Adversarial Defense Through Network Profiling Based Path Extraction

Yuxian Qiu[§] Jingwen Leng[§*] Cong Guo[§]
Quan Chen[§] Chao Li[§] Minyi Guo[§*] Yuhao Zhu[†]
[§]*Department of Computer Science and Engineering, Shanghai Jiao Tong University*
[†]*Department of Computer Science, University of Rochester*
{qiuyuxian,leng-jw,guocong}@sjtu.edu.cn,
{chen-quan,lichao,guo-my}@cs.sjtu.edu.cn, yzhu@rochester.edu

Abstract

Recently, researchers have started decomposing deep neural network models according to their semantics or functions. Recent work has shown the effectiveness of decomposed functional blocks for defending adversarial attacks, which add small input perturbation to the input image to fool the DNN models. This work proposes a profiling-based method to decompose the DNN models to different functional blocks, which lead to the effective path as a new approach to exploring DNNs' internal organization. Specifically, the per-image effective path can be aggregated to the class-level effective path, through which we observe that adversarial images activate effective path different from normal images. We propose an effective path similarity-based method to detect adversarial images with an interpretable model, which achieve better accuracy and broader applicability than the state-of-the-art technique.

1. Introduction

Deep learning (DL) has revolutionized the key application domains such computer vision [16], natural-language processing [35], and automatic speech recognition [1]. DL models have outperformed traditional machine learning approaches and even outperformed human beings. Although most of the current research efforts have been in improving the efficiency and accuracy of DL models, interpretability has recently become an increasingly important topic. This is because many DL-enabled or DL-based systems are mission-critical systems, such as ADAS [9] and online banking systems [11]. However, to date, there is no theoretical understanding of how DL models work, which is a significant roadblock in pushing DL into mission-critical systems.

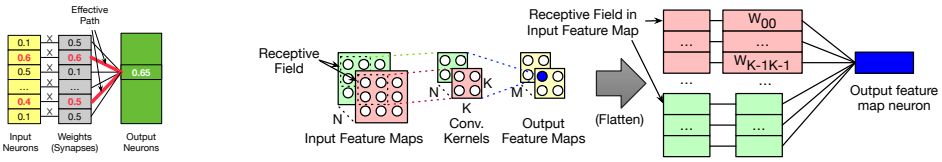
Owing to the lack of interpretability, DL models usually do not have a clear decision boundary and are vulnerable to the input perturbation. Researches have recently been proposed [29, 24, 18, 6], which can all successfully find a small perturbation on the input image to fool the DNN based classifier. There is also prior work that demonstrates the physical attack feasibility by putting a printed image in front of a stop sign to mislead a real DNN based traffic sign detector [10]. Last but not least, a DNN model often fails for inputs that are dramatically different from the training samples. For example, the classification model used in Tesla's autopilot system that incorrectly classified a white truck to cloud [12] and caused the crash accident.

To address the vulnerability challenge in DL models, this work proposes the *effective path* as a new approach to explore the internal organization of neural networks. The effective path for an image is a critical set of synapses and neurons that together lead to the final predicted class. The concept is similar to the execution path of a control-flow based program [3]. We propose an activation based back-propagation algorithm to extract the image's effective path, which preserves the critical information in the neural network and allows us to analyze the inner structures of DNNs.

The derived per-image effective path has direct aggregation capability. For example, we get per-class effective path by aggregating the effective path from all training images in the same class. We can then decompose the entire DNN into multiple components, each pertaining to an inference class. We perform similarity analysis and find the phenomenon called *path specialization* that different classes activate distinctive portions of the neural network in the inference task. On the basis of the observation, we analyze the path similarity between normal and adversarial images, we uncover that when an adversarial image successfully alters the prediction result by small perturbation, the network activates a significantly distinctive set of effective path compared to the training samples, which lays the foundation for defending the DNN using the effective path.

We propose to use the simple linear combination of an im-

* Jingwen Leng and Minyi Guo are co-corresponding authors of this paper.



(a) Linear layer. (b) Convolutional layer.

Figure 1: Examples that illustrate the process of using profiling to extract effective path.

age's per-layer effective path similarity to detect adversarial images. Our work can use a simpler linear model to outperform the state-of-the-art work [37] for six representative attack methods. Besides, we also show that our detection approach generalizes well to those attacks, meaning it can detect adversarial samples from methods that are not used in its training process, while the prior work does not have this level of generalization ability. Moreover, the overhead of using our approach is also much smaller (up to 500 ×) than prior work. In the end, we show that the effective path can not only be used for adversarial image detection but also for explaining the impact of the training process and network structure on the DNN's inference capability.

2. Effective Path via Profiling

Prior work [37] has proposed a method to extract critical data routing path (CDRP) for DNN models and demonstrated its usefulness of defending against adversarial samples. However, deploying it in practice has two major limitations. The first is the extraction process, which requires inserting control gates for each layer's output channel and learning those gates through retraining. This retraining process needs hyper-parameter tuning and takes a long time to process a single image. The second disadvantage is the extracted path representation is still high dimensional (1152 for AlexNet and 15104 for ResNet-50). The high dimensional representation weakens its interpretability and generalization ability to different adversarial attacks, which we will discuss later.

To overcome those limitations, we propose a novel method to extract the DNN's path information for an image. The method is inspired from path profiling in program analysis [3]: a program is represented in the form of control flow graph, where a node is a basic block and an edge is the control flow between basic blocks. Compilers use path profiling to identify the sequences of frequently executed basic blocks, i.e., execution paths in the program. A program's path profiling provides useful insights on its execution and facilitates the understanding of the program. In this work, we treat a neural network as a dataflow graph where a node is a neuron and an edge is a synapse (weight) between two neurons, and apply the profiling technique to extract its execution path, which we call as ***effective path*** to distinguish from the prior work. In the high level, both them represent the critical dataflow inside a DNN but our method doesn't rely on retraining and the derived representation is low-dimensional and generic.

