Defending against Adversarial Attacks through Resilient Feature Regeneration

Tejas Borkar  
Arizona State University  
tsborkar@asu.edu

Felix Heide  
Princeton University  
fheide@cs.princeton.edu

Lina Karam  
Arizona State University  
karam@asu.edu

Abstract

Deep neural network (DNN) predictions have been shown to be vulnerable to carefully crafted adversarial perturbations. Specifically, so-called universal adversarial perturbations are image-agnostic perturbations that can be added to any image and can fool a target network into making erroneous predictions. Departing from existing adversarial defense strategies, which work in the image domain, we present a novel defense which operates in the DNN feature domain and effectively defends against such universal adversarial attacks. Our approach identifies pretrained convolutional features that are most vulnerable to adversarial noise and deploys defender units which transform (regenerate) these DNN filter activations into noise-resilient features, guarding against unseen adversarial perturbations. The proposed defender units are trained using a target loss on synthetic adversarial perturbations, which we generate with a novel efficient synthesis method. We validate the proposed method for different DNN architectures, and demonstrate that it outperforms existing defense strategies across network architectures by more than 10% in restored accuracy. Moreover, we demonstrate that the approach also improves resilience of DNNs to other unseen adversarial attacks.

1. Introduction

DNNs have shown great improvement in state-of-the-art performance for challenging computer vision tasks such as image classification [22, 43, 46, 17], object recognition [41, 40] and semantic segmentation [42, 50]. Despite their continued success and widespread use in practical computer vision applications, these networks have also been shown to make arbitrary high confidence predictions for images that contain random noise and no actual human recognizable object or class (i.e., clutter samples) [35]. Adversarial noise is a small magnitude, carefully crafted image perturbation almost visually imperceptible to humans, which when added to an image causes DNNs to misclassify the image even when the original perturbation-free version of such an image is correctly classified with high confidence [47, 15, 5, 21, 33, 30, 36].

Although the original adversarial attack formulation proposed by Szegedy et al. [47] is computationally expensive to compute, stronger and computationally inexpensive attacks have since been proposed [15, 33, 23]. Most existing adversarial attacks use target network gradients (white-box setting) to construct an adversarial example [47, 15, 23, 33, 37, 5], making them largely image and network specific with limited transferability to other networks [47, 25, 38]. However, some attacks do not rely on accurate gradient information, and instead construct adversarial samples by accessing only the network predictions (black-box attacks) [34, 45] or by utilizing gradients from another similar network [36].

Unlike the previously mentioned adversarial attacks, universal adversarial attacks [30] construct a single image-agnostic perturbation that when added to any image fools DNNs into making erroneous predictions with very high confidence as shown in Fig. 1. These universal perturbations are not unique and many adversarial directions may exist in a DNN’s feature space, further highlighting the need to guard against vulnerable directions of a DNN’s feature...
space [31, 12, 11]. Interestingly, universal perturbations can not only be easily computed for different network architectures (including AlexNet, VGG, GoogLeNet and ResNet) but the perturbation generated for any one DNN also fools other DNNs (black-box setting), making them double universal [30]. Such a pre-computed perturbation that is both image-agnostic and also generalizes well to other unseen networks can be easily used to fool DNN models in real-time on unseen images, circumventing the need for imagespecific computations or even network information used in [13, 27, 1]. Universal adversarial perturbations have also been extended to semantic segmentation [18].

This work proposes a novel defense against unseen universal adversarial perturbations [30] through the following contributions:

- We show the existence of a set of vulnerable convolutional filters, that are largely responsible for erroneous predictions made by a DNN in an adversarial setting. The \( \ell_1 \) norm of the convolutional filter weights can be used to identify such filters, and masking the noise in their activations is able to prevent erroneous predictions.
- A novel defense that operates in the DNN feature space, unlike all existing image domain defenses. Our proposed defense deploys learnable defender units, which operate on the activations of the vulnerable convolutional filters and transform them into noise resilient features.
- A fast method to generate strong synthetic adversarial perturbations for training, computed from a small set of universal adversarial perturbations for a target DNN.
- Extensive evaluations on a variety of DNN architectures, including CaffeNet [22], VGG-F [6], VGG-16 [43], GoogLeNet [46] and ResNet-50 [17], validate that our proposed defense outperforms all other existing defenses. We will make our entire code publicly available for reproducibility and comparison.
- The proposed defender units are also able to improve robustness of a DNN against other unseen attacks without any additional attack specific-training.

2. Related Work

2.1. Adversarial training

Szegedy et al. [47] and Goodfellow et al. [15] show that augmenting adversarial examples in the training stage makes networks more robust to the same adversarial attack but requires computation of an adversarial sample for each training iteration. Tramer et al. improve robustness through adversarial training with an ensemble of attack examples generated for different attack types and networks [48]. Pernot et al. [36] suggest a distillation approach that involves training a shallower network to match softmax predictions of deeper networks. This has the effect of producing smoother and regularized network decision boundaries that are more robust to adversarial perturbations. However, subsequently, Carlini and Wagner [5] proposed a modification to existing gradient-based attacks that result in much stronger adversarial perturbations which are able to fool Pernot et al.’s distillation approach. Buckman et al. [4] show robustness to adversarial examples on MNIST and CIFAR-10 through non-linear discretization of real-valued DNN inputs.

