First Line of Defense: A Robust First Layer Mitigates Adversarial Attacks

Janani Suresh[1], **Nancy Nayak**[2], **Sheetal Kalyani**[1]

[1]Indian Institute Of Technology, Madras
[2] Imperial College London
ee22s079@smail.iitm.ac.in, n.nayak@imperial.ac.uk, skalyani@ee.iitm.ac.in

Abstract

Adversarial training (AT) incurs significant computational overhead, leading to growing interest in designing inherently robust architectures. We demonstrate that a carefully designed first layer of the neural network can serve as an implicit adversarial noise filter (ANF). This filter is created using a combination of large kernel size, increased convolution filters, and a maxpool operation. We show that integrating this filter as the first layer in architectures such as ResNet, VGG, and EfficientNet results in adversarially robust networks. Our approach achieves higher adversarial accuracies than existing natively robust architectures without AT and is competitive with adversarial-trained architectures across a wide range of datasets. Supporting our findings, we show that (a) the decision regions for our method have better margins, (b) the visualized loss surfaces are smoother, (c) the modified peak signal-to-noise ratio (mPSNR) values at the output of the ANF are higher, (d) high-frequency components are more attenuated, and (e) architectures incorporating ANF exhibit better denoising in Gaussian noise compared to baseline architectures.

1 Introduction

In adversarial attacks, a subtle perturbation to the input can cause the model to make erroneous predictions, leading to a significant drop in accuracies (Goodfellow, Shlens, and Szegedy 2014). This necessitates robust defense mechanisms, which have garnered significant research interest (Chakraborty et al. 2021). The white-box attacks like Fast Gradient Sign Method (FGSM) (Goodfellow, Shlens, and Szegedy 2014), Projected Gradient Descent (PGD) (Madry et al. 2017), and Auto Attack (AA) (Croce and Hein 2020b) have full access to model architecture and parameters which leads to significant performance loss. Several defense methods have been proposed to obtain robust Deep Neural Networks (DNN) (Dhillon et al. 2018; Xie et al. 2018) starting from adversarial training (AT) (Madry et al. 2017) to data preprocessing techniques (Zhang et al. 2019). Yan et al. (2021) enhances the adversarial robustness of convolutional neural networks by channel-wise importance-based feature selection during AT. Randomized defense mechanisms, introducing various perturbations during inference to disrupt adversarial attacks, have garnered great interest (Ma, Dong, and Xu 2024).

AT is widely used among different defense methods, and it primarily focuses on improving adversarial robustness by training the network architectures with adversarial samples. AT is computationally intensive because iterative algorithms like PGD or AA require multiple gradient computations per training iteration. In the case of AA, the algorithm uses a combination of different attack techniques (e.g., extended version of PGD, white-box, and decision-based attacks) to generate a wide range of adversarial examples. Running multiple attacks increases computation time as one needs to execute several different adversarial generation processes.

However, the influence of architectural components on adversarial robustness remains under explored compared to AT methods. Salman et al. (2020) propose a denoised smoothing procedure that acts as an image preprocessing technique and when used with a pretrained classifier, gives robustness against Gaussian noise corrupted inputs. The resultant smooth classifier is certifiably robust against ℓ_2-perturbations of its input. The effectiveness of this denoiser depends on the noise level with which it is trained. Verma et al. (2023) propose a certified Zeroth Order preprocessing technique to train a robust UNet denoiser to prepend to the black-box models to defend it against adversarial attacks on high-dimensional datasets. In contrast, the proposed method's focus is on getting implicit robustness from the architecture and defending against white-box attacks under the l_∞ norm without incorporating noise-specific training. Huang et al. (2021) investigate the impact of architectural elements such as topology, depth, and network width on the robustness of adversarially trained DNNs with a focus on ResNets to enhance the adversarial robustness. To circumvent the complexity associated with AT, Lukasik et al. (2023) investigate native robustness. They show that while the models trained with clean samples often prioritize high-frequency information, AT shifts focus from high-frequency to low-frequency details. By leveraging this fact, the authors enhance the native robustness through the regularization of filters with high frequencies. We show that one can achieve native robustness by designing an *Adversarial Noise Filter* (ANF) that inhibits the passage of adversarial noise, thereby achieving a robust neural network by using only our proposed ANF layer as the very first layer. Xie et al. (2019) use

network blocks to denoise features via AT; whereas we focus on native robustness, modifying only the first layer of the architecture for feature denoising.

In designing the ANF, which serves as the first layer of the neural network, we employ a combination of three operations: (a) increased kernel size, (b) higher number of filters, and (c) maxpool, and show that solely by utilizing these operations, one can achieve a substantial degree of robustness without using AT. We hypothesize and then show through extensive empirical results that the proposed combination of kernel, filters, and maxpool implicitly filters out the adversarial noise and reduces its propagation to other layers.

