NoiseCAM: Explainable AI for the Boundary Between Noise and Adversarial Attacks

Wenkai Tan [1a], Justus Renkhoff [2b], Alvaro Velasquez[3c], Ziyu Wang[6g], Lusi Li[6h], Jian Wang[4e]
Shuteng Niu[5f], Fan Yang [1d], Yongxin Liu [1d], Houbing Song[2b]
[1]Embry-Riddle Aeronautical University, FL 32114 USA, [2]University of Maryland, Baltimore County, MD 21250 USA
[3]University of Colorado Boulder, CO 80309 USA, [4]University of Tennessee at Martin, TN 38237 USA
[5]Bowling Green State University, OH 43403 USA, [6]Old Dominion University, VA 23529 USA
[a]tanw1@my.erau.edu, [b]{justusr1, songh}@umbc.edu, [c]alvaro.velasquez@colorado.edu, [d]{LIUY11, YANGF1}@erau.edu,
[e]jwang186@utm.edu, [f]sniu@bgsu.edu [g]zwang007@odu.edu, [h]lusili@cs.odu.edu

2023 IEEE International Conference on Fuzzy Systems (FUZZ) | 979-8-3503-3228-5/23/$31.00 ©2023 IEEE | DOI: 10.1109/FUZZ52849.2023.10309766

***Abstract*—Deep Learning (DL) and Deep Neural Networks (DNNs) are widely used in various domains. However, adversarial attacks can easily mislead a neural network and lead to wrong decisions. Defense mechanisms are highly preferred in safety-critical applications. In this paper, firstly, we use the gradient class activation map (GradCAM) to analyze the behavior deviation of the VGG-16 network when its inputs are mixed with adversarial perturbation or Gaussian noise. In particular, our method can locate vulnerable layers that are sensitive to adversarial perturbation and Gaussian noise. We also show that the behavior deviation of vulnerable layers can be used to detect adversarial examples. Secondly, we propose a novel NoiseCAM algorithm that integrates information from globally and pixel-level weighted class activation maps. Our algorithm is highly sensitive to adversarial perturbations and will not respond to Gaussian random noise mixed in the inputs. Third, we compare detecting adversarial examples using both behavior deviation and NoiseCAM, and we show that NoiseCAM outperforms behavior deviation modeling in its overall performance. Our work could provide a useful tool to defend against certain types of adversarial attacks on deep neural networks.**

I. Introduction

Artificial intelligence and deep learning have great potential to tackle a wide range of engineering and scientific challenges. For example, Deep Learning have been widely applied in medical imaging to reduce delay and human efforts in disease diagnoses [1]. In addition, deep neural networks are extremely good at dealing with complicated scenarios, making DL a promising tool for safety and security domains where complicated patterns are difficult to define explicitly [2], [3].

However, the robustness and reliability of Deep Neural Networks (DNNs) have raised concerns among researchers. Studies have shown that these networks, such as the VGG-16 model [4], can be fooled by manipulated images that are undetectable to the human eye [5]–[7]. In such cases, the altered pixels possess pseudo-random properties, leading to doubts about the reliability and confidence of DNNs in environments with natural Gaussian noise [8], [9].

For mitigation, on the one hand, some solutions are proposed to increase the robustness of DNNs by augmenting their training with perturbed samples or introducing a robust loss term [10]–[12]. These approaches encourage DNNs to treat a slightly perturbed image as its origin. In this context, finding and incrementally training DNNs with adversarial examples is analogous to fuzzy testing. Some representative frameworks have been proposed, such as DLFuzz [13], DeepXplore [5], DeepHunter [14], and TensorFuzz [15]. In particular, in one of our previous works of a white-box fuzz testing framework, DLFuzz, we derive adversarial perturbation by misleading the neural classifier and maximizing the neuron coverage simultaneously.

One common feature of these adversarial example-enabled neural network fuzzy testing frameworks is that they not only discover adversarial examples but also try to maximize the activation rates of neurons, also known as neuron coverage [16], [17]. The neuron coverage ratio describes how many neurons are activated during a prediction. DLFuzz [13] adapts this concept from DeepXplore [5] and tries to optimize this metric by generating adversarial examples and maximizing the prediction difference between the original and the adversarial images. Higher neuron coverage usually contributes positively to the robustness of DNNs.

However, training DNNs on perturbed samples incrementally is computationally expensive and can reduce classification accuracy [18]. Moreover, it is difficult to find a balance between accuracy and adversarial robustness. Defending existing DNNs against adversarial examples without specifically retraining them is preferred. Defense-GAN [19] trains a defensive generative adversary network (GAN) on natural inputs. A noticeable behavior deviation can be detected when adversarial examples are fed into the defensive GAN. Instead of modeling the inputs directly, I-Defender [20] models the output distributions of fully connected hidden layers for each class. Then it uses statistical testing to reject adversarial examples. Adversarial perturbations can be treated as additive noise. Therefore, similar approaches, such as denoising autoencoders,

can be used to purify the input of DNNs [21]–[24].

Most current efforts view DNNs as black-box models and fail to analyze adversarial attacks in an explainable manner. In our prior work [25], we used ImageNet database [26] and then manipulate them with DLFuzz [13] to generate white-box adversarial attacks examples. We also used Grad-CAM [27] enabled class activation maps in VGG-16 [4]. We compare the model's responses to adversarial and statistically similar Gaussian noise mixed images, revealing the potential of class activation maps for detecting adversarial examples. This paper examines the potential of modeling behavior deviation in vulnerable VGG-16 layers to identify adversarial examples and introduces NoiseCAM, a novel method that integrates GradCAM++ [28] and LayerCAM [29] to detect adversarial examples in an interpretable manner. NoiseCAM is more effective than our behavior modeling method in detecting adversarial examples. The contributions of our work are as follows.