2.1. Single Image Extraction

We first explain how to extract the effective path for a single image, denoted as $\mathcal{P} = (\mathcal{N}, \mathcal{S}, \mathcal{W})$, which represents the collection of critical neurons $\mathcal{N}$, synapses $\mathcal{S}$ and weights $\mathcal{W}$. It can be further broken down to the per-layer form $\mathcal{N} = (\mathcal{N}^1, \ldots, \mathcal{N}^L)$, $\mathcal{S} = (\mathcal{S}^1, \ldots, \mathcal{S}^L), \mathcal{W} = (\mathcal{W}^1, \ldots, \mathcal{W}^L)$, where $\mathcal{N}^l$ represents the important output neurons of layer l, while $\mathcal{S}^l$ and $\mathcal{W}^l$ represent important synapses and weights.

The extraction process starts at the last layer L and moves backward to the first layer. In the last layer L, only the neuron corresponding to the predicted class n_p^L is active and thus is included in the effective path, i.e., $\mathcal{N}^L = \{n_p^L\}$. The important weights form the minimum set of weights that can contribute more than θ ratio of the output neuron n_p^L. Equation 1 formalizes the process, where $\tilde{K}_p^L$ is a selected set of weight indices with pre-nonlinearity neuron n_p^L as the output, $w_{k,p}^L$ is the weight value, and n_k^{L-1} is the corresponding input neuron value (also the output neuron of layer $l-1$). To find the minimum $\tilde{K}_p^L$, we can rank the weight and input neuron pairs by the value of their product and choose the minimum number of pairs that contribute to more than threshold $\theta \times n_p^L$.

$$\min_{\tilde{K}_p^L} |\tilde{K}_p^L|, s.t. \sum_{k \in \tilde{K}_p^L} n_k^{L-1} \times w_{k,p}^L \geq \theta \times n_p^L \quad (1)$$

$$\mathcal{W}^L = \{w_{k,p}^L | k \in \tilde{K}_p^L\} \quad (2)$$

$$\mathcal{N}^{L-1} = \{n_k^{L-1} | k \in \tilde{K}_p^L\} \quad (3)$$

After deriving the weight indices set $\tilde{K}_p^L$, we can get the $\mathcal{W}^L$ set using Equation 2. Since the last layer is the fully connected layer and there is a one-to-one mapping between weight and synapses, $\mathcal{S}^L$ can also be derived. Meanwhile, since the output neurons of layer $L-1$ are the input neurons of layer L, it is straightforward to derive $\mathcal{N}^{L-1}$ in Equation 3. We then can repeat the process in Equation 1 for every active neuron in $\mathcal{N}^{L-1}$: each active neuron will result in a set of weights and their union form the $\mathcal{W}^{L-1}$. The process repeats backward until the first layer, and yields the whole neuron set $\mathcal{N}$, synapse set $\mathcal{S}$, and weight set $\mathcal{W}$ for the input image.

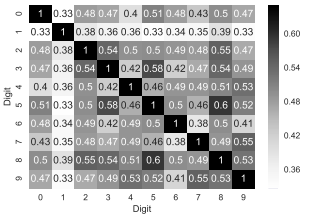


Figure 2: Class-wise path similarity in LeNet.

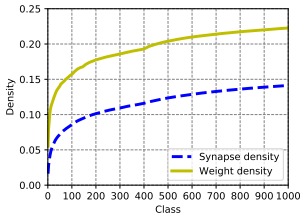


Figure 3: Density growth when merging per-class path.

Note that the above process addresses the fully connected layer. To process the convolutional layer, we need to convert it to FC layer according to each output neuron's receptive field as Fig. 1b shows. There are two caveats on handling the convolutional layer. First, solving of Equation 1 does not require the ranking of all input neurons but only the neurons in the receptive field of the output neuron. Second, there is no one-to-one mapping between synapse and weight because of weight sharing. As a result, multiple synapses can have the same active weights in the effective path.

2.2. Multi-Image Aggregation

The derived effective path is a binary mask that indicates whether a neuron or synapse contributes to the image inference. As such, we can simply aggregate effective paths from an image group, e.g. images of the same class, to obtain a larger effective path that provides a higher level perspective of the whole group. Aggregating the effective path of two images $\mathcal{P}(i)$ and $\mathcal{P}(j)$ is essentially taking the union of $\mathcal{N}$, $\mathcal{S}$ and $\mathcal{W}$ on each layer, represented by $\mathcal{P}(i) \cup \mathcal{P}(j) = (\mathcal{N}(i) \cup \mathcal{N}(j), \mathcal{S}(i) \cup \mathcal{S}(j), \mathcal{W}(i) \cup \mathcal{W}(j))$, where $\mathcal{N}(i) \cup \mathcal{N}(j) = (\mathcal{N}^1(i) \cup \mathcal{N}^1(j), \ldots, \mathcal{N}^L(i) \cup \mathcal{N}^L(j))$ ($\mathcal{N}$ and $\mathcal{W}$ are similar). This approach can create a meaningful representation for an image group without increasing its dimension. In contrast, the feature dimension of CDRPs [37] increase linearly with the number of images in the class because each element in CDRP is a continuous number between 0 and 10 instead of a binary mask.