We provide a detailed discussion on ANF and incorporate it as the first layer in a variety of popular architectures. We then show that a robust first layer such as ours behaves like an adversarial noise filter/denoiser by analyzing a metric closely related to PSNR, which we call the mPSNR (modified Peak Signal-to-Noise Ratio). We show that mPSNR substantially improves once we pass the adversarially perturbed image through the proposed ANF-based first layer. In other words, the ANF serves as a first layer of defense and is sufficient to mitigate adversarial noise. Since our method is natively robust, we compare it with one of the most recent works in this area by Lukasik et al. (2023) and the other native robustness methods cited therein. Note that all existing works on native robustness typically change all/multiple layers of the neural network to be natively robust (Lukasik et al. 2023; Grabinski et al. 2022; Lopes et al. 2019). Ours is the first work to show that native robustness can be achieved by only changing the first layer of the architecture. Furthermore, we achieve substantially better robust accuracies than state-of-the-art (SOTA) native robust architectures despite changing only the very first layer. To further demonstrate the practical utility of our work, we design one single ANF/first layer and then incorporate that as the first layer for a variety of architectures, i.e., we do not change the number of kernels, filters, or maxpool operators across the architectures. Nevertheless, we observe substantially higher adversarial accuracies across architectures and attacks. For example, we are 15.56% and 26% better than Lukasik et al. (2023) against PGD and AA for ResNet20 on CIFAR10. We show results for FGSM, PGD, and AA on VGG, ResNets, WideResNets, and EfficientNet architectures across datasets such as CIFAR10, CIFAR100, TinyImagenet, and Imagenet.

To further investigate and compare our work with the corresponding baseline architectures, we study the loss surfaces and the contour plots inspired by Li et al. (2018) and observe that our method smoothens the adversarial loss surface and achieves lower adversarial loss. Moreover, utilizing the decision boundary visualization technique from Schulz, Hinder, and Hammer (2019), we study the change in decision boundaries with and without ANF, and this further showcases the robustness of our approach. Aside from the visible robustness demonstrated using the above two visualizations, we demonstrate the robustness of our method to high variance Gaussian noise. Further, we study the spectrum of adversarial noise when passed through ANF and show that ANF attenuates higher frequencies, which is what AT typically tries to achieve (Gilmer et al. 2019) and we also show how the proposed method acts as a feature denoiser.

2 Proposed Technique

Drawing inspiration from signal processing research, where one combats noise by designing filters resilient to noise, we propose combining larger kernels, a higher number of filters, and a maxpool operator to construct ANF. We use ANF as the first layer of the network architecture, and this restricts the passage of noise to the other layers.

2.1 Basic Components of ANF

Kernel size: Typically 3×3, 5×5, 7×7 are popular choice for kernel size (Öztürk et al. 2018). However, recently (Ding et al. 2022) have utilized sizes as large as 31×31 to achieve better performance with convolutional neural networks because large kernels have larger receptive fields, which implies an increase in information to subsequent convolution blocks. The larger kernels also allow the network to capture more global patterns in the input data, which can benefit tasks where context matters, such as image classification. Denoising processes also benefit from using larger kernels, as incorporating and averaging more pixels effectively reduces variance (Vogels et al. 2018). We leverage that larger kernels smooth the features and hence should also smooth/mitigate noise. Inspired by Ding et al. (2022) and Vogels et al. (2018), the existing kernel dimension is increased from $k \times k$ to $\tilde{k} \times \tilde{k}$ only for the ANF layer. Replacing every small kernel with a corresponding large kernel is prohibitively expensive. Fortunately, in our work, we change only the first layer; hence, the associated computational cost is less.

Increased number of filters: More filters allow the network to learn a diverse set of features at each layer, which helps the network generalize better to different types of inputs and variations in the data, making it more robust to changes in the inputs. Let the number of filters in the baseline be F, then the number of filters in the first convolution layer (i.e., ANF) is increased to $\tilde{F}$.

Maxpool: Maxpool is a downsampling operation that reduces the spatial dimensions of the feature map while retaining the essential information. It is applied after convolution to reduce computational complexity and control overfitting. Maxpool divides the input into non-overlapping rectangular regions and outputs the maximum value from each region. This reduces the size of the feature map while preserving the most prominent features. Maxpool serves as a down-sampling operator, and hence, we hypothesize that it can downsample or reduce the impact of adversarial noise. See Sec. 5, where we demonstrate how maxpool is crucial for the ANF design.

To show that the combination of the above three operations leads to denoising, we measure the denoising capability of the ANF by calculating the noise in the feature before and after the ANF.

2.2 Measure of Denoising

Peak Signal-to-Noise Ratio (PSNR) is a metric that quantifies the ratio between the maximum possible power of the

original image and the power of noise that corrupts the image, thereby assessing the fidelity of the reconstructed image. As the dimensions of all the input samples are the same, we denote the dimension of sample $\mathbf{x}_i$ by $[W, H, D]$ where D is the number of features in $\mathbf{x}_i$; W and H are the width and height of a single input feature $\mathbf{x}_i^d$. Let the perturbed image be denoted as $\tilde{\mathbf{x}}_i$. The transformed feature for $\mathbf{x}_i$ and $\tilde{\mathbf{x}}_i$ after the ANF is denoted by $\mathbf{h}_i$ and $\tilde{\mathbf{h}}_i$, respectively.

The PSNR is typically computed for one particular input feature. For the adversarial attacks, all N features may not always have non-zero values for the difference between the actual feature $\mathbf{h}_i$ and the corrupted feature $\tilde{\mathbf{h}}_i$. This results in zero in the denominator if one uses PSNR. To handle such a situation, we first compute the inverse of the PSNR as shown below for every sample, where the numerator is a measure of the perturbation in the feature, and the denominator is the peak value of the input image:

$$\text{invPSNR}_i = \frac{\sqrt{\frac{1}{DHW}\sum_{d=1}^{D}\sum_{h=1}^{H}\sum_{w=1}^{W}(h_{idhw} - \tilde{h}_{idhw})^2}}{\frac{1}{D}\sum_{d=1}^{D}\max_{1\le m\le H,\, 1\le n\le D}\mathbf{x}_{imn}^d}. \quad (1)$$

This is followed by an average over N samples and then taking an inverse as follows:

$$\text{mPSNR} = \frac{1}{\frac{1}{N}\sum_{i=1}^{N}\text{invPSNR}_i} \quad (2)$$

We define the above metric as modified PSNR (mPSNR) and measure it at the input layer and after the ANF to understand how well the ANF can denoise the perturbed image. In the next section, we briefly describe various adversarial attacks.

