DOI: 10.1111/cgf.14418
Pacific Graphics 2021
E. Eisemann, K. Singh, and F.-L Zhang
(Guest Editors)

Volume 40 (2021), Number 7

Interactive Analysis of CNN Robustness

Stefan Sietzen[1], Mathias Lechner[2], Judy Borowski[3], Ramin Hasani[1,4], and Manuela Waldner[1]

[1]TU Wien, Vienna, Austria
[2] Institute of Science and Technology Austria (IST Austria), Klosterneuburg, Austria
[3] University of Tübingen, Germany
[4] Massachusetts Institute of Technology (MIT), Cambridge, USA

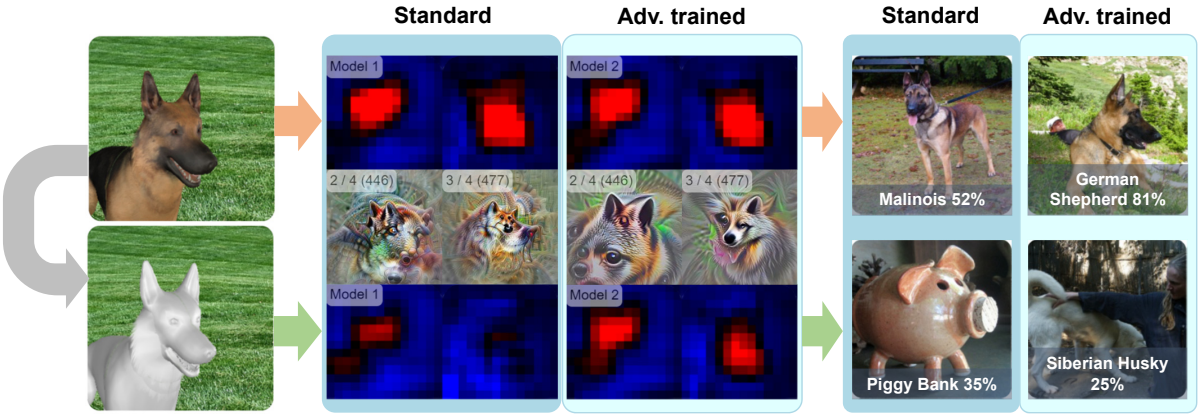


Figure 1: *Rapid interactive comparison of two models' network responses through activations of two selected "dog-relevant" neurons (middle) and predicted target classes (right) to input scene perturbations of a dog (left): Removing the texture leaves the 3D model with a smooth surface and causes a standard model to identify a piggy bank instead of a dog, while an adversarially trained model identifies a white dog breed (Siberian Husky). Activations of dog-relevant neurons in the standard model decrease due to missing texture information.*

Abstract

While convolutional neural networks (CNNs) have found wide adoption as state-of-the-art models for image-related tasks, their predictions are often highly sensitive to small input perturbations, which the human vision is robust against. This paper presents Perturber, a web-based application that allows users to instantaneously explore how CNN activations and predictions evolve when a 3D input scene is interactively perturbed. Perturber offers a large variety of scene modifications, such as camera controls, lighting and shading effects, background modifications, object morphing, as well as adversarial attacks, to facilitate the discovery of potential vulnerabilities. Fine-tuned model versions can be directly compared for qualitative evaluation of their robustness. Case studies with machine learning experts have shown that Perturber helps users to quickly generate hypotheses about model vulnerabilities and to qualitatively compare model behavior. Using quantitative analyses, we could replicate users' insights with other CNN architectures and input images, yielding new insights about the vulnerability of adversarially trained models.

CCS Concepts

• ***Human-centered computing*** → ***Visualization systems and tools;*** • ***Computing methodologies*** → ***Machine learning;***

1. Introduction

Convolutional neural networks (CNNs) achieve impressive results in a variety of applications, such as image classification [KSH17] or object detection [RHGS17]. The performance of a CNN trained on a given training data set is typically assessed in terms of prediction accuracy on a held-out validation dataset. If the statistical distributions of the training and validation set differ, the high performance can drop precipitously. In that case, the model is not robust

against the variability within the validation data. CNN robustness has been shown to be affected by image perturbations such as cropping [ZSLG16], blur [DK16], high-frequency noise [YLS*19], or texture variations [GRM*19]. Prominent examples of noise perturbation are adversarial attacks: Single pixels of an input image are changed slightly such that a CNN misclassifies it [SZS*14]. To humans, these changes are typically imperceptible, and they would still assign the correct labels.

Robustness is a highly safety-critical aspect of CNNs in various applications, such as self-driving cars [LHA*20]. For example, researchers have demonstrated how targeted yet unsuspicious changes of traffic signs can cause CNNs to consistently miss stop signs [EEF*18]. Understanding which factors influence the robustness of CNNs and, consequently, designing and evaluating more robust models are therefore research topics of central importance in the machine learning community [GSS14, SZS*14, MMS*19, LHG*21].

To create a basis for tackling robustness, researchers aim to gain a better general understanding of which image features CNNs are most sensitive to and how this sensitivity differs from human visual perception [BHL*21]. For example, Geirhos et al. [GRM*19] investigated the influence of shape versus texture on humans and CNNs in image classification and found a strong bias for texture in CNNs. This texture bias was also confirmed by Zhang and Zhu [ZZ19] by evaluating the effect of image saturation or random shuffling of image patches. Through systematic analysis of image perturbation in the Fourier domain, Yin et al. [YLS*19] could show that CNNs are highly sensitive to high-frequency perturbations.

A common way to improve robustness is to train models on more variable input images, using data augmentation methods [PW17], image stylization [GRM*19], or adversarial examples [MMS*19]. To evaluate the performance of these improved models, they are then compared to standard CNN models. To perform these analyses, researchers automatically perturb images offline, log the predictions for each perturbed input image, and compare the results between models. According to our collaborating domain experts, such experiments take several hours to set up and include time-consuming parameter searches. For some experiments, it is not even possible to determine network performance fully automatically as the ground truth of the input images also becomes unclear for humans because of the performed image perturbations. Clearly, these factors severely limit machine learning experts to quickly explore factors of image perturbation that may impact CNN performance.

We hypothesize that *being able to interactively manipulate a synthetic input scene with a large and diverse set of visual perturbation parameters and observing the changing activations and predictions instantly will allow researchers to quickly generate hypotheses and build a stronger intuition for potential vulnerabilities of CNNs*. In this work, we therefore introduce Perturber – a novel real-time experimentation interface for exploratory analysis of model robustness (see Figure 1). Our work provides the following contributions:

1. Interactive input perturbations of 3D scenes in combination with feature visualizations, activation maps, and model predictions as a novel approach to interactively explore model robustness.
2. A direct comparison interface for qualitative visual validation of more robust, fine-tuned model variants.
3. Implementation of a publicly accessible web-based VA tool (`http://perturber.stefansietzen.at/`) that supports a large variety of perturbation methods of 3D input scenes and visualizes the model responses in real-time.
4. Observations from case studies with machine learning experts demonstrating that live inspection of input perturbations allows experts to visually explore known vulnerabilities, to compare model behaviors, and to generate new hypotheses concerning model robustness.
5. Quantitative verification of selected user observations from the case study shedding new light on the robustness of adversarially trained models.

2. Related Work

In recent years, a wide spectrum of deep learning visualization methods has emerged. For a comprehensive overview of visual analytics (VA) for deep learning, we refer the reader to a survey by Hohman et al. [HKPC19]. For example, graph structure visualizations, such as the *TensorFlow Graph Visualizer* [WSW*18], help users to get a better understanding of their models' structure, which comprise numerous layers and connections. Others, such as *DeepEyes* [PHVG*18] and *DeepTracker* [LCJ*19], track detailed metrics throughout the training process to facilitate the identification of model problems or anomalous iterations. *ExplAIner* [SSSEA20] goes beyond monitoring and also integrates different steering mechanisms to help users understand and optimize their models. A case study using a VA system to assess a model's performance to detect and classify traffic lights has shown that interactive VA systems can successfully guide experts to improve their training data [GZL*20].