In this work, we use two types of aggregated effective path for the neural network interpretation and defense. For the class-level perspective, we aggregate all correctly predicted training images from the class c, denoted by $\tilde{\mathcal{X}}_c$, to get the ***per-class effective path*** $\tilde{\mathcal{P}}_c = \bigcup_{x \in \tilde{\mathcal{X}}_c} \mathcal{P}(x)$; for the network-level perspective, we aggregate images from the whole training set $\tilde{\mathcal{X}}$ to get the ***overall effective path*** $\tilde{\mathcal{P}} = \bigcup_{x \in \tilde{\mathcal{X}}} \mathcal{P}(x)$.

Path Sparsity The derived overall effective path is highly sparse compared to the full model, indicating that critical information is reserved. We define the weight (synapse) density of the effective path $\mathcal{D}_\mathcal{W}$ ($\mathcal{D}_\mathcal{S}$) as the ratio of its weights (synapses) over the entire weights (synapses). They can be calculated in Equation 4, where $\mathbb{W}^l$ and $\tilde{\mathcal{W}}^l$ ($\mathbb{S}^l$ and $\tilde{\mathcal{S}}^l$) is the layer l's entire weight (synapse) set and weight (synapse) set in overall effective path, respectively.

$$\mathcal{D}_\mathcal{W} = \frac{\sum_{l=1}^{L} \left|\tilde{\mathcal{W}}^l\right|}{\sum_{l=1}^{L} |\mathbb{W}^l|}, \mathcal{D}_\mathcal{S} = \frac{\sum_{l=1}^{L} \left|\tilde{\mathcal{S}}^l\right|}{\sum_{l=1}^{L} |\mathbb{S}^l|} \tag{4}$$

We extracted the overall effective path for popular DNN models including LeNet-5 [20], AlexNet [17], ResNet-50 [15], Inception-v4 [36], and VGG-16 [34]. With $\theta = 0.5$, their synapse densities are 13.8%, 20.5%, 22.2%, 41.7%, 17.2%, respectively. Note that those values are calculated after aggregating all training samples (i.e. overall effective path). Prior work CDRP [37] reported similar sparsity values, which, however, was calculated based on an individual image because different images' CDRPs cannot be aggregated directly while effective path can. We also conduct an experiment which shows the DNN accuracy drops immediately when we start deactivating portions of the effective path, which indicates that the extracted path is not only sparse but also representative.

3. Effective Path Visualization

The per-class path dissects the network to different components and can be used to understand why the neural network can distinguish different classes and study the impact of changing the network structure. We perform the path similarity analysis among different classes, which leads to a finding called ***path specialization***. Different classes activate not only sparse but also a distinctive set of neurons and synapses for the inference task.

We first study the similarity of per-class effective paths. The similarity between class c_1 and c_2 is calculated by the Jaccard coefficient of their synapse set as in Equation 5.

$$J_{c_1,c_2} = J(\tilde{\mathcal{S}}_{c_1}, \tilde{\mathcal{S}}_{c_2}) = \frac{\left|\tilde{\mathcal{S}}_{c_1} \bigcap \tilde{\mathcal{S}}_{c_2}\right|}{\left|\tilde{\mathcal{S}}_{c_1} \bigcup \tilde{\mathcal{S}}_{c_2}\right|} \tag{5}$$

Fig. 2 shows the class-wise path similarity in LeNet, unveiling the existence of path specialization: the averaged similarity between two classes is low (around 0.5). On average, two classes activate about 50% common paths, as well as 50% distinctive paths. We can also conjecture that the degree of path specialization reflects the visual similarity between the two classes. For example, in Fig. 2, digit '1' has the highest degree of specialization (i.e., lowest path similarity against other digits): its average similarity with other classes is around 0.35 (compared to the 0.5 average value). The reason is most likely attributed to its unique shape. In contrast, digit '5' and '8' have the highest path similarity of 0.6, also likely owing to their similar shapes.

We observe the existence of the path specialization in other datasets and networks. Fig. 3 shows the path density

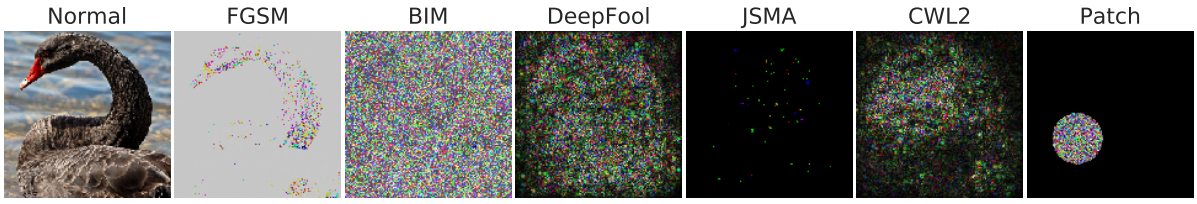
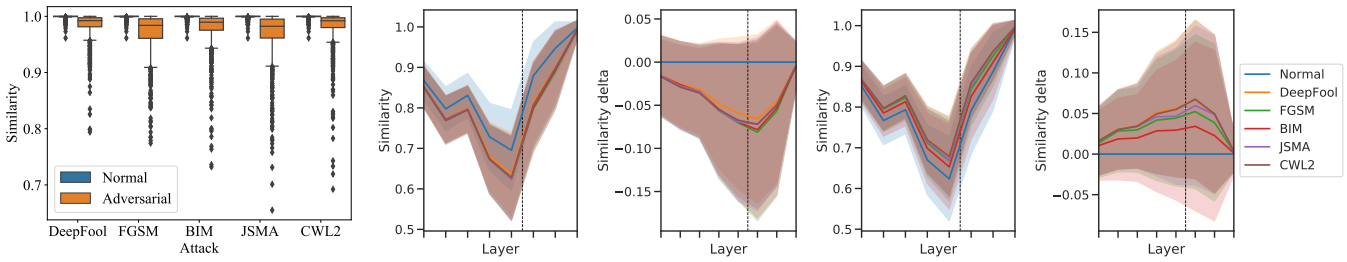


Figure 4: Normal example and perturbations from different attacks. The perturbations are enhanced by 100 times to highlight the differences.