- We propose an interpretable method for detecting compromised layers in a neural network under adversarial attacks by modeling behavior deviation. Our findings show that the deeper layers in VGG-16 are more sensitive to adversarial perturbations and Gaussian random noise.
- We investigate the use of behavior deviation modeling in vulnerable convolution layers under non-adversarial scenarios to obtain decision thresholds for detecting adversarial examples, but found it to be unreliable.
- We propose NoiseCAM, a new algorithm for detecting adversarial attacks on DNNs in an interpretable manner. Our experiments demonstrate that NoiseCAM is highly sensitive to adversarial noise while being impervious to Gaussian random noise even when they have statistically similar properties.

The remainder of this paper is organized as follows: A literature review of related work is presented in Section II. We present the methodology in Section III. Evaluation and discussion are presented in Section IV and conclusions in Section V.

II. Related Work

Fuzzy Testing Enabled Adversarial Example Generation on DNNs: Given a network model, an adversarial example is generated by introducing small and imperceptible perturbations to a given seed image to cause misclassifications. There are several procedures, such as FGSM [30], to generate adversarial examples. In general, this is a white-box fuzzy testing procedure, one has to know the parameters of the target neural classifier and then solve the optimization problem so that the perturbations should maximize the classification loss and minimize the difference between the perturbed and original image. Defending DNNs against adversarial examples can increase the robustness of the model.

Explainalbe AI (XAI) for Visual Explanations of DNNs: Explainable AI (XAI) methods such as Gradient-weighted Class Activation Maps (Grad-CAMs) [31] and Local Interpretable Model-Agnostic Explanations (LIME) [32] aim to improve the interpretability of DNNs. They have the ability to explain the prediction of black-box models and can therefore improve their trustworthiness. Grad-CAM calculates a heatmap highlighting different areas of an image in different colors. These colors visualize how much an area positively contributes to a certain prediction. LIME highlights pixels that contribute positively or negatively based on a threshold. These pixels can be represented in different colors, as in Grad-CAM. With Grad-CAM, it is possible to identify not only which areas contribute positively and negatively to a certain prediction, but also how much an area influences a decision [33]. Numerous efforts use these methods to increase confidence in machine learning models [34] [35] [36].

Fig. 1. The effect of an adversarial noise removal filter on VGG-16 model classification accuracy. We apply a Gaussian Blur filter with radius 1.5 to three distinct scenarios: a) an adversarial input, b) an input with Gaussian noise, and c) the original input (labeled as Space Shuttle). All three inputs subjected to the Gaussian Blur filter are incorrectly predicted by the VGG-16 network.

Countermeasures for Adversarial Attacks: The defense against adversarial attacks has been extensively studied with several methods summarized in [37]. The traditional method to defend test-time evasion (TTE) attacks is to remove all potential perturbations for the input images, representative methods are: Principal Component Analysis (PCA), blur filters, and autoencoders. However, these methods can also potentially decrease the accuracy of the DNN classifier.

For example, in Figure 1, we applied a Gaussian Blur filter with radius 1.5 to three different scenarios to protect the VGG-16 model against adversarial examples: a) an adversarial input, b) input with Gaussian noise and c) the original input. All three inputs with Gaussian Blur filter are erroneously predicted by the VGG-16 network. One reason is that DNN classifiers compress the input image into a relatively low resolution; e.g., VGG-16 compresses the input into 224x224. When we apply a blur filter or PCA to input images, the loss of information will lead to incorrect predictions. Although traditional methods can easily eliminate the perturbation, it also affect the classification results.

III. Methodology

A. Problem Definition

In order to safeguard DNNs against adversarial attacks, it is crucial to understand how these examples lead to misclassifications. To achieve this, we utilize XAI techniques to analyze the classification decision process of the VGG-16 model, layer by

layer. Figure 2 provides a concise overview of our investigation method.

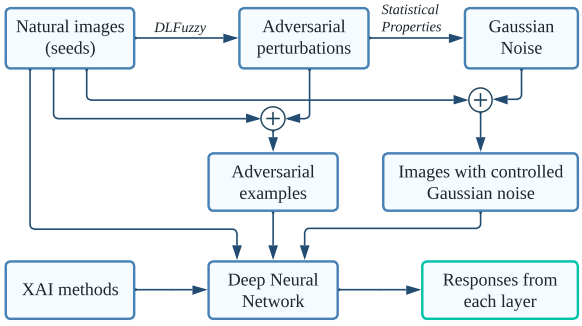


Fig. 2. Data pre-processing and analysis procedure [25]

