
MSc Data Analytics
Stiftung Universität Hildesheim
Marienburger Platz 22
31141 Hildesheim
Prof. Dr. Dr. Lars Schmidt-Thieme
Supervisor: Mofassir ul Islam Arif, mofassir@ismll.de

# Student Research Project Final Report

## "Adversarial Robustness across Representation Spaces"

**Sruthy Annie Santhosh***, **Diego Coello de Portugal** *,
**Heliya Hasani***, **Aditya Nair** *

`* equal contribution`

(santhosh/coello/hasani/nair @uni-hildesheim.de) 17th March 2023

17th March 2023

**Abstract**

The main objective of our student research project is to improve Adversarial Robustness across different Representation Spaces in the domain of image classification. The ability of a neural network to resist adversarial attacks on the input data in various representation spaces is termed as adversarial robustness across representation spaces. We test the model's resistance to input disturbances or perturbations in pixel-level feature spaces like grayscale and color and in other feature spaces like wavelet and Fourier transformations. This project aims to understand the trade-off between robustness and accuracy in neural networks by examining the generalization properties of neural networks, the impact of representation spaces on the model's robustness to several different adversarial attacks, and how different representations affect the effectiveness of adversarial examples. We implement different training strategies and extend the field of adversarial robustness through various experiments. Each training method is tested intricately to discover some key information in this domain.

**Keywords**: Adversarial Robustness Across Representation Spaces, Image Classification, Adversarial Robustness, DCT, FFT, Pixel Space, Neural Networks

# Contents

# 1 Introduction

When training a deep neural network model to understand an image dataset, it occurs that the output of the trained model changes significantly between the original image and that same image with imperceivable perturbations. These perturbations in the original image that interfere with the model performance are called *Adversarial Attacks*. One of the fundamental implications of adversarial attacks is that in some instances where the task corresponds to classification, the model evaluates the image from the adversarial attack as belonging to a class completely different from the correct class even though the original image was classified correctly. It is worth mentioning, that the changes made to the original image are so small that the perturbed image should also belong to the same class, being in most cases imperceivable to the human eye. Thus adversarial attacks prove that these types of models are not foolproof and can be exposed to be inconsistent. The capability of a model to resist being fooled is called Adversarial Robustness.

This research project uses adversarial training in neural networks across various representation spaces to increase adversarial robustness in various data sets. Images cannot be precisely described by pixels. A lot of information is lost in this representation. Hence, the consideration of wave signals is necessary for more knowledge. Since it's crucial that the images don't take up too much room on the computer, DCT is one type of signal that is used in image compression. The major benefit of using Fourier series is that we can analyze a signal more effectively outside of its original domain. Although adversarial robustness does not generalize well across attacks or representation spaces, it is essential in real-world situations to achieve some generalization because numerous parameters cannot be taught and trained before but should be detected by the machine learning algorithms when deployed.

# 2 Related Work

This section describes several methods/components that relate closely to our approach to tackle the Adversarial Robustness problem.

## 2.1 Adversarial Attacks

In the baseline, [1] the authors aimed to achieve a deep learning model robust against adversarial attacks [2]. These attacks populate pain points due to the fact that machine learning models are used in sensitive applications such as speech and image recognition, spam filtering, and virus detection where bypassing the model is a security concern. These attacks have the potential to modify a model's input and could possibly mislead it, generating destructive outputs, which might have far-reaching effects.

In an adversarial assault, skillfully designed perturbations to an input result in a targeted misclassification by the model. Image-based inputs are not the only concern here; text-based models, like spam filters and virus detectors, are also susceptible to these assaults. This is especially troubling because an adversary can create nefarious inputs without having access to the model directly.

Recently, new techniques have been put forth by academics to strengthen machine learning models' defense against hostile attacks. To train a model and make it resilient to the worst-case change of the inputs, [2] suggested the usage of robust optimization. Similar to this, [18] suggested using feature squeezing, which reduces the inputs' dimension and makes it more resistant to hostile assaults. Although adversarial training and other defense strategies are available, the field is still developing, and more investigation is required to comprehend and neutralize these assaults. Adversarial attacks can be classified into *white box* attacks where model structure is known and *black box* attacks where model structures are not known. A brief insight into these two attacks is mentioned in Table 1

- **FGSM**: It is a single-step attack where the gradient of the loss w.r.t the input data, then adjusts the input data to maximize the loss using the sign of the gradient. It determines the gradient with regard to the image's pixels. By adding a slight disturbance to an input leads a machine-learning model to predict something incorrectly, creating adversarial examples.

| Norm | $L_0$ | $L_1$ | $L_2$ | $L_\infty$ |
|---|---|---|---|---|
| White box | SparseFool [3], JSMA [4] | Elastic-net attacks [5] | Carlini-Wagner [8] | PGD [16], i_FGSM [17], Carlini-Wagner [8] |
| Black-box | Adversarial Scratches [6], Sparse-RS [7] | - | GenAttack [9], sim [13] | GenAttack [9], SIMBA [13] |

**Table 1:** Attacks classification table

$$x_{adv} = x + \epsilon \text{sign}(\nabla_x L(\theta, x, y)) \tag{1}$$

Here, x is the input to the model, L is the cost function, $\theta$ denotes the model parameters, y is the target value, i.e., the class the model is aiming to misdirect, $\epsilon$ determines the perturbation's size and the sign operation returns 1 or -1 depending on the input value. The motivation for this method comes from linearizing the cost function and finding the perturbation that optimizes the cost while satisfying a $L_\infty$ constraint.

- **FFGSM**: The FGSM method can be expanded to generate the FFGSM attack [14]. Instead of applying the perturbation directly, the FFGSM algorithm applies some random noise to the input before performing the FGSM attack. In certain circumstances, this strategy is quicker and more efficient than FGSM allowing for complex adversarial images.

$$\begin{aligned} x_0 &= x + \mathcal{U}(-\epsilon, \epsilon) \\ x_{adv} &= \text{clip}\Big(x_0 + \epsilon \cdot \text{sign}\Big(\nabla_{x_0} L(\theta, x_0, y)\Big)\Big) \end{aligned} \tag{2}$$

- **PGD**: The PGD [15] attack seeks to locate a spot in the locality of the original image that maximizes the loss. It is a more effective version of the FGSM attack, that performs the FGSM algorithm repeatedly with s smaller step size $\alpha$ to produce more powerful adversarial samples.

$$\begin{aligned} x_0 &= x + \mathcal{U}(-\epsilon, \epsilon) \\ x^{(t+1)} &= \text{clip}\Big(x^{(t)} + \alpha \cdot \text{sign}\Big(\nabla_{x^{(t)}} L(\theta, x^{(t)}, y)\Big)\Big) \end{aligned} \tag{3}$$

## 2.2 Adversarial Training

Deep learning models' resistance to adversarial attacks can be strengthened through adversarial training. In order to force the model to learn how to correctly categorize these examples, the strategy entails creating hostile examples during the training process and including them in the training set. This strategy can strengthen the model's defense against hostile assaults in a practical setting. In their publication "Towards Deep Learning Models Resistant to Adversarial Attacks" [2] the authors present one of the most pioneering research work on adversarial training. For using adversarial samples for training deep neural networks, the authors suggested a solid optimization methodology. To increase the resilience of the model, they generated adversarial samples using projected gradient descent (PGD) and added them to the training set. Contrary to conventional training methods, the authors demonstrated that adversarial training can greatly increase the model's accuracy against adversarial attacks.

## 2.3 Representation Spaces

The model's robustness and susceptibility to adversarial attacks can be significantly impacted by the choice of representation space in the context of adversarial robustness. Therefore, we will take a look at the main representation spaces for images.

### 2.3.1 Pixel Representation Space

The "pixel representation space" describes an image as collection of pixels, i.e., as a combination of red, green and blue values for each position in the image.

To increase the resiliency of the models, researchers are constantly investigating various representation spaces and quantization techniques.

### 2.3.2 Discrete Cosine Transform (DCT)

Discrete Cosine Transform (DCT) is a popular transformation for images and signal processing. A machine learning model's input space is the focus of adversarial attacks, which aim to change the image so that the model incorrectly classifies it. It is more challenging for the attacker to fool the model with perturbations in the initial input space, compared to when the image perturbations are generated in a different representation space For example, when converting RGB images (pixel space) to the DCT (frequency

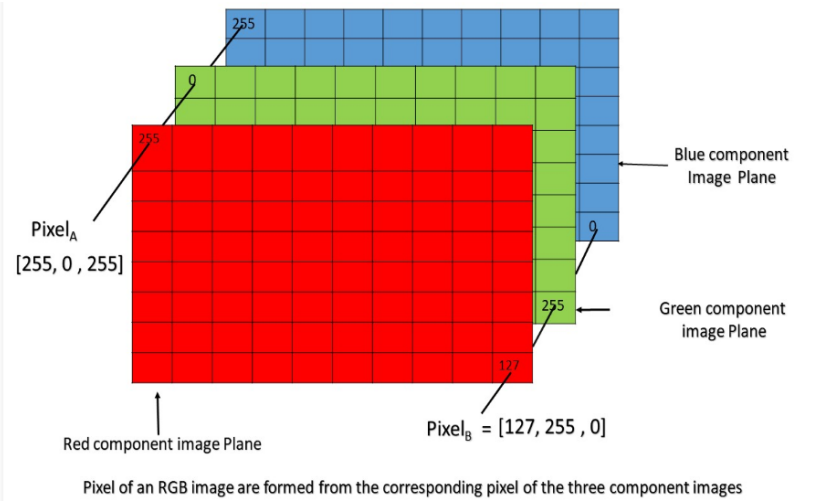



**Figure 1:** Pixel representation space. [Source: [11]]

space), it becomes easier for an attacker, to find an adversarial attack that is effective against the model.