3 Adversarial Attacks

Let $\mathbf{x}_i \in \mathbb{R}^d$ be an input data point belonging to class c_o, and y_i is the corresponding true label. Let the loss function be given by $L(\mathbf{x}, y; \mathbf{w})$, where $\mathbf{x}$ is the input, y is the corresponding true label, and $\mathbf{w}$ is the network parameters. An adversarial attack is a mapping $\mathcal{A} : \mathbb{R}^d \to \mathbb{R}^d$ such that the perturbed data $\tilde{\mathbf{x}}_i = \mathcal{A}(\mathbf{x}_i)$ is misclassified as an adversarial class c_t. We focus on white-box attacks, which assume full knowledge of the classifier and require knowing all weight values and the network structure. We show the results for three popular white-box attacks, i.e., FGSM, PGD, and AA.

FGSM: The loss function is maximized, subject to an upper bound on the input perturbation, i.e., $||\tilde{\mathbf{x}}_i - \mathbf{x}_i||_\infty \le \epsilon$. The FGSM creates an attack $\tilde{\mathbf{x}}_i = \mathbf{x}_i + \epsilon\ \text{sign}(\nabla_{\mathbf{x}} L(\mathbf{x}_i, y_i; \mathbf{w}))$. Here, $\nabla_{\mathbf{x}} L(\mathbf{x}_i, y_i; \mathbf{w})$ is interpreted as the gradient of L w.r.t. $\mathbf{x}$ evaluated at $\mathbf{x}_i$.

PGD: PGD iteratively perturbs input data to maximize the model's loss $L(.; \mathbf{w})$ to find more effective adversarial examples compared to non-iterative attacks. The adversarial sample is found in the same way as FGSM but iteratively starting from $\tilde{\mathbf{x}}^{(0)} = \mathbf{x}_i$, one updates the sample according to $\tilde{\mathbf{x}}^{(k+1)} = \text{Proj}_{B_\xi(\mathbf{x}_i)}\left(\tilde{\mathbf{x}}^{(k)} + \eta\ \text{sign}\left(\nabla_{\mathbf{x}} L(\tilde{\mathbf{x}}^{(k)}, y_i; \mathbf{w})\right)\right)$ for $k = 0, 1, \ldots, K-1$, where K is the number of PGD iterations, η is the step size, ξ is the norm-ball radius, and $\text{Proj}_{B_\xi(\mathbf{x})}(\mathbf{z}) = \arg\min_{\mathbf{x}' \in B_\xi(\mathbf{x})} \|\mathbf{z} - \mathbf{x}'\|_p$ (Madry et al. 2017). The final adversarial sample is $\tilde{\mathbf{x}}_i = \tilde{\mathbf{x}}^{(K)}$. The PGD attack has been widely used to evaluate machine learning models' robustness and develop more effective defense mechanisms against adversarial attacks (Huang et al. 2023).

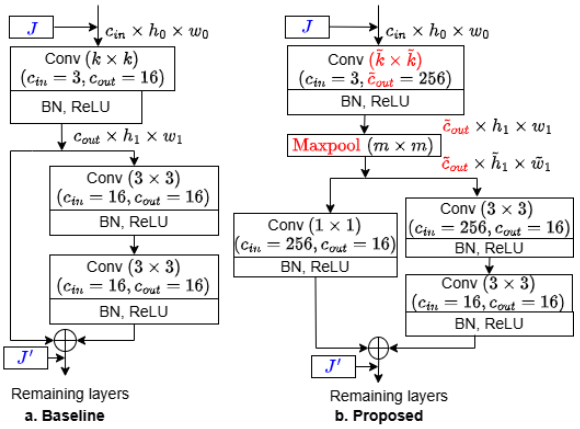


Figure 1: ANF with ResNet20

AA: In Auto attacks, which is one of the most powerful attacks, two different versions of PGD i.e. AutoPGD with Cross Entropy loss and Difference of Logits Ratio loss proposed in Croce and Hein (2020b) are combined with white-box FAB attack (Croce and Hein 2020a), and black-box Square Attack (Andriushchenko et al. 2020) to form a parameter-free ensemble of complementary attacks.

4 Results and Discussions

We now discuss how ANF is incorporated as the first layer of ResNet, WideResNet, VGG, and EfficientNet architectures based on their structures[1].[2] For comparison with very recent work on native robustness (Lukasik et al. 2023), we showcase results on the same baselines, i.e., ResNet20 and EfficientNet-B0. In line with Lukasik et al. (2023), the attack strength for all the attacks is considered to be $\epsilon = 1/255$, and the number of iterations for the PGD attack is considered to be 40, unless specified otherwise. We have used ℓ_∞ norm for AA apart from the default hyper-parameters[3].

4.1 ANF With ResNet and WRN

The ANF in ResNet20 for CIFAR10 and CIFAR100 classification is represented by Fig. 1, where the differences be-

[1]The baseline performance for each architecture has been sourced from the following references;
ResNet20: https://github.com/akamaster/pytorch_resnet_cifar10,
WRN-28-10: https://github.com/bmsookim/wide-resnet.pytorch,
VGG16 and ResNet50: https://github.com/kuangliu/pytorch-cifar/blob/master/models/resnet.py,
EfficientNet-B0: https://github.com/jovitalukasik/filter_freq_reg

[2]Codes for ANF available at https://github.com/janani-suresh-97/first-line-defence.git

[3]https://github.com/fra31/auto-attack.