While these examples all focus on the inspection of a single model, others support model comparisons. For example, using *REMAP* [CPCS20], users can rapidly create model architectures through ablations (i.e., removing single layers of an existing model) and variations (i.e., creating new models through layer replacements) and compare the created models by their structure and performance. Ma et al. [MFH*20] designed multiple coordinated views to help experts analyze network model behaviors after transfer learning. *CNNComparator* [ZHP*17] uses multiple coordinated views to compare the architecture and the prediction of a selected input image between two CNNs. Model comparison is also a crucial aspect of our work. However, the focus of the present work lies on fluid modification of input stimuli and instantaneous analysis and comparison of the network responses. Thus, we let users *generate and perturb* input images from 3D scenes in a "playground-like" manner rather than letting the user select input images with a ground-truth class label.

Interactive "playgrounds" require relatively little underlying deep learning knowledge and can be used for educational purposes. For more informed users, they are valuable for building an intuition and validating knowledge from literature [KTC*19]. Notable examples are *TensorFlow Playground* [SCS*17], which supports the interactive modification and training of DNNs in the browser, and *GANLab* [KTC*19], which is designed in a similar style and

supports experimentation with Generative Adversarial Networks. *CNN Explainer* [WTS*20] provides a visual explanation of the inner workings of a CNN by showing connections between layers and activation maps, allowing the user to choose the input from a fixed collection of images. More similarly to our work, Harley [Har15] provides an online tool where users can draw digits onto a canvas. Then, the responses of all neurons in a simple MNIST-trained network are visualized in real-time. *Adversarial Playground* [NQ17] allows users to interactively generate adversarial attacks and instantly observe the predictions of a simple MNIST-trained network. To support interactive probing of model responses based on input modifications, *Prospector* [KPN16], the *what-if-tool* [WPB*20], and *NLIZE* [LLL*18] allow users to interrogate the model by varying the input in the domains of tabular data and natural language processing, respectively. These works inspired us to build a system that lets users *interactively explore* model robustness through input perturbations. In contrast to prior work, Perturber operates on complex CNN models, such as Inception-V1 trained on ImageNet, and can be used to discover and explain vulnerabilities to complex input scenes, like animals or man-made objects in different environments.

To explain a CNN model, there are powerful methods to reveal the role of a network's hidden units by visualizing their learned features. *Feature visualization* is an activation maximization technique, which was improved by combining a variety of regularization techniques [YCN*15, OMS17]. Feature visualizations have been used to identify and characterize causal connections of neurons in CNNs [CCG*20], and to comprehensively document the role of individual neurons in large CNNs [Ope]. Within VA tools, feature visualizations have been used to compare learned features before and after transfer learning [MFH*20] or to visualize a graph of the most relevant neurons and their connections for a selected target class [HPRC20]. Similarly, *Bluff* [DPW*20] shows a graph containing the most relevant neurons explaining precomputed adversarial attacks, where neurons are represented by their feature visualizations.

Other powerful interpretability methods are *saliency maps* (or *attribution maps*), which show the saliency of the input image's regions with respect to the selected target class or network component [SVZ13]. Saliency maps and other gradient-based methods like *LRP* [LBM*16], *Integrated Gradients* [STY17], or *Grad-CAM* [SCD*17], however, require a computationally expensive backpropagation pass. A very simple solution to reveal relevant image regions for a model's prediction is to directly visualize the forward-propagation activations of selected feature maps in intermediate layers. For example, the *DeepVis Toolbox* [YCN*15] shows live visualizations of CNN activations from a webcam feed. The goal is to get a general intuition what features a CNN has learned. *AEVis* [LLS*18] shows activations of neurons to a pre-defined set of input images along a "datapath visualization". This visualization allows users to trace the effects of adversarial attacks through the hidden layers of a network. Datapaths are formed by critical neurons and their connections that are responsible for the predictions. Like the *DeepVis Toolbox* [YCN*15], Perturber visualizes activations and predictions based on live input. The major difference is that Perturber generates input images from an interactive 3D scene and provides a rich palette of input perturbation methods to gradually explore potential vulnerabilities of a model. In addition, it facilitates direct comparison between a standard model and a more robust variation thereof to explore the benefits and limitations of model variations.

3. Perturber Interface

The high-level goal of Perturber is to interactively explore potential sources of vulnerabilities for CNNs to facilitate the design of more robust models. Through constant exchange between visualization researchers and machine learning experts investigating model interpretability and robustness, we identified four central requirements of a VA application to support exploratory analysis and qualitative validation of model robustness:

R1: Rich **online input perturbation** of a representative scene is essential to quickly generate input images for which model responses can be investigated. The system should provide a large variety of possible perturbation parameters and allow a flexible combination of these perturbations.

R2: To understand what makes a model robust, it is not sufficient to understand *how* a model responds to the perturbed input but also *why* the model's responses change. To this end, looking not only at the input and output, but particularly at the numerous **hidden layers** is inevitable when trying to understand what makes a model react unpredictably for humans.

R3: Interactivity can help users to build intuitions through dynamic experiments [KC19]. We thus aim to make the model responses to input perturbations **instantly** visible to support a fluid feedback loop in a playground-like environment.

R4: Robustness can be achieved by training models on more versatile input images, using data augmentation methods, such as affine image transformations, image stylization, or adversarial training. A direct visual **comparison** between the standard CNN model and its more robust fine-tuned version are necessary for a first qualitative validation whether a fine-tuned model is generally more robust to input perturbations or only selectively more robust to specific perturbations it has been trained on.

Perturber supports these four core requirements through a highly interactive web-based playground consisting of the following components: the 3D input scene (Section 3.1, Figure 2 A), the perturbation control (Section 3.2, Figure 2 B-C), the prediction view (Section 3.3, Figure 2 F), and the (comparative) neuron activation view (Section 3.4, Figure 2 E-D). These views allow for a qualitative inspection of the effects of input scene perturbations on one or two selected models (Section 3.5).

Conceptually, Perturber can handle any CNN architecture. The current version supports models based on the Inception-V1 (also called GoogLeNet) [SLJ*15] architecture. Our standard model was trained on ImageNet [RDS*15]. We use a pre-trained model with weights accessible through the *Lucid* feature visualization library [OMS17].

3.1. Input Scene

The input image is generated from an interactive 3D scene that can be manipulated in numerous ways (Section 3.2). In computer vi-

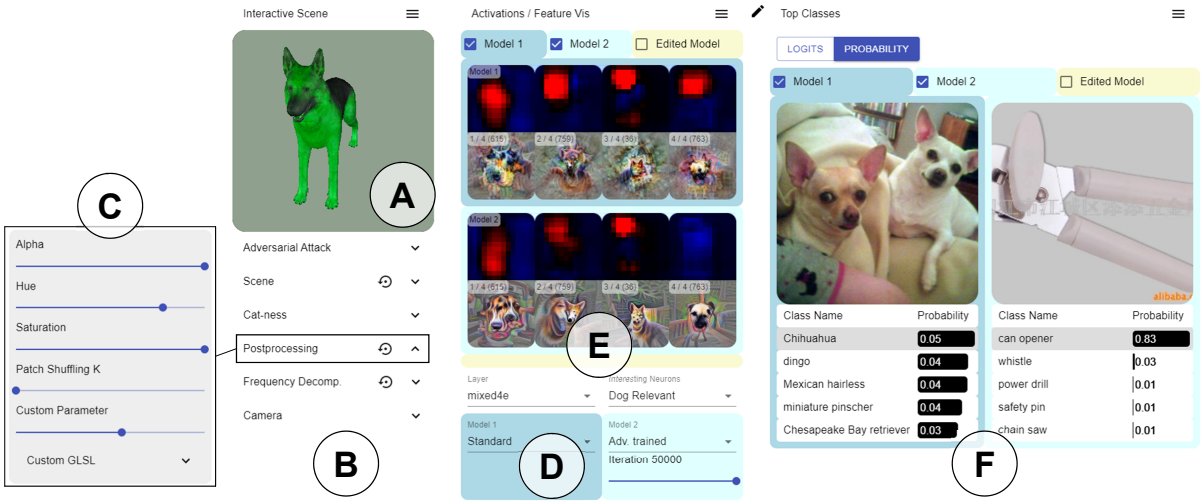


Figure 2: *Perturber interface: users manipulate the 3D scene (A) through a large variety of input perturbation methods that are grouped into functional categories (B). Per category, input perturbations can be seamlessly controlled through sliders (see inset (C)). Users can select two models to compare, as well as a set of neurons from dedicated layers (D). For the selected neurons and models, respectively, neuron responses are visualized live (E). Top-5 predictions for both models are shown as logits or probabilities along with an image example (F).*

sion projects, rendered images are often used instead of – or in addition to – photographs for training CNNs [SQLG15]. Quantitative experiments have shown that, for high-level computer vision algorithms, the gap between synthetic and real images is fairly small [GWCV16]. CNN responses were shown to be consistent between simple rendered 3D models and natural images [AR15]. Interactive exploration of CNN responses based on synthetic scenes therefore seems to be sufficiently representative of real-world applications. The major advantage of a synthetic 3D scene is that perturbation factors can be easily disentangled. That means, input modifications can be flexibly applied independently from each other to assess their isolated effects as well as their interactions.