DCT transformation:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot \cos\left(\frac{\pi}{N}(n+\frac{1}{2})k\right), \quad k = 0, 1, \ldots, N-1 \tag{4}$$

IDCT or inverse DCT transformation:

$$x_n = \frac{2}{N} \sum_{k=0}^{N-1} X_k \cdot \cos\left(\frac{\pi}{N}(n+\frac{1}{2})k\right), \quad n = 0, 1, \ldots, N-1 \tag{5}$$

### 2.3.3 Discrete Fourier Transform (DFT)

The Fourier transform is analogous to decomposing the sound of a musical chord in terms of the intensity of its constituent pitches. The output of the transform is a complex-valued function of frequency, which in theory should allow for more freedom in the perturbations generated. Since images are discrete representation objects, we use the discrete version of the Fourier Transform. An effective method for calculating a sequence's discrete Fourier transform (DFT) is the Fast Fourier Transform (FFT). In order to conduct adversarial training on this transformed space, we use the FFT to transform images into the frequency space [22].

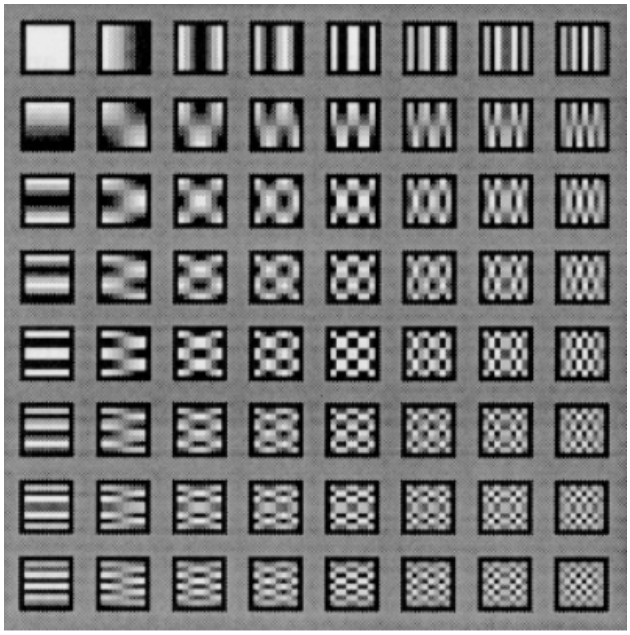


**Figure 2:** Discrete cosine transform representation space. [Source: [? ]]

FFT equation:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi \frac{kn}{N}}, \quad k = 0, 1, \ldots, N-1 \tag{6}$$

IFFT equation:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cdot e^{i2\pi \frac{kn}{N}}, \quad n = 0, 1, \ldots, N-1 \tag{7}$$

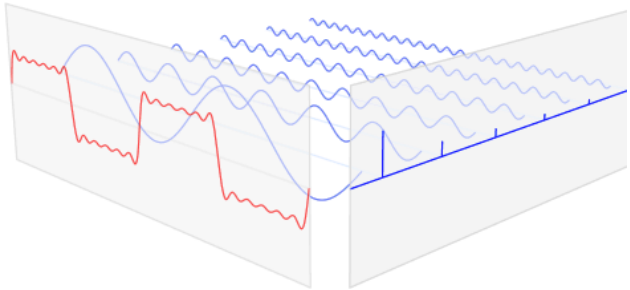


**Figure 3:** Fourier transform representation space. [Source: [12]]

### 2.3.4 JPEG

JPEG transformation is a variation of DCT transformation. The idea is to split the images in small patches of 8x8 pixels and perform DCT transformation independently in each image patch. It is frequently used for image compression after some additional post-processing.

### 2.3.5 Log Transform

The log transform is a mathematical procedure that is frequently used to transform image intensities. The log transform is a non-linear transformation unlike DCT or DFT, that still works in the pixel domain. Nevertheless, the non-linear scaling of the values allows the attacker to have another representation space in order to try to perturb the image and fool the model [23]. When combined with other defense methods, this strategy has been shown to increase the robustness of machine learning models against adversarial attacks.

## 2.4 Models Used

### 2.4.1 ResNet -50

ResNet-50 is a deep convolutional neural network (CNN) architecture commonly used for image classification tasks. "Deep Residual Learning for Image Recognition" [27] presented a 50-layer residual network. ResNet-50 has been utilized as a benchmark model for assessing the efficacy of different defense strategies against adversarial attacks in the setting of adversarial robustness [1]. In order to cause a machine learning model to incorrectly classify an image, adversarial attacks target the input area of the model. Numerous studies have looked into how resilient ResNet-50 is to adversarial assaults and have suggested various defense strategies to increase its resilience. For instance, [28] suggested using random resizing and padding as a protection method to the ResNet-50 architecture.

This model consists of several residual blocks, each of which contains several convolutional and activation layers [27]. Convolutional processing is used in ResNet-50's first layer to extract basic features from the input picture. There are then several chunks of residual data, each of which has a number of convolutional and activation layers. ResNet-50's final layer is a fully connected layer that generates the final classification results. The residual blocks are connected to each other via skip connections, which allow the network to have a better gradient flow [27]. This allows ResNet-50 to use a lot of layers and learn more complex features than traditional CNNs, which are prone to degradation problems when increasing the depth of the network.

### 2.4.2 Auto-Encoders and Decoders

An application of the neural network design known as auto-encoders and decoders is used to reconstruct the input data. The fundamental principle of auto-encoders is to first compress the incoming data into a lower-dimensional representation, which is referred to as the encoding, and then to use a decoder to extract the original data from the encoding. Typically, neural networks like feedforward or convolutional neural networks are used for the encoding and processing. Image denoising and anomaly identification are two issues to which auto-encoders and decoders have been applied.

In the experiments section, we train the autoencoder on clean images and reconstruct those images for training. Then, we reconstruct the attacked/adversarial images using the autoencoder to find their losses. This

way, we can understand how the attack perturbs a clean image and how it deviates from the original manifold to which the images belong.

## 2.5 Adversarial robustness

In this section, we briefly give an overview on Adversarial Robustness and its implementation in the baseline. Most existing approaches to adversarial robustness include training against a particular attack and representation basis, usually pixel (Adversarial training). The FGSM and PGD methods are some of the most common methods when performing adversarial training. There is little to no research on improving robustness across different datasets and attacks simultaneously. Some recent studies [34] trained classifiers that are simultaneously robust to perturbations to different $L_p$ norms. However, they do not consider multiple representation spaces.

The project's baseline paper is *Adversarial Robustness across Representation Spaces* [1]. The authors of this paper emphasized the importance of training against different adversarial attacks on different representation spaces. This is because training against a specific representation space only improves the model robustness against that same representation space. They use Project Gradient Descent Method (PGD) to produce adversarial examples [26]. To acquire an opposing example, they try to make small changes to the image which are restricted by the selected representation space: pixel-based, discrete cosine transform (DCT), etc. They propose the Multiplicative Weights algorithm to make models robust against several representation spaces. We will discuss further about the baseline in the Experiments section.

# 3 Data Foundation

This section briefly describes the datasets we are working with in our research.

### 3.0.1 MNIST

In the Computer Vision and machine learning fields, the MNIST (Modified National Institute of Standards and Technology) dataset is a frequently used dataset. It is a total of 70,000 grayscale photographs of handwri,tten numbers (0–9), each of tehm measuring 28x28 pixels. It is splitted in 10,000 test pictures and 60,000 training images. Particularly in the area of image classification, the MNIST dataset is frequently used as a beginning point for the development and evaluation of machine learning algorithms. It is thought to be a reasonably straightforward and well-understood dataset, making imodel'sexcellent option for testing new methods and algorithms.

The article "Gradient-Based Learning Applied to Document Recognition" [30] is a source in this field. The authors of this paper show a Convolutional Neural Network (CNN) that performed at the cutting edge on the MNIST dataset. This work cleared the way for the widespread use of CNNs for image classification and is widely regarded as a seminal article in the field of deep learning.

### 3.0.2 Fashion-MNIST

The Fashion-MNIST dataset, which includes a training set of 60,000 instances and a test set of 10,000 examples, is a collection of Zalando article images. Each illustration is a 28x28 grayscale image paired with a label drawn from one of ten groups. The ten categories correspond to various categories of apparel, including T-shirts and tops, pants, pullovers, dresses, coats, sandals, shirts, sneakers, bags, and ankle boots. The Fashion-MNIST dataset was created as a more difficult option to the MNIST dataset, which is frequently used for image classification. The Fashion-MNIST images are more intricate and varied than the handwritten digit pictures in the MNIST dataset, despite the fact that both datasets have the same image size and number of classes.

