# Adversarial Robustness Across Representation Spaces

Pranjal Awasthi* George Yu* Chun-Sung Ferng Andrew Tomkins
Da-Cheng Juan
Google Research
[pranjalawasthi, georgeyu, csferng, tomkins, dacheng]@google.com

## Abstract

*Adversarial robustness corresponds to the susceptibility of deep neural networks to imperceptible perturbations made at test time. In the context of image tasks, many algorithms have been proposed to make neural networks robust to adversarial perturbations made to the input pixels. These perturbations are typically measured in an $\ell_p$ norm. However, robustness often holds only for the specific attack used for training. In this work we extend the above setting to consider the problem of training of deep neural networks that can be made simultaneously robust to perturbations applied in multiple natural representations spaces. For the case of image data, examples include the standard pixel representation as well as the representation in the discrete cosine transform (DCT) basis. We design a theoretically sound algorithm with formal guarantees for the above problem. Furthermore, our guarantees also hold when the goal is to require robustness with respect to multiple $\ell_p$ norm based attacks. We then derive an efficient practical implementation and demonstrate the effectiveness of our approach on standard datasets for image classification.*[1]

## 1. Introduction

In recent years deep learning has enjoyed tremendous success in solving a variety of machine learning tasks, even achieving or surpassing human level performance in certain cases [14, 15]. At the same time important vulnerabilities in these systems have also been discovered. One such example is their susceptibility to imperceptible perturbations made to the input at test time [24]. This has led to the new paradigm of *adversarial machine learning*, i.e., making deep neural networks robust to test time perturbations. There has been a flurry of recent works in this area with several proposed defenses [18, 29, 6, 17] and methods to attack and evaluate these defenses [5, 3, 26]. When studying the design of networks robust to adversarial attacks several aspects need to be considered such as a) what perturbations can the adversary apply to the input, and b) what information does the adversary have about the neural network? One widely-studied setting in the current literature is *white box* attacks under $\ell_p$ norm perturbations [10, 18]. Here the adversary has complete knowledge of the neural network and its parameters, and given an input $x$ it can perturb it to $x'$ such that $\|x - x'\|_p \leq \epsilon$ for some $p \geq 1$ specified apriori. In the context of image data this corresponds to applying perturbations to the input pixels. Current approaches for defending against such attacks are based on studying variants of the following robust objective:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim D}\Big[\max_{x':\|x-x'\|_p\leq\epsilon} L(f_\theta(x'), y)\Big]. \qquad (1)$$

Here $(x, y)$ is an example and label pair drawn from the data distribution, $f$ is a neural network parameterized by weights $\theta$ and $L$ is a standard loss function such as the cross entropy loss. As an example the popular projected gradient descent (PGD) method [18] proposes to optimize the above objective by alternately maximizing the inner objective via gradient ascent and then performing the outer minimization via gradient descent. The recent work of [22] combines the above objective with Gaussian smoothing to achieve *certified robustness* guarantees, and another popular method namely the TRADES algorithm [29] adds a regularization term requiring the predictions of the network at $x$ and $x'$ to be close to each other.

In this work we aim to address two main limitations of current approaches to adversarial machine learning. The first concerns the choice of the representation in which the adversary applies the perturbations. Using images as an example, current approaches model the adversary as making small magnitude changes in the pixel representation of the image. However, given that the adversary has full access to the input $x$, apriori there is no reason to restrict the perturbations to only the pixel representations. Real data such as images have many other natural representations, such as the *discrete cosine transform* (DCT) basis for images. One

*Equal contribution and corresponding authors.

[1]Code available at https://github.com/tensorflow/neural-structured-learning/tree/master/research/multi_representation_adversary.

could envision an adversary making changes to the input image in the DCT basis that are still imperceptible but don't satisfy the small $\ell_p$ norm property in the pixel basis. Empirical attacks based on this have been shown to be successful in recent works [4]. Hence it is important to consider adversarial robustness in other representations for a model to be truly robust. Secondly, current approaches fix a representation and the perturbation model, and design an algorithm to achieve robustness for that specific setting. In general such networks do not turn out to be robust to other types of attacks. For example a network trained to be robust to $\ell_\infty$ norm perturbations in the pixel representation may not be robust to $\ell_1$ norm perturbations.

Ideally, one would like to train networks that can be simultaneously robust to multiple attack models in multiple representation spaces. At the same time it is desirable to have a scalable solution with training cost not that much more than standard adversarial training in a fixed attack model. This is precisely the problem that we solve in this work. Our main contributions are listed below.

- We propose and motivate the problem of studying robustness to adversarial perturbations in multiple representation spaces and under multiple attack models.
- We propose a min-max formulation of the above scenario and use ideas from the theory of online learning, in particular the multiplicative weights update method [13] to design an algorithm for our formulation and provide theoretical guarantees to justify our approach.
- We extend our theoretically principled algorithm to design a practical implementation that can scale to multiple representation spaces and multiple attack models with training cost not significantly more than that of standard adversarial training for a fixed attack model and representation space. We demonstrate the effectiveness of our algorithm for image classification tasks on the MNIST [16] and the CIFAR-10 [14] datasets.

## 2. Related Work

There is a vast amount of literature on defenses and attacks for adversarial robustness. See [26] for a survey. Here we discuss the works most relevant to the results of the paper. As mentioned in the introduction most existing defenses for adversarial robustness design customized solution for a fixed attack model ($\ell_p$ norm) and representation space (pixel basis). These methods are aimed at approximately solving the robust optimization objective in (1). The FGSM method [10] solves the inner maximization problem via one step of a gradient ascent whereas the PGD method [18] performs multiple iterations of gradient ascent to better optimize the inner objective. Typically this scales the cost of training linearly with the number of iterations used in the inner maximization. There have been recent works aimed at achieving the same performance as the PGD method but with faster training time [23, 27].