2.2. Image Transformation

Image domain defenses utilize non-differentiable transformations of the input to mitigate the impact of adversarial perturbations [16, 39, 10, 32]. However, such approaches introduce unnecessary artifacts in clean images resulting in accuracy loss [1][39]. Using JPEG compression to eliminate adversarial perturbations has been explored in [10], but compression alone tends to be insufficient as a defense. Song et al. [44] propose filtering high frequency coefficients of an image’s Saak transform [44] to remove adversarial perturbations. Das et al. [7] train an ensemble of networks on varying degrees of JPEG compression to defend against certain attacks on Inception-v4 and ResNet-50 models. Liu et al. [26] uses a DNN-oriented image compression scheme to defend against adversarial attacks.

Guo et al. [16] use image quilting to deconstruct an adversarial image into small patches that are reconstructed using Total Variance Minimization. Similarly, dividing the adversarial image into small patches and denoising these patches have been explored in [32]. Prakash et al. [39] observe that most adversarial attacks are agnostic to object location and leverage this to locally corrupt the image through pixel redistribution, followed by a wavelet denoising stage.

2.3. Detecting Adversarial Examples

Another popular line of defenses explores the idea of first detecting an adversarially perturbed input and then either abstaining from making a prediction or further pre-processing adversarial input for reliable predictions [24, 27, 49, 28, 29]. Li and Li [24] use statistics based on the activations of convolutional filters to first detect adversarial examples, followed by a 3x3 averaging operation on the input image to improve robustness. Similarly, Meng and Chen [28] use two additional networks, one to detect adversarially perturbed images and the other to then pre-process the detected input adversarial images to resemble natural images, before being passed on to the target DNN for predictions. Lu et al. [27] proposed an adversarial image detector based on discrete codes generated using activations from the late
stages of a DNN. Xu et al. [49] use numerous quantization operations on the DNN input to both detect adversarial images and to also squash out adversarial perturbations in the input.

All of the aforementioned defenses in Sections 2.1-2.3 are geared towards image-specific gradient-based attacks and none of them has, as of yet, been shown to defend against image-agnostic attacks like [30] and [20]. Recently, Akhtar et al. [1] proposed a defense against image-agnostic adversarial perturbations, using a learnable Perturbation Rectifying Network (PRN) that denoises the input image. Although adversarial examples are mitigated by the PRN, clean images tend to be over-corrected leading to a loss in accuracy. This over-correction of original images is reduced by adding an adversarial sample detector that detects and passes only adversarially perturbed input images through the PRN. In contrast, our proposed approach works solely in the DNN feature space and outperforms existing defenses. As a result, our approach does not suffer from over-correction of the input and is able to maintain high accuracy on both adversarial and clean images without the need to detect if the input has been adversarially perturbed. An in-depth survey of existing defenses and attacks can be found in [2].

3. Problem Formulation

Let \( \mu_c \) represent the distribution of clean (unperturbed) images in \( \mathbb{R}^d \). \( \mathcal{P}(\cdot) \) be a classifier that predicts a class label \( \mathcal{F}(x) \) for an image \( x \in \mathbb{R}^d \). The universal adversarial perturbation attack seeks a perturbation vector \( v \in \mathbb{R}^d \) under the following constraints [30]:

\[
P(x \sim \mu_c \left( \mathcal{F}(x + v) \neq \mathcal{F}(x) \right) \geq (1 - \delta) \text{ s.t. } \|v\|_p \leq \xi \tag{1}
\]

where \( P(\cdot) \) denotes probability, \( \| \cdot \|_p \) is the \( \ell_p \) norm with \( p \in [1, \infty] \), \( (1 - \delta) \) is the target fooling ratio with \( \delta \in [0, 1] \), (i.e., the fraction of samples in \( \mu_c \) that change labels when perturbed by an adversary) and \( \xi \) controls the magnitude of adversarial perturbations. Moosavi-Dezfooli et al. [30] computed perturbations that achieved a fooling ratio \( \geq 80\% \) with \( \xi = 10 \) for \( \ell_\infty \)-norm and \( \xi = 2000 \) for \( \ell_2 \)-norm.

4. Feature-Domain Adversarial Defense

This section describes the proposed defense framework which operates in the feature domain. Specifically, we propose a defense framework that first identifies convolutional filters whose activations are significantly disrupted by adversarial perturbations (Section 4.1) and then deploy defender units that transform (regenerate) disruptive features into noise-resilient features that restore the network accuracy (Section 4.2). The defender units are trained using a target loss on synthetic adversarial perturbations (Section 4.3), leaving the parameters for the underlying baseline DNN unchanged.

4.1. Stability of Convolutional Filters

Szegedy et al. [47] showed that convolutional layer stability is affected by the operator norm of the layer weights. In this work, we assess the vulnerability of individual filters and show that, for each layer, certain filter activations are significantly more disrupted than others, especially in the early layers of a DNN.

For a given layer, let \( \phi_m(u) \) be the output (activation map) of the \( m \)-th convolutional filter with kernel weights \( W_m \) for an input \( u \). Let \( c_m = \phi_m(u + r) - \phi_m(u) \) be the additive noise (perturbation) that is caused in the output activation map \( \phi_m(u) \) as a result of applying an additive perturbation \( r \) to the input \( u \). It can be shown (see supplementary 6) that \( c_m \) is bounded as follows:

\[
\|c_m\|_\infty \leq \|W_m\|_1 \|r\|_p \tag{2}
\]

where as before \( \| \cdot \|_p \) is the \( \ell_p \) norm with \( p \in [1, \infty] \). Equation 2 shows that the \( \ell_1 \) norm of the filter weights can be used to identify and rank convolutional filter activations in terms of their ability to restrict perturbation in their activation maps. For example, filters with a small weight \( \ell_1 \) norm would result in insignificant small perturbations in their output when their input is perturbed, and are thus considered to be less vulnerable to perturbations in the input. Figure 2 shows observed norms in the first convolution layer of CaffeNet [22] and GoogLeNet [46] for an \( \ell_\infty \)-norm universal adversarial input. We can see that our \( \|W\|_1 \)-based ranking correlates well with the degree of perturbation (max level) that is induced in the filter outputs. Similar observations can be made for other convolutional layers in the network (see supplementary 6, Figure 8).