For image classification, Fashion-MNIST has grown to be a well-liked benchmark for machine learning and deep learning models. It is regarded as a helpful testing ground for comparing how well various algorithms work when applied to a challenging image classification problem. The article "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning

Algorithms" [31] is a source in this field. The authors of this paper present the Fashion-MNIST dataset and contrast how different machine learning algorithms perform on it.

### 3.0.3 CIFAR-10

A common image classification dataset called CIFAR-10 contains 10,000 test images and 60,000 training images. Each of these images being 32x32 RGB images that have been divided into 10 groups: aircraft, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and vehicles are among the ten categories in CIFAR-10. The Canadian Institute for Advanced Research (CIFAR) developed the CIFAR-10 dataset, which was intended to be a more difficult image classification issue than the frequently used MNIST dataset. The handwritten digit pictures in the MNIST dataset are much simpler than the images in the CIFAR-10 dataset, which also has much larger images with a 32x32 resolution as opposed to 28x28.

For machine learning and deep learning models, the CIFAR-10 dataset is frequently used as a benchmark, particularly for picture classification and is more challenging than FashionMNIST dataset. The article "Learning Multiple Layers of Features from Tiny Images" [33]is a source in this field. In this study, the author tests the effectiveness of deep convolutional neural networks for picture classification using the CIFAR-10 dataset. The article has received numerous citations and is regarded as a ground-breaking contribution to deep learning.

# 4 Baseline Assumptions

The paper "Adversarial Robustness across Representation Spaces" by George Yu et al.[1] presents a study of the robustness of deep neural networks (DNNs) to adversarial examples across different representation spaces. The authors make several assumptions in their study:

- The first assumption is that DNNs are vulnerable to adversarial examples, which are inputs that have been slightly modified to fool the network into making a wrong prediction. The authors mentioned that adversarial examples can be crafted in a variety of ways, with different methods of adding small perturbations to the input and optimizing different losses.

- The second assumption is that a model can have different robustness to different representation spaces, such as the pixel space or the DCT space. The authors assume that the robustness of a DNN to adversarial examples can be measured by the size of the adversarial perturbation required to fool the network.

- The third assumption is that the robustness of a DNN can be improved by training the network against adversarial attacks. Nevertheless, training on one space does not make the model robust to the same attack on a different representation. The authors assume that training a DNN in different representation spaces to adversarial examples can lead to a more robust network overall.

- The fourth assumption is that the robustness of a DNN can be trained in different representation spaces by using a training strategy called Multiplicative weights method. This training strategy was proposed by the authors and they claimed to have found a better training method than round robin or greedy method.

The authors of the paper test these assumptions by experimenting with various DNN architectures and adversarial attacks on the CIFAR-10 and MNIST datasets. They show that DNNs can indeed be made more robust to adversarial examples by training them in different representation spaces, using Multiplicative Weights update method.

Overall, the paper provides a thorough analysis of the robustness of DNNs to adversarial examples across different representation spaces and proposed a new training method. The assumptions made by the authors are supported by their experimental results, which suggests that their findings are likely to be valid. However, it is important to note that the results are limited to the specific datasets and architectures used in the study, and further research is needed to investigate the generalisation of their findings. Hence our idea was to extend this approach to more datasets, attacks and representation spaces.

First, we started by implementing the methods in the given paper and verifying the assumptions and results. Then we extended the approach.

# 5 Research Idea

In this research, we extend on the approach proposed in the paper "Adversarial Robustness Across Representation spaces" [1] that is based on training models across different representation spaces. The methodology can be broken down into several key steps:

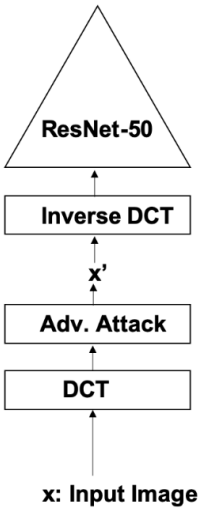



**Figure 4:** The modified network architecture for computing an adversarial perturbation in the DCT basis. The same can be applied to all the representation spaces [1]

- Data Pre-processing: We first pre-process the input data by transforming it into multiple representations. For example, in the case of image classification, we can use different image scales, or different feature spaces such as the Fourier or wavelet transform. Here we have used Pixel, DCT (JPEG), DFT and LogSpace image representations
- Model Training: Next, we train a machine learning model on each representation of the data. For example, we can train a convolutional neural network (CNN) on the original images, as well as on the images in the different scales or feature spaces. We initially experimented with CNN and then moved on to the Resnet50 model. The model is trained using different training strategies like greedy, round robin and multiplicative weights method. Resnet-50 only accepts images in pixel base, hence we convert images to different spaces, apply perturbations according to the attacks and then convert it back before sending to model.

- Adversarial Example Generation: To evaluate the robustness of the models, we generate adversarial examples using different methods. For example, we can use the fast gradient sign method (FGSM) or the projected gradient descent (PGD) method to generate adversarial examples for each representation of the data. We have used the following attacks: PGD, FGSM, FFGSM, PGD_L2.

- Model Evaluation: We evaluate the robustness of the models by measuring the classification accuracy on the adversarial examples. We also compare the results with the results obtained by the paper [1]. The model evaluation is also done using different approaches. We replicated the approach used in the paper, then tried a static and a dynamic approach also.

In our methodology, we use different representations of the data to train models. By training models on multiple representations of the data, we aim to make them more robust to adversarial examples that are crafted for one specific representation. Additionally, by generating adversarial examples tailored for each representation space, we aim to better evaluate the robustness of models.

The paper [1] had proposed that we can improve the robustness of models to adversarial examples by adding more representation spaces, and that the proposed techniques for generating adversarial examples can be used to more effectively evaluate the robustness of models. But while some of our experiments were supporting this arguments we found some irregularities also.

# 6 Training Strategies

The main training strategies for doing adversarial training with different attacks and representation spaces are:

- **Round Robin**: It is a training method that trains a DNN in different representation spaces in a round-robin fashion. This method trains the DNN in one representation space, then switches to another representation space, and so on. We trained the model on each of the different attacks in all spaces. Here all the attacks and spaces are given equal weights. The algorithm can be seen in Figure 5a.

- **Greedy**: It is another training method that trains a DNN in different representation spaces in a greedy fashion. This method starts by training the DNN in one representation space, then selects the next representation space that leads to the largest improvement in robustness. Here also we considered all attacks and spaces, but the attack or space which leads to largest loss is given more weight and is selected for training in the next iteration. The algorithm can be seen in Figure 5b.

**Input:** Training data $\{(x_1, y_1), \ldots, (x_m, y_m)\}$.
**Input:** Mini batch size $B$, time steps $T$.

1. Initialize $index = 1$.
2. For $t = 1 \ldots, T$ do:
   - Get the next mini batch of size $B$.
   - Set $i = index$.
   - Use PGD to optimize $L_i$ on the mini batch to get $\theta_t$.
   - $index = (index + 1) \bmod k + 1$.
3. Output $\hat{\theta} = \theta_T$.

**(a)** The round robin heuristic.[1]

**Input:** Training data $\{(x_1, y_1), \ldots, (x_m, y_m)\}$.
**Input:** Validation data: $\{(x_{m+1}, y_{m+1}), \ldots, (x_{m+s}, y_{m+s})\}$.
**Input:** Mini batch size $B$, time steps $T$, update frequency $r$.

1. For $t = 1 \ldots, T$ do:
   - Set $i = \arg\max_j L_j^{\text{val}}$.
   - Repeat for $r$ epochs:
     - Get the next mini batch of size $B$.
     - Use PGD to optimize $L_i$ on the mini batch to get $\theta_t$.
2. Output $\hat{\theta} = \theta_T$.

**(b)** The greedy heuristic.[1]

- **Multiplicative Weights Update Method (MWUM)**: The paper "Adversarial Robustness across Representation Spaces" by George Yu et al.[1] proposed this optimization method to improve the robustness of deep neural networks (DNNs) to adversarial examples. The MWUM is a method of updating the weights of the DNN that is based on the

multiplicative weights update method in online learning.

In the MWUM, we first define a set of weights for each representation space, which are used to adjust the importance of each representation space during the training process. Then, at each iteration of training, the weights are updated based on the performance of the DNN in each representation space. The weights are increased for representation spaces in which the DNN performs well, and decreased for representation spaces in which the DNN performs poorly. This method lays in between the round robin and greedy methods. During the training epoch, attacks are selected in a probabilistic way depending on the weights. The algorithm can be seen in Figure 6.

**Input:** Training data $\{(x_1, y_1), \ldots, (x_m, y_m)\}$, Validation data $\{(x_{m+1}, y_{m+1}), \ldots, (x_{m+s}, y_{m+s})\}$, mini batch size $B$, time steps $T$, update frequency $r$, window size $h$, Scaling factor $\eta$.