Arch.	FGSM	PGD	AA	CorptnA	Clean acc
ResNet20 with CIFAR10					
Baseline	42.86	27.03	12.41	73.32	**91.26**
ANF	**59.56**	**59.98**	**55.14**	**78.43**	83.09
(Lukasik et al. 2023)	53.12	44.42	29.14	–	90.54
AT (Lukasik et al. 2023)	49.93	46.34	36.47	–	70.31
FLP (Grabinski et al. 2022)	52.49	30.25	8.48	–	91.52
GA (Lopes et al. 2019)	50.36	31.50	11.38	–	91.29
ResNet20 with CIFAR100					
Baseline	12.28	3.83	1.01	34.93	**65.34**
ANF	**26.8**	**26.43**	**21.58**	**48.13**	54.86
Lukasik et al. (2023)	17.2	12.24	5.11	–	58.19
WRN-28-10 with CIFAR100					
Baseline	29.51	20.67	10.88	51.1	**77.9**
ANF	**43.8**	**44.38**	**41.09**	**58.24**	62.22
VGG16 with CIFAR10					
Baseline	60.60	54.94	43.83	**75.06**	**92.42**
ANF	**62.27**	**63.11**	**60.42**	67.14	80.44
EfficientNet-B0 with CIFAR10					
Baseline	53.05	52.20	42.24	44.08	**92.29**
ANF	**64.95**	**66.23**	**62.27**	**80.18**	87.14
Lukasik et al. (2023)	57.83	59.68	53.50	–	89.18

Table 1: Comparison of different architectures with CIFAR10 and CIFAR100. Top accuracies are highlighted.

Arch.	FGSM	PGD	AA	Clean acc
ResNet50 with TinyImagenet				
Baseline	34.07	31.82	24.08	**69.7**
ANF	**35.64**	**35.17**	**28.08**	60.98
ResNet50 with ImageNet				
Baseline with AT	42.36	26.17	1.05	**64.37**
ANF with AT	**55.09**	**55.46**	**52.95**	61.67
AT (Lukasik et al. 2023)	36	37	24.32	58.09

Table 2: Comparison of ResNet50 with TinyImagenet and ImageNet. Top accuracies are highlighted.

tween the baseline and the proposed are highlighted in red. The kernel size is increased from $k \times k$ to $\tilde{k} \times \tilde{k}$, where $k = 3$, and $\tilde{k} = 15$. The number of output features from the first convolution layer is increased from $F = c_{out} = 16$ to $\tilde{F} = \tilde{c}_{out} = 256$. Further, the maxpool is introduced with a kernel size of $m = 5$ and a stride of 1.

The ANF in ResNet50 used for TinyImageNet classification includes some modifications to the first layer of the baseline architecture compared to ResNet20. For ResNet50, a key difference in the first layer is that the number of filters is 64 instead of 16. In our proposed architectures, we increase the filters to 256, the kernel size increased from 3×3 to 15×15, and use a max-pooling operation with a 5×5 dimension. The WRN considered in our work for CIFAR100 classification has pre-activation, which implies that the batch normalization (BN) and ReLU are before the convolution in every basic block. Otherwise, the first layer transformation is just the same as ResNet20.

In ResNets and WRNs, a common architectural characteristic includes residual blocks, typically positioned after the first convolutional layer. We convert this first convolution layer to ANF by introducing maxpool and by increasing the kernel size and the number of filters. To explain it further, the image samples in CIFAR10, CIFAR100, TinyImagenet, and Imagenet datasets have three features, e.g., 'R', 'G', and 'B'; therefore $c_{in} = 3$. For the ResNet and WRN architectures we have chosen, the number of output features from the first convolution layer is 16, which means $c_{out} = 16$. Note that, with the proposed ANF for the ResNet and WRN architectures, the number of features input to the first residual block changes from 16 to 256 as we increase the number of filters from 16 to 256 at the first convolution layer. So, to match the dimensions in the succeeding layers, the signal in the direct path is passed via a 1×1 convolution before summing it up with the features from the residual path. The usual ResNet architectures already have this 1×1 convolution, which matches the dimensions before adding every residual block.

From Table 1, for ResNet20 with CIFAR10 dataset, the proposed ANF achieves PGD and AA accuracy of 59.98% and 55.14% respectively compared to 27.03% and 12.41% in the baseline. By using frequency filter regularization Lukasik et al. (2023) show that their method outperforms frequencylowcut-pooling (FLP) (Grabinski et al. 2022) and patch Gaussian augmentation (GA) (Lopes et al. 2019) by a significant margin in terms of native robustness. AT (Lukasik et al. 2023) in Table 1 represents AT results that the authors have produced by training their architecture with FGSM attacks with the attack strength of $\epsilon = 8/255$ for CIFAR10. However, our proposed ANF significantly outperforms both Lukasik et al. (2023) and AT in (Lukasik et al. 2023) without using AT. We also evaluate the accuracy of the proposed ANF against corruption noise (CorptnA), which is Gaussian noise with a standard deviation of $\sigma = 16/255$. We achieve an accuracy of 78.43% against corruption noise with the help of ANF, whereas the baseline has an accuracy of 73.32%. A model that has better adversarial accuracy often suffers from clean accuracy. However, with ANF, one does not have to perform AT, and therefore, we can maintain a clean accuracy of 83.09% compared to 70.31 with AT. Our approach achieves an accuracy of 21.58% compared to 5.11% reported by Lukasik et al. (2023) for ResNet20 on CIFAR100 for AA, a powerful attack. Additionally, training time comparisons are detailed in Sec. 8 of the supplementary file (Suppl).[4] For all ResNet variants and WRN-28-10, we observe significant accuracy improvement for all attacks and across datasets in Tables 1 and 2 when compared with the corresponding baselines and Lukasik et al. (2023).