Out of the 1000 ImageNet classes, 90 of them are dog breeds. Therefore, there are numerous neurons in our Inception-V1 model that specialize in dog-related patterns, such as snouts, head-orientations, fur, eyes, ears, etc. As a consequence, we chose a dog as our main foreground 3D object to represent this special significance and to feed an input image that many hand-identified neuron circuits [CCG*20] respond to strongly. As a second 3D model, we provide a vehicle, which has very different characteristics from a dog and is a commonly analyzed object in the machine learning community (e.g., [GWCV16, AR15]). In addition, users can upload custom 3D models. This way, users can verify observations they have made also with other models.

3.2. Perturbation Control

The core principle of Perturber is that the user can flexibly vary the perturbation factors they want to explore (R1) and observe their effects instantly (R3). Below the input scene view, Perturber provides sliders for seamless control of the perturbations (Figure 2 C). Perturber provides more than 20 scene perturbation factors, as well as all possible combinations between these factors. We group these perturbation methods into the following categories:

Geometry perturbations, such as rotation, translation, or cropping an image, are classic perturbations a CNN might be sensitive to [ZSLG16, ACW18, ETT*19]. Therefore, affine image transformations are also a common data augmentation method for more robust training [PW17]. We support geometric perturbations through simple camera controls, which allow the user to arbitrarily orbit around the 3D model, as well as to freely zoom and pan the scene to create image cropping effects. In addition, users can rotate the camera around the z-axis to simulate image rotation.

Scene perturbations that may seem irrelevant for human observers may easily confuse a standard CNN model. For example, changing the background [XEIM20] can have a tremendous effect on the prediction. We, therefore, support various scene perturbation operations, such as changing the background image, as well as modifying the lighting and texture parameters of the main object (see Figure 3). Through combinations of these parameters, specialized perturbations, such as silhouette images, can be generated (see Figure 3 bottom right).

Shape perturbations let users morph the shape of the main scene object into another one. Perturber currently supports morphing between dog and cat, as well as between a firetruck and a race car. By morphing the shape of an object independently from its texture, the texture-shape cue conflict [GRM*19] can be investigated.

Color perturbations act on the rasterized 2d image. We support individual post-processing effects, such as alpha blending to black, hue shifting, and (de-)saturation. In addition to these parameters, users are provided with a text field where they can write their own GLSL code, taking a single parameter which is controlled by the slider. The code snippet defaults to code for contrast adjustment.

Figure 3: *Scene perturbations – first row: original scene, no lighting influence, full background blur, different background image; second row: no texture influence, full texture blur, lowest background saturation, combination of background saturation, lighting influence, texture influence, and background blur. All perturbation parameters can be controlled seamlessly.*

Using these post-processing effects, users can assess the models' surprisingly high robustness against low contrast [DK16] and low image saturation [SLL20].

Frequency perturbations selectively modify different image frequencies. Perturber provides three-parameter frequency decomposition that splits the image into low and high-frequency bands, which is achieved by Laplacian decomposition. Through selective frequency suppression, users can investigate phenomena such as the one described by Yin et al. [YLS*19], where the authors show that adversarially trained networks are robust against high-frequency perturbations but very sensitive to low-frequency perturbations.

Spatial perturbations are post-processing effects systematically changing the image's pixel order. Perturber supports patch shuffling, which was used by Zhang and Zhu [ZZ19] to reveal a model's sensitivity to global structure, which gets highly disturbed for human observers by patch shuffling. The image is divided into a grid of $k \times k$ cells, which are then randomly re-ordered.

Adversarial perturbations, finally, are a core feature to understanding model robustness. Users can perform projected gradient descent (PGD) adversarial attacks [MMS*19] on the current scene image. To do that, they choose a model to generate the attack from, a target class or the option to suppress the original prediction, as well as the attack ε and L_p norm (L_2 or L_∞). With each button press, the user performs one PGD step with a step length of $\varepsilon/8$. This perturbation is costly and cannot be performed instantaneously. The first step typically takes around 10 seconds on a powerful consumer PC as the gradient function needs to be computed for the current input image first. To let users inspect the effect of an adversarial attack on the model fluidly like the other perturbation parameters, we overlay the current input image with the perturbation vector once it has been computed. This way, users can interactively fade the attack alpha using a slider and observe the system's response instantaneously. They can also fade the original image to inspect the perturbation vector itself.

3.3. Prediction View

The prediction view shows the top-5 classification results for each model (Figure 2 F) either as probability or as logits. This allows the user to observe the classification changes resulting from input perturbations in real-time (R3). Each model's top result is represented by a class image example. Perturber shows the first image of the respective class in the ImageNet validation dataset.

3.4. Neuron Activation View

The neuron activation view is the central interface for the analysis of how input perturbations affect the models' hidden layers (R2). Perturber represents neurons through feature visualizations. Feature visualizations aim at providing a lens into networks to visualize what patterns certain network units respond to [OMS17]. These patterns might be, for example, edges in a particular direction in earlier layers (cf., Figure 4), or specific objects, like dog heads (cf., Figure 1) or car windows, in later layers.

Activation maps show which regions of the input image cause the respective neuron to be activated. After experimenting with an implementation of the gradient-based visualization method *Grad-CAM* [RSG16], we decided to focus on the computationally far less expensive activation maps showing the neuron activations for a forward propagation pass. This way, neuron activations can be observed instantly (R3). Input regions that highly activate the neuron are shown in red, while blue regions cause a negative activation (see Figure 4). As the user manipulates the input scene, the activation maps update instantaneously so that the user can observe in real-time to which image features a neuron responds in one of the models. For example, in Figure 4, neurons 412 and 418 of layer mixed4a respond to oriented patterns. Rotating the textured race car causes these two neurons to strongly change their activations.

Inception-V1 contains thousands of neurons. Clearly, it is impossible to show feature visualizations and interactive activation maps for all neurons concurrently. Instead, we provide pre-selected neuron groups and categories for each layer, which were characterized into semantically meaningful neuron families in the course

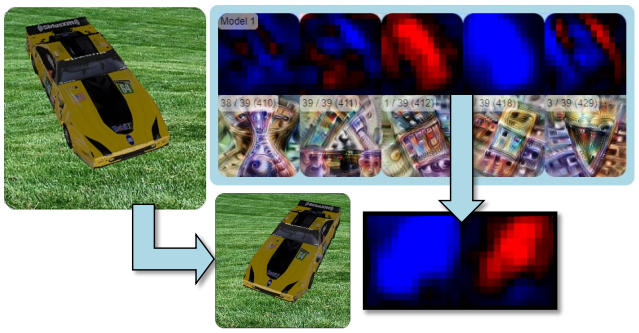


Figure 4: *Feature visualizations for neurons associated with complex shapes and curvatures in layer mixed4a for the standard model, as well as their activation maps for the input on the left. Note how rotating the input model causes an activation change for oriented shape detectors (insets on the bottom).*

of the *Circuits* project [CCG*20]. In addition to these previously presented neuron groups, we identified further sets of dog-, cat-, race car-, and firetruck-related neurons in later layers using *Summit* [HPRC20]. These pre-selected neuron groups consisting of one to up to around 70 neurons can be selected from drop-down menus (Figure 2 D).

3.5. Model Comparison

After exploring potential sources of vulnerability, model designers commonly fine-tune existing models using augmented training data covering the identified vulnerability. To leverage Perturber for direct comparison of robustness (R4), we employ a transfer learning approach, which is initialized with the weights from the standard model. This is necessary to guarantee that feature correspondence of individual network units is kept intact. Only this way, we can assume that units across models respond similarly to identical input images.