1. Initialize $w_i = 1$ for all $i \in [k]$.
2. For $t = 1 \ldots, T$ do:
   - Repeat for $r$ epochs:
     - Get the next mini batch of size $B$. Sample loss $L_i$ with probability $p_i = \frac{w_i}{\sum_{j=1}^{k} w_j}$.
     - Run the PGD based algorithm to optimize $L_i$ on the mini batch.
   - For all $i$ set $w_i = w_i \cdot e^{\eta L_i^{\text{val}}(\theta_t)}$. Here $L^{\text{val}}$ is the loss evaluated on the validation set.
3. Output $\hat{\theta} = \frac{1}{h} \sum_{t=T-h+1}^{T} \theta_t$.

**Figure 6:** The multiplicative weights update method proposed by the paper[1].

# 7 Resources

This project was done in Python language (version 3.7), using jupyter notebooks IDE. We have trained our models on the University of Hildesheim cluster. The datasets were pre-downloaded along with pretrained weights for our experiments. Training was done with and without pretrained weights. Some of the major libraries used are numpy (version 1.21.6), matplotlib (version 3.2.2), torch (Version = 1.7.1), torchvision (version 0.8.2), torchattacks (version 3.2.6) and torchjpeg. The torchattacks library was used to simulate the adversarial attacks and the dct and fft functions were also taken from torch.

# 8 Our approach

We started our research project by defining a CNN model and training it on MNIST dataset. We mainly focussed on PGD and FGSM attacks on pixel and DCT spaces. Initially we performed adversarial training, i.e, trained only in the given attack and it was found to have bad performance against other attacks. Thus we verified the baseline assumptions on MNIST dataset also. Evaluation was also done on the MNIST dataset to make sure that reconstruction error is less while converting from idct images to pixel images.

Then we changed the model to Resnet-50 and introduced more attacks and spaces like FFGSM, SLIDE, DCT(+JPEG) and FFT. The dataset was changed to FashionMNIST and training was done using round robin, greedy and multiplicative weights update methods. Here also we verified the baseline assumptions before proceeding with the training. AutoEncoder was also used to verify the reconstruction errors and understand the importance of high frequency domains in misclassification.

In the next stage of our research we included CIFAR10 dataset and verified the baseline assumptions for them also. We also included PGD_L2 attack and Log space also in our experiments. Initially we trained the model and conducted experiments on all 3 training strategies. Then we performed experiments on static and dynamic variations of the algorithms. Instead of choosing all the attacks and spaces, we trained only on specific sets which we named as the static variations. This is similar to holdout testing, since we train only on some sets and test on all sets. In the dynamic variations, different attacks and spaces were chosen for each epoch and training was done. Testing was done on all the attacks and spaces. We also calculated the average accuracy across all combinations and standard accuracy on clean images to get a base idea.

In the following sections, the experiments conducted and results obtained are discussed in detail.

# 9 Experiments

For the experiments we device 2 principal settings. The first one on FashionMNIST [24], where we try to grasp a better understanding of the problem with visualizations on the transformations and the attacks, and tests on the different ideas before experimenting on a more complex setting. The second part, is done on CIFAR10 [25] were we conduct the main experiment to check the performance of the baseline on a more realistic setting.

## 9.1 FashionMNIST

FashionMNIST [24] is a benchmark dataset with grayscale images with 28x28 pixels and 10 different cloth classes (shirt, trouser, bag, sneaker...). It has 70000 examples (60000 for training and 10000 for testing) with 10% (7000 images) of it corresponding to each of the 10 classes.

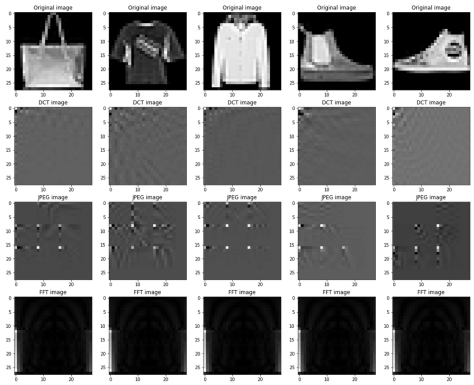
The simplicity of the dataset and the small image input size, allows us to run several experiments on a computationally cheap setting to prove properties of the model, data behaviour or problem setting that we could later generalize to a more complex scenario. In this regard, we will start by studying the impact of the Representation Space and its transformations.

### 9.1.1 Representation Spaces and transformations impact

The first experiment we did was to prove that the Representation Space transformations are indeed invertible, i.e., the difference between the original input and the output obtained after doing a transformation and its inverse is negligible. In order to prove this, we do the transformations and inverse transformations from pixel representation to any other representation space and average the MSE loss over all the training examples (Fig. 7).

From this experiment we notice that the error of the transformation is smaller than $10^{-12}$ for DCT, JPEG and FFT transformation in average. With this we conclude that the transformation functions should not affect the images enough to produce a different prediction for any given model. Notice that a change in the pixel values is $1/255 \approx 4 * 10^{-3}$ which is 9 orders of magnitude higher than the perturbation achieved with the transformations.

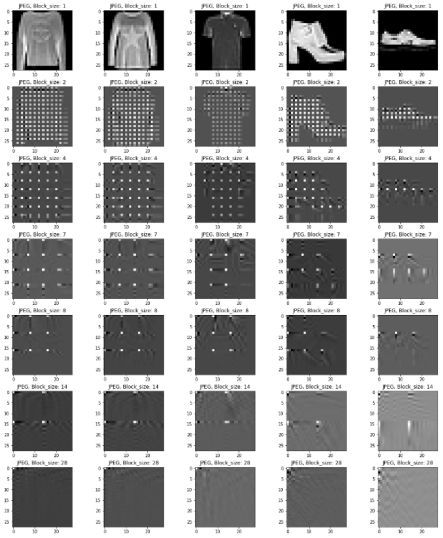
We also did a small study on the impact of changing the size of the JPEG transformation from 7 to other different values. The reason for this is because standard JPEG transformation uses DCT transformation in patches of 8x8 pixels in the image. However, since the original image is 28x28 which is not

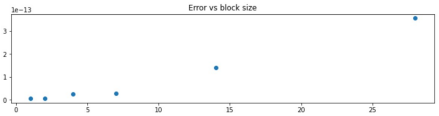


| Transformation | MSE Loss |
|---|---|
| DCT | 3.5082800922160706e-13 |
| JPEG | 2.7177738209341502e-14 |
| FFT | 2.011194606444137e-15 |

**Figure 7:** Representation space transformation visualizations and average MSE for the transformation.

divisible by 8, we decide to change it to 7 in order to avoid a black stripe on the right and bottom side of the image after the transformations, since it won't be considered due to the fact that a 4 pixel band is left and not divisible in patches of size 8.






| JPEG block size | MSE Loss |
|---|---|
| 1 | 4.8358885492772504e-15 |
| 2 | 5.475080363582553e-15 |
| 4 | 2.4232663944044608e-14 |
| 7 | 2.7397946446836886e-14 |
| 14 | 1.4177050646716621e-13 |
| 28 | 3.573205099860477e-13 |

**Figure 8:** JPEG space transformations visualization and average MSE for the transformation.

In Fig. 8, we show the impact of changing the patch size to different values. Notice that the MSE is still below $10^{-12}$ and that the case with patch size 28x28 corresponds to DCT and the patch size 1x1 corresponds to multiplying by an exponential factor and then dividing by the same value. This trend

of doing less multiplication and additions when the patch size is smaller and more when the patch size is bigger, can explain the trend of having higher MSE Loss for bigger block size when compared to smaller ones.

### 9.1.2 Adversarial robustness between representation spaces

In order to improve the robustness of the model against adversarial attacks, one of the main procedures is to do adversarial training, i.e., generate adversarial examples in which the models are trained.

| | Test STD | Test w/PGD_Pixel | Test w/PGD_DCT | Test w/PGD_JPEG | Test w/PGD_FFT |
|---|---|---|---|---|---|
| STD | **88.37+0.29** | 0.0±0.0 | 0.0±0.0 | 0.0±0.0 | 23.9±2.26 |
| Test w/PGD_Pixel | 71.22±1.97 | **52.79±0.45** | 38.64±1.0 | 46.73±1.2 | 68.83±0.3 |
| Test w/PGD_DCT | 71.27±3.1 | 50.53±1.02 | **49.86±0.86** | 50.22±0.9 | 65.92±1.64 |
| Test w/PGD_JPEG | 72.78±0.9 | 51.87±0.42 | 46.8±1.08 | **51.64±0.42** | 67.13±0.79 |
| Test w/PGD_FFT | 85.69±0.54 | 14.48±1.39 | 5.83±0.51 | 6.34±0.68 | **77.88±0.43** |

**Table 2:** Robustness on attacks with adversarial training in a specific representation space (FashionMNIST). STD refers to standard/no adversarial examples.

We can observe that training against a specific adversarial attack only improves the robustness against that specific attack. This hypothesis seem to also apply for different representation space attack. In order to show this, we train the model with a specific attack in a specific representation space and see how it performs against different representation spaces (Table 2).