Effect of maxpool stride: In the ANF filter, the maxpool operation has stride of one. We have also considered maxpool with stride of two and given results in Sec. 3 of

[4]The Suppl file is found in the arxiv version of the paper.

the Suppl. Coarser downsampling with stride two provides even better adversarial accuracy than stride one. Further, the performance of ResNet20 is tested with attacks of different strengths and given in Sec. 3 of the Suppl.

ImageNet dataset: Models trained with ImageNet dataset barely withstand adversarial attacks (Lukasik et al. 2023) and even ANF alone cannot provide native robustness. However, ANF provides significant robustness when trained with adversarial samples. We use ResNet50, where the variant designed for ImageNet has a kernel size of 7×7 in the first layer, 64 filters, and a max-pooling kernel size of 3×3. Our proposed architecture with ANF modifies these parameters to a kernel size of 15×15, 256 filters, and a max-pooling kernel size of 5×5 in the first layer. In line with the work by Lukasik et al. (2023), who performed AT on ImageNet using ResNet50, we also trained the ANF incorporated ResNet50 adversarially with a 1-step PGD attack at $\epsilon = 4/255$. Our results demonstrate a significant improvement in accuracy achieving 55.09%, 55.46%, and 52.95%, compared to 36%, 37% and 24.32% of AT (Lukasik et al. 2023) for FGSM, PGD, and AA, respectively, as shown in Table 2. These results were obtained using early stopping at around 200 epochs.

4.2 ANF With VGG

The VGG16 architecture was proposed with multiple maxpool layers, primarily to reduce the spatial dimensions of the input feature maps, thus decreasing the computational complexity of the network. As the ANF has a maxpool layer with a kernel size of $m = 5$, the spatial feature dimension is already reduced after ANF. If the remaining maxpool layers from the baseline architecture are retained, the dimensions of the features decrease significantly in the deeper layers, leading to a substantial reduction in the feature information transmitted to subsequent layers. Therefore, when we integrate ANF with VGG, we remove the other maxpool layers and use only the ANF, which includes a maxpool layer with a stride of two to keep the input feature dimensions similar to the baseline[5]. From Table 1, we observe that the robust accuracy of the VGG16 baseline is comparatively better than other baselines because of the non-linearity caused by maxpool and the usage of multiple layers of a higher number of convolution filters. This further justifies the usage of maxpool in the ANF for robustness. Compared to this baseline, the proposed architecture with ANF has the following differences. The baseline has five maxpool layers, each having a kernel size $m = 2$ and a stride of 2. For ANF, we use only one maxpool layer as a part of ANF just after the first convolution layer with kernel size $m = 5$ and a stride of 2. The number of output features for the filters at the first convolution layer is increased from 64 to 256. The kernel size is increased from 3 to 15.

With ANF, we achieve accuracies of 63.11% for PGD and 60.42% for AA compared to the baseline accuracies of 54.94% for PGD and 43.83% for AA. The resultant architecture after integrating ANF and the impact of different stride values for maxpool is shown in Sec. 1 in the Suppl.

[5]For VGG16, we opt for a stride of 2 for maxpool to have a similar complexity as the baseline while for other architectures the maxpool stride is 1.

4.3 ANF With EfficentNet-B0

The baseline EfficientNet-B0 has 32 filters in the first layer with a kernel size of 3×3, and there is no maxpool in that layer. The proposed architecture converts the first layer to ANF with 256 filters with a kernel size of 15×15. In the ANF layer, the convolution, batchnorm, and the swish activation are followed by maxpool operation with a kernel size of 5×5. Although the clean accuracy of ANF is marginally lesser than the baseline, the baseline exhibits significant vulnerability to adversarial attacks. The proposed ANF demonstrates superior robustness with minimal degradation in clean accuracy, as seen from Table 1.

5 Why Does the ANF Work?

In the previous section, we have shown through extensive experiments that the ANF-based architectures exhibit significant implicit robustness to adversarial attacks without requiring AT. Further, all our architectural changes are restricted to only the first layer. In all architectures, we have chosen convolution kernel size (K) as 15×15 , filter number (F) as 256, and maxpool (M) as 5×5 to show even without tuning these numbers according to the architecture, one can still get excellent adversarial robustness. In Sec. 2 in the Suppl., we detail the impact of tuning K, F, and M values in the ANF for PGD, FGSM, and AA, respectively, and sometimes even better results are obtained.

