STARS

University of Central Florida
STARS

Electronic Theses and Dissertations, 2020-

2022

Methods For Defending Neural Networks Against Adversarial Attacks

Sharvil Shah
University of Central Florida

 Part of the Computer Sciences Commons
Find similar works at: https://stars.library.ucf.edu/etd2020
University of Central Florida Libraries http://library.ucf.edu

This Masters Thesis (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Electronic Theses and Dissertations, 2020- by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation
Shah, Sharvil, "Methods For Defending Neural Networks Against Adversarial Attacks" (2022). *Electronic Theses and Dissertations, 2020-*. 1287.
https://stars.library.ucf.edu/etd2020/1287

METHODS FOR DEFENDING NEURAL NETWORKS AGAINST ADVERSARIAL ATTACKS.

by

SHARVIL VIPUL SHAH
B.Tech Ahmedabad University, 2017

A thesis submitted in partial fulfilment of the requirements
for the degree of Master of Science
in the Department of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2022

Major Professor: Rickard Ewetz

© 2022 Sharvil Vipul Shah

ABSTRACT

Convolutional Neural Networks (CNNs) have been at the frontier of the revolution within the field of computer vision. Since the advent of AlexNet in 2012, neural networks with CNN architectures have surpassed human-level capabilities for many cognitive tasks. As the neural networks are integrated in many safety critical applications such as autonomous vehicles, it is critical that they are robust and resilient to errors. Unfortunately, it has recently been observed that deep neural network models are susceptible to adversarial perturbations which are imperceptible to human vision. In this thesis, we propose a solution to defend neural networks against white box adversarial attacks. The proposed defense is based on activation pattern analysis in the frequency domain. The technique is evaluated and compared with state-of-the-art techniques on the CIFAR-10 dataset.

I would like to dedicate this work to my parents and my brother who have have always supported me in my goals to conduct research.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Rickard Ewetz for his guidance and support throughout the duration of my thesis. I would also like to thank my thesis advisory committee members Dr. Yogesh Singh Rawat and Dr. Sharma Thankachan for their critical input.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOTATION

x : Original Image

x^{adv} : Adversarial Image

f : Classifier

y : Label of image x

$f(x)$: Output logits

y_{pred} : Predicted label of the model

y^{adv} : Output label for adversarial image

ϵ : Perturbation budget

$||.||_p$: Lp norm

$\mathcal{L}()$: Cross-entropy loss function

$\nabla_x \mathcal{L}()$: Gradient of loss with respect to the input image

KL(): Kullback Leibler divergence

CHAPTER 1: INTRODUCTION

The field of Computer Vision has seen significant improvements in the past 2 decades across variety of tasks [Deng et al., 2009]. Most of these improvements have been led by the Convolutional Neural Network(CNN) architecture. CNNs have also exceeded human performance in some of the tasks. This progress and applications of CNNs is not only limited to the academia. CNN based models have also been used by the industry to solve similar Computer Vision problems on a much larger scale like autonomous driving. Some of these real-world applications require extreme safety critical systems. It has been long assumed that due its good empirical performance CNNs are safe-to-deploy in wild. However, since past few years this assumption has come under a scrutiny. These otherwise accurate models fail miserably when the input is perturbed smartly. These perturbations are so small in the pixel values that it's almost imperceptible to us humans and yet the CNNs fail to predict the same output.

These almost invisible perturbations are called adversarial attacks. This perturbation is added to the original image to create the adversarial image. One such example of adversarial image is shown in Figure 1.1. The adversarial image in the Figure 1.1b has a l_∞ distance of 8/255 which means the maximum perturbation to original value for any pixel is 8. These adversarial image under a L_p norm x_p^{adv} of an input image x can be mathematically defined as follows.

$$\exists x_p^{adv} : argmax(f(x)) \neq argmax(f(x_p^{adv})) \wedge ||x^{adv} - x||_p \leq \epsilon \tag{1.1}$$

where $f(x)$ is output logits of a Neural Network and $argmax(f(x))$ will give us the predicted label y_{pred}. This definition can be applied to other Machine Learning models but in this work we're only focused on the Neural Network models applied at test time. Also, adversarial attacks

(a) (b)

Figure 1.1: Example of image and its adversarial counterpart from ImageNet dataset [Deng et al., 2009]. The image in (a) was correctly classified by ResNet50 [He et al., 2016] as "Tiger". However, image in (b) was classified as "Egyptian Cat" by the same network.

do not necessarily have to be inside a hyper-sphere around an input image. It can also be an image for which humans can classify it as "None" class but the model outputs a label with high confidence. However, we're only concerned with the adversarial examples defined in Equation 1.1. They're called the ϵ-adversarial examples and we'll refer to them simply as adversarial examples.