(a) (b) (c) (d) (e)

Figure 5: (a) Path similarity for LeNet. (d-e): Distribution of per-layer similarity for AlexNet on ImageNet. Each line plot represents the mean of each kind of adversarial examples' similarity, with the same-color band around to show the standard deviation. The dashed line split convolutional layers and FC layers. (b): Rank-1 similarity. (c): Rank-1 similarity delta. (d): Rank-2 similarity. (e): Rank-2 similarity delta.

growth when merging per-class (ImageNet) paths for ResNet-50. The growth of both weight and synapse follow the same trend (weight density is greater owing to weight sharing). The density increases rapidly initially, indicating the high degree of path specialization. After 50 classes, the density still increases but at a much slower pace. This matches the class hierarchy in the ImageNet dataset, which has around 100 basic categories: different categories have a larger degree of path specialization while classes in the same categories have a smaller specialization degree.

In summary, we find the existence of path specialization phenomenon in trained DNNs, which unveils that DNNs activate different blocks when handling different classes. Inspired by the observation, we study the possibility of using the effective path to detect adversarial samples.

4. Adversarial Samples Defense

In this section, we study how to exploit the observed path specialization phenomenon to detect adversarial samples. Adversarial samples are generated by adding a small perturbation to normal images. The perturbation is small and imperceptible by human beings but can lead to an incorrect prediction of the neural network. We evaluate 6 different attacks (i.e. methods to generate misleading perturbation for a given input image), whose examples are shown in Fig. 4.

For each attack, we always choose the canonical implementation. We use Foolbox [30] implementations and its default parameters in version 1.3.2 for Fast Gradient Sign Method (FGSM) [13], Basic Iterative Method(BIM) [18], DeepFool [24], Jacobian-based Saliency Map Attack(JSMA) [28]. For Carlini and Wagner(C&W) attacks [6], we use the open-source code released by the paper authors. We use adversarial patch [4] implementation provided in CleverHans [27] and extend it to support AlexNet without modification to its settings.

We first explore the distribution of effective path for normal and adversarial examples, and show that adversarial images activate distinctive effective path to fool the DNN. Our further analysis indicates that effective path similarity provides a generic detection metric across all studied adversarial attacks. Based on the analysis result, we propose a low-dimensional and uniform metric to detect adversarial samples from different kinds of attacks.

4.1. Adversarial Samples Similarity Analysis

On the basis of path specialization, we study the similarity of the effective path between normal images and adversarial images. We introduce another similarity metric called ***image-class path similarity***, which indicates how many synapses in the image's effective path come from the predicted class's effective path. It can be calculated as $J_{\mathcal{P}} = J(\mathcal{S}, \mathcal{S} \cap \tilde{\mathcal{S}}_p) = |\mathcal{S} \cap \tilde{\mathcal{S}}_p| / |\mathcal{S}|$, where p is the image's predicted class, $\mathcal{S}$ is the synapse set of image effective path, and $\tilde{\mathcal{S}}_p$ is the synapse set of class p's effective path. Because the per-class effective path is far larger than the image's effective path, their Jaccard coefficient will be nearly zero. As such, the image-class path

similarity is essentially the Jaccard coefficient between the image's effective path and the intersection set of effective path between the image and predicted class.

Fig. 5a shows the distribution of image-class path similarity for both normal images and a rich set of adversarial images in MNIST for LeNet. The similarity values for normal images are almost all 1, and note that they are not used in the training and per-class path extraction. In contrast, the similarity values for adversarial images are mostly smaller than 1, indicating effective path as a great metric to distinguish between normal and adversarial images.

For deeper and more complicated DNNs, we breakdown the image-class path similarity metric to different layers. It can be calculated as $J^l_{\mathcal{P}} = |\mathcal{S}^l \cap \tilde{\mathcal{S}}^l_p|/|\mathcal{S}^l|$ for layer l. Fig. 5b compares the per-layer similarity for normal images (from test set) and adversarial images on AlexNet, and show that normal images demonstrate a higher similarity degree than adversarial images. We further calculate the similarity delta, which equals to the similarity value of a normal image minus the similarity value of its corresponding adversarial image. Fig. 5c shows that all adversarial attacks cause almost identical similarity decrease pattern, where the largest decrease occurs in the middle layers, i.e. boundary between convolutional layers and fully connected layers.

Recall that we extract the effective path starting from the predicted class, i.e. rank-1 class, which we call rank-1 effective path. We also study the rank-2 effective path which starts from the rank-2 class. Fig. 5d compares the rank-2 effective path similarity for normal and adversarial images. Different from the rank-1 effective path, adversarial images demonstrate a higher similarity degree than normal images. The reason is that the predicted rank-2 class for an adversarial image is often the rank-1 class of its corresponding normal image (i.e. without adding the perturbation). In comparison, the predicted rank-2 class for a normal image has no such relationship, and therefore has a lower degree of similarity than adversarial image. Moreover, different adversarial attack methods cause a similar pattern as Fig. 5e shows.

In summary, extending class-wise path similarity to the image-class case opens the door of using effective path to detect adversarial images: mainstream adversarial attacks modify the normally inactive path to fool the DNN and their impact indicates a uniform pattern. In the next subsection, we propose a simple and highly interpretable method to exploit these observations for detecting adversarial samples.