We evaluate the impact of masking the adversarial noise in such ranked filters on the overall top-1 accuracy of CaffeNet [22], VGG-16 [43] and GoogLeNet [46]. Specifically, we randomly choose a subset of 1000 images (1 image per class) from the ILSVRC2012 [8] training set and add an \( \ell_\infty \) norm universal adversarial perturbation (generated using [30]) to each image to generate adversarially perturbed images. The top-1 accuracies for perturbation-free images of our subset are 0.58, 0.70 and 0.69 for CaffeNet, GoogLeNet and VGG-16, respectively. Similarly the top-1 accuracies for adversarially perturbed images of the same subset are 0.10, 0.25 and 0.25 for CaffeNet, GoogLeNet and VGG-16, respectively. From Figure 3, we see that just masking the adversarial perturbations in 50% of the most vulnerable filter activations is able to recover most of the lost performance for each DNN resulting in top-1 accuracies of 0.56, 0.68 and 0.67 for CaffeNet, GoogLeNet and VGG-16, respectively.
Figure 2. Observed $\ell_\infty$ norm for universal adversarial noise in the ranked convolutional filters (ordered from most vulnerable to least) of the first layer of CaffeNet [22] and GoogLeNet [46]. The $\ell_\infty$-norm attack is used with $\xi \leq 10$, i.e., $\|r\|_\infty \leq 10$.

Figure 3. Effect of masking $\ell_\infty$ norm universal adversarial noise in ranked convolutional filter activations of the first layer in CaffeNet [22], GoogLeNet [46] and VGG-16 [43], evaluated on a 1000 image subset of the ILSVRC2012 [8] training set. Top-1 accuracies for perturbation-free images are 0.58, 0.70 and 0.69 for CaffeNet, GoogLeNet and VGG-16, respectively. Similarly, top-1 accuracies for adversarially perturbed images with no noise masking are 0.1, 0.25 and 0.25 for CaffeNet, GoogLeNet and VGG-16, respectively. Masking the noise in just 50% of the ranked filter activations restores most of the lost accuracy for all three DNNs.

4.2. Resilient Feature Regeneration Defense

Our proposed defense is illustrated in Figure 4. We learn a task-driven feature restoration transform that acts as a **defender** for convolutional filter activations severely disrupted by adversarial input. This **defender unit** does not modify the remaining activations of the baseline DNN. A similar approach of learning corrective transforms has been explored in a different context in [3], with the goal of making networks more resilient to image blur and additive white gaussian noise.

Let $S_l$ represent a set consisting of indices for convolutional filters in the $l^{th}$ layer of a DNN. Furthermore, let $S_{def}$ be the set of indices for filters we wish to defend (Section 4.1) and let $S_{adv}$ be the set of indices for filters whose activations are not regenerated (i.e., $S_l = S_{def} \cup S_{adv}$). If $\Phi_l$ represents the set of convolutional filters in the $l^{th}$ layer, then our **defender units** perform a feature regeneration transform $D_l(\cdot)$ under the following conditions:

$$D_l(\Phi_{S_{def}}(u + r)) \approx \Phi_{S_{def}}(u) \quad (3)$$

and

$$D_l(\Phi_{S_{def}}(u)) \approx \Phi_{S_{def}}(u) \quad (4)$$

where $u$ is the unperturbed input to the $l^{th}$ layer of convolutional filters and $r$ is an additive perturbation that acts on $u$. In Equations 3 and 4, $\approx$ denotes similarity based on classification accuracy in the sense that features are restored to regain the classification accuracy of the original perturbation-free activation map. Equation 3 forces $D_l(\cdot)$ to pursue task-driven feature denoising that restores lost accuracy of the DNN while Equation 4 ensures that prediction accuracy on unperturbed activations is not decreased, without any additional adversarial perturbation detector. Similar to Borkar and Karam [3], we implement $D_l(\cdot)$ (i.e., **defender unit**) as a shallow residual block [17], consisting of two stacked $3 \times 3$ convolutional layers sandwiched between a couple of $1 \times 1$ convolutional layers and a single skip connection. $D_l(\cdot)$ is then estimated using a target loss from the baseline network, through backpropagation as shown in Figure 4, but with much less number of parameters.