The above approaches provide robustness to first order attacks that are of the same type that are used in training. There has also been a lot of recent work on *provably* certifying the robustness of neural networks via approaches such as interval bound propagation [11], semi-definite programming [20], and randomized smoothing [6, 17].

Relatively little work exists on studying robustness to multiple types of attacks simultaneously and in multiple representation spaces. The recent work of [25] studies training classifiers that are simultaneously robust to perturbations to the input pixels of different $\ell_p$ norms. However they do not consider multiple representation spaces. Furthermore, their approach does not come with theoretical guarantees and scales linearly with the number of perturbations considered. In contrast our algorithm comes with theoretical guarantees and has a training cost that is not much more than that of adversarial training for a fixed attack. The recent work of [4] motivates the problem of studying certified robustness in other representations such as the DCT basis. However they do not consider training classifiers that are simultaneously robust to multiple attack models.

## 3. Adversarial Robustness in Multiple Representations

In this section we motivate the need for studying adversarial robustness in representations other than the one that is input to the network. Real world data can be represented in many natural representations, each with their own appealing properties. For instance, in the context of images, the DCT basis is a popular choice and it is well known that signals when represented in this basis are sparse. This has been exploited in recent works [4] to achieve better robustness to $\ell_\infty$ perturbations in this representations.

In the context of adversarial learning a fundamental question to ask is: *what constitutes an imperceptible perturbation?* Is it enough for an adversarially perturbed example to have a small $\ell_p$ norm in the pixel representation? As one can imagine, this is not a sufficient condition for imperceptibility. Many works have notice that images and their adversarial perturbations made in the pixel basis have distinct spectral signatures when viewed in other bases such as the discrete cosine transform (DCT). This has led to the proposal of many learning systems for detecting pixel based adversarial attacks using properties of images in other representations [2, 28, 9]. In particular, the work of [19] shows that one can achieve high accuracy in detecting adversarial perturbations made in the pixel representation by training a binary classifier to separate real and perturbed images. Hence an adversary has to naturally think about attacking

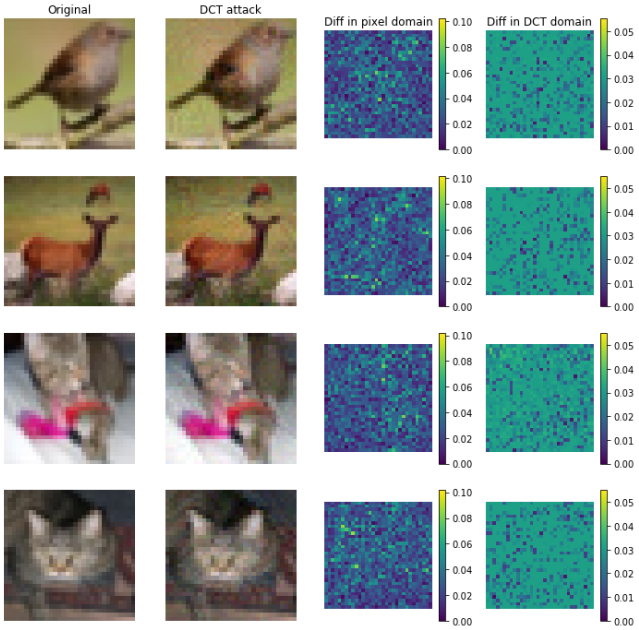


Figure 1. The figure shows examples of images from the CIFAR-10 dataset with their adversarially perturbed counterparts computed by launching a PGD based attack in the DCT basis. The perturbed images, although imperceptible, are far from the original images in the pixel basis in $\ell_\infty$ norm.

the model in multiple representations simultaneously in order to fool such systems.

Additionally, working in multiple representation spaces can help an adversary craft stronger attacks. As an example, the recent work of [4] provides examples where one can generate imperceptible examples by perturbing the image in the DCT basis and at the same time the perturbed examples are far way from the original image in the original pixel basis. Such an attack can fool classifiers that are only trained for defending against small norm $\ell_p$ attacks in the pixel representation. We further illustrate this in Figure 1. The figure shows examples of images from the CIFAR-10 dataset and corresponding adversarial perturbations computed by launching a PGD based adversarial attack in the DCT basis. For the case of CIFAR-10 it is generally accepted that $\ell_\infty$ perturbations in the pixel basis upto a magnitude of $\epsilon = 0.03$ constitute imperceptible perturbations. However, the adversarial images obtained in the Figure via working in the DCT basis, while being imperceptible, have much higher $\ell_\infty$ distance from the true images in the pixel representation.

From the above discussion we conclude that it is an important problem to design classifiers that are simultaneously robust against adversarial attacks in multiple representation spaces. Unfortunately, simply performing standard adversarial training in a fixed space is not enough in order to achieve this goal. As an example in Table 1 we show the performance of two neural networks, one trained adversarially in the pixel representation and the other in the DCT representation. As can be seen the trained networks have very poor robustness against the attacks that were not considered during training.

| | Test w/pixel $\ell_\infty$ | Test w/DCT $\ell_\infty$ | Nat. Acc. |
|---|---|---|---|
| Train w/pixel $\ell_\infty$ | 44.02 $\pm$1.02 | 27.30 $\pm$1.44 | 80.24 $\pm$0.40 |
| Train w/DCT $\ell_\infty$ | 11.80 $\pm$0.68 | 51.92 $\pm$0.43 | 74.92 $\pm$0.61 |

Table 1. The rows of the table correspond to two classifiers that have been adversarially trained via the PGD method for $\ell_\infty$ robustness either in the pixel basis or the DCT basis. The first two columns show the adversarial accuracies achieved by the classifiers against $\ell_\infty$ attacks in the pixel and the DCT basis. The last column displays the natural accuracy. As can be seen no classifier is simultaneously robust to both types of attacks.