4.2. Defense Model

Based on the per-layer similarity analysis, we propose to use the rank-1 and rank-2 effective path similarity to detect adversarial samples. We study four different detection models, including linear model, random forest, AdaBoost, and gradient boosting. Among them, the linear model is the simplest one with the strongest interpretability. As we will show later, the linear model also achieves similar accuracy of other more complex models, proving that the selected input features (effective path similarity values) are strong indicators for detecting adversarial images.

Linear Model For the linear model, we propose ***jointed similarity*** as the defense metric. It can be calculated as $\tilde{J}_{\mathcal{P}} = \sum_{l=1}^{L} \omega^l J^l_{\mathcal{P}} - \sum_{l=1}^{L} \omega^{l'} {J^l_{\mathcal{P}}}'$, where $J^l_{\mathcal{P}}$ and ${J^l_{\mathcal{P}}}'$ are respectively rank-1 and rank-2 similarity for layer l, ω^l and $\omega^{l'}$ are their coefficients that satisfy $\omega^l \geq 0, \omega^{l'} \geq 0$. The joint similarity reflects the low rank-1 similarity degree and high rank-2 similarity degree of adversarial images. An image is detected as an adversarial image if its joint similarity is less than a threshold. The simple linear model avoids overfitting and offers strong interpretability.

We use LeNet-5 on MNIST, AlexNet on ImageNet and ResNet-50 v2 on ImageNet for evaluation. For each dataset, effective path extraction is performed on the overall training set with $\theta = 0.5$. For each model, adversarial examples from all evaluated attacks are aggregated, shuffled and split into 10% for the training of joint similarity's coefficients and 90% for defense performance evaluation. Notice that we only generate adversarial examples for the first test image in each class of ImageNet due to the significant computational cost. The training of joint similarity's coefficients is performed by SGD running 10000 epochs, with elastic-net regularization (l_1 ratio is 0.5) for sparsity.

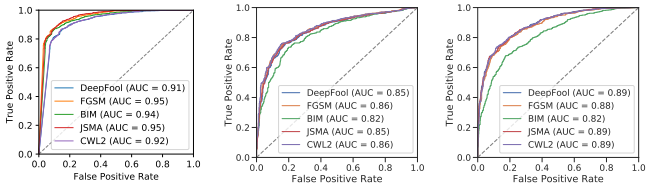
Other Models We also study to use other more complex models, including random forest, AdaBoost, and gradient boosting. These three approaches are also used to construct models based on CDRPs [37]. However, our input features for those approaches are a vector formed by each layer's rank-1 and rank-2 effective path similarity while prior work's input features have a much larger dimension (e.g. 1152 for AlexNet and 15104 for ResNet-50). For sake of consistency, we apply the same adversarial example preprocessing with the linear model. We use 100 estimators for random forest and gradient boosting, while AdaBoost is limited to 50 estimators. All unmentioned configurations of these models stay the same with the default values in scikit-learn v0.19.2.

5. Evaluation

In this section, we evaluate the adversarial sample detection accuracy based on effective path. We first focus on the highly interpretable linear detection model and show its detection performance on a wide range of different attacks, datasets, and models. We then compare our approach with prior work CDRP [37] and show that our approach achieves better accuracy, requires less training samples, and generalizes well to different types of adversarial attacks.

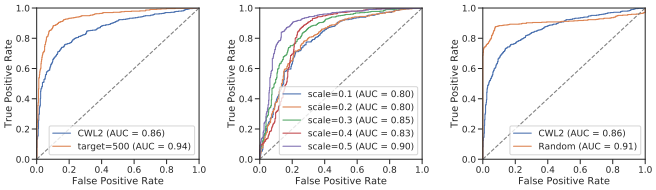
5.1. Linear Model

We first evaluate the detection accuracy of the linear model in Sec. 4.2 for a wide range of adversarial attacks.



(a) LeNet. (b) AlexNet. (c) ResNet-50.

Figure 6: Detection results for LeNet (on MNIST), AlexNet (on ImageNet), and ResNet-50 with joint similarity.



(a) Targeted attack. (b) Universal perturbation attack. (c) Unrecognizable examples.

Figure 7: Linear model detection results for AlexNet on different attack methods.

Non-targeted Attacks We evaluate non-targeted attacks, which are free to use any class as the adversarial image's inference result, with three different norms: FGSM and BIM with l_∞ norm, DeepFool and C&W $l2$ (CWL2) attack with l_2 norm, and JSMA with l_0 norm. For LeNet, we achieves an area under the curve (AUC) value up to 0.95 in Fig. 6a. Even the lowest AUC value is 0.92, because of significant path similarity distinction between adversarial and normal images of MNIST. On ImageNet, we achieve AUC of 0.85~0.86 for AlexNet and AUC of 0.88~0.89 for ResNet-50, which has more layers to provide richer information for detection, leading to better accuracy. The BIM has a low AUC value of 0.82. The reason is that BIM iteratively modifies all pixels (Fig. 4), which makes its rank-2 effective path behave slightly different from other attacks.

Targeted Attack Targeted attacks are designed to mislead the prediction to a specific target class. Fig. 7a shows the result of evaluating targeted C&W l_2 attack for AlexNet. We achieve AUC of 0.94, which is better than the non-targeted version. It is reasonable since the targeted attack's stricter constraint for target class requires larger perturbation, which eases our detection.

Universal Perturbation Attack Universal perturbation attacks generate perturbations that fool models on a large range of examples. Adversarial Patch [4] is an attack that generates universal perturbations in the form of image patches, which is robust against patch transformations such as translations, rotations or scaling. The result of adversarial patches in Fig. 7b indicates that the detection becomes more accurate when the patch becomes larger. Our method can reach AUC of 0.9 when the patch scale relative to image size rises to 0.5.