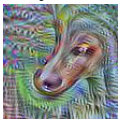
To illustrate what individual network units respond to, Perturber relies on feature visualizations. Feature visualizations are independent of the input image, but they differ between models. Indeed, Table 1 shows noticeable differences between a single neuron's feature visualizations of three different model variations.

To showcase model comparison, Perturber already includes two robust model variations of Inception-V1: The first model was adversarially trained with projected gradient descent (PGD), which was described by Madry et al. [MMS*19]. Adversarial training increases model robustness by incorporating strong adversarial attacks into the training procedure. The second model was trained by Stylized-ImageNet, which is a variation of ImageNet, where images were transformed into different painting styles using a style transfer algorithm [GRM*19]. The purpose of Stylized-ImageNet was to induce a shape bias, while the standard ImageNet-trained models were found to be biased unproportionally towards texture [GRM*19]. Users can choose two models for comparison from the interface (Figure 2 D). In addition, they can choose multiple checkpoints along the incremental fine-tuning process to analyze the development of the models during training. Corresponding neuron activation views are then juxtaposed in the Perturber interface for direct comparison (Figure 2 E).

4. Web-Based Implementation

Perturber runs purely on the client-side in the user's browser and without any server-side computations at runtime. We precomputed all data that does not depend on interactively changeable elements to make the UI as efficient as possible. For transfer learning, we initialized the weights with those of the standard model (i.e., Inception-V1 trained on ImageNet), which were obtained through the *Lucid* library [OMS17]. For adversarial fine-tuning, we used the open-source implementation by Tsipras et al. [TSE*19]. No layers were frozen for transfer learning. For both fine-tuned models, we used a learning rate of 0.003, a batch size of 128, and we trained until the models reached a top-5 training error of around 0.5, which took approximately 50K iterations for the adversarial fine-tuning and around 90K iterations for the Stylized-ImageNet fine-tuning.

Table 1: *Comparison between naive pixel (top) and Fourier basis (bottom) parametrized feature visualizations of neuron 222 in layer mixed4a for three model variations.*

	standard	**adversarial**	**stylized**
naive pixel param.			
Fourier basis param.			

To generate feature visualization, we employed the implementation provided by the *Lucid* [OMS17] library. For each model, we generated feature visualizations for 5808 neurons from the most relevant layers (i.e., the first three convolutional layers and the concatenation layers at the end of each mixed-block). During fine-tuning, we obtained feature visualizations for 17 checkpoints of adversarial fine-tuning and seven checkpoints of Stylized-ImageNet fine-tuning. For each neuron, we computed feature visualizations using two parametrization methods – naive pixel or Fourier basis [OMS17] (see Table 1). The second performs gradient ascent in Fourier space and leads to a more equal distribution of frequencies, resulting in more naturally looking feature visualizations. We use *transformation robustness* [OMS17] in addition to both methods. Without Fourier basis parametrization, the differences between the models are more visually distinct (Table 1 top row). The computation of all ∼300K generated feature visualizations required around one month on a machine with two NVIDIA GTX 1070 GPUs. All generated feature visualizations can be downloaded from `https://github.com/stefsietz/perturber/`.

Manipulation of the 3D scene, model inference based on the rendered image, and computation of activation maps are performed live in the web browser. The client relies on GPU acceleration for both, 3D scene rendering and CNN inference. We use the WebGL-based libraries *Three.js* and *TensorFlow.js* [STA*19] for these tasks, respectively. The front-end GUI is based on *React.js*. Input perturbations based on post-processing effects are implemented as multiple sequential render passes with custom GLSL shaders. For model inference, we use *Tensorflow.js* [STA*19], which enables fast GPU-accelerated CNN inference. *Tensorflow.js* is also used for computing adversarial attacks.

A major requirement of Perturber is that the effects of input perturbations can be observed instantaneously (R3). To assess requirement R3, we measured the client's performance while constantly orbiting the camera around the object. Figure 5 shows the recorded frame rates for two client notebooks: a MacBook Pro 13" 2018 with Intel Iris Plus Graphics 655 (MBP) and an AORUS 15G Gaming Notebook with an NVIDIA GeForce GTX 2080 Super GPU (AORUS). It is clearly visible that the GPU of the client machine has a strong influence on the frame rate. The application performance also depends on the enabled visualizations, with the prediction view (Section 3.3) being more computationally expensive than

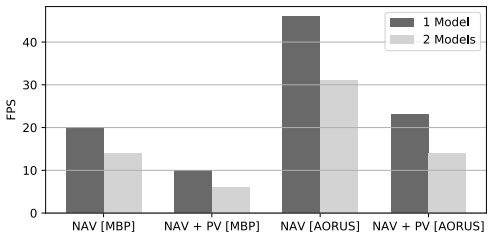


Figure 5: *Performance benchmarks for four different output configurations, measured on two machines (MBP or AORUS): neuron activation view (NAV) only, or neuron activation view in combination with prediction view (NAV+PV) visualized for one or two visualized models concurrently.*

showing four live activation maps (Section 3.4). When using less powerful client machines, users can selectively switch off views.

5. Exploration Scenario: Texture-Shape Conflict

To demonstrate the capabilities of Perturber, we selected one important aspect of robustness: the influence of shape and texture on CNN classification results. It has been shown that CNNs are biased towards texture, which affects robustness [GRM*19]. In this scenario, we first aim to investigate whether this effect could have been discovered using Perturber. Secondly, we aim to qualitatively validate whether a more robust model variation, which was fine-tuned using Stylized-ImageNet [GRM*19], can improve upon this bias.

For our first analysis, we explore the texture-shape cue conflict through shape and texture perturbations. Figure 6 illustrates that shape perturbations alone do not cause the standard model to predict a cat breed. However, when morphing the texture, the model predictions switch to cat breeds – even when the object shape remains unchanged. This is a first indicator that the model is indeed much more sensitive to texture than to shape perturbations.

To further investigate the texture sensitivity of the standard model, we replicated the patch shuffling experiment by Zhang and Zhu [ZZ19]. Using Perturber, we can inspect single neurons' activations during this experiment. We illustrate our observations on a hand-picked neuron in Figure 7, which is strongly activated by dog faces looking to the left. Note how this neuron is activated by strong texture contrasts, especially around the mouth of the dog. Image regions containing ears do not activate this particular neuron. For this scene, the standard model still predicts a dog breed up to $k = 7$ randomly shuffled image patches. This illustrates that the decomposed shape has indeed very little influence on the model prediction.

In the next step, we analyze if the standard model is indeed more sensitive to texture than the model fine-tuned on Stylized-ImageNet [GRM*19] by gradually removing the texture of the main object. The predictions in Figure 8 confirm that the Stylized-ImageNet trained model consistently predicts a dog breed, even in the absence of a texture. The standard model, on the other hand,

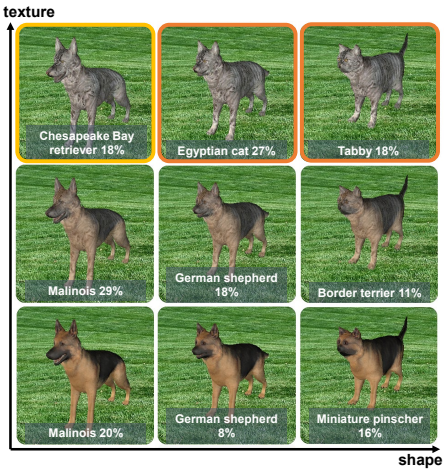


Figure 6: *Object morphing between dog (bottom left) and cat (top right) along the texture dimension (y axis) and shape dimension (x axis): Top-1 standard model predictions with their probabilities are shown as text labels. Input images leading to cat breeds within the top-5 or top-1 predicted classes are indicated by a yellow and orange frame, respectively (top row).*

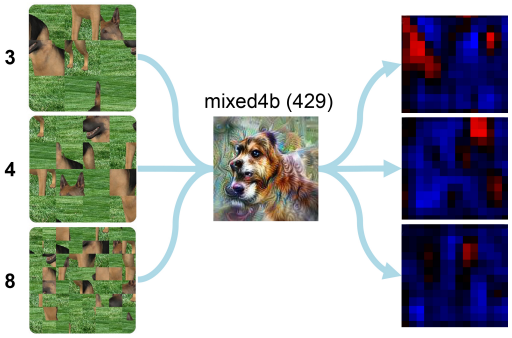


Figure 7: *Randomly shuffling the dog scene into 3 (top), 4 (middle), and 8 (bottom) patches: A dog-related neuron in mixed4b of the standard model is activated by patches containing parts of the dog's face and textured body parts.*

seems to rely much more on the texture. The model trained with Stylized-ImageNet is also more sensitive to patch shuffling, which is another indication that it relies less on texture information than the standard model (see Section A in the Supplemental Document for image examples).