As a result of the above observations what is needed is a general algorithmic approach for such scenarios. We next formulate and present such an approach.

## 4. Formulation and Algorithms

We next formulate the above scenario and design a near optimal algorithm for simultaneously achieving robustness across multiple representation spaces. We fix a canonical representation (say the pixel basis) and denote $x \in \mathbb{R}^d$ as examples and $y$ being the label. We assume that the ex-

ample and label pairs $(x, y)$ are drawn from an unknown joint distribution $D$. We then consider a given set of $k$ representation spaces with corresponding maps being given by $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_k$. Hence, given an example $x \in \mathbb{R}^d$ its representation in space $i$ is given by $\mathcal{R}_i(x) \in \mathbb{R}^{d_i}$. It would be instructive to think of $\mathcal{R}_i$ as the DCT basis although in general these maps could be non-linear. The only assumption we require is that the maps be surjective, i.e. $\mathcal{R}_i^{-1}$ exists. We will overload notation and denote by $\mathcal{R}_i$ both the $i$th representation and the map corresponding to it. For each $\mathcal{R}_i$ and any $x \in \mathbb{R}^d$, we denote by $B_i(x)$ the set of allowed perturbations to $x$ in the representation space $\mathcal{R}_i$. For example if we are modeling an $\ell_\infty$ attack of radius $\epsilon$ in the representation $\mathcal{R}_i$ then we have $B_i(x) = \{z \in \mathbb{R}^{d_i} : \|\mathcal{R}_i(x) - z\|_\infty \leq \epsilon\}$. In this work we will be able to deal with very general perturbation sets. Given a fixed representation space $\mathcal{R}_i$, the problem of learning a robust classifier specific to $\mathcal{R}_i$ can be written as that of minimizing:

$$\min_\theta L_i(\theta) = \mathbb{E}_{(x,y)\sim D}\big[\max_{z \in B_i(x)} L(f_\theta(\mathcal{R}_i^{-1}(z)), y)\big]. \quad (2)$$

Then given $k$ representation spaces $\mathcal{R}_1, \ldots, \mathcal{R}_k$ our goal is to solve the following:

$$\min_\theta \max_i L_i(\theta). \quad (3)$$

The above min-max formulation lends itself naturally to techniques from online learning. In particular, consider a two player game with the row player as the one that chooses the network parameter $\theta$ and the column player as the one that chooses the loss functions $L_i$ with the payoff for the column player being $L_i(\theta)$. From the minimax theorem [13] for two player games, we know that if for every distribution over the $k$ columns there exists a good solution $\theta$, then there exists a distribution over solutions that is simultaneously good for all the columns, i.e., the $k$ loss functions. This immediately provides a way to solve the min-max formulation via solving a simple *cost sensitive* adversarial optimization problem. Such techniques have been widely used in the literature to solve a variety of constrained problems in machine learning [1, 8]. Here we demonstrate their applicability for adversarial robustness. There has also been recent work on algorithms for solving optimization of the form $\min_\theta \max_{\lambda \in \Lambda} \sum_i \lambda_i L_i(\theta)$ [7] for $\Lambda$ being a convex set and functions $L_i$ being convex in $\theta$. Our loss functions are non-convex in $\theta$ and hence we need access to a cost sensitive optimization oracle to provide overall guarantees for our formulation. We next define the *adversarial cost sensitive* optimization problem.

**Definition 4.1.** *Given weights $w_1, w_2, \ldots, w_k$ with $w_i \geq 0$ and non-negative losses $L_1, L_2, \ldots, L_k$ the adversarial cost sensitive optimization corresponds to finding an approximately optimal solution $\hat{\theta}$ such that*

$$\sum_{i=1}^k w_i L_i(\hat{\theta}) \leq \min_\theta \sum_{i=1}^k w_i L_i(\theta) + \delta. \quad (4)$$

*Here $\delta$ quantifies the additive error in approximating the cost sensitive objective.*

We will show how to convert an algorithm for solving the adversarial cost sensitive optimization problem above to provably optimize (3). The algorithm is based on the popular multiplicative weights update method [13] and is described in Figure 2. For the proposed algorithm we show the following guarantee

**Theorem 4.2.** *For a given set of non-negative losses bounded in $[0, R]$, if the adversarial cost sensitive optimization in (4) can be solved to additive error $\delta$ for any setting of non-negative weights then the algorithm in Figure 2 when run with $\eta = O(\epsilon/R)$ and $T = O(\frac{R^2 \log k}{\epsilon^2})$ outputs a uniform distribution $P$ over solutions $\theta_1, \theta_2, \ldots, \theta_T$ such that*

$$\max_i \mathbb{E}_{\theta \sim P} L_i(\theta) \leq \min_\theta \max_i L_i(\theta) + \epsilon + \delta. \quad (5)$$

*Furthermore if the loss function $L$ in (2) is convex in its first argument, such as the cross-entropy loss, squared loss and hinge loss to name a few, then the average hypothesis $f_{\hat{\theta}}$ satisfies*

$$\max_i L_i(f_{\hat{\theta}}) \leq \min_\theta \max_i L_i(\theta) + \epsilon + \delta. \quad (6)$$

*Here $L_i(f_{\hat{\theta}})$ refers to the loss incurred by the ensembled hypothesis as output by the algorithm in Figure 2.*

The proof can be found in Appendix 8 in the supplementary material.