Given an $L$ layered DNN $\Phi$, pre-trained for an image classification task, $\Phi$ can be represented as a function that maps network input $x$ to an N-dimensional output label vector $\Phi(x)$ as follows:

$$\Phi = \Phi_L \circ \Phi_{L-1} \circ \ldots \circ \Phi_2 \circ \Phi_1 \quad (5)$$

where $\Phi_l$ is a mapping function (set of convolutional filters) representing the $l^{th}$ DNN layer and $N$ is the output dimensionality of the DNN (i.e., number of classes). Without any loss of generality, the resultant DNN after deploying a **defender unit** that operates on the set of filters represented by $S_{def}$ in layer $l$ is given by:

$$\Phi_{def} = \Phi_L \circ \Phi_{L-1} \circ \ldots \circ \Phi_{l_{def}} \ldots \circ \Phi_2 \circ \Phi_1 \quad (6)$$

where $\Phi_{l_{def}}$ represents the new mapping function for layer $l$, such that $D_l(\cdot)$ defends only activations of the filter subset $\Phi_{S_{def}}$ and all the remaining filter activations (i.e., $\Phi_{S_{adv}}$) are left unchanged. If $D_l(\cdot)$ is parameterized by $\theta_l$, then the **defender unit** can be trained by minimizing:

$$J(\theta_l) = \frac{1}{K} \sum_{k=1}^{K} \mathcal{L}(y_k, \Phi_{l_{def}}(x_k)) \quad (7)$$

where $\mathcal{L}$ is the same target loss function of the baseline DNN (e.g., cross-entropy classification loss), $y_k$ is the target output label for the $k^{th}$ input image $x_k$, $K$ represents the total number of images in the training set consisting of both clean and perturbed images. As we use both clean and
perturbed images during training, $x_k$ in Equation 7, represents a clean or an adversarially perturbed image. Equations 6 and 7 show defender unit estimation for a single layer, however, it is possible to add such units in more than one DNN layer as shown in Figure 4. In all our experiments (Section 5), the defender units only regenerate 50% of the convolutional filter activations in a layer.

4.3. Generating Synthetic Perturbations

The defender units in Figure 4 are trained using a combination of clean and adversarially perturbed images in each training mini-batch. However, in order to avoid overfitting, generating a reasonable variety of adversarial perturbations (≥ 100) using the algorithm of Moosavi-Dezfooli et al. [30] can be computationally prohibitive, as noted by Akhtar et al. [1]. We propose a single efficient method (Algorithm 1) to construct synthetic adversarial perturbations for both $\ell_\infty$-norm and $\ell_2$-norm attacks from a small set of adversarial perturbations $V \subseteq \mathbb{R}^d$ computed using [30]. Starting with the synthetic perturbation $v_{syn}$ set to zero, we iteratively select a random perturbation $v_{new} \in V$ and a random scale factor $\alpha \in [0, 1]$ and update $v_{syn}$ as:

$$v_{syn} = \alpha v_{new} + (1 - \alpha)v_{syn}$$

(8)

till the $\ell_2$-norm of $v_{syn}$ exceeds a threshold $\eta$. We set the threshold $\eta$ to be the minimum $\ell_2$-norm of perturbations in the set $V$.

Unlike the approach of Akhtar et al. [1], which uses an iterative random walk along pre-computed adversarial directions, the proposed algorithm has two distinct advantages: 1) The same algorithm can be used for $\ell_2$-norm and $\ell_\infty$-norm attacks without any modification and 2) instead of having to check and ensure that the $\ell_\infty$-norm of the perturbation does not violate the constraint of the attack (i.e., $\ell_\infty$-norm ≤ $\xi$), Equation 8 (step 5 in Algorithm 1) automatically ensures this.

Synthetic samples generated using Algorithm 1 achieve a fooling rate equivalent to that achieved by sampling from $V$ (e.g., 0.8 for a set $V$ fooling rate of 0.8), while Akhtar et al. [1] achieves a significantly lower fooling rate (e.g., 0.67 for a set $V$ fooling rate of 0.8). The synthetic perturbations are not identical to their closest match in the set $V$, yet they achieve similar fooling rates as the original perturbations and on average achieve a linear correlation ≥ 0.9 with their closest match compared to 0.83 achieved by Akhtar et al. [1]. See the supplementary material for visualizations (Figure 10).

5. Assessment

We use the ImageNet validation set (ILSVRC 2012) [8] consisting of 50000 images with a single crop evaluation (unless specified otherwise) in our experiments. The baseline DNNs and our proposed approach are implemented
using Caffe [19] and we use publicly provided code by Moosavi-Dezfooli et al. [30] to generate the universal adversarial perturbations for testing. For effective comparison with prior works, we follow the same evaluation setup as Akhtar et al. [1] and report our results in terms of top-1 accuracy and the restoration accuracy proposed by Akhtar et al. [1]. Given a set \( I_c \) containing clean images, set \( I_{p/c} \) containing clean and perturbed images in equal numbers, then:

\[
\text{Restoration accuracy} = \frac{\text{acc}(I_{p/c})}{\text{acc}(I_c)} \tag{9}
\]

where acc(\( \cdot \)) is the top-1 accuracy. We use all 50000 images during evaluation for easy reproducibility of results. We compute 5 independent universal adversarial test perturbations during evaluation for easy reproducibility of results.

5.1. Defense Training Methodology

In our proposed defense (Figure 4), only the parameters for defender units have to be trained and these parameters are updated to minimize the cost function given by Equation 7. Although a defender unit can be deployed for each convolutional layer in a DNN, we limit the number of deployed defender units as \( \min(\#\text{DNN layers}, 6) \)\(^2\). The filter depth of convolutional layers in our defender units is upper bounded to the number of regeneratable feature maps (i.e., 50% filters in a DNN layer). Using Algorithm 1, we generate 2000 synthetic perturbations from a set \( V \) of 25 original perturbations [30] and train defender units on a single Nvidia Titan-X using a standard SGD optimizer, momentum of 0.9 and a weight decay of 0.0005 for 4 epochs of the ImageNet training set [8]. The learning rate is dropped by a factor of 10 after each epoch with an initial learning rate of 0.1.