Finally, we compare shape sensitivity between the two models. To this end, we combine various scene perturbations to generate a silhouette image of the dog. We then gradually change the pose of the dog. Figure 9 shows the predictions of the standard model and the model fine-tuned with Stylized-ImageNet. Clearly, the predictions of the standard model are unstable, especially when the dog is directly facing the camera. The model trained with Stylized-ImageNet, however, reliably predicts a white wolf with high probability. The activations of relevant neurons indicate that the Stylized-

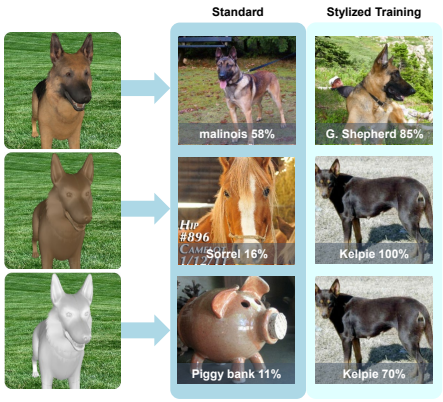


Figure 8: *Inspecting the shape vs. texture conflict through scene perturbations for the standard (left) and the Stylized-ImageNet trained model (right): while the standard model gets confused by missing textures, the Stylized-ImageNet trained model predicts a dog breed even if the 3D model is untextured.*

ImageNet-based model has learned a more stable representation of a frontal dog head, which also get activated by a silhouette image.

These examples illustrate that Perturber provides flexible ways to explore and better understand potential threats to robustness, such as the texture-shape conflict of CNNs. Section A in the supplemental document contains more exploration scenarios.

6. Case Studies

While the previous scenario investigated a known vulnerability, we further explored whether Perturber can help to discover unknown threats to robustness through case studies with machine learning experts. We conducted case studies with five machine learning researchers (four PhD students, one post-doctoral researcher; one female, four males). Except for one user, sessions were conducted individually through video conferencing and lasted approximately one hour each. One user preferred to perform the case study offline and sent a textual text report instead.

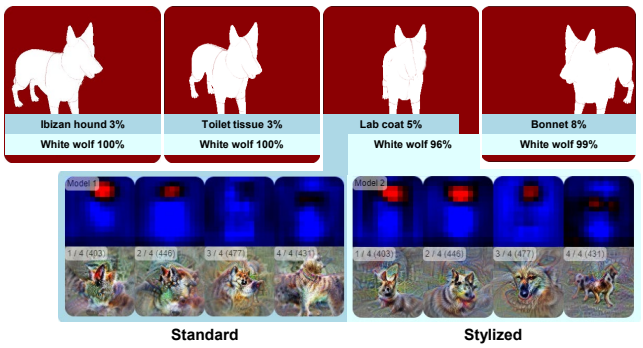


Figure 9: *Rotating a dog silhouette: the corresponding predictions of the standard model and the model trained on Stylized-ImageNet, as well as activations of dog-related neurons for the front-facing dog image in mixed4d of the standard model (left) and the robust model (right).*

The researchers cover various topics of expertise in the field of AI interpretability and the design of robust machine learning models. The concrete research areas are listed in Section B of the supplemental document. Two users were involved in the co-design process of Perturber and are co-authors of the paper. Three users were unfamiliar with the system before the evaluation. One of these users entered the co-design iteration process after the case study and is also a co-author. Paper co-authorship is a common collaboration role in design studies [SMM12].

During the video conference, the participant used the online tool while sharing his/her screen with the first and last author. Every session was recorded while conversations and observations were transcribed on-the-fly or in retrospective through automatic speech-to-text.

Every video conferencing session started with a short introduction by the participant describing his/her research focus. Afterwards, we gave a short demonstration of Perturber's features. We then asked the participant to shortly comment on his/her first impression and his/her expected insights from the analysis. Then, the participant freely played around with Perturber while thinking aloud. In particular, we asked the user to always state his/her intent before performing an action and whether he/she would have any particular hypotheses about the response and behavior of the network based on the chosen input. If a user could not find the respective functionality of the tool, the first and/or last author provided oral assistance. At the end of the study, the user was encouraged to summarize his/her impressions, the potential benefits of the tool, and to provide suggestions for improvements.

6.1. Observations and Feedback

Users praised the fact that Perturber works "live" and therefore allows for ad-hoc **exploratory analyses**. They liked that Perturber allows to play around with simple examples to quickly find patterns and form hypotheses. Generally, the main focus of the exploratory analyses was trying to identify input perturbations where a model would respond unexpectedly. One user described this process as trying to answer the following questions: *"How can I break a model? What do I need to do so that the resulting prediction is wrong?"* For example, three users were surprised to see how vulnerable the adversarially trained model seems to be to some geometric changes, such as zooming or rotating. Two users also discovered a high sensitivity of adversarially trained models to background modifications, as illustrated in Figure 10. The live approach was praised in particular for cases without a clear ground truth, such as object morphing (Figure 6). Such scenarios would not be possible to assess quantitatively without human subject studies. Being able to quickly generate hypotheses that could then be formally tested in a more controlled setting was considered very useful.

We also observed indications that Perturber is practically helpful for **visual confirmation**. For example, using the model predictions and activations of selected neurons, one user verified that adversarial attacks only affect the standard model. He also observed how

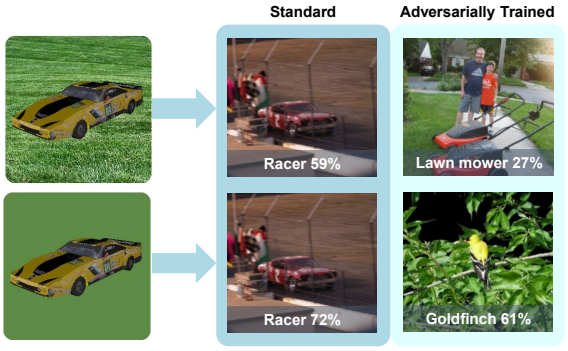


Figure 10: *Blurring the background of the race car has little influence on the standard model's predictions (left). The adversarially trained model (right) is very sensitive to the chosen background. The probability for "racer" is 11% with a grass background and 7% with a uniform green background.*

animal-related neurons of the standard model got activated during an attack with the target class "badger" (Figure 11). Not all assumptions were actually confirmed. For example, one user expected that the model trained on Stylized-ImageNet would be noticeably more vulnerable to image blur than the standard model. Unexpectedly, the model's responses were not more sensitive to blur than the standard model's for the chosen input scene (see Section A in the supplemental document).

Finally, a user pointed out that playing around with Perturber would **let non-experts get an intuition** how easily CNNs are fooled. For example, one user demonstrated how the standard

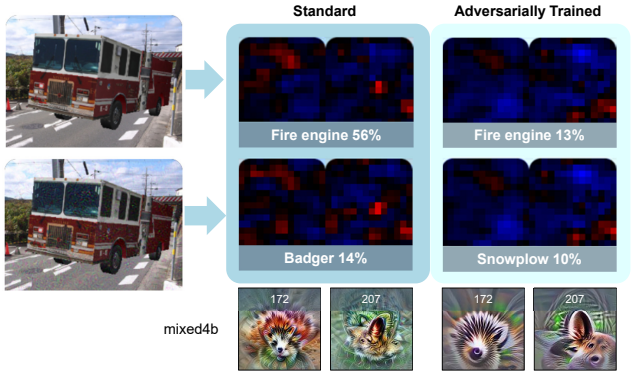


Figure 11: *Adversarial attack with target class "badger" on the fire truck scene (top) and the effect on two cat-related neurons in layer mixed4b for the standard model (left) and adversarially trained model (right): As the attack strength is increased (bottom), activations of some standard model neurons increase while the activations of the adversarially trained model are hardly affected. The prediction probabilities for the fire engine decrease to 0.1% for the standard model and to 6% for the adversarially trained model after the attack.*

model sometimes rapidly changes its predictions during a simple translation of the dog. Another expert demonstrated that a rotation of the dog along the z-axis in combination with an unusual background (street) was sufficient to disturb the adversarially trained model. Accordingly, he stated that Perturber could be informative for "people fearing that AI will take over the world".