## 5. A Practical Implementation

While the algorithm in Figure 2 and the associated guarantee in Theorem 4.2 provide a principled way to approach the optimization in (3), we need to make a number of modifications to the core algorithm in order to obtain a practical and scalable implementation. In particular, we do not want the cost of training the robust classifiers to scale linearly with $k$ the number of representation spaces. We first discuss solving the adversarial cost sensitive optimization in (4). In practice, each $L_i(\theta)$ itself represents a hard optimization problem (of the form (2)). Luckily, there exists first order algorithms such as the PGD method [18] to optimize each $L_i$ separately that work well in practice. We now show how to combine them to solve (4). We follow the methodology of stochastic optimization and proceed in epochs. In each epoch, we sample a mini batch of $B$ data

**Input:** Training data $\{(x_1, y_1), \ldots, (x_m, y_m)\}$, Scaling factor $\eta$.

1. Initialize $w_i = 1$ for all $i \in [k]$.
2. For $t = 1 \ldots, T$ do:
   - Compute $\theta_t$ by approximately optimizing (4) with normalized weights $\frac{w_i}{\sum_{j=1}^{k} w_j}$ as inputs.
   - For all $i$ set $w_i = w_i \cdot e^{\eta L_i(\theta_t)}$.
3. Output $f_{\hat{\theta}} = \frac{1}{T} \sum_{t=1}^{T} f_{\theta_t}$.

Figure 2. An algorithm achieving robustness simultaneously across representation spaces.

**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 3. A scalable variant of the algorithm in Figure 2.

points, sample a loss $L_i$ with probability proportional to its current $w_i$ and then run the corresponding PGD based algorithm for optimizing $L_i$ on the current mini batch. After a few epochs of optimization we update the weights $w_i$ of the losses as described in the algorithm in Figure 2. In order to evaluate the losses for the weight update we use a separate validation set. This significantly reduces the variance in our estimates.

Next we consider approximating the output $f_{\hat{\theta}}$. Notice that the guarantee of Theorem 4.2 applies to an ensemble of $T$ neural networks provided by parameters $\theta_1, \theta_2, \ldots, \theta_T$. Maintaining this ensemble requires a high storage cost and makes the final output model impractically big. We first notice that if the losses $L_i$ were convex, then the guarantee of Theorem 4.2 will also hold for the average parameter, i.e., $\hat{\theta} = \frac{1}{T} \sum_t \theta_t$. To get a practical implementation we make a *near convexity* assumption on the losses and simply take the average of the model weights. Furthermore, in our experiments we observe that taking the average of the last few model parameters performs better than the uniform average of all the model weights. Fixing these choices leads to a scalable variant as shown in Figure 3.

## 6. Experimental Evaluation

We next demonstrate the effectiveness of our approach on the task of learning a classifier for image classification that is simultaneously robust to adversarial attacks in multiple representations spaces.

**Datasets.** We perform the experimental evaluation on two public datasets namely the MNIST dataset [16] and the CIFAR-10 dataset [14]. The MNIST dataset consists of $60,000$ training images with each being a $28 \times 28 \times 1$ tensor. The CIFAR-10 dataset consists of $50,000$ training images each of dimensionality $32 \times 32 \times 3$. Both the datasets consist of $10,000$ test images and correspond to a multi class classification problem with 10 class labels. In each case we reserve $10\%$ of the training data to be used as the validation set in the Algorithm from Figure 3. This validation set will be used to evaluate the loss $L^{\text{val}}$ in the algorithms. In the Appendix we also include experiments on the Tiny ImageNet dataset [21].

**Representation Spaces and Attack Models.** To demonstrate the scalability of our approach we consider two dif-

ferent representation spaces namely the pixel basis and the DCT basis. In each representation space we consider three types of $\ell_p$ norm based attacks for $p = 1, 2, \infty$. Hence, in total we have 6 loss functions $L_i$ to optimize as in (3). All our experiments are conducted on a ResNet-50 deep neural network that is a popular architecture for training on image classification tasks [12]. To compute adversarial examples in the pixel basis for norm bounded $\ell_\infty$ and $\ell_2$ perturbations we use the standard PGD based attack as proposed in [18]. For computing a norm bounded $\ell_1$ perturbation we use the sparse ascent algorithm namely the SLIDE method as proposed in [25]. To compute an adversarial attack in the DCT basis we append the ResNet-50 architecture with a linear DCT transformation follows by an inverse DCT transformation as shown in Figure 4. Notice that both the DCT and the Inverse DCT are fixed linear layers and in the absence of any perturbations the output of the network in Figure 4 is exactly the same as the original ResNet-50 network.

Using the modified architecture we first compute the DCT representation of the image and then launch an adversarial perturbation in the DCT basis using either the PGD method (for $\ell_2, \ell_\infty$ attacks) or the SLIDE method (for $\ell_1$ attacks). In this way we get the perturbed image after taking the inverse DCT transform of the perturbed example $x'$ as shown in Figure 1. After computing the adversarial perturbation and passing it through the inverse DCT transform we clip the pixel values in $[0, 1]$ to make the example a valid input for the ResNet-50 network. We pick pixel and DCT spaces as examples to demonstrate the effectiveness of our algorithm. Our approach is general and can be applied to other natural spaces. In the appendix we also include experiments with other perturbations/spaces such as rotations.

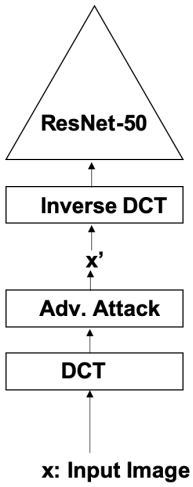


Figure 4. The modified network architecture for computing an adversarial perturbation in the DCT basis.

**Baselines.** We next describe the baselines that we use when comparing our proposed approach. The problem of being simultaneously robust to multiple adversarial attacks has been largely ignored in the literature so far. The recent work of [25] studies being robust in pixel basis to different $\ell_p$ norm based attacks. However the proposed method is not scalable to a large number of attacks.

We instead compare our proposed algorithm with following two baseline heuristics. We choose these heuristics due to their simplicity and more importantly due to their scalability. Furthermore, the reader should also compare our results to those in Table 1 that shows the results of training a state-of-the-art model trained using the PGD [18] method for a single representation space.