5.2. Validation for Varying DNN Architectures

Top-1 accuracy of adversarially perturbed test images for various DNNs (no defense) and our proposed defense for respective DNNs is reported in Table 1 under both white-box and black-box settings. Target fooling ratio for ResNet-50 is 0.6 and 0.8 for all other DNNs. DNNs in column one are tested with attacks generated for DNNs in row one.

| Methods | CaffeNet VGG-F GoogLeNet VGG-16 Res50 |
|---------|--------------------------------------|
| Baseline | 0.596 0.628 0.661 0.681 0.709 |
| PRN [1] | 0.936 0.903 0.956 - - |
| PRN-det [1] | 0.952 0.922 0.964 - - |
| PD+WD [39] | 0.873 0.813 0.884 0.818 0.900 |
| JPEG comp. [10] | 0.554 0.697 0.830 0.693 0.849 |
| Feat. Distill. [26] | 0.671 0.689 0.851 0.717 0.867 |
| Ours | 0.976 0.967 0.970 0.963 0.946 |

Table 1. Cross-DNN evaluation: Top-1 accuracy of different classifiers against a \( \ell_\infty \)-norm universal adversarial attack with \( \xi = 10 \).

For ResNet-50, the provided code in [30] was able to achieve a maximum fooling ratio of only 0.6

\(^2\)See supplementary material for results supporting this choice (Figure 9)

Table 2. Same-norm evaluation: Restoration accuracy of DNNs and associated defenses against an \( \ell_\infty \)-norm universal adversarial attack with \( \xi = 10 \).

| Methods | CaffeNet VGG-F GoogLeNet VGG-16 Res50 |
|---------|--------------------------------------|
| Baseline | 0.593 0.605 0.644 0.648 0.790 |
| Ours | 0.652 0.654 0.677 0.644 0.677 |

5.3. Analysis and Comparisons

We use the restoration accuracy metric given by Equation 9 to provide a comparison of our defense with existing defenses (Tables 2 and 3), using both clean and perturbed
Table 3. Cross-norm evaluation: Restoration accuracy against an $\ell_2$-norm attack using defense (PRN [1] and Ours) trained on $\ell_\infty$-norm attack examples with $\xi = 10$.

| Methods          | CaffeNet | VGG-F | GoogLeNet | VGG-16 |
|------------------|----------|-------|-----------|--------|
| $\ell_2$-norm attack, $\xi = 2000$ |          |       |           |        |
| Baseline         | 0.677    | 0.671 | 0.682     | 0.697  |
| PRN [1]         | 0.922    | 0.880 | 0.971     | -      |
| PRN+det [1]     | 0.936    | 0.900 | 0.975     | -      |
| PD+WD [39]      | 0.782    | 0.784 | 0.857     | 0.809  |
| Ours            | 0.964    | 0.961 | 0.912     | 0.876  |
| Stronger $\ell_2$-norm attack, $\xi = 3500$ |          |       |           |        |
| Baseline         | 0.548    | 0.548 | 0.613     | 0.617  |
| PD+WD [39]      | 0.782    | 0.640 | 0.774     | 0.756  |
| Ours            | 0.916    | 0.921 | 0.880     | 0.809  |

images since a defense should not only recover the DNN accuracy against adversarial images but must also maintain a high accuracy on clean images. Even though Akhtar et al. [1] are the only ones who have reported defense results on universal adversarial attacks, we also compare results with the defense proposed by Prakash et al. [39] as it has one of the best reported defense results for all other attacks. Additionally, defenses that perform pre-processing of the input such as JPEG compression are also compared with [10, 26], with a quality factor of 50 for compression as suggested by the respective authors.

In Table 2, we report results for an $\ell_\infty$-norm attack against various DNNs and show that our defender units outperform all the other defenses for all networks and achieve a 97% restoration rate for GoogLeNet [46]. Even without a perturbation detector, our defense outperforms the existing defense with a perturbation detector (PRN+det) of Akhtar et al. [1] for all networks and by almost 4% for VGG-F. Without a perturbation detector, PRN [1] is consistently outperformed by our approach with a significant margin. Although Prakash et al. [39] (PD+WD) report an effective defense against many attacks, their approach does not defend well against a universal adversarial attack.

In Table 3, we also evaluate how well a defense trained on an $\ell_\infty$-norm attack (Akhtar et al. [1] and ours) defends against an $\ell_2$-norm attack for the same DNN. Our defender units are able to effectively generalize to even cross-norm attacks and outperform the other two defenses on a majority of DNNs. Akhtar et al. [1] do not report results for stronger $\ell_2$-norm attacks (i.e., $\xi \geq 2000$). From Table 3, it can be seen that our defender units are still able to effectively recover 91% accuracy even against much stronger $\ell_2$-norm attacks.

5.4. Robustness to Secondary Attacks

Although in practical situations, an attacker may not have full or even partial knowledge of a defense, for completeness, we also evaluate our proposed defense against a secondary attack (type II attack [27]) that has full access to the gradient information of our defender units. Figure 6 shows the robustness of our defender units (trained on samples with fooling rate 0.85) to a secondary attack seeking to achieve a target fooling rate of 0.85, whilst having full access to defense gradients. While the attack can easily converge against a baseline DNN in less than 2 epochs and eventually achieving a final fooling rate of 0.9, it struggles to achieve a fooling rate of even 0.8 against our defense even after 20 epochs, plateauing at 0.78. Although it is possible to design a converging attack for a much smaller fooling rate, it is fairly straightforward to defend against this by simply augmenting defender unit training with such samples.