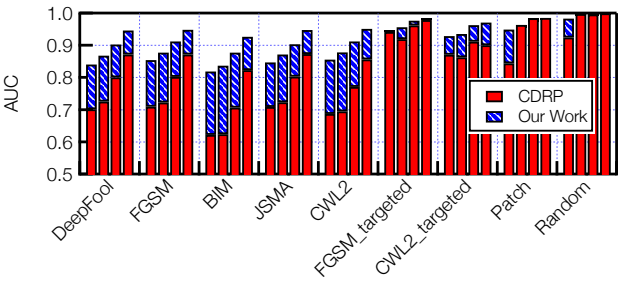


Figure 8: Detection accuracy comparison between effective path and CDRP. Note that the four bars in an attack type represent linear model, AdaBoost, gradient boosting, and random forest in order. The blue bars indicate the AUC delta between our work and CDRP, no matter which is higher. Our work outperforms CDRP except on the linear model for the patch and random attack.

Unrecognizable Examples Adversarial examples are usually human-recognizable, however, unrecognizable images can also fool neural networks [26]. We find that effective path can also be used to detect unrecognizable examples by evaluated on LeNet and AlexNet. For LeNet, our detector can recognize 93.85% randomly generated images. For AlexNet, our method achieves AUC of 0.91 as shown in Fig. 7c. In this sense, effective path offers the DNNs the ability to identify its recognizable inputs' distribution.

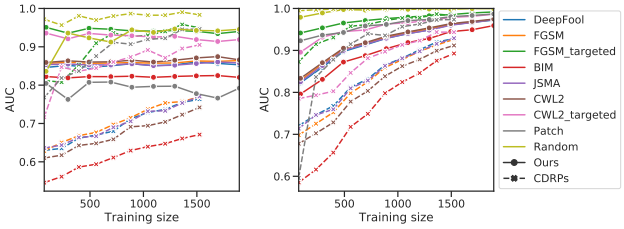
To summarize, the simple linear model constructed with effective path achieves high detection accuracy without requiring attack-specific knowledge.

5.2. Comparison with Prior Work

We now compare effective path based detection with prior work CDRP [37] with the linear model as well as three different kinds of models described in Sec. 4.2.

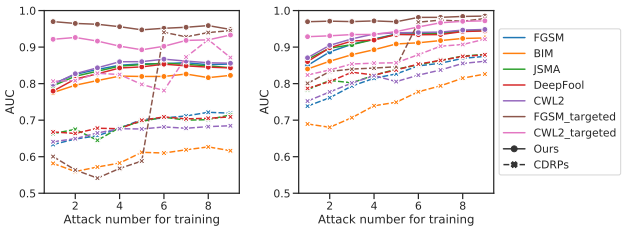
Detection Accuracy Fig. 8 compares the detection accuracy between effective path based models and CDRP based models. For both approaches, we find that random forest performs the best while the linear model performs worst among all models. However, the effective path approach has a much smaller gap between random forest and linear model than the CDRP approach. With the exception of patch and random attack, the effective path based linear model can outperform the CDRP based random forest model. In particular, the accuracy improvements for our approach are much more significant for the first five attack methods that are non-targeted and smaller for the two targeted attack method in the middle. Note that the effective path based linear model performs slightly worse on the patch and random attack, which generate much different perturbation patterns (see Fig. 4).

Training Size We also study how the size of the training set impacts the detection accuracy. We choose the linear model and random forest model and gradually increase their



(a) Linear model. (b) Random forest.

Figure 9: Impact of training set size on the AUC.



(a) Linear model. (b) Random forest.

Figure 10: Impact of attack number in the training set.

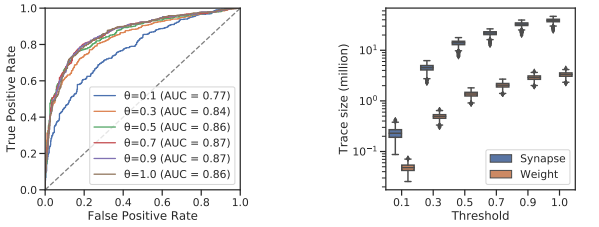
training set size. Fig. 9 compares our approach with the CDRP approach. For the linear model, our approach stabilizes with a small number of training samples (around 100 images) while the CDRP requires much larger training set size. For the random forest model, both approaches require a larger training set while our approach is less sensitive because our input feature is low dimensional and effective.

Generalizability Generalizability measures a defense's ability to withstand unknown attacks. To study the generalizability of our detection model, we perform a control experiment: we gradually add the adversarial samples from different attack types for training the detection model and observe the model detection accuracy all on attack types. Fig. 11 shows the experiment results where we add the adversarial samples in the order of legend shown in the right. For both linear model and random forest model, our work generalizes well to unseen attacks because effective path captures their common behavior. The CDRP based linear model performs worse for all non-targeted attacks, and its accuracy on targeted attack (FGSM_targeted and CWL2_targeted) shows an abrupt increase once incorporating the corresponding samples for training the model.

5.3. Sensitivity Study

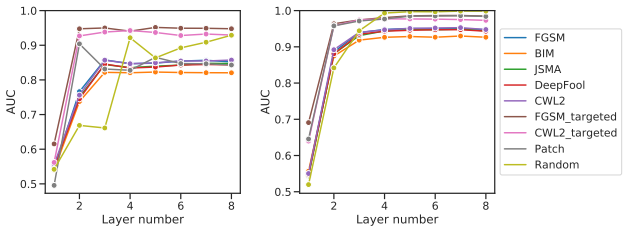
After demonstrating the accuracy of using effective path to detect adversarial samples, we now perform sensitivity study on its parameters, including the θ in Equation 1 and the number of extracted layers. Our results further unveil optimization opportunity to make effective path more practical.