A comprehensive list of observations reported by the individual users can be found in Section B of the supplemental document. Overall, the most frequently performed perturbations leading to the most interesting observations in our case studies were 1) geometric transformations, such as object rotation, zooming, and translation, 2) modification of the background, 3) combinations of (1) and (2), as well as 4) object morphing. The prediction view was perceived as giving instant, easily interpretable, and useful feedback. It was thus the primary view to observe model behavior. Feature visualizations were considered useful to characterize the difference between the models. Participants described feature visualizations of the adversarially trained model as more "intuitive" or "cartoonish" compared to the corresponding standard model's neurons. One participant found feature visualizations sometimes hard to interpret. One recommendation therefore was to additionally show strongly activating natural image examples from a dataset.

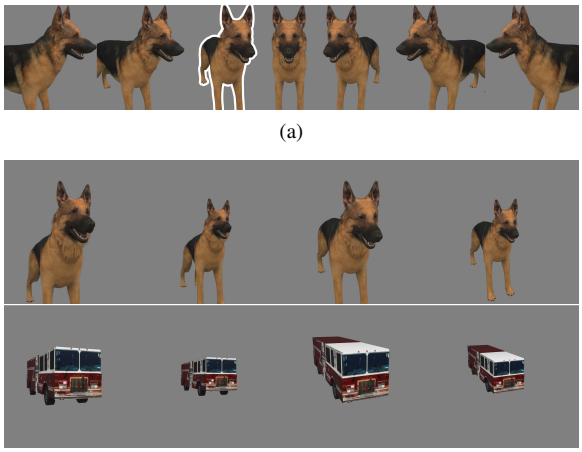
6.2. Quantitative Evaluation of User Observations

We performed quantitative measurements to see whether what users observed visually can be replicated with systematically varied input scenes, natural images, and a different network architecture. Specifically, we used the pre-trained ResNet50 from the *torchvision* library of *PyTorch* [PGM*19] as the standard model. For comparison, we used weights of an adversarially trained version of the same model from the *robustness* library [EIS*19] (ResNet50 ImageNet L_2-norm ε 3/255).

First, we investigated the adversarially trained model's sensitivity to background changes, which was reported by two users in the case study. For verification, we performed the *Background Challenge* by Xiao et al. [XEIM20], where a model is tested with (natural image) adversarial backgrounds. The adversarially trained model can only correctly predict 12.3% under background variations. The standard model achieves 22.3% accuracy. Using this test dataset, random guessing would yield 11.1% accuracy [XEIM20]. This shows that the robustified network is considerably more susceptible to adversarial backgrounds than the standard model.

Second, we tested if the adversarially trained model is indeed more sensitive to geometric scene transformations than the standard model (reported by three users). In particular, we looked into camera rotation. We generated a synthetic dataset, where we rendered the four 3D models provided in Perturber from seven yaw angles, ranging from -70°to 70° (Figure 12a), two pitch angles, and two distances of the camera to the object, as shown in Figure 12b. In total, we generated 28 views for each of the four 3D models.

To compare how much the predictions fluctuate, we chose a prototype view with a yaw angle of -23.3°for each of the four pitch / distance combinations and 3D model (Figure 12b). For each of the 16 resulting prototypes, the logits of the top-10 classes served as



(a)

(b)

Figure 12: *Seven yaw angles variations (a) tested for four prototype views (b) per model (two of the four models are shown here). The prototype view in (a) is highlighted.*

ground truth vector $\mathbf{l}^{\star}_{10}$. For the 16 prototype views, we then calculated a fluctuation score f_p:

$$f_p = \sum_y \frac{\|\mathbf{l}^{\star}_{10} - \mathbf{l}^{y}_{10}\|_2}{s(\mathbf{l}^{\star}_{n})}, \tag{1}$$

where $\mathbf{l}^{\star}_{10}$ are the top-10 predictions of the prototype view, $\mathbf{l}^{y}_{10}$ is the logit vector of these 10 classes for the view associated with yaw y, and $s(\mathbf{l}^{\star}_{n})$ is the standard deviation of the logits of all n classes in the prototype view. In other words, the fluctuation score measures how strongly the logits of the top-10 prototype classes diverge in the rotated input images.

Figure 13 shows the average fluctuation score of all 28 prototype views. The fluctuation scores are considerably higher for the adversarially trained model for three of the four 3D models. This verifies that the adversarially trained model can be indeed more vulnerable to rotations of the main object.

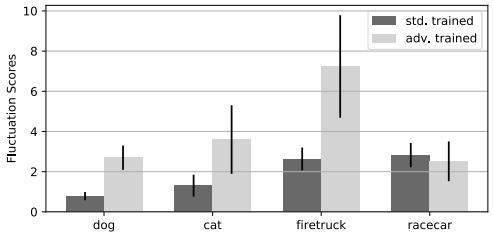


Figure 13: *Average yaw fluctuation scores for the standard model and the adversarially trained model across the four 3D models. Error bars show the standard deviation.*

7. Discussion & Conclusions

We showed that interactive perturbations in combination with live activations can be an effective method to explore potential vulnerabilities of CNNs and to perform qualitative evaluations of more robust model variations. In an exploration scenario, we could replicate the known texture-shape conflict [GRM*19] through multiple perturbation examples. Machine learning experts participating in our case study observed a variety of known but also unexpected network behaviors. They could successfully replicate known CNN properties, such as models' varying sensitivity to adversarial attacks or patch shuffling, using our synthetic input scenes. In addition, our quantitative post-study experiments of selected user observations could replicate their observations using a different network architecture and natural images. Through these experiments, we demonstrated vulnerabilities of adversarially trained models to background modifications and yaw rotations of the main object that, to the best of our knowledge, have not been discussed yet in the machine learning community.

The majority of insights reported by our users were based on observations how the model predictions changed when perturbing the image. This implies that the principle of live input perturbations could also be useful for pure black-box models. To get a better intuition about which image features are influential for the model's final decision, our exploration scenarios indicate that neuron activations are also essential. But due to the large number of logical neuron units distributed among multiple layers, it can be time-consuming to find a set of neurons that eventually is affected strongly by the performed input perturbation. In the future, we thus plan to investigate alternative methods to select the displayed neurons in the neuron activation view. Also visual guidance to support the discovery of potentially harmful perturbation factors would be helpful. However, traditional guidance mechanisms may require costly computation, which will hamper interactivity (R3). Effective guidance mechanisms can therefore be considered interesting future work. Another interesting line of future work would be the support for encoder-decoder architectures, used prominently for semantic segmentation and image translation tasks among others. This could be facilitated by replacing the prediction view with a continuously updating display of the generated image.

Acknowledgments

We thank Robert Geirhos and Roland Zimmermann for their participation in the case study and valuable feedback, Chris Olah and Nick Cammarata for valuable discussions in the early phase of the project, as well as the Distill Slack workspace as a platform for discussions. M.L. is supported in part by the Austrian Science Fund (FWF) under grant Z211-N23 (Wittgenstein Award). J.B. is supported by the German Federal Ministry of Education and Research (BMBF) through the Competence Center for Machine Learning (TUE.AI, FKZ 01IS18039A) and the International Max Planck Research School for Intelligent Systems (IMPRS-IS). R.H. is partially supported by Boeing and Horizon-2020 ECSEL (grant 783163, iDev40).