Notice that training on any adversarial attack has a trade-off with the natural accuracy, i.e., making the model robust against attacks makes it loose some accuracy on the original task.

We can notice that even if general training in different representations maximizes the performance on that representation space, there are some instances were this is not the case. This happens in simple cases with a lot of randomness involved. In Table 3, we can observe this due to the fact that FashionMNIST is relatively simple dataset and that FFGSM attack has an initial random perturbation that has a high impact in the results. This randomness, is the reason we believe that the previous statement is broken in this setting.

| | STD | Test FFGSM_Pixel | Test FFGSM_DCT | Test FFGSM_JPEG | Test FFGSM_FFT |
|---|---|---|---|---|---|
| STD | 88.56+0.55 | 12.07±1.93 | 9.42±1.43 | 9.48±1.57 | 71.83±0.64 |
| Train FFGSM_Pixel | 83.96±0.27 | **74.99±0.48** | 71.1±0.39 | **73.54+0.58** | 81.99+0.27 |
| Train FFGSM_DCT | 83.41±0.35 | 73.93±0.09 | **72.19±0.29** | 73.11±0.11 | 81.63+0.36 |
| Train FFGSM_JPEG | 82.98±0.46 | 74.02±0.55 | 71.18±0.59 | 73.28±0.47 | 81.1±0.49 |
| Train FFGSM_FFT | **89.32±0.4** | 37.84±3.27 | 30.5±2.95 | 31.11±3.17 | **84.4+0.57** |

**Table 3:** Training with FFGSM attack in a certain representation space (rows) and testing with FFGSM in another representation space (columns) in FashionMNIST.

### 9.1.3 On the efficacy of different representation spaces

From our experiments, DFT/FFT representation space seems to be the one performing the weakest perturbations, i.e., the perturbation that drops the accuracy the least when generating adversarial attacks. In order to prove this, we first generate FGSM attacks in different representations and see the visualization (Fig. 9) as well as a graph with the mean perturbation values (Fig. 10) to show how the perturbations in different representation spaces have a higher distortion than FFT.

In the Figure 9 we can observe that adversarial attacks on pixel, DCT and JPEG representation have a high impact on the FFT representation. Meanwhile, the attacks on FFT don't have that same magnitude of perturbation.

To further check this, we calculated the mean perturbation over the values of the image in Figure 10. Notice that in some representations spaces like DCT, each position has a different impact on the final image. For example the first position/pixel of the image has a higher impact on the perturbation.

Lastly, we create an autoencoder to check the impact of the different representation spaces. The idea is to show that the images generated with the FFT adversarial attacks are not so 'far away' from the initial image distribution when compared to other representation spaces attack. First, we show the reconstruction of natural images in Figure 11 to prove the correctness of the autoencoder model.

This autoencoder will be used to calculate the MSE loss between the reconstructed image and the reconstructed attacked image. This way, a

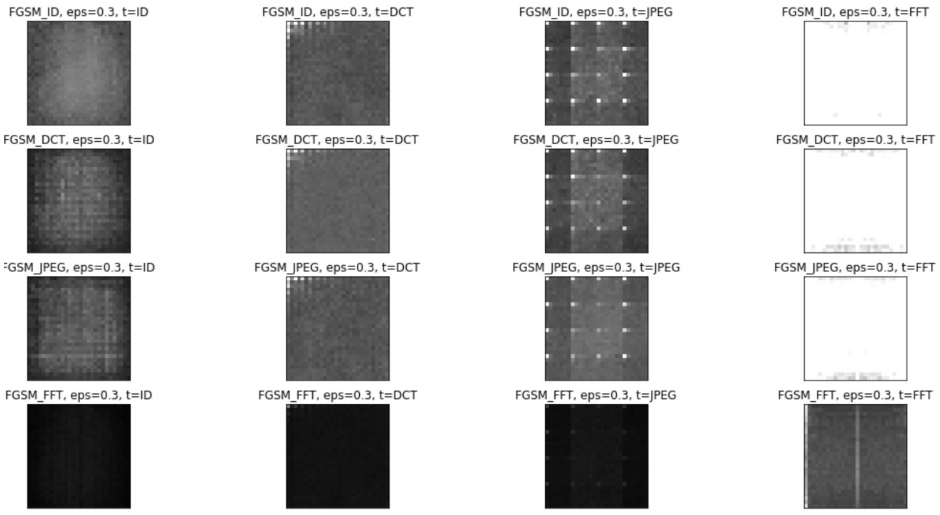



**Figure 9:** Visualization of the mean MSE for FGSM attack perturbations on different representation spaces. Rows perform the FGSM attack on different representation spaces, while the columns visualize the same perturbation on different representation spaces.

subjacent distance notion of pseudo-distance to the original manifold of the images can be calculating via the autoencoder. Thus, we show from the results in Table 4 that the perturbations of the FFT lie much closer to the original manifold compared to the other attacks.

| Sets of Inputs Used | MSLoss (Between Input and Reconstructed original pixel image) |
|---|---|
| Adversarial images in Pixel (PGD) | 0.01150194 |
| Adversarial images in DCT (PGD) | 0.01101580 |
| Adversarial images in FFT (PGD) | 0.00050876 |

**Table 4:** Reconstruction error of the attack after passing it trough the autoencoder.

### 9.1.4 Adversarial Training

The last experiment we did on FashionMNIST is to check the robustness of the model when trained with different adversarial training methods for a diverse set of attacks and representation spaces.

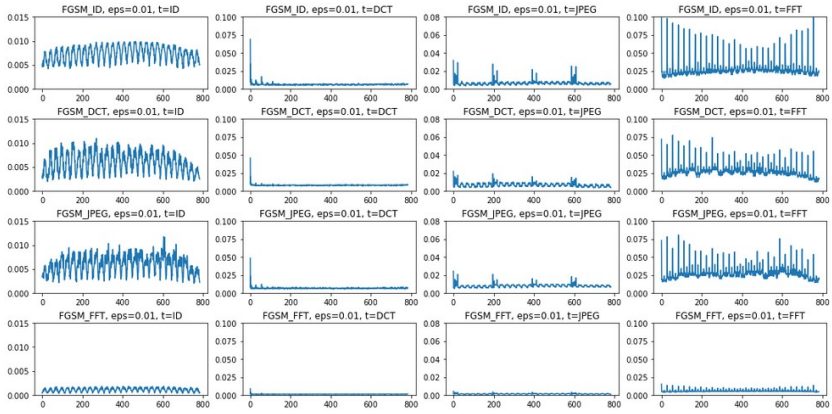



**Figure 10:** Mean MSE of the FGSM attack perturbations on different representation spaces. Rows perform the FGSM attack on different representation spaces, while the columns visualize the same perturbation on different representation spaces. The x-axis refers to the pixel position.

From the experiment in Table 5, we can conclude that:

- Training a model on adversarial attacks has a negative impact on the performance on clean/non-attacked inputs. This is something we have observed previously in section 9.1.2, where the performance also decreased when training for only 1 attack. In the current section, we showed that this is the case even when trained over different attacks and representation spaces.

- Greedy sacrifices performance on most "softer"/"weaker" attacks in order to increase only the performance on "harder"/"stronger" attacks. With "stronger" attacks we refer to those that are most difficult to be robust against, i.e., those that have the lowest accuracy even when trained on them. Due to the behaviour of the Greedy method, it only trains on the most difficult attacks (PGD, PGD DCT and PGD JPEG). This makes it drop the performance not only on the clean images but also on the other attacks.

- Round Robin is not able to perform well on "strong" attacks. Due to iteratively training on all the attacks, the most difficult cases are not trained enough while the simpler cases are trained too much. This allows Round Robin to perform well on "softer"/"weaker" attacks but it drops a lot the performance on more difficult cases.

- Multiplicative weights seems to be a good balance between the Greedy approach and the Round Robin method. It focuses more on the "com-

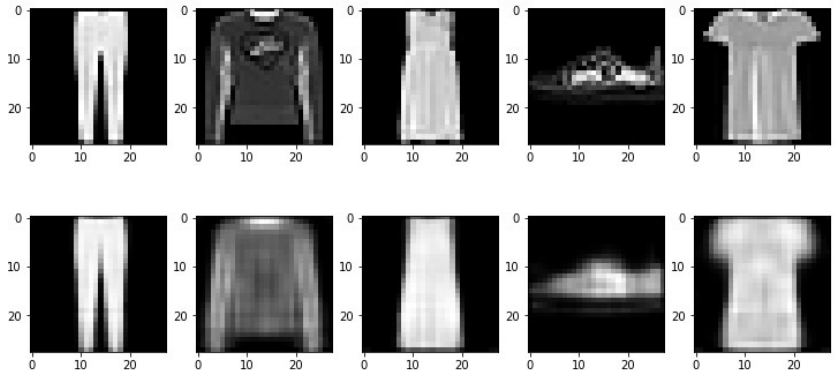


**Figure 11:** Autoencoder reconstruction. The top row are the original images, while the bottom row is the reconstruction of the autoencoder.