5.5. Resilience to Unseen Attacks

Although our proposed defender units effectively defend against universal adversarial perturbations, here, we evalu-
Table 4. Resilience across various attacks on CaffeNet: Restoration accuracy against various other attacks using defender units trained on just $\ell_\infty$-norm universal adversarial examples. Attacks are constructed for the baseline DNN and are assumed to have no knowledge of the defense.

| Defense          | Univ. adv [30] | AWGN | Local adv. [45] | FGSM [15] | DeepFool [33] | IGSIM [22] | Singular-UAP [20] | Sensor noise [9] | Average |
|------------------|----------------|------|-----------------|-----------|---------------|------------|-------------------|------------------|---------|
| Baseline         | 0.596          | 0.921| 0.741           | 0.715     | 0.611         | 0.553      | 0.852             | 0.836            | 0.728   |
| PD+WD [39]       | 0.873          | 0.899| 0.761           | 0.819     | 0.894         | 0.657      | 0.870             | 0.841            | 0.827   |
| Ours             | 0.975          | 0.936| 0.816           | 0.765     | 0.918         | 0.584      | 0.914             | 0.870            | 0.847   |

Figure 7. Visualization of various attacks on CaffeNet: Adversarially perturbed images for respective attacks are shown in row 1 with an enlarged view of a chosen image region (red square). Row 2 shows the 2D-FFT magnitude plots of the adversarial perturbations. Each FFT magnitude is normalized to a range of $[0,1]$. Row 3 shows predicted label and confidence (green=correct, red=incorrect) of respective images for a baseline DNN (no defense) and defender units trained only on universal adversarial perturbations.

6. Conclusion

We propose a novel approach that operates solely in the DNN feature domain and effectively defends against universal adversarial perturbations [30], unlike existing adversarial defenses which work mainly in the image domain or use a modified loss function. The proposed defense first identifies convolutional features that are most disrupted by adversarial noise and deploys trainable defender units which transform these filter activations into noise-resilient features. The defender units can efficiently be trained using a set of synthetic adversarial perturbations, avoiding the need for a large set of original adversarial perturbations that are computationally expensive to generate. The proposed defense not only effectively generalizes across different DNNs as well as attack norms, but also makes secondary white-box attacks more difficult. Additionally, defender units trained solely on universal adversarial perturbations are shown to improve the resilience of baseline DNNs to other types of unseen (adversarial) noise. Our DNN feature domain-based defense is complimentary to existing work such as image domain defenses [39, 16], and adversarial training [47] and can be easily combined with such defenses. Although we show results for the task of image classification, we expect the proposed defense to be immediately applicable to DNNs for other semantically-related tasks such as object detection [14] and semantic segmentation [42].

References

[1] N. Akhtar, J. Liu, and A. Mian. Defense against universal adversarial perturbations. In IEEE Conference on Computer Vi-
[34] N. Narodytska and S. P. Kasiviswanathan. Simple black-box adversarial perturbations for deep networks. CoRR, abs/1612.06299, 2016. 1

[35] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In IEEE Conference on Computer Vision and Pattern Recognition, pages 427–436, June 2015. 1

[36] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In ACM Asia Conference on Computer and Communications Security, pages 506–519, 2017. 1, 2

[37] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, C. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In IEEE European Symposium on Security and Privacy, pages 372–387, 2016. 1

[38] N. Papernot, P. D. McDaniel, and I. J. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. CoRR, abs/1607.02533, 2016. 1

[39] A. Prakash, N. Moran, S. Garber, A. DiLillo, and J. Storer. Deflecting adversarial attacks with pixel deflection. In IEEE Conference on Computer Vision and Pattern Recognition, pages 8571–8580, June 2018. 2, 6, 7, 8

[40] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In IEEE Conference on Computer Vision and Pattern Recognition, pages 779–788, 2016. 1

[41] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In Advances in Neural Information Processing Systems, pages 91–99, 2015. 1

[42] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. IEEE Trans. Pattern Anal. Mach. Intell., 39(4):640–651, Apr. 2017. 1, 8

[43] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2014. 1, 2, 3, 4, 6, 11, 12, 14

[44] S. Song, Y. Chen, N. Cheung, and C. J. Kuo. Defense against adversarial attacks with Saak transform. CoRR, abs/1808.01785, 2018. 2

[45] J. Su, D. V. Vargas, and K. Sakurai. One pixel attack for fooling deep neural networks. CoRR, abs/1710.08864, 2017. 1, 8, 13

[46] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In IEEE Conference on Computer Vision and Pattern Recognition, pages 1–9, 2015. 1, 2, 3, 4, 6, 7, 11, 12, 14

[47] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199, 2013. 1, 2, 3, 8

[48] F. Tramr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel. Ensemble adversarial training: Attacks and defenses. In International Conference on Learning Representations, 2018. 2

[49] W. Xu, D. Evans, and Y. Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In Network and Distributed System Security Symposium, 2018. 2, 3

[50] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. CoRR, abs/1511.07122, 2015. 1
Supplementary Material