Finding an adversarial example around a given input sample using Equation 1.1 is intractable since there can be infinite points inside the ϵ-radius hyper-sphere around the sample. To resolve this a lot of the literature use a surrogate loss function to generate the adversarial example as follows.

$$x_p^{adv} = \underset{x'}{\text{argmax}}\ \mathcal{L}(f(x'), y),\ ||x' - x||_p \leq \epsilon \tag{1.2}$$

Equation 1.2 tries to find a sample inside the hyper-sphere around the original sample which maximize the cross-entropy loss of the model. This approximation to the Equation 1.1 works really well in practice. Since the introduction of Adversarial examples in [Szegedy et al., 2014], a lot of defense methods have been proposed. However, these defenses have been shown to be at odds with

the natural accuracy [Zhang et al., 2019]. There have been a large array of research already done to explain why these phenomena happens. One such interesting work was presented in [Wang et al., 2020] which talks about the importance of lower and higher frequency components in generalization of CNN models.

Motivated by this we present an end-to-end pipeline to train a robust model by selecting only some frequencies for image classification. The frequency selection is done using the method described in [Xu et al., 2020] by creating attention map over the frequency spectrum. We combine that method with Adversarial Training described in [Zhang et al., 2019] to achieve a robust model. The goal of this work is to shed some light on the frequency aspect of the images to explain robust generalization.

CHAPTER 2: LITERATURE REVIEW

According to [Chakraborty et al., 2018] there can be different types of adversarial attacks depending on the threat surface. In this work we will only focus on the attacks that are applied on the inference after the model is trained. The very first works which kicked off the adversarial attacks field were presented in [Szegedy et al., 2014, Biggio et al., 2013]. Although these methods were computationally expensive to generate the adversarial attacks. Soon Goodfellow et al. [2015] presented a method called FGSM which generated the adversarial attacks quickly. They also presented Adversarial Training by using the adversarial examples to train the model more robustly. Since then a lot of attacks and defenses have been proposed by the community. To overcome FGSM's less effectiveness, iterative attacks like Projected Gradient Descent [Madry et al., 2018] and Basic Iterative Method [Kurakin et al., 2017] were presented. These attacks that require access to the model's parameters to perform the backward pass are called white box. In real life application an adversary might not have access to the weights. However, it was shown in [Szegedy et al., 2014, Goodfellow et al., 2015] that these samples can be generated on one model and can be transferred to other model. This transferable property of the attacks paves way to the black box methods.

There have been number of defenses approaches proposed by the community to defend the models against these attacks. One of the most effective attacks is the adversarial training introduced in [Goodfellow et al., 2015] uses the adversarial samples from attacks in training to train a robust model. Every white box attack can be used to train a more robust model and more sophisticated attacks like PGD can arguably generate more robust models. Another effective defense method called Randomized Smoothing was shown theoretically to guarantee robustness against adversarial attacks by [Cohen et al., 2019]. Randomized Smoothing based methods add random noise to input image and takes the maximum of all classifier decisions to give final decision. There have also

been some methods proposed in [Dhillon et al., 2018] which adds stochasticity to the model to defend against adversarial attacks. Although it was shown by [Athalye et al., 2018] that stochastic methods like this can be defeated by taking the expected gradient over random gradients sampled by adding noise. Recently, some work have been done to create robust models which are inspired from human perception like the ones presented in [Reddy et al., 2020, Dapello et al., 2020]. Both of these methods are very recent have shown promising results. Some researchers like [Xiao et al., 2019] have also proposed modification in the activation functions to increase robustness.