plex" cases (PGD, PGD DCT and PGD JPEG) while not sacrificing too much the performance on the "weaker" cases. It is not able to performa as well as Greedy on the "complex" cases or as well as Round Robin for the "easier" cases (Std and Rand), but it has the best or second best performance on every attack keeping a good balance over all of them.

| Method | Std | Rand | FGSM | FGSM DCT | FGSM JPEG | FGSM DFT | FFGSM |
|---|---|---|---|---|---|---|---|
| Std | **88.29±0.47** | 61.5±6.38 | 5.28±1.5 | 13.26±2.56 | 10.56±3.1 | 33.1±3.14 | 10.12±2.23 |
| Round Robin | 80.37±1.61 | **79.71±1.96** | 31.76±1.27 | 57.76±1.54 | 49.1±1.88 | 68.03±1.47 | 70.26±1.71 |
| Greedy | 68.54±0.3 | 68.22±0.4 | 31.12±2.29 | **58.58±0.42** | **53.05±1.14** | 62.68±0.37 | 64.32±0.26 |
| Mult. Weight | 79.64±0.59 | 78.66±0.57 | **35.92±1.96** | 58.56±1.08 | 50.81±1.48 | **68.7±0.62** | **70.94±0.66** |

| Method | FFGSM DCT | FFGSM JPEG | FFGSM DFT | PGD | PGD DCT | PGD JPEG | PGD DFT |
|---|---|---|---|---|---|---|---|
| Std | 7.75±1.65 | 7.47±1.68 | 72.03±0.78 | 0.0±0.0 | 0.0±0.0 | 0.0±0.0 | 23.26±4.08 |
| Round Robin | 68.11±1.73 | 69.2±1.82 | **78.33±1.69** | 42.79±1.65 | 32.92±1.85 | 36.54±2.04 | 72.35±1.99 |
| Greedy | 64.23±0.18 | 64.29±0.24 | 67.56±0.34 | **50.09±1.06** | **48.76±1.65** | **49.37±1.38** | 65.0±0.46 |
| Mult. Weight | **69.04±0.66** | **70.15±0.51** | 77.83±0.52 | 46.22±1.24 | 35.36±1.67 | 40.06±1.65 | **72.92±0.52** |

**Table 5:** Adversarial training on FashionMNIST. The table has been split in 2 tables in order to accomodate the values for the 12 different attacks (columns). Bold results refers to the best method results for that attack, while underlined refers to the second best method for the given attack. Rand refers to a random perturbation and STD refers to the model with no adversarial training/adversarial attacks.

## 9.2 CIFAR-10

CIFAR-10 [25] contains 10,000 test images and 60,000 32x32 color train images that have been divided into 10 groups. Aircraft, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and vehicles are among the ten categories in CIFAR-10. CIFAR-10 is more complex dataset used popularly in image classification tasks. Due to it being more complex and bigger, it takes longer to train than FashionMNIST. It is a good dataset to validate and test the scalability of algorithms. Now let us discuss more in detail about the experiments conducted.

### 9.2.1 Reproducing Baseline Experiment

Here we aim to confirm the initial hypothesis by the baseline that adversarial robustness does not scale across representation spaces. Adversarial accuracy was determined by averaging 5 times over the given procedure. The attacks used were FGSM, PGD, and PGD ($L_2$ norm) in their Pixel and DCT representations. Here we are only considering these attacks and spaces since the baseline paper experiments were conducted only on them. Standard accuracy (trained and tested without attacked images) and an AVG_ALL (row-wise average of all attacks metric) were also included for clarity. In addition to the 3 adversarial training procedures, Greedy, Round Robin and Multiplicative

Weights method, a standard training procedure was also used for benchmark.

| Procedures | Std | FGSM | FGSM DCT | PGD | PGD DCT | PGDL2 | PGDL2 DCT | AVG ALL |
|---|---|---|---|---|---|---|---|---|
| Std | **86.39+0.64** | 6.43+1.29 | 9.01+1.55 | 0.0+0.0 | 0.0+0.0 | 40.25+1.03 | 40.25+1.07 | 26.05+29.46 |
| Round Robin | 61.45+9.07 | <u>21.77+1.26</u> | <u>26.14+1.99</u> | 1.5+0.87 | 1.48+0.87 | <u>49.38+7.89</u> | <u>49.33+7.87</u> | <u>30.15+22.19</u> |
| Greedy | 32.47+4.73 | 21.08+1.67 | 25.08+3.06 | **25.08+3.06** | **21.79+2.34** | 30.79+4.41 | 30.79+4.42 | 25.95+4.94 |
| Mult. Weight | <u>62.9+15.43</u> | **21.97+3.26** | **26.76+4.76** | <u>3.32+2.72</u> | <u>3.61+3.13</u> | **49.38+11.52** | **49.34+11.48** | **31.04+21.74** |

**Table 6:** Adversarial training on CIFAR-10. Rand refers to a random perturbation and STD refers to the model with no adversarial training/adversarial attacks. Bold results refers to the best method results for that attack, while underlined refers to the second best method for the given attack.

It is no surprise that the standard procedure yields the highest accuracy at 86.39 as it does not have any adversarial attacks involved. Observing the other columns in the standard procedure, it is clear that they have fallen prey to the damage from the 3 attacks we used, dipping in accuracy by a considerable amount.

In the case of Round Robin procedure, it shows decent robustness to FGSM and PGDL2 while it struggles with a heavy attack like PGD irrespective of representation spaces.On the other end, the Greedy procedure does an appreciable job with PGD in both Pixel and DCT while it doesn't perform so well with FGSM and PGDL2 in comparison to Round Robin procedure. This can be explained by the design of the Greedy method, which emphasizes training on the heaviest attack. Finally, the proposed Multiplicative weights method performs the best or the second best in every case. It performs slightly better than Round Robin in most attacks and gets the highest average accuracy for all 3 methods here. Thus we have confirmed the results of the baseline paper.

### 9.2.2 Baseline Variation : Fixed set of 6 attacks

Progressing from the baseline experiments and aiming to further our findings, we perform training a fixed set of 6 attacks i.e with a slightly tweaked multiplicative weights method. Adversarial accuracy was determined by averaging 5 times over the given procedure. The attacks used for training this fixed setting were FGSM-FFT, FGSM-LOG, FFGSM-LOG, FFGSM-ID, PGD-FFT and PGDL2-DCT. ( The prefix is the attack and the postfix is the corresponding representation space). The postfix 'ID' denotes pixel representation space. Standard accuracy ( trained and tested without attacked images) and an AVG-ALL (row-wise average of all attacks metric ) were also included. Like the baseline experiment, we use the Greedy, Round Robin and Multiplicative Weights method, and the standard training procedure for the benchmarking. Here the basic idea is similar to cross validation, where we train and test on different sets of data. We have randomly chosen 6 attacks and trained only on them. We experimented with different sets of attacks and we have only reported the results of one case since they reported similar trends. We train only one the selected 6 attacks but testing is done on all the attacks.

Taking a look at the first row of Standard accuracies, we observe that LOG representations barely have dropped in accuracies and may not be well fit for adversarial attacks or learning. Other than this, the trend is the same as our baseline experiment where standard accuracy is the highest and the other attacks drop in accuracy.

Moving to our training procedures, Round Robin comes in as the 3rd best as the Greedy procedure takes the 2nd place. This can be explained by the fact that the attacks we chose for this experiment are relatively weak, greedy method localizes around them and performs well. With this observation we find that there could be niche situations where Greedy is suitable. Once again, the proposed multiplicative weights method performs the best with the highest average accuracy of the 3 methods.

| Procedures | STD | RAND ID | FGSM ID | FGSM FFT | FGSM DCT | FGSM JPEG | FGSM LOG | FFGSM ID |
|---|---|---|---|---|---|---|---|---|
| Std | **86.68+0.42** | **81.75+0.5** | 11.35+3.07 | 24.96+1.59 | 12.69+2.98 | 11.82+2.94 | 84.72+0.43 | 8.29+2.31 |
| Round Robin | 86.04+0.72 | 83.27+1.6 | 11.02+2.54 | 38.37+13.46 | 14.6+3.08 | 13.39+2.84 | **84.74+0.36** | 14.39+6.44 |
| Greedy | 79.75+8.91 | 77.81+7.84 | **15.88+7.2** | 44.73+14.21 | **21.48+10.05** | **20.66+10.54** | 78.81+8.41 | **23.02+13.29** |
| Mult. Weight | 81.04+8.03 | 79.32+7.28 | 15.52+6.31 | **45.88+12.5** | 20.94+8.79 | 19.9+9.26 | 80.11+7.63 | 22.86+11.56 |