A. Maximum adversarial perturbation

We show in Section 4.1 of the paper that the maximum possible adversarial perturbation in a convolutional filter activation map is proportional to the \( \ell_1 \)-norm of its corresponding filter kernel weights. Here, we provide a detailed proof for Equation 2 in Section 4.1 of the paper. For a given convolutional layer, let \( \mathcal{W}_m \in \mathbb{R}^{d \times d \times C} \) be the \( m \)th convolutional filter kernel, where \( C \) is the number of input feature maps to the layer and \( d \) is the spatial kernel size. Let \( \mathcal{Z} \in \mathbb{R}^{X \times Y \times C} \) be the adversarial perturbation in the layer input, where \( X \) and \( Y \) represent the height and width of the input feature maps, respectively. Then, the adversarial perturbation in the \( m \)th filter activation map is given as:

\[
\mathcal{Z} \cdot \mathcal{W}_m = \sum_{c=1}^{C} \sum_{x'=1}^{X} \sum_{y'=1}^{Y} \mathcal{Z}_c(x', y') \mathcal{W}_m(x-x', y-y')
\]  

(10)

where the subscript \( c \) is the index of the input feature map. The maximum adversarial perturbation in the \( m \)th output activation map can be given by:

\[
\max |\mathcal{Z} \cdot \mathcal{W}_m| = \max \left( \sum_{c=1}^{C} \sum_{x'=1}^{X} \sum_{y'=1}^{Y} \mathcal{Z}_c(x', y') \mathcal{W}_m(x-x', y-y') \right)
\]  

(11)

\[
\leq \sum_{c=1}^{C} \sum_{x'=1}^{X} \sum_{y'=1}^{Y} \mathcal{Z}_c(x', y') \mathcal{W}_m(x-x', y-y')
\]  

(12)

\[
\leq z_m \sum_{c=1}^{C} \sum_{x'=1}^{X} \sum_{y'=1}^{Y} \mathcal{Z}_c(x', y') \mathcal{W}_m(x-x', y-y')
\]  

(13)

where \( z_m = \max |\mathcal{Z}| \) and \( \mathbb{I} \) is a 3D tensor with each of its elements being 1 and having the same size as \( \mathcal{Z} \). From Equation 13, we can easily show:

\[
\max |\mathcal{Z} \cdot \mathcal{W}_m| \leq z_m \sum_{c=1}^{C} \sum_{x'=1}^{X} \sum_{y'=1}^{Y} \mathcal{W}_m(x-x', y-y')
\]  

(14)

If \( \text{vec}(\cdot) \) represents a vectorization operation, Equation 14 can be rewritten as:

\[
\|\text{vec}(\mathcal{Z} \cdot \mathcal{W}_m)\|_\infty \leq \|\text{vec}(\mathcal{W}_m)\|_1 \|\text{vec}(\mathcal{Z})\|_\infty
\]  

(15)

Since \( \ell_\infty \)-norm \( \leq \ell_p \)-norm where \( p \in [1, \infty) \), we have,

\[
\|\text{vec}(\mathcal{Z} \cdot \mathcal{W}_m)\|_\infty \leq \|\text{vec}(\mathcal{W}_m)\|_1 \|\text{vec}(\mathcal{Z})\|_p
\]  

(16)

B. Masking perturbations in other layers

In Section 4.1 of the paper (Figure 3 in the paper), we evaluate the effect of masking \( \ell_\infty \)-norm adversarial perturbations in a ranked subset (using \( \ell_1 \)-norm ranking) of convolutional filter activation maps of the first convolutional layer of a DNN. Here, in Figure 8, we evaluate the effect of masking \( \ell_\infty \)-norm adversarial perturbations in ranked filter activation maps of the convolutional layers 2, 3, 4 and 5 of CaffeNet [22] and VGG-16 [43]. We use the same evaluation setup as in Section 4.1 of the paper (i.e., 1000 image random subset of ILSVRC2012 [8]). The top-1 accuracy for perturbation-free images of the subset are 0.58 and 0.69 for CaffeNet and VGG-16, respectively. Similarly, the top-1 accuracies for adversarially perturbed images of our subset are 0.10 and 0.25 for CaffeNet and VGG-16, respectively. Similar to our observations in Section 4.1 of the paper, for most DNN layers, masking the adversarial perturbations in just the top 50% most susceptible filter activation maps (identified by using the \( \ell_1 \)-norm ranking measure, Section 4.1 of the paper), is able to recover most of the accuracy lost by the baseline DNN (Figure 8). Specifically, masking the adversarial perturbations in the top 50% ranked filters of VGG-16 is able to restore at least 84% of the baseline accuracy on perturbation-free images.

C. Defender units: An ablation study

In general, defender units can be added at the output of each convolutional layer in a DNN. However, this may come at the cost of increased computations, trainable parameters and also over-fitting. As mentioned in Section 5.1 of the paper, we constrain the number of defender units added to the DNN, in order to avoid drastically increasing the training and inference cost for larger DNNs (i.e., VGG-16, GoogLeNet and ResNet-50). Here, we perform an ablation study to identify the least number of defender units needed to at least achieve a 95% restoration accuracy across most DNNs. Specifically, we use VGG-16 [43] and GoogLeNet [46] for this analysis. We evaluate restoration accuracy on the ILSVRC2012 [8] validation set (50000 images) by adding increasing number of defender units, starting from a minimum value of 2 towards a maximum value of 10 in steps of 2. In Figure 9, we report the results of this ablation study and observe that for GoogLeNet, adding just two defender units achieves a restoration accuracy of 97% and adding any more defender units does not have any significant impact on the restoration accuracy. However, for VGG-16, adding only 2 defender units achieves a restoration accuracy of only 91%. For VGG-16, adding more defender units improves performance with the best restoration accuracy of 96.2% achieved with 6 defender units. Adding more than 6 defender units resulted in a minor drop in restoration accuracy and this may be due to data over-fitting. As a result, we restrict the number of defender units deployed for any DNN to \( \min(\#\text{DNN layers}, 6) \).