**Round Robin.** The round robin heuristic sketched in Figure 5 follows the same outline as the algorithm in Figure 3 except that instead of maintaining and updating weights, it simply picks the loss function $L_i$ to apply to a mini batch in a fixed order. It is easy to see that this method scales very well.

**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 $\hat{\theta} = \theta_T$.

Figure 5. The round robin heuristic.

**Greedy.** The greedy heuristic also follows the same outline as the algorithm in Figure 3. However, for each time step it chooses the loss with the worst error (on the validation set) to apply next. See Figure 6.

**Hyperparameter Configurations.** Next we discuss the hyperparameters we use when computing the adversarial perturbations for the different $\ell_p$ norm based attacks. When running our proposed algorithm in Figure 3 and the greedy heuristic, we set $T = 40$, $r = 5$ for CIFAR-10 (200 epochs total), and $T = 20$, $r = 3$ for MNIST (60 epochs total). We train the round robin heuristic for the same number of epochs.

For the case of $\ell_\infty$ and $\ell_2$ attacks, during training we use 10 steps of gradient ascent to optimize the inner maximization in (2). During evaluation we again run the PGD based attack on our model across all the representation spaces and use 40 steps of the PGD method to solve the inner maxi-

| | Greedy | Round Robin | Mult. Weights ($h = 1$) | Mult. Weights ($h = 3$) |
|---|---|---|---|---|
| Pixel ($\ell_\infty$) | $36.03 \pm 6.69$ | $29.99 \pm 1.49$ | $35.70 \pm 6.36$ | $\mathbf{39.13 \pm 1.74}$ |
| Pixel ($\ell_2$) | $69.03 \pm 1.58$ | $\mathbf{71.92 \pm 0.36}$ | $69.51 \pm 2.21$ | $\mathbf{71.27 \pm 0.22}$ |
| Pixel ($\ell_1$) | $45.84 \pm 2.85$ | $\mathbf{53.45 \pm 0.70}$ | $44.22 \pm 4.79$ | $46.90 \pm 1.46$ |
| DCT ($\ell_\infty$) | $44.72 \pm 8.60$ | $\mathbf{45.63 \pm 1.75}$ | $39.44 \pm 6.76$ | $42.27 \pm 2.77$ |
| DCT ($\ell_2$) | $68.99 \pm 1.58$ | $\mathbf{72.01 \pm 0.28}$ | $69.66 \pm 2.21$ | $71.24 \pm 0.12$ |
| DCT ($\ell_1$) | $37.51 \pm 6.78$ | $\mathbf{41.44 \pm 0.82}$ | $39.62 \pm 5.71$ | $\mathbf{42.96 \pm 1.12}$ |
| Min. Accuracy | $34.62 \pm 5.60$ | $29.99 \pm 1.49$ | $\mathbf{35.70 \pm 6.36}$ | $\mathbf{39.13 \pm 1.74}$ |
| Union Attack | $31.95 \pm 4.55$ | $29.70 \pm 1.42$ | $\mathbf{33.17 \pm 5.95}$ | $\mathbf{36.18 \pm 1.89}$ |
| Nat. Acc. | $78.56 \pm 1.46$ | $\mathbf{81.80 \pm 0.38}$ | $79.64 \pm 0.51$ | $\mathbf{80.66 \pm 0.89}$ |

Table 2. Comparison of the adversarial accuracies achieved on the CIFAR-10 dataset.

**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 $\hat{\theta} = \theta_T$.

Figure 6. The greedy heuristic.

mization in (1). For the case of $\ell_1$ attacks we use 20 iterations of the SLIDE method during training to compute adversarial perturbations and 100 iterations of the method during evaluation. We experiment with both running the PGD method with 20 random restarts, and a simpler attack with no restarts. The experiments we report here are for the latter case. The qualitative conclusions of our experiments remain the same when using 20 random restarts. See the supplementary material for details.

For the MNIST dataset we use perturbation magnitudes of $0.4, 1$ and $5$ for $\ell_\infty$, $\ell_2$ and $\ell_1$ norm based attacks respectively. The corresponding magnitudes for the CIFAR-10 dataset are $0.06, 0.1$ and $7.84$. We keep the perturbation magnitudes the same across both the pixel and the DCT basis. In our experiments when performing gradient ascent for $\ell$ steps to compute a perturbation, we use a step size of $2.5\frac{\epsilon}{\ell}$, where $\epsilon$ is the perturbation magnitude.

**Metrics.** For each of our trained classifiers we report the adversarial accuracy for each of the 6 individual attacks launched separately on the trained model. In addition we also report the worst adversarial accuracy among the 6 attacks on the same model. Notice that this is the metric that our proposed algorithm in Figure 3 aims to optimize. Finally, we also report the accuracy of our trained models on a *union attack*, i.e., for each example we produce all 6 adversarial perturbations and consider the attack successful if any one of them succeeds in making the prediction of the model incorrect. Finally, notice that our proposed algorithm in Figure 3 has a parameter $h$ namely the window size. We report results for $h = 1$ and $h = 3$. These correspond to either using the parameters of the last time step or using the model averaged over the last three time steps.

**Results.** We compare our algorithm to the baseline as shown in Table 2 for the CIFAR-10 dataset and in Table 4 for the MNIST dataset. In both the cases the performance of the multiplicative weights update based algorithm is significantly better than the baseline on the minimum accuracy metric and the union attack metric. This difference is significantly higher for the MNIST dataset where both the greedy and the round robin heuristics are unstable and have much higher variances. The round robin heuristic switches among different losses much more often and pays unnecessary attention to the adversaries which it already covers well. The greedy heuristic, on the other hand, switches less frequent than our proposed algorithm. But greedy fails to address the runner-up adversary which may be almost as difficult as the chosen one, causing instability. We also notice that using the average of the last three model parameters in the multiplicative weights method is slightly better than simply using the parameters of the last time step.