B. Data Preparation

We used randomly selected images as seeds from the ImageNet validation dataset, and we derive their adversarial perturbations, or adversarial noise, denoted as:

$$\mathbf{N}_a = \mathrm{ADV}(\mathbf{S}, \varepsilon) \tag{1}$$

where $\mathrm{ADV}(\cdot)$ is the adversarial perturbation function, ε is the strength of the perturbation, and $\mathbf{S}$ denotes a seed image. In this work, DLFuzz is configured to generate adversarial noise and maximize neuron coverage simultaneously. We generate the same amount of Gaussian noise $\mathbf{N}_g$, defined as random noise, with statistical properties similar to adversarial perturbations. Consequently, we use the expected value, standard deviation and shape of $\mathbf{N}_a$ to generate $\mathbf{N}_g$ as:

$$\mathbf{N}_g \sim \mathcal{N}(\mu_a, \Sigma_a) \tag{2}$$

where $\mathcal{N}(\cdot)$ denotes a normal distribution, μ_a denotes the expected value, and Σ_a denotes the standard deviation. We let $\mu_a = \overline{N_a}$ and $\Sigma_a = \sigma(N_a)$. Based on this distribution, we generate random noise $\mathbf{N}_g$ represented as a matrix with the same shape as $\mathbf{N}_a$. In this way, the augmented input $\mathbf{I}$ to the DNN becomes:

$$\mathbf{I} = \{\mathbf{S}, \mathbf{S} + \mathbf{N}_a, \mathbf{S} + \mathbf{N}_g\} \tag{3}$$

where $\mathbf{S} + \mathbf{N}_a$ and $\mathbf{S} + \mathbf{N}_g$ are adversarial examples and the noisy version of the seed image.

C. Adversarial Example Generation

We use DLFuzz [13] to calculate perturbations and apply it to images from the ImageNet [38] dataset. Mathematically, we are conducting a white-box attack on the neural network. The procedure of deriving additive perturbations is regarded as an optimization problem:

$$\underset{\mathbf{N}_a,\ ||\mathbf{N}_a|| \le \delta}{\operatorname{argmax}} \left[\sum_{i=1}^{K} c_i - c + \lambda \sum_{i=0}^{m} n_i \right] \tag{4}$$

where δ restricts the magnitude of the adversarial perturbation, c is the prediction on a given seed image, c_i is one of the top K candidate predictions, n_i is one of the activation values in the m selected neurons, and λ is a constant to balance between activating more neurons and obtaining adversarial examples.

DLFuzz manipulates the perturbations to mislead a tested network to make wrong predictions and simultaneously activate the selected neurons. DLFuzz ensures that adversarial examples are generated more efficiently during a test by maximizing neuron coverage. This is based on the assumption that this will trigger more logic in the network and thus provoke and detect more erroneous behaviors [5].

After obtaining an effective adversarial perturbation for each seed, we amplify the perturbations with a total of five different ratios, defined as the perturbation strength. These are 25%, 50%, 100%, 200% and 400%.

The additive perturbation is then added to the original image to form adversarial examples. Unlike the Gaussian random noise that is evenly distributed on the image, adversarial perturbations are usually presented clustered with a certain pattern. These adversarial perturbations can lead to the misactivation of the wrong convolutional filters and can even be visualized on class activation maps.

D. Network Behavior Deviation Modeling

For a given convolution layer l, its response to an input image, e.g., a seed image, can be visualized using its grad-CAM heatmap, defined as:

$$\mathbf{H}(\mathbf{S}) = R\left[sign\left(\frac{\mathbf{J}(\mathbf{S})}{\partial \theta_{\mathbf{l}}} \right) \right] \tag{5}$$

where $\mathbf{J}(\mathbf{S})$ denotes the classification score of a selected category $\mathbf{S}$ and $\theta_{\mathbf{l}}$ denotes the parameters of layer l. The function $R(\cdot)$ denotes the operation that reshapes and interpolates the derived gradients to the same dimension and size as $\mathbf{S}$. The derivative values display the focal areas of l in $\mathbf{S}$. Our hypothesis is that the focal regions under attack in the neural network could differ under adversarial and natural images. Moreover, we use cosine similarity to calculate the degree of behavior deviation as:

$$\mathbf{G}[\mathbf{S_1}, \mathbf{S_2}] = \frac{vec(\mathbf{H}(\mathbf{S_1})) \cdot vec(\mathbf{H}(\mathbf{S_2}))}{\|vec(\mathbf{H}(\mathbf{S_1}))\| \cdot \|vec(\mathbf{H}(\mathbf{S_2}))\|} \tag{6}$$

where $\mathbf{H}(\mathbf{S_1})$ and $\mathbf{H}(\mathbf{S_2})$ denote Grad-CAM heatmaps of different images. These are vectorized and normalized for similarity calculation. For a specific layer and a seed image $\mathbf{S}$, we calculate and compare two types of behavior deviation.

$$\mathbf{D}_a = \mathbf{G}[\mathbf{S}, \mathbf{S} + \mathbf{N}_a] \tag{7}$$

$$\mathbf{D}_g = \mathbf{G}[\mathbf{S}, \mathbf{S} + \mathbf{N}_g] \tag{8}$$

Our observations revealed that both Gaussian noise and adversarial perturbations can cause behavior deviations, while Gaussian noise does not easily cause a wrong classification result. For each layer, we used the median value of its degree of behavior deviation under Gaussian noise as the reference threshold to quantify whether it is compromised, i.e., whether its behavior deviates more severely than when it is under the same amount and strength of Gaussian noise.

