One interesting avenue for future research is to monitor the online testing process, detect potentially adversarial examples, and remove them from any further processing. The key is to design a set of streaming visualizations that can incrementally integrate the incoming log data with existing data. We would also like to continue working with the machine learning experts to conduct several field experiments, with the aim of designing more targeted and effective defense solutions based on the discovered root cause. Another important direction is to analyze the robustness of other types of DNNs, such as RNNs and DGMs. For these types of DNNs, exciting research topics include more efficient datapath extraction algorithms and suitable visualizations for different types of DNNs.

Acknowledgments

This research was funded by National Key R&D Program of China (No. SQ2018YFB100002), the National Natural Science Foundation of China (No.s 61761136020, 61672308, 61620106010, 61621136008, 61332007), Beijing NSF Project (No. L172037), Microsoft Research Asia, Tiangong Institute for Intelligent Computing, NVIDIA NVAIL Program, and Tsinghua-Intel Joint Research Institute. The authors would like to thank Yinpeng Dong for insightful discussions in the case studies and Jie Lu for help in the development of AEVis.

References

[1] N. Akhtar and A. Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *arXiv preprint arXiv:1801.00553*, 2018.

[2] E. Alexander and M. Gleicher. Task-driven comparison of topic models. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):320–329, 2016.

[3] B. Alper, N. Riche, G. Ramos, and M. Czerwinski. Design study of linesets, a novel set visualization technique. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2259–2267, 2011.

[4] B. Alsallakh, L. Micallef, W. Aigner, H. Hauser, S. Miksch, and P. Rodgers. Visualizing sets and set-typed data: State-of-the-art and future challenges. In *Eurographics Conference on Visualization*, pages 1–21, 2014.

[5] B. Alsallakh and L. Ren. Powerset: A comprehensive visualization of set intersections. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):361–370, 2017.

[6] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[7] A. Bilal, A. Jourabloo, M. Ye, X. Liu, and L. Ren. Do convolutional neural networks learn class hierarchy? *IEEE Transactions on Visualization and Computer Graphics*, 24(1):152–162, 2018.

[8] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[9] M. Brehmer, S. Ingram, J. Stray, and T. Munzner. Overview: The design, adoption, and analysis of a visual document mining tool for investigative journalists. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2271–2280, 2014.

[10] M. Brehmer, B. Lee, B. Bach, N. H. Riche, and T. Munzner. Timelines revisited: A design space and considerations for expressive storytelling. *IEEE Transactions on Visualization and Computer Graphics*, 23(9):2151–2164, 2017.

[11] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier. Parseval networks: Improving robustness to adversarial examples. In *International Conference on Machine Learning*, pages 854–863, 2017.

[12] T. H. Cormen. *Introduction to algorithms*. MIT press, 2009.

[13] W. Cui, S. Liu, Z. Wu, and H. Wei. How hierarchical topics evolve in large text corpora. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2281–2290, 2014.

[14] Y. Dong. Non targeted adversarial attacks. `https://github.com/dongyp13/Non-Targeted-Adversarial-Attacks`, 2017.

[15] Y. Dong, F. Liao, T. Pang, H. Su, X. Hu, J. Li, and J. Zhu. Boosting adversarial attacks with momentum, arxiv preprint. *arXiv preprint arXiv:1710.06081*, 2017.

[16] M. Gleicher. Considerations for visualizing comparison. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):413–423, 2018.

[17] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[18] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[19] Google. Tensorflow. `https://www.tensorflow.org`, 2017.

[20] A. W. Harley. An interactive node-link visualization of convolutional neural networks. In *ISVC*, pages 867–877, 2015.

[21] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[22] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, page 3, 2017.

[23] J. Intriligator and P. Cavanagh. The spatial resolution of visual attention. *Cognitive psychology*, 43(3):171–216, 2001.

[24] M. Kahng, P. Y. Andrews, A. Kalro, and D. H. P. Chau. ActiVis: Visual exploration of industry-scale deep neural network models. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):88–97, 2018.

[25] B. Kim, B. Lee, and J. Seo. Visualizing set concordance with permutation matrices and fan diagrams. *Interacting with Computers*, 19(5-6):630–643, 2007.

[26] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[27] D. Liu, W. Cui, K. Jin, Y. Guo, and H. Qu. Deeptracker: Visualizing the training process of convolutional neural networks. *To appear in ACM Transactions on Intelligent Systems and Technology*.