**Comparison of Training Times.** We next demonstrate the scalability of our proposed algorithm. Table 3 shows the training time of our method, measured in wall clock time, as compared to the baselines when optimizing over all 6 loss functions. Moreover, the first two columns represent the training times for optimizing a single loss function ($\ell_\infty$ attack) in either the pixel or the DCT basis. As can be seen our the training cost of our approach scales sublinearly with the number of representation spaces.

**On the Convexity Assumption.** Recall that the guarantees of Theorem 4.2 apply to the algorithm in Figure 2 that requires one to produce a hypothesis that is an ensemble of

| Dataset | Pixel ($\ell_\infty$) | DCT ($\ell_\infty$) | Round Robin | Greedy | Mult. Weights |
|---|---|---|---|---|---|
| MNIST | 2.25 | 2.09 | 3.36 | 2.67 | 2.90 |
| CIFAR-10 | 7.66 | 7.06 | 10.61 | 8.70 | 9.51 |

Table 3. Training time in hours (wall clock time) for the baselines and our proposed method. The first two columns represent the training time for optimizing a single loss, i.e., $\ell_\infty$ attack in the pixel and the DCT basis respectively. The next three columns represent the training time of the three methods when optimizing over all the 6 losses simultaneously. The reported numbers are averaged over 5 runs.

| | Greedy | Round Robin | Mult. Weights ($h = 1$) | Mult. Weights ($h = 3$) |
|---|---|---|---|---|
| Pixel ($\ell_\infty$) | $63.12 \pm 19.50$ | $64.40 \pm 25.12$ | $\mathbf{67.73 \pm 14}$ | $\mathbf{66.56 \pm 13.45}$ |
| Pixel ($\ell_2$) | $22.51 \pm 17.66$ | $10.23 \pm 8.40$ | $\mathbf{71.23 \pm 2.68}$ | $\mathbf{71.77 \pm 4.25}$ |
| Pixel ($\ell_1$) | $64.33 \pm 30.73$ | $43.14 \pm 23.57$ | $\mathbf{73.20 \pm 10.54}$ | $\mathbf{73.66 \pm 9.47}$ |
| DCT ($\ell_\infty$) | $61.53 \pm 21.45$ | $59.47 \pm 32.83$ | $\mathbf{60.65 \pm 10.60}$ | $\mathbf{60.32 \pm 11.34}$ |
| DCT ($\ell_2$) | $25.80 \pm 18.82$ | $12.28 \pm 12.16$ | $\mathbf{84.58 \pm 2.52}$ | $\mathbf{84.46 \pm 4.18}$ |
| DCT ($\ell_1$) | $66.00 \pm 31.83$ | $38.23 \pm 21.47$ | $\mathbf{72.39 \pm 8.43}$ | $\mathbf{73.13 \pm 7.33}$ |
| Min. Accuracy | $22.13 \pm 17.17$ | $9.76 \pm 8.69$ | $\mathbf{57.64 \pm 7.83}$ | $\mathbf{57.57 \pm 8.46}$ |
| Union Attack | $12.32 \pm 9.80$ | $3.72 \pm 4.96$ | $\mathbf{35.30 \pm 4.48}$ | $\mathbf{35.87 \pm 6.69}$ |
| Nat. Acc. | $77.16 \pm 37.64$ | $60.93 \pm 32.31$ | $\mathbf{91.00 \pm 10.50}$ | $\mathbf{91.43 \pm 9.61}$ |

Table 4. Comparison of the adversarial accuracies achieved on the MNIST dataset.

| | Mult. Weights ($h = 3$) (wt. avg.) | Mult. Weights ($h = 3$) (ensemble) |
|---|---|---|
| Pixel ($\ell_\infty$) | $39.13 \pm 1.74$ | $40.52 \pm 1.28$ |
| Pixel ($\ell_2$) | $71.27 \pm 0.22$ | $71.41 \pm 0.24$ |
| Pixel ($\ell_1$) | $46.90 \pm 1.46$ | $48.14 \pm 0.99$ |
| DCT ($\ell_\infty$) | $42.27 \pm 2.77$ | $43.95 \pm 2.05$ |
| DCT ($\ell_2$) | $71.24 \pm 0.12$ | $71.38 \pm 0.20$ |
| DCT ($\ell_1$) | $42.96 \pm 1.12$ | $44.49 \pm 0.58$ |
| Min. Accuracy | $39.13 \pm 1.74$ | $40.52 \pm 1.28$ |
| Union Attack | $36.18 \pm 1.89$ | $37.76 \pm 1.30$ |
| Nat. Acc. | $80.66 \pm 0.89$ | $80.46 \pm 0.81$ |

Table 5. Adversarial accuracies on the CIFAR-10 dataset by when using the average of the last three model parameters (convexity assumption) vs. ensembling the outputs of the last three models.

| | Mult. Weights ($h = 3$) (wt. avg.) | Mult. Weights ($h = 3$) (ensemble) |
|---|---|---|
| Pixel ($\ell_\infty$) | $66.56 \pm 13.45$ | $66.61 \pm 13.77$ |
| Pixel ($\ell_2$) | $71.77 \pm 4.25$ | $71.89 \pm 3.79$ |
| Pixel ($\ell_1$) | $73.66 \pm 9.47$ | $73.38 \pm 9.45$ |
| DCT ($\ell_\infty$) | $60.32 \pm 11.34$ | $59.87 \pm 11.32$ |
| DCT ($\ell_2$) | $84.46 \pm 4.18$ | $84.34 \pm 4.13$ |
| DCT ($\ell_1$) | $73.13 \pm 7.33$ | $73.08 \pm 7.38$ |
| Min. Acc. | $57.57 \pm 8.46$ | $57.23 \pm 8.51$ |
| Union Attack | $35.87 \pm 6.69$ | $35.54 \pm 6.69$ |
| Nat. Acc. | $91.43 \pm 9.61$ | $91.44 \pm 9.62$ |

Table 6. Adversarial accuracies on the MNIST dataset when using the average of the last three model parameters (convexity assumption) vs. ensembling the outputs of the last three models.

the intermediate trained models. If the loss functions were convex then one could replace the ensembling with simply averaging the model weights and retain the theoretical guarantees. Even though we have non-convex losses we still make the near convexity assumption and average the model weights to produce a scalable implementation. In Table 5 and Table 6 we compare the performance of our weight averaging strategy with that of the ideal one that ensembles the models. As can be seen the loss in making the near convexity assumption is negligible.