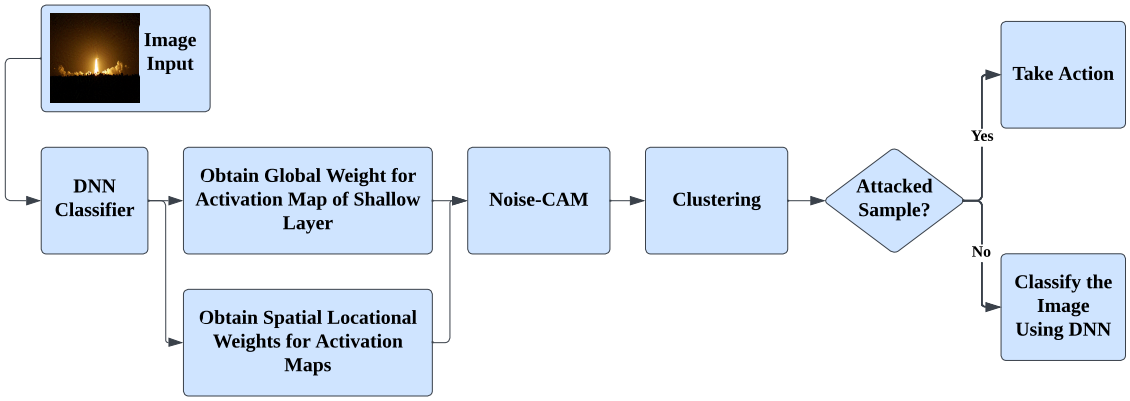


Fig. 3. Detecting adversarial examples with NoiseCAM

We also employ behavior deviation to detect adversarial examples. Specific steps are as follows:

Step 1: *Image cleaning:* Given an input image, $\mathbf{I_o}$, we generated a clean version $\mathbf{I_c}$ of it using PCA to compress and retain 99% of its original information.

Step 2: *Noise extraction:* We extract the noise component from the original input as $\mathbf{I_n} = \mathbf{I_o} - \mathbf{I_c}$.

Step 3: *Generating benign noisy samples:* We derive the statistical properties of $\mathbf{I_n}$ as in Equation (2) and generate statistically similar Gaussian random noise $\mathbf{N_g}$ and add it back to $\mathbf{I_c}$. In this way, we generate a benign noisy image. The process is repeated at least 50 times to obtain sufficient samples.

Step 4: *Obtaining input-specific statistical model:* We let benign noisy samples and $\mathbf{I_c}$ pass through the model and obtain the statistical properties of the behavior deviation of the most sensitive (vulnerable) convolution layer. In the VGG-16 model, the layer selected as a probe is the first convolution layer in block 5 according to our experiments in Section IV-B.

Step 5: *Comparing:* We feed the original image into the model. If the original image compromises the last convolution layer by causing sufficient drift. We then decide that this input is an adversarial example.

E. NoiseCAM for Adversarial Example Detection

Along with behavior deviation modeling, we present an XAI-based algorithm, called NoiseCAM, for detecting adversarial examples as shown in Fig. 3. Our proposed workflow capitalizes on gradient information from the DNN classifier's internal layers. To extract necessary information, we combine GradCAM++ [28] and LayerCAM [29] as follows:

1) *Globally Weighted CAM:* We let y^c be the prediction score of the DNN classifier of the target category c, and for a selected convolution layer, $\mathbf{A}_k$ is the k-th output feature map of the layer output tensor. The spatial gradient g_{ij}^{kc} with respect to $\mathbf{A}_k$ is as:

$$g_{ij}^{kc} = \frac{\partial y^c}{\partial \mathbf{A}_{ij}^k} \tag{9}$$

where g_{ij}^{kc} is the partial derivative of the prediction score of category c with respect to the pixels of the k-th feature map of the selected convolution layer. In the NoiseCAM algorithm, we use the category with the highest prediction score to generate g_{ij}^{kc}. We further process this spatial gradient tensor using similar method as in GradCAM++ [28]. Specifically, we first compute the enhanced spatial

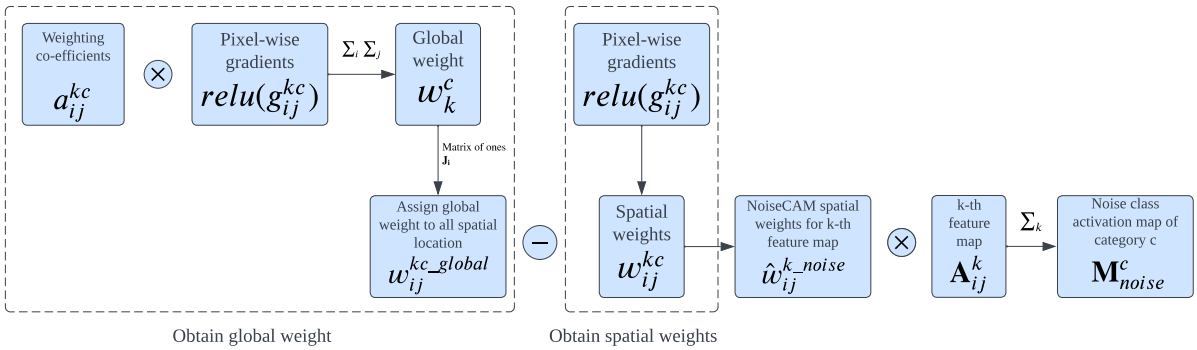


Fig. 4. The workflow of NoiseCAM

gradients as:.

$$a_{ij}^{kc} = \frac{(g_{ij}^{kc})^2}{2(g_{ij}^{kc})^2 + \sum_a \sum_b \mathbf{A_{ab}^k}(g_{ij}^{kc})^3} \tag{10}$$

where both $(a,\ b)$ and $(i,\ j)$ are the spatial location of the feature map. We then compute the weights of each channel in the output tensor w_k^c as in Equation (10). The final class activation map that captures activations positively contribute to the classification score Y^C is obtained by a linear aggregation along all k feature maps as in Equation (12)

$$w_k^c = \sum_i \sum_j a_{ij}^{kc} \cdot relu(g_{ij}^{kc}) \tag{11}$$

$$M^c = \mathbf{ReLU}(\sum_k w_k^c \cdot \mathbf{A}^k) \tag{12}$$

According to our experiments, adversarial perturbations severely affect feature maps in shallow layers. The global weights used in Grad-CAM++ algorithm will treat all activated pixels as equal. In our experiments, the Grad-CAM++ algorithm generally outperforms Grad-CAM in deriving w_k^c when multiple objects are in the input.