Some of these adversarial defenses are really effective against strong adversarial attacks. However, they have also shown to reduce the natural accuracy - accuracy on natural images. [Zhang et al., 2019] showed theoretically that robust accuracy is at odds with the natural accuracy. They also proposed a method to achieve the best trade-off. Historically speaking newer defenses are later broken by stronger attacks and stronger attacks can be defended against by even better methods. But to better defend it and end the cycle we also need to look at the underlying reason which causes this phenomena to occur.

There have been great deal of study to explain the existence of these adversarial examples. With the help of topology [Shafahi et al., 2019] recently proved a lower bound on probability of adversarial example's existence which shows that they can not be completely avoided. There's also an ongoing debate about the hypothesis that one explanation of these adversarial example is the fact that Neural Networks, as opposed to humans, focus more on the texture of the object rather than the shape. This hypothesis was presented by [Brendel and Bethge, 2019, Hermann and Lampinen, 2020] initially. Another explanation was presented by [Szegedy et al., 2014] who showed that Neural Networks have high Local Lipschitzness which can be one of the cause for the existence of the adversarial examples. Recently, [Wang et al., 2020] showed the effect high(HFC) and low frequency components(LFC) have on a Neural Network's generalization on image classification task. According to their work Naturally trained Neural Networks relies on HFCs to make decision.

If there's an alteration in the HFC it won't be visible to the human eye but the Neural Network will change its decision. Their work has been the primary motivation for our research presented here.

CHAPTER 3: METHOD

Our method utilizes DCTNet architecture [Xu et al., 2020] and TRADES algorithm [Zhang et al., 2019] to adversarially train a model in end-to-end fashion which selects frequencies based on input to maximize the robust accuracy. In this chapter we have described the crucial components of the method.

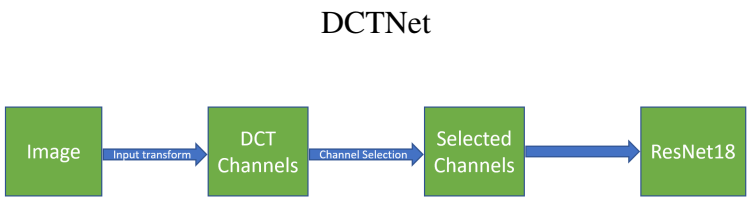


Figure 3.1: Overview of the DCTNet architecture

DCTNet as defined in Xu et al. [2020] can be applied to any existing CNN architecture that takes input as images. The overall architecture is shown in Figure 3.1. It takes input image in frequency domain instead of usual spatial domain. To do so, the original backbone architecture has to be modified slightly to accommodate the input. We will go over the most important changes of the DCTNet as opposed to a typical Convolutional Neural Network in this section.

Input Module

The overall flow of the input is shown in Figure 3.2. As described earlier, the model takes input in frequency domain. To convert from spatial domain of size HxWx3(H: height, W: width) to frequency domain it uses Discrete Cosine Transform. The reason to choose DCT instead of Fast Fourier Transform is that FFT introduces complex numbers which increases the computational

complexity of the model. The module first transforms the RGB input image into YCbCr space and then computes the DCT of YCbCr image by dividing the image in each Y, Cb and Cr channels into blocks of size 8x8 and computing 8x8 DCT coefficients for each block. Each of these 64 DCT coefficients represent a frequency in certain direction. If the YCbCr image is of $H \times W \times 3$ shape the DCT will have $\frac{H}{8} \times \frac{W}{8}$ blocks for each channels. Within one block the top left corner coefficient will represent the weight of the DC component.

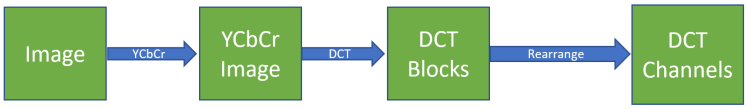


Figure 3.2: Overview of input pipeline

Once we have this DCT blocks we rearrange it to represent all same frequency components in the same channels. So for example, all DC component from each block of Y channel's DCT will be grouped together in channel 0, 2^{nd} coefficient from each block of Y channel in channel 1 and so on. This will be done for each channel from YCbCr so there will be $3 \times 64 = 192$ frequency channels where first 64 will be for DCT of Y channel, 64-127 for DCT of Cb and 128-191 for DCT of Cr. An example of this rearrangement module is shown in Figure 3.3. The example given in the figure is for an input with single channel of size $H \times W = 32 \times 32$. Notice as we rearrange the block the spatial size will change to $\frac{H}{8} \times \frac{W}{8}$.