## 7. Discussion

We motivated the problem of designing neural networks that are simultaneously robust to multiple types of adversarial attacks in multiple representation spaces. We provided a theoretically sound algorithm with training cost that grows sublinearly with the number of representation spaces. We designed a scalable implementation and showed that it significantly outperforms strong baselines with training cost not much more than that of standard adversarial training. Several future directions emerge from this work. Notice that in our proposed algorithm we use the PGD based method of [18] to optimize the individual losses. There has been very recent work proposing faster training methods that achieve similar performance to that of the PGD method [23, 27]. It would be interesting to incorporate them in our framework to drive down the training cost even further. The benefits of this could be significant as the number of representation spaces grows.

Finally, we hope that future work on adversarial robustness will consider multiple representation spaces for evaluation of robustness of classifiers to adversarial perturbations.

## References


[1] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna Wallach. A reductions approach to fair classification. *arXiv preprint arXiv:1803.02453*, 2018.

[2] Naveed Akhtar, Jian Liu, and Ajmal Mian. Defense against universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3389–3398, 2018.

[3] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

[4] Pranjal Awasthi, Himanshu Jain, Ankit Singh Rawat, and Aravindan Vijayaraghavan. Adversarial robustness via robust low rank representations. *Advances in Neural Information Processing Systems*, 33, 2020.

[5] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[6] Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. Certified adversarial robustness via randomized smoothing. *arXiv preprint arXiv:1902.02918*, 2019.

[7] Corinna Cortes, Mehryar Mohri, Javier Gonzalvo, and Dmitry Storcheus. Agnostic learning with multiple objectives. *Advances in Neural Information Processing Systems*, 33, 2020.

[8] Andrew Cotter, Heinrich Jiang, and Karthik Sridharan. Two-player games for efficient non-convex constrained optimization. In *Algorithmic Learning Theory*, pages 300–332. PMLR, 2019.

[9] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M Roy. A study of the effect of jpg compression on adversarial images. *arXiv preprint arXiv:1608.00853*, 2016.

[10] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[11] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] Satyen Kale. *Efficient algorithms using the multiplicative weights update method*. Princeton University, 2007.

[14] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[16] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[17] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 656–672. IEEE, 2019.

[18] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[19] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*, 2017.

[20] Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems*, pages 10877–10887, 2018.

[21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[22] Hadi Salman, Greg Yang, Jerry Li, Pengchuan Zhang, Huan Zhang, Ilya Razenshteyn, and Sebastien Bubeck. Provably robust deep learning via adversarially trained smoothed classifiers. *arXiv preprint arXiv:1906.04584*, 2019.

[23] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *Advances in Neural Information Processing Systems*, pages 3353–3364, 2019.

[24] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[25] Florian Tramèr and Dan Boneh. Adversarial training and robustness for multiple perturbations. In *Advances in Neural Information Processing Systems*, pages 5866–5876, 2019.

[26] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. *arXiv preprint arXiv:2002.08347*, 2020.

[27] Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020.

[28] Dong Yin, Raphael Gontijo Lopes, Jon Shlens, Ekin Dogus Cubuk, and Justin Gilmer. A fourier perspective on model robustness in computer vision. *Advances in Neural Information Processing Systems*, 32:13276–13286, 2019.

[29] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P Xing, Laurent El Ghaoui, and Michael I Jordan. Theoretically principled trade-off between robustness and accuracy. *arXiv preprint arXiv:1901.08573*, 2019.

## 8. Proof of Theorem 4.2

*Proof.* Given the weights $w_1, w_2, \dots w_k$ for the $k$ losses, we denote by $\mathbf{p}$ the normalized probabilities, i.e., $p_i = w_i / \sum_{j=1}^{k} w_j$. Let $\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_T$ be the sequence of probability vectors produced by the column players namely the player that chooses among the $k$ losses. Denote by $p_{t,j}$ the $j$th coordinate of the vector $\mathbf{p}_t$. Since the player is performing multiplicative weights updates, by the standard guarantee of the multiplicative weights update (see Theorem 2 in [13]) we get that for any $i \in [k]$ the following holds

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{k} p_{t,j} L_j(\theta_t) \geq (1-\eta)\frac{1}{T}\sum_{t=1}^{T} L_i(\theta_t) - 2R\frac{\log k}{\eta T}. \tag{7}$$

Setting $\eta = O(\epsilon/R)$ and $T = O(R^2 \frac{\log k}{\epsilon^2})$ we get that

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{k} p_{t,j} L_j(\theta_t) \geq \frac{1}{T}\sum_{t=1}^{T} L_i(\theta_t) - \epsilon. \tag{8}$$

Hence, denoting by $P$ the uniform distribution over the $T$ parameters, we get that

$$\mathbb{E}_{\theta\sim P} L_i(\theta) \leq \frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{k} p_{t,j} L_j(\theta_t) + \epsilon. \tag{9}$$