2) *Pixel-wisely weighted CAM:* This CAM uses the same mathematical process as in LayerCAM [29]. Each feature map $\mathbf{A}_{ij}^k$ is first weighted pixel-wisely as:

$$w_{ij}^{kc} = relu(g_{ij}^{kc}) \tag{13}$$

$$\hat{\mathbf{A}}_{\mathbf{ij}}^{\mathbf{k}} = w_{ij}^{kc} \cdot \mathbf{A}_{\mathbf{ij}}^{\mathbf{k}} \tag{14}$$

where w_{ij}^{kc} is the pixel-wise spatial weight matrix that has the same size with the k-th feature map with respect to y^C. LayerCAM [29] showed that the spatial weight method could eliminate most of the noise and preserve fine-grained information.

Our proposed NoiseCAM combines the information from globally and pixel-wisely weighted CAMs. The brief workflow of NoiseCAM is given in Fig 4. We first obtain the global weight of the k-th activation map from shallow layers. Then, we obtain the pixel-wise spatial weight of the k-th activation map from the same layer. To obtain the noise weight $\hat{w}_{ij}^{k\_noise}$, we first assign the global weight to all spatial locations (i, j) on the activation map, which is formulated as:

$$w_{ij}^{kc\_global} = w_k^c \cdot \mathbf{\Omega}^k \tag{15}$$

where $\mathbf{\Omega}^k$ is an all-one matrix with the same size as the k-th feature map. Then we subtract $w_{ij}^{kc\_global}$ with the pixel-wise spatial weight $relu(g_{ij}^{kc})$ and form a spatial noise weight matrix $\hat{w}_{ij}^{k\_noise}$ as:

$$\begin{aligned} \hat{w}_{ij}^{k\_noise} &= w_{ij}^{kc\_global} - w_{ij}^{kc} \\ &= w_{ij}^{kc\_global} - relu(g_{ij}^{kc}) \end{aligned} \tag{16}$$

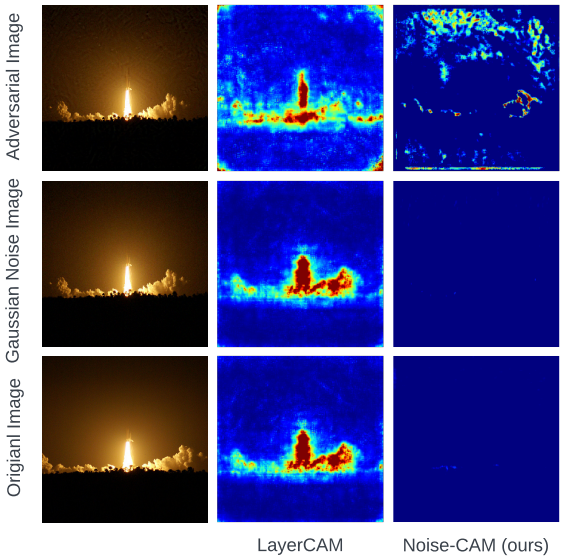


Fig. 5. Comparison of adversarial perturbation detection on NoiseCAM and GradCAM++. Top row: the adversarial example (prediction: lampshade) and its class activation map and NoiseCAM map. Center row: original image mixed with Gaussian noise (prediction: space shuttle) and corresponding maps. Bottom row: original image (prediction: space shuttle) and corresponding maps.

where w_{ij}^{kc} is defined in Equation (13). A linear summation can be performed to obtain the final noise activation map with respect to the target category c.

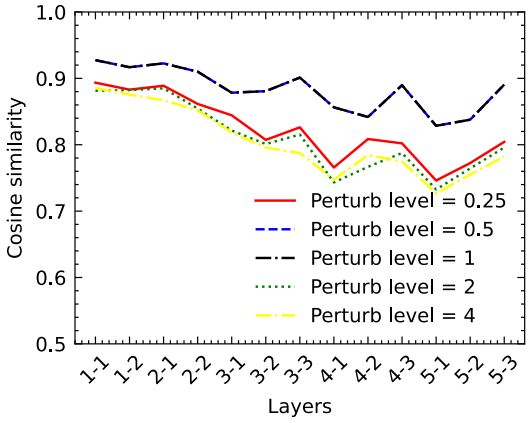
$$\mathbf{M_{noise}^c} = \mathbf{ReLU}(\sum_k \hat{w}_{ij}^{k\_noise} \cdot \mathbf{A}_{ij}^k) \tag{17}$$

Mathematically, for an input image with adversarial noise, a globally weighted CAM contains both adversarial perturbations and fine-grained class activation details, while a pixel-wisely weighted CAM only contains the fine-grain details without perturbation noise. Consequently, the subtraction of the two CAMs could expose adversarial perturbations as in Figure 5. As depicted, NoiseCAM exposes and highlights adversarial noise patterns that do not belong to the subject of interest. On the other hand, NoiseCAM generates a blank map for input with Gaussian noise and original image input.