To generate adversarial examples in end-to-end manner we have to generate them on the spatial domain and the whole module has to be differentiable. In our work we use the torchjpeg module[Ehrlich et al., 2020] which computes DCT as described above using PyTorch's[Paszke et al., 2019] differentiable operations.

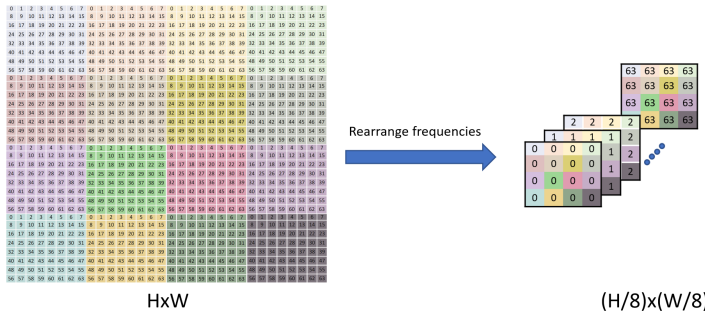


Figure 3.3: DCT blocks to DCT frequency channels rearrangements

Channel Selection

Once the image is transformed into DCT coefficient channels the DCTNet selects frequencies dynamically which maximizes the classification performance. The module creates attention maps using a squeeze-excitation block [Hu et al., 2018] like branch which then samples from Bernoulli distribution to create attention map using a Gumble Softmax module [Maddison et al., 2017, Jang et al., 2017, Tucker et al., 2017]. The Gumble Softmax module is used to make the sampling differentiable. The training also adds a regularization term λ which sums up the attention map to keep the attention map sparse. The attention map when multiplied with the channels will either turn a frequency on or off. The module is called the Gate Module in [Xu et al., 2020].

Backbone Modification

The output of the Gate Module is passed on to the ResNet-18 architecture. However, the original DCTNet architecture from [Xu et al., 2020] modifies the ResNet-18 architecture. The input to the CNN model after all processing described above becomes $\frac{H}{8} \times \frac{W}{8} \times 192$. So the DCTNet removes the first sequence of 'Conv-BatchNorm-ReLU-MaxPool' layers from backbone as the resolution is already downsampled. The method selects the input height and width of the RGB image such

that the spatial size of the input to the modified backbone is similar to spatial size of the output of the original backbone's Maxpool layer. In our work we have worked on CIFAR-10 dataset [Krizhevsky, 2009].

The original backbone architecture ResNet-18 for CIFAR-10 is taken from [Phan, 2021]. To match the spatial shape of the first maxpool layer we resize the original RGB images of CIFAR-10 from 32×32 to 128×128. This will keep the output shape after the Gate Module as $16 \times 16 \times 192$ which will go to the first residual block of the modified backbone.

TRADES Adversarial Training

To defend Convolutional Neural Networks against adversarial attacks many defenses have been proposed. The most effective among all these defenses is Adversarial Training. Adversarial Training uses adversarial images at training time to train the model. This effectively leads the model to learn weights that are robust against adversarial attacks. AT can be formalised as follows.

$$w^* = \underset{w}{\operatorname{argmin}} \underset{x'}{\operatorname{argmax}} \ \mathcal{L}(f(x'), y), \ ||x' - x||_p \leq \epsilon \tag{3.1}$$

The inner loop in Equation 3.1 generates the adversarial example by maximizing the loss and the outer loop trains on those adversarial examples to minimize the loss. This simple yet effective training method was first introduced in [Szegedy et al., 2014]. Since then there has been a lot of improvement made in the AT methods. Stronger attacks can be used to generate more effective adversarial examples which can be used to train robust models. However, [Zhang et al., 2019] showed with theory that there's a trade-off between robust accuracy and the natural accuracy of a classifier. They also derived a loss to tune the trade-off using a hyperparameter β. The loss function is shown in Equation 3.2. The KL in the equation represents the Kullback Leibler divergence. The

hyperparameter β will control the trade-off between the robust and the natural accuracy of the model. Higher the value of β higher the robustness and lesser the natural accuracy. We can use any adversarial attack to generate the x^{adv} which will also affect the robustness of the model.