In this section, we would like to better understand the workings of ANF using various metrics and visualization tools. We focus on the ResNet20 architecture for a detailed analysis, as Lukasik et al. (2023) extensively use ResNet20 in their study and restricts their investigation to the CIFAR10 dataset. Unless otherwise stated, the baseline and the proposed architecture with ANF are models where the data is not normalized. Note that not much change in performance was observed upon introducing data normalization. In Table 3, we consider eight different scenarios for ResNet20 with combinations of three choices: (i) increased kernel size (K), (ii) increased number of filters (F), and (iii) maxpool (M).

mPSNR: To study the effectiveness of the filter to denoise the features, we study the mPSNR before and after the ANF. Just after the ANF, the dimension of the features is not the same as the baseline, as shown in Fig. 1. Therefore, we report the mPSNR only after the first residual block when the feature dimension is the same as the baseline. We report the mPSNR at the input, i.e., at point J, and after the first residual block, i.e., J' in Fig. 1. The mPSNR values are calculated with all the 10k test samples of the CIFAR10 dataset. Note that the initial mPSNR at the input, i.e., at J, is relatively high, as expected, as adversarial noise is a minute perturbation. However, these are carefully structured perturbations, and for them to be effective, they should lead to a drop in mPSNR after propagating through the layers of the network. This is an expected trend and is also evident in the last column of Table 3. However, the drop in mPSNR is the least for our architecture, demonstrating its implicit adversarial

noise-denoising capability. The mPSNR at the input (at J) is almost in the same range for all eight versions. However, the mPSNR at J' varies across different versions. ANF with only maxpool in Type 1 has an mPSNR of 65.65, and the mPSNR increases to 78.17 and 89.92 by combining operations K and F with M, respectively. Further, the mPSNR is maximum when K, F, and M are combined for designing ANF. One can observe that the most robust architecture is one where the mPSNR value is maximum at J'. The trend in mPSNR values across different configurations corroborates the PGD accuracy as shown in Table 3.

We present the clean accuracy and the adversarial accuracy when tested with a PGD attack. While looking at the individual contribution of the three operations, K, F, and M in Type 1, Type 2, and Type 4, respectively, we observe that Type 1 and Type 4 have almost the same adversarial accuracy, i.e., 45.37% and 45.54%, respectively, against PGD attack which is higher than Type 2, i.e., 29.91%. This indicates that among the three operations, introducing maxpool and increasing kernel size gives the highest robustness. In Type 3, when maxpool is combined with increased filter operations, it improves the robustness further to 49.92%. In Type 5, when kernel size is increased along with the introduction of maxpool, the adversarial accuracy improves and becomes 51.71. Using all three operations, K, F, and M, together in Type 7 improves the accuracy to 59.93%, significantly higher than 27.22% of the baseline.

Arch.	K	F	M	PGD	Clean acc	mPSNR at J	mPSNR at J'
Baseline	✗	✗	✗	27.22	91.26	160.42	22.66
Type 1	✗	✗	✓	**45.37**	88.35	158.18	**65.65**
Type 2	✗	✓	✗	29.91	91.16	160.23	24.23
Type 3	✗	✓	✓	**49.92**	89.72	156.63	**61.79**
Type 4	✓	✗	✗	45.54	85.68	156.08	59.08
Type 5	✓	✗	✓	**51.71**	80.99	153.06	**78.17**
Type 6	✓	✓	✗	40.12	86.64	157.47	29.80
Type 7	✓	✓	✓	**59.93**	83.09	151.62	**89.92**

Table 3: mPSNR in ResNet20 for CIFAR10. For column K, ✓increases the kernel size from 3×3 to 15×15; for column F, ✓increases filters from 16 to 256; for column M, ✓introduces a 5×5 maxpool operation.

The mPSNR values clearly demonstrate that the proposed ANF effectively filters out a significant amount of adversarial noise. To better understand the impact of this filtering, we will visualize the decision surface for both the baseline and the proposed approach next.

5.1 Visualization of the Decision Surface

In the DeepView method, Schulz, Hinder, and Hammer (2019) propose a discriminative dimensionality reduction (DR) method called Fisher UMAP, enabling the DR to focus on the aspects of the data that are relevant to the classifier. They also develop a scheme based on inverse dimensionality reduction to obtain predictions only for a relevant subspace, which is then used to visualize the decision function in two dimensions. The steps to perform the visualization are

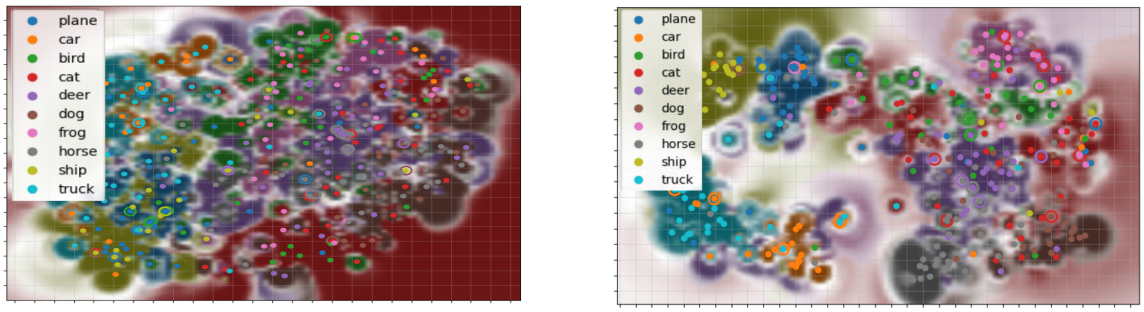
1. Apply Fisher UMAP, based on the underlying deep network, to project points $\mathbf{x}_i$ to two dimensions, yielding $\mathbf{y}_i = \pi(\mathbf{x}_i)$ so that π preserves as much information as possible in $\mathbf{y}_1, \ldots, \mathbf{y}_n \in \mathbb{R}^d, d = 2, 3$ from $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbf{S}$.
2. Create a tight regular grid of samples $\mathbf{r}_i$ in 2D space and map it to high dimensional space by $\mathbf{s}_i = \pi^{-1}(\mathbf{r}_i)$
3. Apply the neural network f to $\mathbf{s}_i$ in order to obtain predictions and certainties.
4. Visualize the label with the entropy of the certainty for each position $\mathbf{r}_i$ in the background of the projection space to get an approximation of the decision function.