References

[ACW18] ATHALYE A., CARLINI N., WAGNER D.: Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. *arXiv:1802.00420 [cs]* (July 2018). arXiv: 1802.00420. URL: http://arxiv.org/abs/1802.00420. 4

[AR15] AUBRY M., RUSSELL B. C.: Understanding deep features with computer-generated imagery. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 2875–2883. doi:10.1109/ICCV.2015.329. 4

[BHL*21] BABAIEE Z., HASANI R., LECHNER M., RUS D., GROSU R.: On-off center-surround receptive fields for accurate and robust image classification. In *International Conference on Machine Learning* (2021), PMLR, pp. 478–489. 2

[CCG*20] CAMMARATA N., CARTER S., GOH G., OLAH C., PETROV M., SCHUBERT L.: Thread: Circuits. *Distill 5*, 3 (Mar. 2020). URL: https://distill.pub/2020/circuits, doi:10.23915/distill.00024. 3, 4, 6

[CPCS20] CASHMAN D., PERER A., CHANG R., STROBELT H.: Ablate, Variate, and Contemplate: Visual Analytics for Discovering Neural Architectures. *IEEE Transactions on Visualization and Computer Graphics 26*, 1 (Jan. 2020), 863–873. doi:10.1109/TVCG.2019.2934261. 2

[DK16] DODGE S., KARAM L.: Understanding how image quality affects deep neural networks. In *2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX)* (2016), pp. 1–6. doi:10.1109/QoMEX.2016.7498955. 2, 5

[DPW*20] DAS N., PARK H., WANG Z. J., HOHMAN F., FIRSTMAN R., ROGERS E., CHAU D. H.: Bluff: Interactively Deciphering Adversarial Attacks on Deep Neural Networks. *arXiv:2009.02608 [cs]* (Sept. 2020). arXiv: 2009.02608. URL: http://arxiv.org/abs/2009.02608. 3

[EEF*18] EYKHOLT K., EVTIMOV I., FERNANDES E., LI B., RAHMATI A., XIAO C., PRAKASH A., KOHNO T., SONG D.: Robust Physical-World Attacks on Deep Learning Visual Classification. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (June 2018), pp. 1625–1634. ISSN: 2575-7075. doi:10.1109/CVPR.2018.00175. 2

[EIS*19] ENGSTROM L., ILYAS A., SALMAN H., SANTURKAR S., TSIPRAS D.: Robustness (Python Library), 2019. URL: https://github.com/MadryLab/robustness. 9

[ETT*19] ENGSTROM L., TRAN B., TSIPRAS D., SCHMIDT L., MADRY A.: Exploring the landscape of spatial robustness. In *International Conference on Machine Learning* (2019), PMLR, pp. 1802–1811. URL: https://arxiv.org/abs/1712.02779. 4

[GRM*19] GEIRHOS R., RUBISCH P., MICHAELIS C., BETHGE M., WICHMANN F. A., BRENDEL W.: Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. *International Conference on Learning Representations (ICLR)* (May 2019). URL: https://openreview.net/forum?id=Bygh9j09KX. 2, 4, 6, 7, 10

[GSS14] GOODFELLOW I. J., SHLENS J., SZEGEDY C.: Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014). URL: https://arxiv.org/abs/1412.6572. 2

[GWCV16] GAIDON A., WANG Q., CABON Y., VIG E.: Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 4340–4349. doi:10.1109/CVPR.2016.470. 4

[GZL*20] GOU L., ZOU L., LI N., HOFMANN M., SHEKAR A. K., WENDT A., REN L.: Vatld: a visual analytics system to assess, understand and improve traffic light detection. *IEEE transactions on visualization and computer graphics* (2020). doi:10.1109/TVCG.2020.3030350. 2

[Har15] HARLEY A. W.: An Interactive Node-Link Visualization of Convolutional Neural Networks. In *Advances in Visual Computing*, vol. 9474. Springer International Publishing, Cham, 2015, pp. 867–877. doi:10.1007/978-3-319-27857-5_77. 3

[HKPC19] HOHMAN F., KAHNG M., PIENTA R., CHAU D. H.: Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers. *IEEE Transactions on Visualization and Computer Graphics 25*, 8 (Aug. 2019), 2674–2693. doi:10.1109/TVCG.2018.2843369. 2

[HPRC20] HOHMAN F., PARK H., ROBINSON C., CHAU D. H. P.: Summit: Scaling Deep Learning Interpretability by Visualizing Activation and Attribution Summarizations. *IEEE Transactions on Visualization and Computer Graphics 26*, 1 (Jan. 2020), 1096–1106. doi:10.1109/TVCG.2019.2934659. 3, 6

[KC19] KAHNG M., CHAU D. H.: How does visualization help people learn deep learning? evaluation of GAN Lab. In *Workshop on EValuation of Interactive VisuAl Machine Learning systems* (2019). 3

[KPN16] KRAUSE J., PERER A., NG K.: Interacting with Predictions: Visual Inspection of Black-box Machine Learning Models. pp. 5686–5697. doi:10.1145/2858036.2858529. 3

[KSH17] KRIZHEVSKY A., SUTSKEVER I., HINTON G. E.: ImageNet classification with deep convolutional neural networks. *Communications of the ACM 60*, 6 (May 2017), 84–90. URL: https://dl.acm.org/doi/10.1145/3065386, doi:10.1145/3065386. 1

[KTC*19] KAHNG M., THORAT N., CHAU D. H., VIÉGAS F. B., WATTENBERG M.: GAN Lab: Understanding Complex Deep Generative Models using Interactive Visual Experimentation. *IEEE Transactions on Visualization and Computer Graphics 25*, 1 (Jan. 2019), 310–320. doi:10.1109/TVCG.2018.2864500. 2

[LBM*16] LAPUSCHKIN S., BINDER A., MONTAVON G., MÜLLER K.-R., SAMEK W.: The LRP Toolbox for Artificial Neural Networks. *Journal of Machine Learning Research 17*, 114 (2016), 1–5. 3

[LCJ*19] LIU D., CUI W., JIN K., GUO Y., QU H.: DeepTracker: Visualizing the Training Process of Convolutional Neural Networks. *ACM Transactions on Intelligent Systems and Technology 10*, 1 (Jan. 2019), 1–25. URL: https://dl.acm.org/doi/10.1145/3200489, doi:10.1145/3200489. 2

[LHA*20] LECHNER M., HASANI R., AMINI A., HENZINGER T. A., RUS D., GROSU R.: Neural circuit policies enabling auditable autonomy. *Nature Machine Intelligence 2*, 10 (2020), 642–652. doi:10.1038/s42256-020-00237-3. 2

[LHG*21] LECHNER M., HASANI R., GROSU R., RUS D., HENZINGER T. A.: Adversarial training is not ready for robot learning. *arXiv preprint arXiv:2103.08187* (2021). 2

[LLL*18] LIU S., LI Z., LI T., SRIKUMAR V., PASCUCCI V., BREMER P.-T.: Nlize: A perturbation-driven visual interrogation tool for analyzing and interpreting natural language inference models. *IEEE Transactions on Visualization and Computer Graphics 25*, 1 (2018), 651–660. doi:10.1109/TVCG.2018.2865230. 3

[LLS*18] LIU M., LIU S., SU H., CAO K., ZHU J.: Analyzing the Noise Robustness of Deep Neural Networks. In *2018 IEEE Conference on Visual Analytics Science and Technology (VAST)* (Oct. 2018), pp. 60–71. doi:10.1109/VAST.2018.8802509. 3

[MFH*20] MA Y., FAN A., HE J., NELAKURTHI A. R., MACIEJEWSKI R.: A Visual Analytics Framework for Explaining and Diagnosing Transfer Learning Processes. *IEEE Transactions on Visualization and Computer Graphics* (2020), 1–1. doi:10.1109/TVCG.2020.3028888. 2, 3

[MMS*19] MADRY A., MAKELOV A., SCHMIDT L., TSIPRAS D., VLADU A.: Towards Deep Learning Models Resistant to Adversarial Attacks. *arXiv:1706.06083 [cs, stat]* (Sept. 2019). arXiv: 1706.06083. URL: http://arxiv.org/abs/1706.06083. 2, 5, 6

[NQ17] NORTON A. P., QI Y.: Adversarial-Playground: A visualization suite showing how adversarial examples fool deep learning. In *2017 IEEE Symposium on Visualization for Cyber Security (VizSec)* (2017), IEEE, pp. 1–4. doi:10.1109/VIZSEC.2017.8062202. 3

[OMS17] OLAH C., MORDVINTSEV A., SCHUBERT L.: Feature Visualization. *Distill 2*, 11 (Nov. 2017), e7. URL: https://distill.pub/2017/feature-visualization, doi:10.23915/distill.00007. 3, 5, 6