$$\mathcal{L}(f(x), y) + \beta KL(f(x^{adv}), f(x)) \tag{3.2}$$

In this work we have used Projected Gradient Descent(PGD) [Madry et al., 2018] attack to generate adversarial example at each iteration of the training. The PGD method is an iterative way of generating the adversarial examples in a robust way to attack the model effectively. Each step in the iteration is defined as shown in Equation 3.3. The "clip" function in the equation is to clip the adversarial image's values in the $x + \epsilon$ where ϵ is the perturbation budget. The α represents the steps size in direction of maximum loss. PGD initialize the x_0^{adv} by randomly adding gaussian noise around the original sample.

$$x_{t+1}^{adv} = clip(x_t^{adv} + \alpha.sign(\nabla_x \mathcal{L}(f(x_t^{adv}), y))) \tag{3.3}$$

CHAPTER 4: EXPERIMENTS

Dataset

The entire model is trained on the CIFAR-10 dataset[Krizhevsky, 2009] which contains 50,000 RGB training images with 32x32 dimension. Each of these images belong to one of the 10 classes. We resize these images to 146×146 shape and randomly crop resized image to 128×128 dimension.

Adversarial Training

We set the hyperparameters β = 2.0, step size α = 0.007, number of steps as 10 and perturbation budget ϵ = 0.031 for TRADES adversarial training based on PGD. To train the model we set the initial learning rate as 0.01 with Warmup Cosine LR scheduler to update it at each epoch. We train the model for 200 epochs. The regularization term is λ = 0.1 for the attention module.

Robustness Test

We test our robust model against multiple adversarial attacks which includes FGSM [Goodfellow et al., 2015], BIM [Kurakin et al., 2017], PGD-20 [Madry et al., 2018], and AutoAttack [Croce and Hein, 2020b]. Each attacks is described in the following sub-sections. We also describe the test setup detail which uses these attacks to test the trained model rigorously. In our work we use the l_∞ metric to generate the adversarial attacks.

FGSM

The method was proposed in [Goodfellow et al., 2015] to generate adversarial samples in less time complexity. The method can be formalized as shown in Equation 4.1. It takes step of size ϵ in the direction of the derivative of the cross entropy loss to maximize the loss while keeping the perturbation size below the constraint. The only hyperparameter this method has is the perturbation budget ϵ.

$$x^{adv} = x + \epsilon.sign(\nabla_x \mathcal{L}(f(x), y)) \tag{4.1}$$

PGD and BIM

Linear methods like FGSM to generate adversarial samples are not really effective. The adversarial samples generated by them can be easily evaded. To overcome this iterative methods like PGD and Basic Iterative Method(BIM) [Kurakin et al., 2017] have been proposed. These method work as described earlier in the section "TRADES Adversarial Training". The only difference between PGD and BIM is that PGD initialize x_0^{adv} as adding gaussian noise around the original sample while BIM doesn't add any noise. These methods are also sometimes called FGSM-k as it applies the FGSM attacks iteratively k times. According to [Dong et al., 2020] both of these methods have negligible difference in terms of attack effectiveness.

AutoAttack

The biggest issue with most of the attacks used is that they have a lot of hyperparameters that need to be chosen carefully. Sometimes these choice can give a wrong idea of robustness of the

model. To deal with this issue authors in [Croce and Hein, 2020b] proposed a hyperparameter free version of the iterative attacks. It is one of the strongest attack in existence. The attack uses ensemble of 4 underlying attacks: APGD-CE, APGD-DLR, FAB [Croce and Hein, 2020a], and Square Attack [Andriushchenko et al., 2020]. The first 2 attacks are modified version of PGD to make it parameter free. APGD-DLR uses a novel Difference of Logits Ratio loss function inplace of a typical Cross Entropy loss.