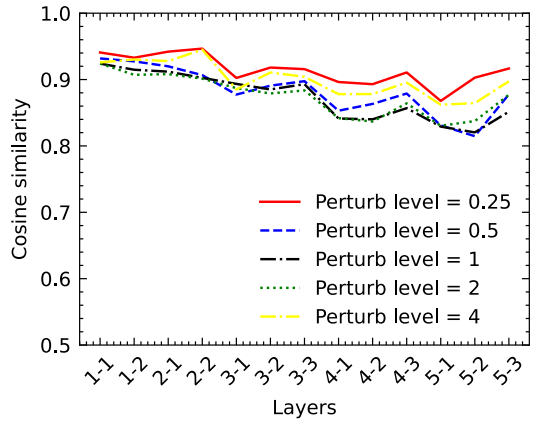
Finally, the noise activation map, $\mathbf{M_{noise}^c}$, is processed by the DBSCAN [39] clustering algorithm, we extract the number of effective noise clusters from the noise activation map, if the number of effective noise cluster exceeds 3, we then judge that the input is an adversarial example. Specifically, we set the scanning radius to 2 and the number of neighboring points to 3 as well.

IV. Evaluation

This section examines the behavior deviation of the VGG-16 classifier when subjected to varying degrees of adversarial attacks and Gaussian noise. The goal is to determine the boundary between adversarial and natural noise. Moreover, we compare the efficacy of NoiseCAM and behavior deviation modeling in detecting adversarial examples.



(a)



(b)

Fig. 6. Comparison of behavior deviation on: (a) adversarial examples and (b) images with Gaussian noise with different attack strength (x-axis 1-1 represents the Block1_Conv1 layer in VGG-16).

A. Seed Selection

The ImageNet2012 validation dataset [40] contains 5,000 images in 1,000 categories. We use all 5,000 images to derive their adversarial examples. Unfortunately, not all images can find their corresponding adversarial examples within the predefined magnitude of perturbation. Ultimately, we derived at least one adversarial example from 48% of the dataset.

B. Behavior Deviation Under Different Attacks

Using the methods in Section III-D, we found that Gaussian noise and adversarial perturbations can cause deviations in the VGG-16 model's behavior. Adversarial perturbations can further drift the network behavior and play a vital role in misleading the classifier.

According to Figure 6a and 6b, the comparison of behavior deviation indicates that adversarial perturbation causes more severe behavior deviations than statistically similar Gaussian noise. Although it is generally believed that a stronger adversarial attack strength can cause the behavior of the neural network to deviate further, our experiments show that the behavior deviation at perturbation level 0.25 is even more

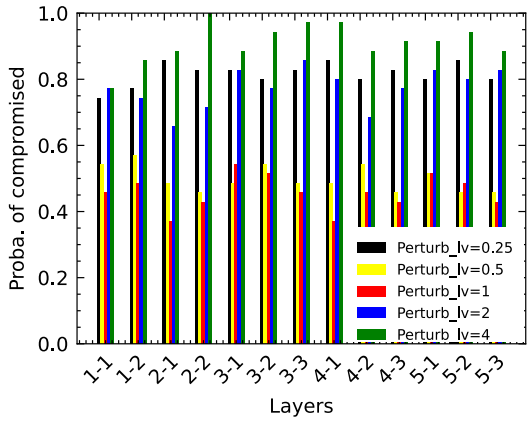


Fig. 7. Compromise probability of convolution layers in the VGG-16 network.

significant than at levels 2 and 4. Gaussian noise can cause a deviation in the behavior of the neural network, but this deviation is not significant to lead to a classification error.

Further behavior deviations can be observed in deeper layers of the network, as shown in Figures 6a and 6b. However, we do notice some fluctuations, which indicates that there are some specific layers that are more vulnerable and easier to compromise. For example, the first convolution layer in the fifth block is sensitive to adversarial perturbations and Gaussian noise simultaneously.

We also discover that the VGG-16 network can be misled by compromising only a few intermediate layers. We analyze the probability of compromise on each network layer and derive Figure 7. As depicted, the adversarial examples with the lowest attack strength have approximately 40% of the chance to compromise any layer. This probability increases significantly and reaches 80% when we increase the attack strength to be greater than 2 or less than 0.25.

C. Adversarial example detection

We evaluated the adversarial example detection performance using an augmented dataset consisting of our seed images and their modifications. This test set comprises seed images mixed with adversarial perturbations and Gaussian noise of varying intensities. Our approach involves feeding the input image into a pre-trained VGG-16 model on the ImageNet dataset, applying NoiseCAM on the VGG-16 network's third convolution layer, and using the behavior deviation modeling method on the same test set but on the most vulnerable convolution layer (Block5_1). Figure 9 showcases examples of adversarial example detection with intermediate results. By combining the responses of GradCAM++ and LayerCAM, adversarial perturbations are effectively highlighted. Figure 8 provides a brief comparison of the two methods, demonstrating that NoiseCAM has a higher detection accuracy than behavior deviation analysis, as shown in Figure 8a. Notably, the behavior deviation modeling method has a lower true positive rate and a higher true negative rate than NoiseCAM, as seen in Figure 8. Interestingly, increasing the attack strength from

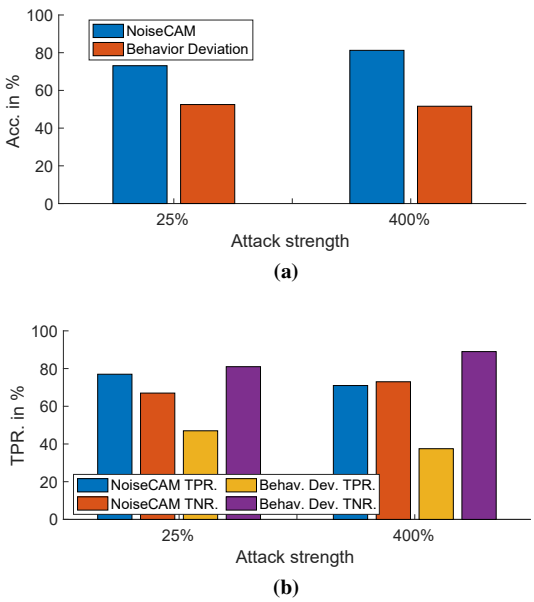


Fig. 8. Performance comparison of adversarial example detection

25% to 400% slightly reduces the true positive rate of the behavior deviation modeling method, but has no significant impact on the performance of the NoiseCAM approach.