For better representation, both baseline and proposed architecture with ANF are trained with normalized data without any change in performance. Fig. 2a illustrates the decision boundary of the baseline architecture using 300 adversarial samples from the test data. In comparison, Fig. 2b displays the decision boundary of the proposed architecture with the same 300 samples. Each point represents an image, with its colour indicating the original label. The background colours are the predictions learned by UMAP, with intensity reflecting the confidence level of the predictions. Additionally, there are encircled representations where the outer circle of the encircled point has the colour of the class predicted by the model. The encircled points are present when the model predictions do not match the UMAP representation, i.e., the outer circle colours are different than the background colour. Among these encircled points, two cases may arise; in the first case, the model prediction is different than the actual label, in which case the outer circle is of a different colour than the colour of the inner point, and in the second case, the model prediction is same as the actual label, and the inner point and the outer circle have the same colours. Ideally, the colour of a particular point should match the background colour for the UMAP to be a correct representation of the decision boundaries. There are very few encircled points in all the figures of Fig. 2, indicating that UMAP is a good representation of the decision region.

When the samples are perturbed with adversarial noise, most of the samples are misclassified with baseline, and the decision regions are scattered, as seen from Fig. 2a. Misclassified samples are those whose colours are different from the background colour in the case of non-encircled points and different from the outer circle colour for the encircled points. However, a significantly smaller number of adversarial points are misclassified while using ANF, as shown in Fig. 2b. The corresponding plots with only clean samples are provided in Sec. 4 in the Suppl.

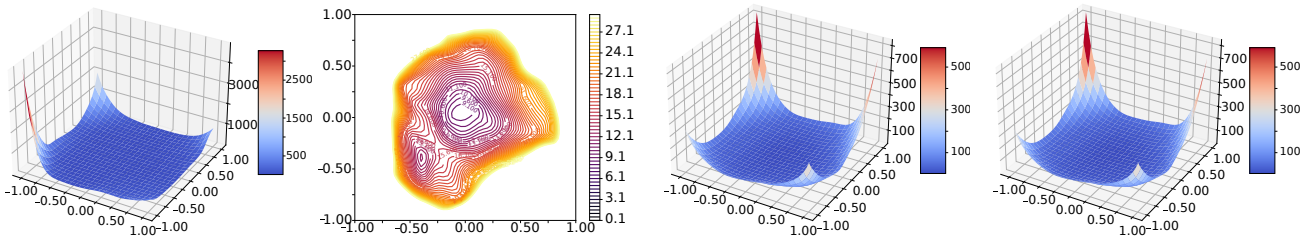
5.2 Visualization of Loss Surfaces

We look at the loss surface visualization utilizing the ideas in (Li et al. 2018), which demonstrate that visualization is only possible in lower dimensions, such as using line or surface plots. After the training, the clean loss surface should take the form of a convex surface and have minima at the center of the plot $\mathbf{w}^*$ for both the baseline and the proposed architecture with ANF. In Fig. 3, we study the loss surface when



(a) Baseline with adversarial samples (b) ANF with adversarial samples

Figure 2: Visualization of the decision regions



(a) Baseline loss surfaces (b) Baseline contour plots (c) ANF loss surfaces (d) ANF contour plots

Figure 3: Visualization of 3D loss surface (a,c) and contour plots (b, d) with adversarial samples.

tested with adversarial samples. Fig. 3a and Fig. 3c represent the 3D adversarial loss surface of baseline and proposed ANF, respectively, demonstrating that ANF has a smoother loss surface and better convergence. Further, we present the contour plots of baseline and ANF in Fig. 3b and Fig. 3d, respectively. The contour plots depict that the loss with adversarial samples in the proposed model is 3.10, which is much less compared to the loss in the baseline model, which is 9.10. Moreover, the contour plot of the adversarial loss for baseline demonstrates more than one minima, whereas the proposed ANF has one distinct minima.

5.3 Spectrum of Adversarially Perturbed Features

This study aims to elucidate the impact of ANF on noise and assess its potential as a filter for attenuating high-intensity frequency components. We observed that ANF effectively suppresses high-frequency components, unlike the baseline model. Due to space constraints, the spectrum with adversarial noise and corruption noise for ANF is discussed in Sec. 5 of the Suppl. Adversarial noise is considered structured noise because it is intentionally designed to exploit model vulnerabilities through targeted optimization, inducing specific errors. In contrast, unstructured noise, such as random or normal noise, lacks a specific pattern or intent, being randomly generated from a probability distribution without targeting model weaknesses. Gilmer et al. (2019) claim that improving adversarial robustness against structured noise improves the robustness against image corruption, too. We observe that ANF does not only have better adversarial accuracy with respect to FGSM, PGD, and AA but is also resilient towards corruption noise, as highlighted in Table 2. Accuracies against corruption noise for various standard deviation σ are also highlighted in Sec. 5 of the Suppl.

5.4 Feature Denoising Using ANF

Xie et al. (2019) suggest that the adversarial perturbations on images lead to noise in the features constructed by these networks. The authors have developed new network architectures that increase adversarial robustness by performing feature denoising. We show that ANF also removes the noises in the features, which we detail in Sec. 6 of the Suppl.