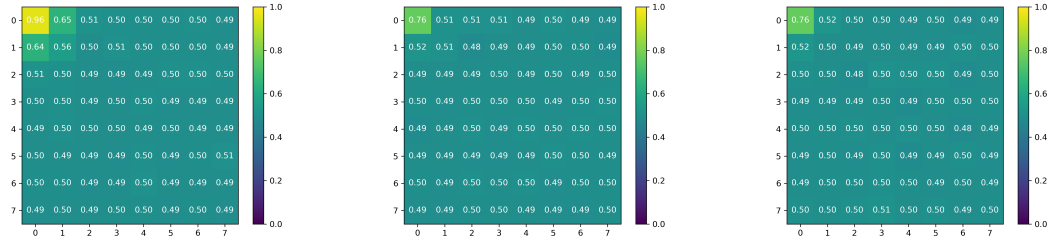
Evaluation and Results

To compare the trained model with TRADES we tested it against PGD and AutoAttack attacks. For PGD the iterations were set as 20, α = 0.003 and ϵ = 8/255. The results are shown in table Table 4.1. One thing to note here is that since our method used ResNet-18 as the backbone, we picked the results from Table 4 of the [Zhang et al., 2019] to make a fair comparison. However, we couldn't find the results for robust accuracy using ResNet-18 based TRADES against AutoAttack. The AutoAttack result in the table is from the [Croce et al., 2020]. The model used in that is the popular WideResNet-34 model [Zagoruyko and Komodakis, 2016]. As shown in the table the to get the same robust accuracy the natural accuracy of our method drops to 80%.

We also performed another experiment against FGSM and BIM attacks to see the robust accuracy of our method against different perturbation budgets. The budget was set to be (1-8, 16, 32)/255. We plot its robust accuracy against ϵ in Figure 4.1. As we can see the our DCT based method with Adversarial Training does better than same method without the TRADES adversarial training. These experiments shows the effectiveness of the model trained. However, the main goal of this work is not to get the state-of-the-art robustness but rather see what frequencies are used to achieve the robustness. The average attention maps of frequency selection module are shown in Figure 4.2 and Figure 4.3 for the naturally trained and adversarially trained models respectively. The maps

(a) (b)

Figure 4.1: Results for our DCT based model with and without TRADES training against FGSM attack in (a) and BIM attack in (b) for different perturbation budget.

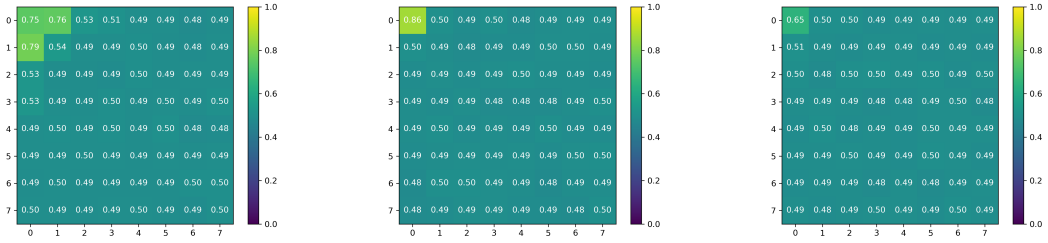


(a) (b) (c)

Figure 4.2: This figures show the frequency selection attention map averaged over test set for the naturally trained DCT based model. Figure (a), (b) and (c) are for Y, Cb and Cr components respectively.

show that for the Y map, which represents the image information in grey scale, the robust model lets the lowest frequency pass only 75% of the time while the naturally trained model lets it pass 96% time. The robust model gives a little higher weight to other lower frequencies too. Another surprising thing to note is that the robust model turns on the Cb channel's lowest frequnecy with 86% probability which just represent the color information.

We believe that the robustness and natural accuracy tradeoff can still be optimized with proper hyperparameter fine-tuning.



(a) (b) (c)

Figure 4.3: This figures show the frequency selection attention map averaged over test set for the DCT based model trained with TRADES AT. Figure (a), (b) and (c) are for Y, Cb and Cr components respectively.

Table 4.1: Robustness comparison with [Zhang et al., 2019]

	Models		
Threat Model	TRADES+DCT	DCT	TRADES
Clean	80	84	89
PGD-20	37	1	37
AutoAttack	41	0	53

CHAPTER 5: CONCLUSION

In this thesis, we demonstrated and end-to-end pipeline which trains a robust CNN model by creating a frequency filter. This model was rigorously evaluated against some adversarial strong attacks. The results presented here are competitive to other robust models. The pipeline also gave us insight into the importance of the frequency spectrum using Discrete Cosine Transform. From results we can see that the lowest 3-4 frequencies which contains the highest information should be almost equally important to train a robust model rather than the lowest only.