| Procedures | FFGSM DCT | FFGSM JPEG | FFGSM LOG | PGD ID | PGD FFT | PGD DCT | PGD JPEG | PGD LOG |
|---|---|---|---|---|---|---|---|---|
| Std | 8.51+2.37 | 8.08+2.45 | **85.6+0.36** | 3.78+3.39 | 7.79+3.73 | 3.74+3.41 | 3.88+3.49 | 84.69+0.43 |
| Round Robin | 15.75+7.55 | 16.0+8.19 | 85.32+0.42 | 2.28+2.88 | 21.07+13.57 | 2.23+2.91 | 2.35+2.97 | **84.72+0.36** |
| Greedy | **25.51+15.12** | **26.0+15.64** | 79.23+8.62 | **6.14+5.96** | **32.14+19.18** | **7.32+7.59** | **8.75+9.38** | 78.79+8.39 |
| Mult. Weight | 25.12+13.17 | 25.56+13.63 | 80.52+7.8 | 4.81+5.65 | 31.53+16.71 | 5.65+7.18 | 6.76+8.83 | 80.1+7.62 |

| Procedures | PGDL2 ID | PGDL2 FFT | PGDL2 DCT | PGDL2 JPEG | PGDL2 LOG | AVG ALL |
|---|---|---|---|---|---|---|
| Std | 37.21+1.45 | 21.34+1.82 | 37.21+1.47 | 37.23+1.46 | **86.51+0.44** | 35.74+32.13 |
| Round Robin | 52.7+15.53 | 19.36+2.6 | 52.71+15.54 | 52.71+15.53 | 85.94+0.67 | 40.47+31.5 |
| Greedy | 55.98+13.51 | **28.97+13.77** | 55.99+13.52 | 55.99+13.5 | 79.68+8.87 | 43.52+26.32 |
| Mult. Weight | **58.15+12.29** | 25.28+13.58 | **58.15+12.3** | **58.15+12.29** | 80.96+8.0 | **43.77+27.51** |

**Table 7:** Adversarial training with a set of fixed attacks on CIFAR-10. The table has been split into 3 tables in order to accomodate the values for the 19 different attacks (columns). Bold results refers to the best method results for that attack, while underlined refers to the second best method for the given attack. Rand refers to a random perturbation and STD refers to the model with no adversarial training/adversarial attacks.

### 9.2.3 Baseline Variation : Dynamic set of 6 attacks

This variation is an interesting take on the baseline assumption. Here, instead of a fixed set of attacks while training, we dynamically change the set of attacks for each iteration.To be more precise, 6 alternating sets of attacks are trained during each epoch. Hence the training occurs in all the possible combinations of attacks. As seen in the above experiment, the testing is done on all the attacks. The postfix 'ID' denotes pixel representation space. Standard accuracy ( trained and tested without attacked images) and an AVG-ALL (row-wise average of all attacks metric ) were also included.

Standard accuracy is once again the highest metric on the table aided by clean samples as expected. The Greedy method does well for strong attacks like PGD and FGSM while not performing so well in a weak attack like PGD-l2. This trend is built by design and we have discussed in the previous experiment.

In a strange turn of events, the Round Robin method shows the best robustness in this experiment. Dynamically changing the attacks in each epoch, is inconsistent with the Baseline findings. This makes the Multiplicative weights method vulnerable in a not-so-ideal situation and hence it ends up behind Round Robin for this experiment.

| Procedures | Std (Nat. Acc.) | RAND | FGSM ID | FGSM DCT | FGSM JPEG | FGSM FFT | FFGSM ID |
|---|---|---|---|---|---|---|---|
| Std | **86.9+0.2** | **80.71+2.11** | 7.84+1.26 | 10.31+1.52 | 8.46+1.26 | 25.76+1.1 | 6.52+1.17 |
| Round Robin | 77.96+4.1 | 76.72+4.14 | 18.18+3.34 | 25.84+3.03 | 24.48+3.28 | **54.65+2.87** | 29.44+2.58 |
| Greedy | 46.26+4.66 | 46.07+4.55 | **24.04+3.56** | **32.35+3.91** | **32.04+3.88** | 41.18+4.17 | **31.88+4.08** |
| Mult. Weight | 77.22+7.17 | 76.56+6.99 | 17.51+3.65 | 24.67+4.72 | 23.16+4.78 | 52.57+2.2 | 27.49+4.3 |

| Procedures | FFGSM JPEG | FFGSM FFT | PGD ID | PGD DCT | PGD JPEG | PGD FFT | PGDL2 ID |
|---|---|---|---|---|---|---|---|
| Std | 6.64+1.31 | 39.04+0.7 | 0.01+0.02 | 0.0+0.01 | 0.0+0.0 | 1.95+0.71 | 40.72+0.93 |
| Round Robin | **33.59+2.58** | **63.58+3.39** | 2.04+1.34 | 2.35+1.77 | 2.93+1.9 | **42.52+3.53** | **66.23+3.21** |
| Greedy | 36.5+4.2 | 43.23+4.24 | **18.5+4.5** | **25.12+6.23** | **26.38+5.91** | 39.63+4.46 | 43.46+4.5 |
| Mult. Weight | 31.47+4.77 | 62.29+3.61 | 1.58+1.32 | 2.12+2.16 | 2.52+2.49 | 38.82+4.67 | 65.24+4.13 |

| Procedures | PGDL2 DCT | PGDL2 JPEG | PGDL2 FFT | AVG ALL |
|---|---|---|---|---|
| Std | 40.8+0.94 | 40.87+0.94 | 4.53+0.74 | 22.73+26.21 |
| Round Robin | **66.2+3.25** | **66.24+3.3** | 36.78+3.32 | **40.18+24.7** |
| Greedy | 43.46+4.51 | 43.48+4.51 | **38.32+4.22** | 36.01+8.09 |
| Mult. Weight | 65.2+4.18 | 65.24+4.16 | 33.55+5.37 | 38.81+24.66 |

**Table 8:** Adversarial training with a dynamic set of 6 attacks on CIFAR-10. The table has been split into 3 tables in order to accomodate the values for the 15 different attacks (columns). Bold results refers to the best method results for that attack, while underlined refers to the second best method for the given attack. Rand refers to a random perturbation and STD refers to the model with no adversarial training/adversarial attacks.

### 9.2.4 Training on all attacks and representation spaces

The final experiment conducted on the CIFAR-10 dataset points us to some key conclusions. In this setting, we explore all adversarial attacks and representation spaces, training them all at one go with the 3 respective methods. Adversarial Accuracy(↑) is Obtained by averaging 5 times over the given procedure. In the baseline paper, they claimed that the Multiplicative weights methods is scalable to multiple attacks and spaces. Through this experiment we are trying to confirm that. The postfix 'ID' denotes pixel representation space. Standard accuracy (trained and tested without attacked images) and an AVG-ALL (row-wise average of all attacks metric ) were also included.

Supporting our findings from the last experiment ( dynamic sets of attacks ), we find the Multiplicative weights method getting dominated by Round

| Procedures | Std (Nat. Acc.) | RAND ID | FGSM ID | FGSM DCT | FGSM JPEG | FGSM FFT | FGSM LOG |
|---|---|---|---|---|---|---|---|
| Std | **86.78+0.26** | **82.04+1.55** | 6.95+1.09 | 9.78+1.45 | 7.81+1.35 | 24.22+1.92 | **84.98+0.36** |
| Round Robin | 79.14+1.84 | 77.89+1.87 | **19.98+1.27** | **27.7+1.44** | **26.11+1.54** | **55.6+0.88** | 78.55+1.74 |
| Greedy | 28.69+2.59 | 28.68+2.64 | 18.88+1.52 | 22.34+1.77 | 22.5+1.44 | 26.28+1.98 | 28.57+2.55 |
| Mult. Weight | 77.64+2.22 | 76.23+2.87 | 19.56+2.42 | 27.07+1.95 | 25.38+2.11 | 52.4+0.54 | 77.03+2.18 |

| Procedures | FFGSM ID | FFGSM DCT | FFGSM JPEG | FFGSM FFT | FFGSM LOG | PGD ID | PGD DCT | PGD JPEG |
|---|---|---|---|---|---|---|---|---|
| Std | 5.29+1.05 | 6.58+1.16 | 5.41+1.14 | 37.95+1.49 | **85.78+0.29** | 0.0+0. 0 | 0.0+0.0 | 0.0+0.0 |
| Round Robin | **30.5+1.53** | **34.68+1.5** | **34.59+1.48** | **64.78+0.85** | 78.82+1.75 | 1.4+0.17 | 1.69+0.43 | 2.12+0.48 |
| Greedy | 22.39+0.96 | 23.97+1.69 | 24.42+1.57 | 27.09+2.14 | 28.64+2.59 | **16.39+2.13** | **18.5+2.38** | **19.73+1.79** |
| Mult. Weight | 28.6+1.63 | 32.53+1.66 | 32.4+1.4 | 61.66+1.02 | 77.31+2.2 | 1.33+0.42 | 1.43+0.47 | 1.75+0.56 |

| Procedures | PGD FFT | PGD LOG | PGDL2 ID | PGDL2 DCT | PGDL2 JPEG | PGDL2 FFT | PGDL2 LOG | AVG ALL |
|---|---|---|---|---|---|---|---|---|
| Std | 1.62+0.36 | **84.94+0.37** | 39.94+0.95 | 39.98+0.88 | 39.94+0.9 | 4.27+0.24 | **86.64+0.24** | 33.68+34.21 |
| Round Robin | **42.09+1.33** | 78.55+1.74 | **67.32+1.01** | **67.31+0.94** | **67.31+1.01** | **36.29+1.46** | 79.09+1.81 | **47.8+27.0** |
| Greedy | 25.42+1.86 | 28.57+2.55 | 27.16+2.08 | 27.17+2.1 | 27.18+2.1 | 25.07+1.69 | 28.67+2.58 | 24.83+3.71 |
| Mult. Weight | 38.09+0.99 | 77.02+2.16 | 64.82+1.08 | 64.84+1.05 | 64.85+1.13 | 34.55+2.3 | 77.59+2.22 | 46.09+26.5 |

**Table 9:** Adversarial training on all attacks and representation spaces on CIFAR-10. The table has been split into 3 tables in order to accomodate the values for the 20 different attacks (columns). Bold results refers to the best method results for that attack, while underlined refers to the second best method for the given attack.Rand refers to a random perturbation and STD refers to the model with no adversarial training/adversarial attacks.