V. Conclusion and Future Work

In this work, we model the behavior of a VGG-16 model using an explainable AI approach when its input is mixed with adversarial perturbations from white-box attacks and statistically similar Gaussian noise. We proposed two approaches, behavior deviation modeling and NoiseCAM to detect adversarial examples and prevent the model from being misled. NoiseCAM can highlight adversarial perturbations mixed in the input while it's not sensitive to random noise. We found that NoiseCAM is more reliable than behavior deviation modeling. Going forward, we aim to extend our approach as a vulnerability detection and securing tool for neural networks across various models and apply it to time-series dataset.

Acknowledgment

This research was partially supported by the National Science Foundation under Grant No. 2309760 and Grant No. 2317117.

References

[1] S. Niu, M. Liu, Y. Liu, J. Wang, and H. Song, "Distant domain transfer learning for medical imaging," *IEEE Journal of Biomedical and Health Informatics*, vol. 25, no. 10, pp. 3784–3793, 2021.

[2] Y. Liu, J. Wang, J. Li, S. Niu, and H. Song, "Machine learning for the detection and identification of internet of things devices: A survey," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 298–320, 2021.

[3] J. Wang, Y. Liu, and H. Song, "Counter-unmanned aircraft system (s)(c-uas): State of the art, challenges, and future trends," *IEEE Aerospace and Electronic Systems Magazine*, vol. 36, no. 3, pp. 4–29, 2021.

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

[5] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," in *Proceedings of the 26th Symposium on Operating Systems Principles*, ser. SOSP '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1–18. [Online]. Available: https://doi.org/10.1145/3132747.3132785

[6] D. J. Miller, Z. Xiang, and G. Kesidis, "Adversarial learning targeting deep neural network classification: A comprehensive review of defenses against attacks," *Proceedings of the IEEE*, vol. 108, no. 3, pp. 402–433, 2020.

[7] B. Luo, Y. Liu, L. Wei, and Q. Xu, "Towards imperceptible and robust adversarial example attacks against neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

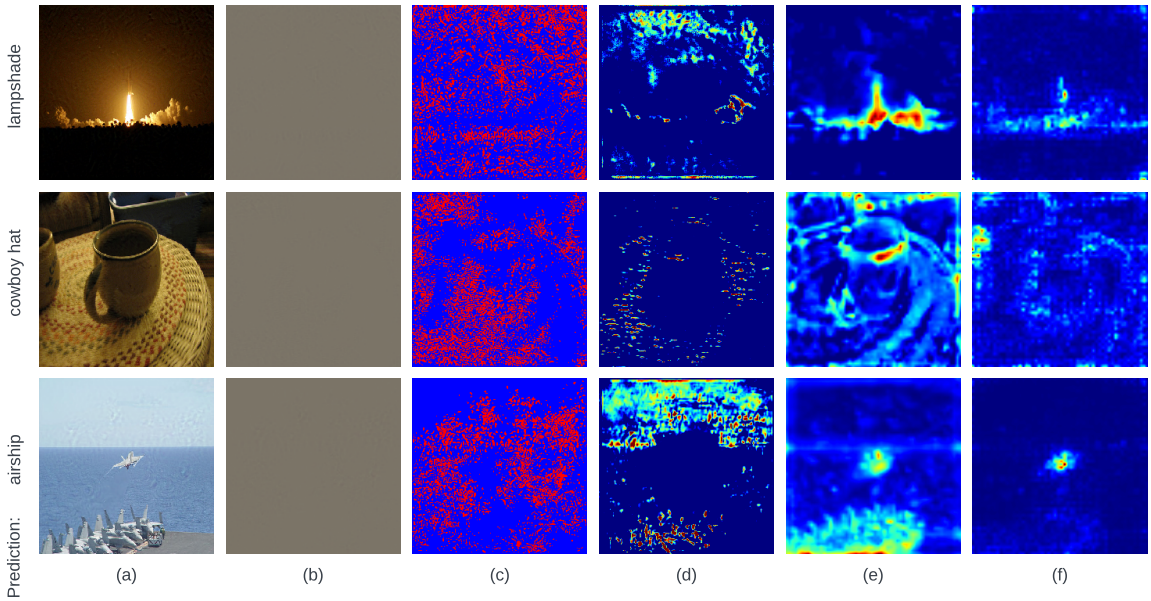


Fig. 9. The class activation map on adversarial perturbations. (a) Adversarial images. (b) Adversarial perturbation. (c) Adversarial perturbation visualization. (d) Activation map generated by NoiseCAM. (e-f) Class activation maps from the third convolution layer of VGG-16 model generated by GradCAM++ and LayerCAM.