D. Adversarial attack details

Here, we provide a brief overview of the different adversarial attacks and noise sources used for the evaluation presented in Section 5.5 of the paper.
Figure 8. Effect of masking $\ell_\infty$ norm universal adversarial noise in ranked convolutional filter activations of CaffeNet [22] and VGG-16 [43], evaluated on a 1000 image subset of the ILSVRC2012 [8] training set. Top-1 accuracies for perturbation-free images are 0.58, 0.69 for CaffeNet and VGG-16, respectively. Similarly, top-1 accuracies for adversarially perturbed images with no noise masking are 0.1 and 0.25 for CaffeNet and VGG-16, respectively. For VGG-16, masking the noise in just 50% of the ranked filter activations restores more than $\approx 80\%$ of the baseline accuracy on perturbation-free images.

Figure 9. Effect of adding defender units on the restoration accuracy of our proposed defense. Adding just two defender units in GoogLeNet [46] achieves a restoration accuracy of 97% and adding more defender units to the DNN does not improve results any further. For VGG-16 [43], adding 6 defender units provides best results.

Fast Gradient Sign Method (FGSM) [15] is a single iteration white-box adversarial attack that uses just the sign of the gradient of the loss function with respect to the input image to construct an adversarial perturbation. Typically, the signed gradient perturbation is scaled by a factor $\epsilon$ that can be used to control the strength of the adversarial perturbation. For our experiments, we set $\epsilon = 0.5$ and the image input to the DNN has a pixel value range of $[0, 255]$.

Iterative Gradient Sign Method (IGSM) [23] is an iterative white-box adversarial attack based on FGSM [15], where at each iteration, an adversarial perturbation is constructed by using the sign of the gradient of the loss function with respect to the image, and then added to the image, followed by a clipping operation that restricts the per pixel perturbation to be within a $[-\epsilon, \epsilon]$ neighborhood of the original image. The iterative process is continued till the target DNN misclassifies the input image or the number of iterations exceeds a maximum limit. We use $\epsilon = 2$ in our experiments and restrict the maximum number of iterations to 10.

DeepFool [33] is an untargeted white-box adversarial attack that seeks to find the smallest possible perturbation which when added to an input image causes the target
DNN to misclassify the perturbed image, even thought the perturbation-free image is classified correctly. This attack approximates a DNN’s decision function as a linear decision boundary and minimizes the \( \ell_2 \)-norm of the the perturbation with respect to the original image. We use the code and default parameters provided by the original authors at https://github.com/LTS4/DeepFool

**Local adversarial attack (Single-pixel attack)** [45] is a black-box attack that restricts the number of pixels (perturbation budget) that can be adversarially perturbed in the original image. Unlike, most other gradient-based attacks, this attack only corrupts a small fraction of the pixels in the original image and still manages to fool a DNN into making erroneous predictions. The attack uses differential evolution and a greedy search to construct adversarial perturbations from only the DNN’s output predictions without using any gradient information. The perturbation budget (i.e., number of pixels perturbed) is constrained to just 500 pixels for our experiments with 224×224 pixel RGB images (i.e. 0.9% of total pixels).

**Singular-UAP** [20] is a recently proposed universal adversarial attack that uses the singular values of the Jacobian matrices of the hidden layers of a DNN to construct universal adversarial perturbations. Unlike [30], this attack utilizes far less number of images to construct an attack. Using the code and parameters provided at https://github.com/KhrulkovV/singular-fool, we construct attacks using the singular vectors of the Jacobian matrices for ”conv3” (third convolutional layer) and ”fc6” (first fully-connected layer) layers of CaffeNet [22] and report the average restoration accuracy.

**Sensor noise** [9] models realistic noise and blur induced during the image formation process in a real camera. Specifically, a poisson-gaussian model parameterized by \((\alpha_p, \sigma_b)\) is used to model the noise and blur observed in the captured image. The poisson component (parameterized by \(\alpha_p\)) represents the noise introduced by the image sensor when an image is captured in low-light settings, while the gaussian component (parameterized by \(\sigma_b\)) models the blur induced during image formation. We evaluated the poission-gaussian image formation model under three different ISO light settings (ISO=100, 200 and 400) representing increasingly noisy imaging conditions with \(\alpha_p = 0.00038\) and a fixed blur standard deviation \(\sigma_b = 0.0025\). We report results averaged over all three ISO settings.

**Additive White Gaussian Noise (AWGN)** with three different levels of noise standard deviation \(\sigma_n\) was used in our evaluation. We use \(\sigma_n = \{5, 10, 15\}\) for our experiments and report results averaged over all three noise levels.

**E. Examples of synthetic perturbations**

Sample visualizations of synthetic adversarial perturbations generated using our algorithm proposed in Section 4.3 (Algorithm 1) of the paper are provided in Figure 10.
Figure 10. Visualization of synthetic perturbations (center) computed for CaffeNet [22], GoogLeNet [46], ResNet-50 [17] and VGG-16 [43] along with their closest match in the set of original perturbations (left) and a per pixel difference map between the two (right).