[28] 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*, 24(1):77–87, 2018.

[29] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu. Towards better analysis of deep convolutional neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):91–100, 2017.

[30] S. Liu, W. Cui, Y. Wu, and M. Liu. A survey on information visualization: recent advances and challenges. *The Visual Computer*, 30(12):1373–1393, 2014.

[31] S. Liu, Y. Wu, E. Wei, M. Liu, and Y. Liu. Storyflow: Tracking the evolution of stories. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2436–2445, 2013.

[32] D. Lloyd and J. Dykes. Human-centered approaches in geovisualization design: Investigating multiple methods through a long-term case study. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2498–2507, 2011.

[33] Y. Ming, S. Cao, R. Zhang, Z. Li, Y. Chen, Y. Song, and H. Qu. Understanding hidden memories of recurrent neural networks. In *IEEE VAST*, 2017.

[34] S. M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 86–94, 2017.

[35] S. M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. DeepFool: A simple and accurate method to fool deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.

[36] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, pages 849–856, 2002.

[37] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015.

[38] C. Olah, A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev. The building blocks of interpretability. *Distill*, 3(3):e10, 2018.

[39] T. Pang, C. Du, Y. Dong, and J. Zhu. Towards robust detection of adversarial examples. *arXiv preprint arXiv:1706.00633*, 2017.

[40] N. Pezzotti, T. Höllt, J. Van Gemert, B. P. Lelieveldt, E. Eisemann, and A. Vilanova. DeepEyes: Progressive visual analytics for designing deep neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):98–108, 2018.

[41] P. E. Rauber, S. G. Fadel, A. X. Falcao, and A. C. Telea. Visualizing the hidden activity of artificial neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):101–110, 2017.

[42] N. H. Riche and T. Dwyer. Untangling euler diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1090–1099, 2010.

[43] N. Rodrigues and D. Weiskopf. Nonlinear dot plots. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):616–625, 2018.

[44] A. S. Ross and F. Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. *arXiv preprint arXiv:1711.09404*, 2017.

[45] W. Rudin et al. *Principles of Mathematical Analysis*, volume 3. McGraw-hill New York, 1964.

[46] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

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

[48] M. Steinberger, M. Waldner, M. Streit, A. Lex, and D. Schmalstieg. Context-preserving visual links. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2249–2258, 2011.

[49] 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*, 24(1):667–676, 2018.

[50] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125, 1981.

[51] N. I. P. Systems. Nips 2017: Adversarial attack. `https://nips.cc/Conferences/2017/CompetitionTrack`, 2017.

[52] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[53] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[54] F. Y. Tzeng and K. L. Ma. Opening the black box - data driven visualization of neural networks. In *IEEE VIS*, pages 383–390, 2005.

[55] X. Wang, S. Liu, J. Liu, J. Chen, J. Zhu, and B. Guo. Topicpanorama: A full picture of relevant topics. *IEEE Transactions on Visualization and Computer Graphics*, 22(12):2508–2521, 2016.

[56] L. Wilkinson. Dot plots. *The American Statistician*, 53(3):276–281, 1999.

[57] K. Wongsuphasawat, D. Smilkov, J. Wexler, J. Wilson, D. Mané, D. Fritz, D. Krishnan, F. B. Viégas, and M. Wattenberg. Visualizing dataflow graphs of deep learning models in tensorflow. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):1–12, 2018.

[58] S. Wright and J. Nocedal. Numerical optimization. *Springer Science*, 35(67-68):7, 1999.

[59] Y. Wu, X. Xie, J. Wang, D. Deng, H. Liang, S. C. Hui Zhang, and W. Chen. Forvizor: Visualizing spatio-temporal team formations in soccer. *IEEE Transactions on Visualization and Computer Graphics*, 2018.

[60] X. Xie, X. Cai, J. Zhou, N. Cao, and Y. Wu. A semantic-based method for visualizing large image collections. *IEEE Transactions on Visualization and Computer Graphics*, 2018.

[61] S. Yantis. Multielement visual tracking: Attention and perceptual organization. *Cognitive psychology*, 24(3):295–340, 1992.

[62] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833, 2014.

[63] S. Zheng, Y. Song, T. Leung, and I. Goodfellow. Improving the robustness of deep neural networks via stability training. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4480–4488, 2016.

[64] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene CNNs. In *International Conference on Learning Representations*, 2015.