Parameter Sensitivity The first tunable parameter of effective path extraction is θ. We test C&W l_2 attack with θ value varying from 0.1 to 1.0 in Fig. 11a. The detection performance remains almost unchanged when θ is in range of



(a) ROC for C&W l_2 attack with different θ. (b) Effective path size with different θ.

Figure 11: Effective path θ sensitivity study.



(a) Linear model. (b) Random forest.

Figure 12: Effective path layer number impact on AUC.

Table 1: Effective path extraction time (second).

Method	Effective Path (Full)	Effective Path (Partial)	CDRP
AlexNet	1.43 ± 0.09	0.43 ± 0.17	106.4 ± 5.2
ResNet-50	68.32 ± 2.43	0.83 ± 0.21	406.3 ± 6.3

0.5 and 1.0, and decreases from $\theta = 0.3$. Fig. 11b shows that the effective path size under $\theta = 0.3$ decrease by one order of magnitude compared with $\theta = 1.0$, with slightly lower detection accuracy. We choose $\theta = 0.5$ as default value to save storage space and improve extraction performance without accuracy loss.

Layer Sensitivity Another tunable parameter of effective path extraction is the number of layers as it is extracted layer by layer. We perform experiments to study the layer number's impact on the adversarial sample detection accuracy and show the result in Fig. 12. Note that we extract the effective path starting from the last layer and the last three layers are fully connected layers in AlexNet. For both linear model and random forest, we observe that the AUC performance for all attacks except random attack saturates after three layers, i.e. FC layers. The random attack detection accuracy saturates after four layers, i.e. one additional CONV layer.

With the layer sensitivity insight, we can extract effective path for just enough layers instead of the full network, which can significantly reduce the extraction time. Tbl. 1 compares the extraction time for AlexNet and ResNet-50. Extracting the full effective path for the entire network is still much less expensive (70× for AlexNet and 6× for ResNet-50) than extracting CDRP which requires the retraining process. Moreover, extracting the partial effective path can lead to even faster process time, which translate to 240× and 500×

Table 2: Comparison with other defenses.

Type	Defense	l_0	l_2	l_∞	Attack	Generalizability	Scale
Detector	**Effective Path**	**Y**	**Y**	**Y**	**all discussed**	**strong**	**ImageNet**
	[23]	-	**Y**	**Y**	(non-)targeted	weak	CIFAR-10
	[22]	-	**Y**	**Y**	(non-)targeted	-	CIFAR-10
	[37]	-	-	**Y**	targeted	-	**ImageNet**
Adversarial Training	[21]	-	**Y**	**Y**	(non-)targeted	weak	CIFAR-10
	[25]	-	-	**Y**	(non-)targeted	weak	CIFAR-10
Input Transformation	[14]	-	**Y**	**Y**	(non-)targeted	-	**ImageNet**
	[5]	-	-	**Y**	(non-)targeted	-	CIFAR-100
Randomization	[38]	-	**Y**	**Y**	(non-)targeted	-	**ImageNet**
	[8]	-	-	**Y**	(non-)targeted	-	CIFAR-10
Generative Model	[32]	**Y**	**Y**	**Y**	(non-)targeted + random	**strong**	MNIST
	[31]	-	**Y**	**Y**	(non-)targeted	-	MNIST

time reduction compared to CDRP extraction.

In summary, effective path enables the use of the highly interpretable linear model to detect a broad range of adversarial attacks, and can achieve great accuracy on different datasets and models. Compared to prior work CDRP [37], our approach achieves better accuracy, requires less training samples, and generalizes well to different adversarial attacks.

6. Related Work

To compare our defense method with prior work, we first categorize various defenses methods to the five types listed in Tbl. 2. Since almost all the compared work reported a similar detection accuracy (AUC value 0.9 - 0.95), we focus the comparison on the comprehensiveness, attack method, generalizability, and scale of their evaluation. The "-" in the table indicates that there are not enough details or experimental results to deduce an appropriate conclusion.

Detector Our work fits in the detector category, which does not require any modification to inputs, models, or training process. Prior work [23] trained a DNN from network activations to detect adversarial examples. The detector subnetwork doesn't generalize well across different attack parameters or attack types because the activation values are highly attack-specific, which motivates [22] to propose MagNet. MagNet uses a reformer to move adversarial examples to normal examples' manifold. However, [7] shows that MagNet can be defeated by a little increase of perturbation.

The closest work to ours is [37], which uses the importance coefficients of different channels in the network (named critical data routing paths, abbr. CDRPs) to detect adversarial examples. However, CDRPs do not have aggregation capability as a single channel can have different significance values for different images. As such, CDRPs fail to defend non-targeted attacks and have weak generalizability. In comparison, we use the effective path, which is essentially a binary value for each neuron/synapse, and therefore can be directly aggregated. Our method generalizes well for different attacks and provides strong transferability.

Adversarial Training Adversarial training requires additional training step to protect the DNNs. It has two known disadvantages: it is difficult to perform in the large-scale dataset like ImageNet [19], at the same time easy to overfit to the trained kinds of adversarial examples. Even adversarial training proposed by [21], considered as the only effective defense among white-box-secure defenses at ICLR 2018 [2], is found overfitting on the trained l_∞ metric [33].