6 Conclusion

We have shown that adversarial robustness can be achieved by designing a robust first layer, which acts as a first line of defense. We term the proposed first layer adversarial noise filter since it filters/suppresses adversarial noise implicitly. The ANF is composed of a combination of simple existing operations: a larger kernel, a higher number of filters, and a maxpool operation. We show that it leads to (a) higher adversarial accuracies for FGSM, PGD, and AA, (b) higher mPSNR, (c) better loss surfaces, d) robust decision boundaries, e) more robustness to corruption noise, etc., for a wide range of datasets and architectures. Note that all the SOTA architectures in native robustness typically change multiple layers, whereas, with only the first layer, we significantly outperform existing SOTA natively robust architectures.

References

Andriushchenko, M.; Croce, F.; Flammarion, N.; and Hein, M. 2020. Square attack: a query-efficient black-box adversarial attack via random search. In *European conference on computer vision*, 484–501. Springer.

Chakraborty, A.; Alam, M.; Dey, V.; Chattopadhyay, A.; and Mukhopadhyay, D. 2021. A survey on adversarial attacks and defences. *CAAI Transactions on Intelligence Technology*, 6(1): 25–45.

Croce, F.; and Hein, M. 2020a. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*, 2196–2205. PMLR.

Croce, F.; and Hein, M. 2020b. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, 2206–2216. PMLR.

Dhillon, G. S.; Azizzadenesheli, K.; Bernstein, J. D.; Kossaifi, J.; Khanna, A.; Lipton, Z. C.; and Anandkumar, A. 2018. Stochastic activation pruning for robust adversarial defense. In *International Conference on Learning Representations*.

Ding, X.; Zhang, X.; Han, J.; and Ding, G. 2022. Scaling Up Your Kernels to 31x31: Revisiting Large Kernel Design in CNNs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 11963–11975.

Gilmer, J.; Ford, N.; Carlini, N.; and Cubuk, E. 2019. Adversarial examples are a natural consequence of test error in noise. In *International Conference on Machine Learning*, 2280–2289. PMLR.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Grabinski, J.; Jung, S.; Keuper, J.; and Keuper, M. 2022. Frequencylowcut pooling-plug and play against catastrophic overfitting. In *European Conference on Computer Vision*, 36–57. Springer.

Huang, H.; Wang, Y.; Erfani, S.; Gu, Q.; Bailey, J.; and Ma, X. 2021. Exploring architectural ingredients of adversarially robust deep neural networks. *Advances in Neural Information Processing Systems*, 34: 5545–5559.

Huang, S.; Lu, Z.; Deb, K.; and Boddeti, V. N. 2023. Revisiting residual networks for adversarial robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8202–8211.

Li, H.; Xu, Z.; Taylor, G.; Studer, C.; and Goldstein, T. 2018. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31.

Lopes, R. G.; Yin, D.; Poole, B.; Gilmer, J.; and Cubuk, E. D. 2019. Improving robustness without sacrificing accuracy with patch gaussian augmentation. *arXiv preprint arXiv:1906.02611*.

Lukasik, J.; Gavrikov, P.; Keuper, J.; and Keuper, M. 2023. Improving Native CNN Robustness with Filter Frequency Regularization. *Transactions on Machine Learning Research*.

Ma, Y.; Dong, M.; and Xu, C. 2024. Adversarial Robustness through Random Weight Sampling. *Advances in Neural Information Processing Systems*, 36.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Öztürk, Ş.; Özkaya, U.; Akdemir, B.; and Seyfi, L. 2018. Convolution kernel size effect on convolutional neural network in histopathological image processing applications. In *2018 International Symposium on Fundamentals of Electrical Engineering (ISFEE)*, 1–5. IEEE.

Salman, H.; Sun, M.; Yang, G.; Kapoor, A.; and Kolter, J. Z. 2020. Denoised smoothing: A provable defense for pretrained classifiers. *Advances in Neural Information Processing Systems*, 33: 21945–21957.

Schulz, A.; Hinder, F.; and Hammer, B. 2019. Deepview: Visualizing classification boundaries of deep neural networks as scatter plots using discriminative dimensionality reduction. *arXiv preprint arXiv:1909.09154*.

Verma, A.; Subramanyam, A.; Bangar, S.; Lal, N.; Shah, R. R.; and Satoh, S. 2023. Certified Zeroth-order Black-Box Defense with Robust UNet Denoiser. *arXiv preprint arXiv:2304.06430*.

Vogels, T.; Rousselle, F.; McWilliams, B.; Röthlin, G.; Harvill, A.; Adler, D.; Meyer, M.; and Novák, J. 2018. Denoising with kernel prediction and asymmetric loss functions. *ACM Transactions on Graphics (TOG)*, 37(4): 1–15.

Xie, C.; Wang, J.; Zhang, Z.; Ren, Z.; and Yuille, A. 2018. Mitigating Adversarial Effects Through Randomization. In *International Conference on Learning Representations*.

Xie, C.; Wu, Y.; Maaten, L. v. d.; Yuille, A. L.; and He, K. 2019. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 501–509.

Yan, H.; Zhang, J.; Niu, G.; Feng, J.; Tan, V.; and Sugiyama, M. 2021. Cifs: Improving adversarial robustness of cnns via channel-wise importance-based feature selection. In *International Conference on Machine Learning*, 11693–11703. PMLR.

Zhang, H.; Yu, Y.; Jiao, J.; Xing, E.; El Ghaoui, L.; and Jordan, M. 2019. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*, 7472–7482. PMLR.