[Ope] OpenAI Microscope. https://microscope.openai.com/models. (Accessed on 10/12/2020). 3

[PGM*19] PASZKE A., GROSS S., MASSA F., LERER A., BRADBURY J., CHANAN G., KILLEEN T., LIN Z., GIMELSHEIN N., ANTIGA L., DESMAISON A., KOPF A., YANG E., DEVITO Z., RAISON M., TEJANI A., CHILAMKURTHY S., STEINER B., FANG L., BAI J., CHINTALA S.: PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. 9

[PHVG*18] PEZZOTTI N., HOLLT T., VAN GEMERT J., LELIEVELDT B. P., EISEMANN E., VILANOVA A.: DeepEyes: Progressive Visual Analytics for Designing Deep Neural Networks. *IEEE Transactions on Visualization and Computer Graphics 24*, 1 (Jan. 2018), 98–108. doi:10.1109/TVCG.2017.2744358. 2

[PW17] PEREZ L., WANG J.: The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621* (2017). URL: https://arxiv.org/abs/1712.04621. 2, 4

[RDS*15] RUSSAKOVSKY O., DENG J., SU H., KRAUSE J., SATHEESH S., MA S., HUANG Z., KARPATHY A., KHOSLA A., BERNSTEIN M., ET AL.: ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision 115*, 3 (2015), 211–252. doi:10.1007/s11263-015-0816-y. 3

[RHGS17] REN S., HE K., GIRSHICK R., SUN J.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence 39*, 6 (June 2017), 1137–1149. doi:10.1109/TPAMI.2016.2577031. 1

[RSG16] RIBEIRO M., SINGH S., GUESTRIN C.: "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations* (June 2016), Association for Computational Linguistics, pp. 97–101. doi:10.18653/v1/N16-3020. 5

[SCD*17] SELVARAJU R. R., COGSWELL M., DAS A., VEDANTAM R., PARIKH D., BATRA D.: Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *2017 IEEE International Conference on Computer Vision (ICCV)* (Oct. 2017), pp. 618–626. ISSN: 2380-7504. doi:10.1109/ICCV.2017.74. 3

[SCS*17] SMILKOV D., CARTER S., SCULLEY D., VIÉGAS F. B., WATTENBERG M.: Direct-Manipulation Visualization of Deep Networks. *arXiv:1708.03788 [cs, stat]* (Aug. 2017). arXiv: 1708.03788. URL: http://arxiv.org/abs/1708.03788. 2

[SLJ*15] SZEGEDY C., LIU W., JIA Y., SERMANET P., REED S., ANGUELOV D., ERHAN D., VANHOUCKE V., RABINOVICH A.: Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 1–9. doi:10.1109/CVPR.2015.7298594. 3

[SLL20] STURMFELS P., LUNDBERG S., LEE S.-I.: Visualizing the impact of feature attribution baselines. *Distill 5*, 1 (2020), e22. URL: https://distill.pub/2020/attribution-baselines/. 5

[SMM12] SEDLMAIR M., MEYER M., MUNZNER T.: Design study methodology: Reflections from the trenches and the stacks. *IEEE transactions on visualization and computer graphics 18*, 12 (2012), 2431–2440. doi:10.1109/TVCG.2012.213. 8

[SQLG15] SU H., QI C. R., LI Y., GUIBAS L. J.: Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 2686–2694. doi:10.1109/ICCV.2015.308. 4

[SSSEA20] SPINNER T., SCHLEGEL U., SCHÄFER H., EL-ASSADY M.: explAIner: A Visual Analytics Framework for Interactive and Explainable Machine Learning. *IEEE Transactions on Visualization and Computer Graphics 26*, 1 (Jan. 2020), 1064–1074. doi:10.1109/TVCG.2019.2934629. 2

[STA*19] SMILKOV D., THORAT N., ASSOGBA Y., YUAN A., KREEGER N., YU P., ZHANG K., CAI S., NIELSEN E., SOERGEL D., BILESCHI S., TERRY M., NICHOLSON C., GUPTA S. N., SIRAJUDDIN S., SCULLEY D., MONGA R., CORRADO G., VIÉGAS F. B., WATTENBERG M.: TensorFlow.js: Machine Learning for the Web and Beyond. *arXiv:1901.05350 [cs]* (Feb. 2019). arXiv: 1901.05350. URL: http://arxiv.org/abs/1901.05350. 6

[STY17] SUNDARARAJAN M., TALY A., YAN Q.: Axiomatic Attribution for Deep Networks. *arXiv:1703.01365 [cs]* (June 2017). arXiv: 1703.01365. URL: http://arxiv.org/abs/1703.01365. 3

[SVZ13] SIMONYAN K., VEDALDI A., ZISSERMAN A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034* (2013). URL: https://arxiv.org/abs/1312.6034. 3

[SZS*14] SZEGEDY C., ZAREMBA W., SUTSKEVER I., BRUNA J., ERHAN D., GOODFELLOW I., FERGUS R.: Intriguing properties of neural networks. *arXiv:1312.6199 [cs]* (Feb. 2014). arXiv: 1312.6199. URL: http://arxiv.org/abs/1312.6199. 2

[TSE*19] TSIPRAS D., SANTURKAR S., ENGSTROM L., TURNER A., MADRY A.: Robustness May Be at Odds with Accuracy. *arXiv:1805.12152 [cs, stat]* (Sept. 2019). arXiv: 1805.12152. URL: http://arxiv.org/abs/1805.12152. 6

[WPB*20] WEXLER J., PUSHKARNA M., BOLUKBASI T., WATTENBERG M., VIÉGAS F., WILSON J.: The What-If Tool: Interactive Probing of Machine Learning Models. *IEEE Transactions on Visualization and Computer Graphics 26*, 1 (Jan. 2020), 56–65. doi:10.1109/TVCG.2019.2934619. 3

[WSW*18] WONGSUPHASAWAT K., SMILKOV D., WEXLER J., WILSON J., MANÉ D., FRITZ D., KRISHNAN D., VIÉGAS F. B., WATTENBERG M.: Visualizing Dataflow Graphs of Deep Learning Models in TensorFlow. *IEEE Transactions on Visualization and Computer Graphics 24*, 1 (Jan. 2018), 1–12. doi:10.1109/TVCG.2017.2744878. 2

[WTS*20] WANG Z. J., TURKO R., SHAIKH O., PARK H., DAS N., HOHMAN F., KAHNG M., CHAU D. H.: CNN Explainer: Learning Convolutional Neural Networks with Interactive Visualization. *IEEE Transactions on Visualization and Computer Graphics* (2020). doi:10.1109/TVCG.2020.3030418. 3

[XEIM20] XIAO K., ENGSTROM L., ILYAS A., MADRY A.: Noise or signal: The role of image backgrounds in object recognition. *ArXiv preprint arXiv:2006.09994* (2020). URL: https://arxiv.org/abs/2006.09994. 4, 9

[YCN*15] YOSINSKI J., CLUNE J., NGUYEN A., FUCHS T., LIPSON H.: Understanding neural networks through deep visualization. In *Deep Learning Workshop, International Conference on Machine Learning (ICML)* (2015). URL: https://arxiv.org/abs/1506.06579. 3

[YLS*19] YIN D., LOPES R. G., SHLENS J., CUBUK E. D., GILMER J.: A Fourier Perspective on Model Robustness in Computer Vision. *arXiv:1906.08988 [cs, stat]* (Oct. 2019). arXiv: 1906.08988. URL: http://arxiv.org/abs/1906.08988. 2, 5

[ZHP*17] ZENG H., HALEEM H., PLANTAZ X., CAO N., QU H.: Cncomparator: Comparative analytics of convolutional neural networks. *arXiv preprint arXiv:1710.05285* (2017). URL: https://arxiv.org/abs/1710.05285. 2

[ZSLG16] ZHENG S., SONG Y., LEUNG T., GOODFELLOW I.: Improving the robustness of deep neural networks via stability training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 4480–4488. doi:10.1109/CVPR.2016.485. 2, 4

[ZZ19] ZHANG T., ZHU Z.: Interpreting Adversarially Trained Convolutional Neural Networks. *arXiv:1905.09797 [cs, stat]* (May 2019). arXiv: 1905.09797. URL: http://arxiv.org/abs/1905.09797. 2, 5, 7