Input Transformation Many image transformations like rescaling, bit-depth reduction and compression can disturb attacks and increase the perturbation lower bound, with the sacrifice of classification accuracy. This kind of defense method works less well for patch-based attacks and does not provide the ability to filter unrecognizable examples.

Randomization Randomization-based defense methods apply random modifications to model weights. They can increase required distortion by forcing attacks to generate transferable adversarial examples over a series of possible modified models. However, they also stochastically alter the prediction results, leading to the overhead of more forward passes or retraining steps.

Generative Model Generative model based defenses change the classification model. They project the inputs onto the manifold before classification. [32] propose a classification model that shows good generality and transferability on MNIST, but its performance on large dataset like ImageNet is still obscure. GAN-based defenses are also hard to apply in ImageNet scale due to its computational cost.

7. Conclusion and Future Work

In this work, we propose a novel profiling based method to extract the deep neural network's (DNN) path information when inferring an image. This method does not modify the DNN structure and can extract meaningful path information that represents the critical dataflow inside the DNN. We study how to use the extracted path information to decompose a DNN model into different functional blocks corresponding to different inference classes. Through analysis, we find that adversarial images can activate functional blocks different from normal images to fool the DNN's prediction results because all the blocks are connected. We propose a defense method that only uses the information from the training set and the image itself, without requiring any knowledge of a specific attack. The defense method achieves high accuracy and broad coverage of mainstream attacks.

Besides adversarial defense, the effective path can also be used to understand the DNN's working mechanism. In the appendix, we report our preliminary result on how the training process and different DNN topology affects the effective path density and similarity. We believe that the functionality based decomposition is a promising direction for understanding DNNs.

Acknowledgments. This work is sponsored in part by the National Basic Research 973 Program of China (No. 2015CB352403), the National Natural Science Foundation of China under Grant Nos. 61702328 and 61602301, and partially funded by Microsoft Research Asia Collaborative Research Grant.

References

[1] Ossama Abdel-Hamid, Abdel rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22:1533–1545, 2014.

[2] Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 274–283, 2018.

[3] Thomas Ball and James R. Larus. Efficient path profiling. In *Proceedings of the 29th Annual ACM/IEEE International Symposium on Microarchitecture*, MICRO 29, pages 46–57, Washington, DC, USA, 1996. IEEE Computer Society.

[4] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *CoRR*, abs/1712.09665, 2017.

[5] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *International Conference on Learning Representations*, 2018.

[6] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. *CoRR*, abs/1608.04644, 2016.

[7] Nicholas Carlini and David A. Wagner. Magnet and "efficient defenses against adversarial attacks" are not robust to adversarial examples. *CoRR*, abs/1711.08478, 2017.

[8] Guneet S. Dhillon, Kamyar Azizzadenesheli, Jeremy D. Bernstein, Jean Kossaifi, Aran Khanna, Zachary C. Lipton, and Animashree Anandkumar. Stochastic activation pruning for robust adversarial defense. In *International Conference on Learning Representations*, 2018.

[9] Ivan Dynov. Is Deep Learning Really the Solution for Everything in Self-Driving Cars? `http://bit.ly/ivan_dynov_adas`, 2016.

[10] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1625–1634, 2018.

[11] Ugo Fiore, Alfredo De Santis, Francesca Perla, Paolo Zanetti, and Francesco Palmieri. Using generative adversarial networks for improving classification effectiveness in credit card fraud detection. *Information Sciences*, 2017.

[12] Jordan Golson. Tesla driver killed in crash with Autopilot active, NHTSA investigating. `http://bit.ly/tesla_crash`, 2016.

[13] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2014.

[14] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. Countering adversarial images using input transformations. In *International Conference on Learning Representations*, 2018.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 1106–1114, 2012.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[18] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016.

[19] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *CoRR*, abs/1611.01236, 2016.

[20] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[21] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *CoRR*, abs/1706.06083, 2017.

[22] Dongyu Meng and Hao Chen. Magnet: A two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 135–147, New York, NY, USA, 2017. ACM.

[23] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. *CoRR*, abs/1702.04267, 2017.

[24] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. *CoRR*, abs/1511.04599, 2015.

[25] Taesik Na, Jong Hwan Ko, and Saibal Mukhopadhyay. Cascade adversarial machine learning regularized with a unified embedding. *CoRR*, abs/1708.02582, 2017.

[26] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 427–436, 2015.

[27] Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Alexey Kurakin, Cihang Xie, Yash Sharma, Tom Brown, Aurko Roy, Alexander Matyasko, Vahid Behzadan, Karen Hambardzumyan, Zhishuai Zhang, Yi-Lin Juang, Zhi Li, Ryan Sheatsley, Abhibhav Garg, Jonathan Uesato, Willi Gierke, Yinpeng Dong, David Berthelot, Paul Hendricks, Jonas Rauber, and Rujun Long. Technical report on the cleverhans v2.1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768*, 2018.

[28] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. *CoRR*, abs/1511.07528, 2015.

[29] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pages 1–18, New York, NY, USA, 2017. ACM.

[30] Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131*, 2017.

[31] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*, 2018.

[32] Lukas Schott, Jonas Rauber, Wieland Brendel, and Matthias Bethge. Robust perception through analysis by synthesis. *CoRR*, abs/1805.09190, 2018.

[33] Yash Sharma and Pin-Yu Chen. Attacking the madry defense model with l_1-based adversarial examples. *CoRR*, abs/1710.10733, 2017.

[34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[35] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112, 2014.

[36] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016.

[37] Yulong Wang, Hang Su, Bo Zhang, and Xiaolin Hu. Interpret neural networks by identifying critical data routing paths. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8906–8914, 2018.

[38] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. In *International Conference on Learning Representations*, 2018.