Robin with Greedy finishing at the last place. Greedy coming in the last is not a surprise as it does well only for the strongest PGD attacks and falls behind for every other attack.

Round Robin performs consistently and as per design gives equal importance to each attack. However, the proposed Multiplicative weights method falls behind as it fails to properly estimate the impact of each attack and assign weights accordingly. This method worked well in the baseline with a specific set of attacks but when new attacks and representation spaces are added to the ensemble it fails to reproduce the results. It can also be due to the complexity of the dataset. If more 'strong' attacks are added, the model tends to train more on those attacks and hence drop the overall accuracy. Hence giving more weight to only complex attacks may not be the right approach. Hence this method does not seem to extend to all scenarios and does not achieve true robustness across representation spaces.

# 10 Conclusions

After doing all these experiments and studying the results, we have arrived at the following conclusions:

- A model trained in a specific representation space doesn't translate its robustness to other representation spaces. This is similar to the idea that training against a specific attack doesn't make the model robust against any attack. It only makes the model robust against the attack it has been trained on.

- The performance of Multiplicative Weights method is better than Round Robin for small/medium amount of attacks/representation spaces (<17) in less complex data-sets (MNIST, FashionMNIST). It seems that in the settings where the number of "strong" attacks is not very high, Multiplicative Weights method can outperform Round Robin since it would train mostly on "harder"/"stronger" attacks while still training on "softer"/"weaker" attacks.

- In contrast to the previous conclusion, the claim of Multiplicative Weights being *scalable* does not hold true for a higher amount of attacks (>20) and for more complex data-sets (CIFAR10, CIFAR100). Multiplicative Weights method tends to collapse to a method more similar to the Greedy approach in the complex cases. More probability is given for attacks with low robustness ("harder"/"stronger" attacks). Therefore, there is no longer a significant amount of training on "softer"/"weaker" attacks in Multiplicative Weights method. This makes the model focus on harder attacks and drop the overall accuracy in favor of the more challenging cases. Due to this, Round Robin training ends up outperforming Multiplicative Weights.

  One hypothesis with regard to this, is that the constant increment of the weight given to the more complex attacks is hindering the performance. The model is only able to visit them a certain amount of times before it updates the weights in the Mult. Weights method. But due to the updation of the weights before the model has been able to train on a wide diversity of attacks, the error keeps increasing before the model is robust against them. By the later iterations, the weight has increased so much in the "harder" cases that the "weaker" ones have a really small probability which makes the method be closer to the Greedy approach.

- Not all representation spaces have the same impact for adversarial training. For example, Log space and DFT tend to squish down the perturbation after doing the inverse transformation, reducing the impact of the attack. It could be argued that using the same $\epsilon$-sphere than other representations is not the way to go and that it should be increased due to the spatial behaviour of those representation spaces. Nevertheless, a deeper study should be done in order to prove if there is a need to change the $\epsilon$ value or not.

With these findings, we conclude that the current state-of-the-art methods for training against several adversarial attacks do not deliver a satisfactory result. More research is needed in this area, which could be correlated with importance sampling. We have shown that training equitably on every attack doesn't deliver a satisfactory model as shown with the results on the Round Robin method. Nevertheless, focusing only on the more 'complex' attacks also doesn't solve the issue. A new method is needed in the direction of Multiplicative Weights but with better scalability properties in regard to the number of attacks.

# References


[1] Yu. George et al. Adversarial robustness across representation spaces. arXiv:2012.00802, 2020.

[2] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083.

[3] Modas, Apostolos and Moosavi-Dezfooli, Seyed-Mohsen and Frossard, Pascal. *SparseFool: a few pixels make a big difference*. 2018, arXiv:1811.02248 [cs.CV].

[4] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, Ananthram Swami.*Practical Black-Box Attacks against Machine Learning*. 2016. arXiv:1602.02697 [cs.CR].

[5] Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh. *EAD: Elastic-Net Attacks to Deep Neural Networks via Adversarial Examples* 2017, arXiv:1709.04114v3 [stat.ML].

[6] Malhar Jere, Briland Hitaj, Gabriela Ciocarlie, Farinaz Koushanfar. *Scratch that! An Evolution-based Adversarial Attack against Neural Networks.* 2019, arXiv:1912.02316v1 [cs.NE].

[7] Francesco Croce, Maksym Andriushchenko, Naman D. Singh, Nicolas Flammarion, Matthias Hein *Sparse-RS: a versatile framework for query-efficient sparse black-box adversarial attacks*. 2020, arXiv:2006.12834v3 [cs.LG].

[8] N. Carlini and D. Wagner. *Towards Evaluating the Robustness of Neural Networks*. 2017 IEEE Symposium on Security and Privacy (SP), 2017, pp. 39-57, doi: 10.1109/SP.2017.49.

[9] Moustafa Alzantot, Yash Sharma, Supriyo Chakraborty, Mani Srivastava *GenAttack: Practical Black-box Attacks with Gradient-Free Optimization*. 2018, arXiv:1805.11090v1 [cs.LG].

[10] Dave Marshall, *The Discrete Cosine Transform (DCT)*, 2001.

[11] ihritik, MATLAB, *RGB image representation*, 2018, GeeksforGeeks.

[12] Shuhao Cao, Replicate the Fourier transform time-frequency domains correspondence illustration using TikZ, 2013, stackexchange.

[13] Chuan Guo, Jacob R. Gardner, Yurong You, Andrew Gordon Wilson, Kilian Q. Weinberger. *Simple Black-box Adversarial Attacks*. 2019, arXiv:1905.07121v2 [cs.LG].

[14] Kurakin, A., Goodfellow, I., and Bengio, S. (2017). Adversarial Machine Learning at Scale. In Proceedings of the 2017 Conference on Neural Information Processing Systems (NIPS), pages 5767–5775.

[15] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2018). Towards Deep Learning Models Resistant to Adversarial Attacks. In Proceedings of the 35th International Conference on Machine Learning (ICML), pages 3323-3332.

[16] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, Adrian Vladu *Towards Deep Learning Models Resistant to Adversarial Attacks*. 2017, arXiv:1706.06083v4 [stat.ML].

[17] Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy *Explaining and Harnessing Adversarial Examples*. 2014, arXiv:1412.6572v3 [stat.ML].

[18] Akhtar, N., Mian, A., & Rakin, M. (2018). Threat of adversarial attacks on deep learning in computer vision: A survey. arXiv preprint arXiv:1801.00553.

[19] Perez-Gonzalez, F., & Tramer, F. (2020). Quantized adversarial training: An efficient defense against adversarial attacks. arXiv preprint arXiv:2010.11339.

[20] Guo, C., Rana, M., Cisse, M., & Van Den Oord, A. (2020). Countering adversarial attacks using generative models. arXiv preprint arXiv:2002.05709.

[21] Li, Y., & Chen, Y. (2018). Theoretical foundations of adversarial examples and defenses. arXiv preprint arXiv:1805.12152.

[22] Xu, W., Liu, D., & Jiao, J. (2019). Feature squeezing: Detecting adversarial examples in deep neural networks. arXiv preprint arXiv:1904.01108.

[23] Hendrycks, D., & Gimpel, K. (2019). A baseline for adversarial robustness. arXiv preprint arXiv:1902.06705.

[24] Han Xiao et. al. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. arXiv:1708.07747v2 [cs.LG] 15 Sep 2017.

[25] Alex Krizhevsky *Learning Multiple Layers of Features from Tiny Images*. Computer Science University Toronto, 8 April 2009.

[26] Ian J. Goodfellow et al. Explaining and harnessing adversarial examples. arXiv:1412.6572, 2014

[27] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385.

[28] Guo, C., Rana, M., Cisse, M., & Van Den Berg, R. (2017). Countering adversarial images using input transformations. arXiv preprint arXiv:1711.00117.

[29] Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., & Frey, B. (2015). Adversarial autoencoders. arXiv preprint arXiv:1511.05644.

[30] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

[31] Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-MNIST: a novel image dataset for benchmarking

[32] machine learning algorithms. arXiv preprint arXiv:1708.07747.

[33] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[34] Florian Trame‘r and Dan Boneh. Adversarial training and robustness for multiple perturbations. In Advances in Neural Information Processing Systems, pages 5866–5876, 2019. 2, 5