[8] C. Englert, P. Galler, P. Harris, and M. Spannowsky, "Machine learning uncertainties with adversarial neural networks," *The European Physical Journal C*, vol. 79, no. 1, pp. 1–10, 2019.

[9] A. Ignatiev, "Towards trustable explainable ai." in *IJCAI*, 2020, pp. 5154–5158.

[10] B. Wu, J. Chen, D. Cai, X. He, and Q. Gu, "Do wider neural networks really help adversarial robustness?" *Advances in Neural Information Processing Systems*, vol. 34, pp. 7054–7067, 2021.

[11] C. Gong, T. Ren, M. Ye, and Q. Liu, "Maxup: Lightweight adversarial training with data augmentation improves neural network training," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2474–2483.

[12] R. Gao, T. Cai, H. Li, C.-J. Hsieh, L. Wang, and J. D. Lee, "Convergence of adversarial training in overparametrized neural networks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[13] J. Guo, Y. Zhao, H. Song, and Y. Jiang, "Coverage guided differential adversarial testing of deep learning systems," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 933–942, 2021.

[14] X. Xie, L. Ma, F. Juefei-Xu, M. Xue, H. Chen, Y. Liu, J. Zhao, B. Li, J. Yin, and S. See, "Deephunter: a coverage-guided fuzz testing framework for deep neural networks," in *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2019, pp. 146–157.

[15] A. Odena, C. Olsson, D. Andersen, and I. Goodfellow, "Tensorfuzz: Debugging neural networks with coverage-guided fuzzing," in *International Conference on Machine Learning*. PMLR, 2019, pp. 4901–4911.

[16] Z. Yang, J. Shi, M. H. Asyrofi, and D. Lo, "Revisiting neuron coverage metrics and quality of deep neural networks," *arXiv preprint arXiv:2201.00191*, 2022.

[17] F. Harel-Canada, L. Wang, M. A. Gulzar, Q. Gu, and M. Kim, "Is neuron coverage a meaningful measure for testing deep neural networks?" in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 851–862.

[18] A. Lamb, V. Verma, J. Kannala, and Y. Bengio, "Interpolated adversarial training: Achieving robust neural networks without sacrificing too much accuracy," in *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, 2019, pp. 95–103.

[19] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-gan: Protecting classifiers against adversarial attacks using generative models," *arXiv preprint arXiv:1805.06605*, 2018.

[20] Z. Zheng and P. Hong, "Robust detection of adversarial attacks by modeling the intrinsic properties of deep neural networks," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[21] L. Gondara, "Medical image denoising using convolutional denoising autoencoders," in *2016 IEEE 16th international conference on data mining workshops (ICDMW)*. IEEE, 2016, pp. 241–246.

[22] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," *arXiv preprint arXiv:1412.5068*, 2014.

[23] A. Yadav, A. Upadhyay, and S. Sharanya, "An integrated auto encoder-block switching defense approach to prevent adversarial attacks," *arXiv preprint arXiv:2203.10930*, 2022.

[24] U. Hwang, J. Park, H. Jang, S. Yoon, and N. I. Cho, "Puvae: A variational autoencoder to purify adversarial examples," *IEEE Access*, vol. 7, pp. 126 582–126 593, 2019.

[25] J. Renkhoff, W. Tan, A. Velasquez, W. Y. Wang, Y. Liu, J. Wang, S. Niu, L. B. Fazlic, G. Dartmann, and H. Song, "Exploring adversarial attacks on neural networks: An explainable approach," in *2022 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, 2022, pp. 41–42.

[26] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[27] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[28] A. Chattopadhay, A. Sarkar, P. Howlader, and V. N. Balasubramanian, "Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks," in *2018 IEEE winter conference on applications of computer vision (WACV)*. IEEE, 2018, pp. 839–847.

[29] P.-T. Jiang, C.-B. Zhang, Q. Hou, M.-M. Cheng, and Y. Wei, "Layercam: Exploring hierarchical class activation maps for localization," *IEEE Transactions on Image Processing*, vol. 30, pp. 5875–5888, 2021.

[30] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[31] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[32] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[33] D. Cian, J. van Gemert, and A. Lengyel, "Evaluating the performance of the lime and grad-cam explanation methods on a lego multi-label image classification task," *arXiv preprint arXiv:2008.01584*, 2020.

[34] R. Fu, Q. Hu, X. Dong, Y. Guo, Y. Gao, and B. Li, "Axiom-based grad-cam: Towards accurate visualization and explanation of cnns," 2020. [Online]. Available: https://arxiv.org/abs/2008.02312

[35] N. Poerner, H. Schütze, and B. Roth, "Evaluating neural network explanation methods using hybrid documents and morphosyntactic agreement," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 340–350. [Online]. Available: https://aclanthology.org/P18-1032

[36] F. Gabbay, S. Bar-Lev, O. Montano, and N. Hadad, "A lime-based explainable machine learning model for predicting the severity level of covid-19 diagnosed patients," *Applied Sciences*, vol. 11, no. 21, 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/21/10417

[37] D. J. Miller, Z. Xiang, and G. Kesidis, "Adversarial learning targeting deep neural network classification: A comprehensive review of defenses against attacks," *Proceedings of the IEEE*, vol. 108, no. 3, pp. 402–433, 2020.

[38] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[39] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "Dbscan revisited, revisited: why and how you should (still) use dbscan," *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 3, pp. 1–21, 2017.

[40] "imagenet2012 — tensorflow datasets," https://www.tensorflow.org/datasets/catalog/imagenet2012. (Accessed on 02/08/2023).