LIST OF REFERENCES

Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *ECCV*, 2020.

Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*, 2018.

B. Biggio, I. Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, P. Laskov, G. Giacinto, and F. Roli. Evasion attacks against machine learning at test time. *ArXiv*, abs/1708.06131, 2013.

Wieland Brendel and M. Bethge. Approximating cnns with bag-of-local-features models works surprisingly well on imagenet. *ArXiv*, abs/1904.00760, 2019.

Anirban Chakraborty, Manaar Alam, Vishal Dey, A. Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey. *ArXiv*, abs/1810.00069, 2018.

Jeremy M. Cohen, Elan Rosenfeld, and J. Z. Kolter. Certified adversarial robustness via randomized smoothing. In *ICML*, 2019.

Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *ICML*, 2020a.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. *ArXiv*, abs/2003.01690, 2020b.

Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020.

Joel Dapello, Tiago Marques, Martin Schrimpf, Franziska Geiger, D. Cox, and J. DiCarlo. Simulating a primary visual cortex at the front of cnns improves robustness to image perturbations. *bioRxiv*, 2020.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Guneet S. Dhillon, K. Azizzadenesheli, Zachary Chase Lipton, Jeremy Bernstein, Jean Kossaifi, A. Khanna, and Anima Anandkumar. Stochastic activation pruning for robust adversarial defense. *ArXiv*, abs/1803.01442, 2018.

Yinpeng Dong, Qi-An Fu, Xiao Yang, Tianyu Pang, Hang Su, Zihao Xiao, and Jun Zhu. Benchmarking adversarial robustness on image classification. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 318–328, 2020.

Max Ehrlich, Larry Davis, Ser-Nam Lim, and Abhinav Shrivastava. Quantization guided jpeg artifact correction. *Proceedings of the European Conference on Computer Vision*, 2020.

I. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2015.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

Katherine L. Hermann and A. Lampinen. What shapes feature representations? exploring datasets, architectures, and training. *ArXiv*, abs/2006.12433, 2020.

Jie Hu, L. Shen, and Gang Sun. Squeeze-and-excitation networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7132–7141, 2018.

Eric Jang, S. Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *ArXiv*, abs/1611.01144, 2017.

A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.

A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *ArXiv*, abs/1607.02533, 2017.

Chris J. Maddison, A. Mnih, and Y. Teh. The concrete distribution: A continuous relaxation of discrete random variables. *ArXiv*, abs/1611.00712, 2017.

A. Madry, Aleksandar Makelov, Ludwig Schmidt, D. Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *ArXiv*, abs/1706.06083, 2018.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

Huy Phan. huyvnphan/pytorch_cifar10, 2021. URL `https://zenodo.org/record/4431043`.

M. V. Reddy, Andrzej Banburski, Nishka Pant, and T. Poggio. Biologically inspired mechanisms for adversarial robustness. *ArXiv*, abs/2006.16427, 2020.

A. Shafahi, W. R. Huang, Christoph Studer, S. Feizi, and T. Goldstein. Are adversarial examples inevitable? *ArXiv*, abs/1809.02104, 2019.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2014.

G. Tucker, A. Mnih, Chris J. Maddison, John Lawson, and J. Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. In *NIPS*, 2017.

Haohan Wang, Xindi Wu, Pengcheng Yin, and E. Xing. High-frequency component helps explain the generalization of convolutional neural networks. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8681–8691, 2020.

Chang Xiao, Peilin Zhong, and Changxi Zheng. Resisting adversarial attacks by k-winners-take-all. *ArXiv*, abs/1905.10510, 2019.

Kai Xu, Minghai Qin, Fei Sun, Yuhao Wang, Yen kuang Chen, and Fengbo Ren. Learning in the frequency domain. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1737–1746, 2020.

Sergey Zagoruyko and N. Komodakis. Wide residual networks. *ArXiv*, abs/1605.07146, 2016.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, E. Xing, L. Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. In *ICML*, 2019.