Next we will write down the right hand side in terms of the optimal value of the objective as in (1). Given the guarantee that we can solve the adversarial cost sensitive optimization upto additive error $\delta$, we have that for any $t \in [T]$,

$$\sum_{j=1}^{k} p_{t,j} L_j(\theta_t) \leq \min_{\theta} \sum_{j=1}^{k} p_{t,j} L_j(\theta) + \delta. \tag{10}$$

Substituting into (9) we get

$$\mathbb{E}_{\theta\sim P} L_i(\theta) \leq \frac{1}{T}\sum_{t=1}^{T} \min_{\theta} \sum_{j=1}^{k} p_{t,j} L_j(\theta) + \epsilon + \delta \tag{11}$$

$$\leq \min_{\theta} \frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{k} p_{t,j} L_j(\theta) + \epsilon + \delta. \tag{12}$$

Next for any $j \in [k]$ define $\tilde{p}_j = \frac{1}{T}\sum_{t=1}^{T} p_{t,j}$. Then we can rewrite the above as

$$\mathbb{E}_{\theta\sim P} L_i(\theta) \leq \frac{1}{T}\sum_{t=1}^{T} \min_{\theta} \sum_{j=1}^{k} p_{t,j} L_j(\theta) + \epsilon + \delta$$

$$\leq \min_{\theta} \sum_{j=1}^{k} \tilde{p}_j L_j(\theta) + \epsilon + \delta. \tag{13}$$

It is easy to check that $\tilde{p}_j \in [0,1]$ and $\sum_{j=1}^{k} \tilde{p}_j = 1$. Hence we get that $\sum_{j=1}^{k} \tilde{p}_j L_j(\theta) \leq \min_\theta \max_j L_j(\theta)$. Substituting into (13) we get the guarantee of the theorem that for all $i \in [k]$,

$$\mathbb{E}_{\theta\sim P} L_i(\theta) \leq \min_{\theta} \max_{j} L_j(\theta) + \epsilon + \delta. \tag{14}$$

Next we prove the consequence when the loss function $L$ in (2) is convex in its first argument. This is true for commonly used loss functions in practice such as the cross entropy loss, squared loss and the hinge loss. We denote by $f_{\theta_t}$ the network corresponding to the parameter $\theta_t$ and by $f_{\hat{\theta}}$ the ensembled network i.e.,

$$f_{\hat{\theta}} = \frac{1}{T}\sum_{t=1}^{T} f_{\theta_t}.$$

By expanding out $\mathbb{E}_{\theta\sim P} L_i(\theta)$ we get

$$\mathbb{E}_{\theta\sim P} L_i(\theta) = \frac{1}{T}\sum_{t=1}^{T} L_i(f_{\theta_t}) \tag{15}$$

$$= \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{(x,y)\sim D}\big[\max_{z\in B_i(x)} L(f_{\theta_t}(\mathcal{R}^{-1}(z)), y)\big] \tag{16}$$

$$\geq \max_{z\in B_i(x)} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{(x,y)\sim D}\big[L(f_{\theta_t}(\mathcal{R}^{-1}(z)), y)\big] \tag{17}$$

$$= \max_{z\in B_i(x)} \mathbb{E}_{(x,y)\sim D}\mathbb{E}_{\theta\sim P} L(f_\theta(\mathcal{R}^{-1}(z)), y) \tag{18}$$

$$\geq \max_{z\in B_i(x)} \mathbb{E}_{(x,y)\sim D} L(\mathbb{E}_{\theta\sim P} f_\theta(\mathcal{R}^{-1}(z)), y) \tag{19}$$

$$= \max_{z\in B_i(x)} \mathbb{E}_{(x,y)\sim D} L(f_{\hat{\theta}}(\mathcal{R}^{-1}(z)), y). \tag{20}$$

Here in the last inequality we have used the fact that $L$ is convex in its first argument. Substituting back into (14) we get that $f_{\hat{\theta}}$ satisfies that for all $i \in [k]$,

$$L_i(f_{\hat{\theta}}) \leq \min_{\theta} \max_{j} L_j(\theta) + \epsilon + \delta. \tag{21}$$

□

## 9. Further Experiments

In this section we present the results of evaluating our trained multiplicative weights method based algorithm from Figure 3, as well as the greedy and the round robin heuristics, against the PGD based attack where the PGD method is run with 20 random restarts in order to find an adversarial example. Tables 7 shows the results for the CIFAR-10

| | Greedy | Round Robin | Mult. Weights ($h = 1$) | Mult. Weights ($h = 3$) |
|---|---|---|---|---|
| Pixel ($\ell_\infty$) | $34.16 \pm 6.57$ | $28.28 \pm 1.53$ | $34.01 \pm 6.38$ | $\mathbf{37.24 \pm 1.89}$ |
| Pixel ($\ell_2$) | $64.74 \pm 1.73$ | $\mathbf{67.33 \pm 0.27}$ | $64.97 \pm 3.03$ | $\mathbf{66.94 \pm 0.28}$ |
| Pixel ($\ell_1$) | $45.26 \pm 3.11$ | $\mathbf{53.08 \pm 0.72}$ | $43.46 \pm 4.99$ | $46.25 \pm 1.44$ |
| DCT ($\ell_\infty$) | $42.64 \pm 9.02$ | $\mathbf{43.62 \pm 1.74}$ | $37.17 \pm 6.92$ | $39.93 \pm 3.01$ |
| DCT ($\ell_2$) | $64.66 \pm 1.71$ | $\mathbf{67.26 \pm 0.38}$ | $65.03 \pm 2.88$ | $66.98 \pm 0.30$ |
| DCT ($\ell_1$) | $37.07 \pm 6.84$ | $\mathbf{41.01 \pm 0.81}$ | $39.16 \pm 5.86$ | $\mathbf{42.52 \pm 1.17}$ |
| Min. Accuracy | $32.50 \pm 5.31$ | $28.28 \pm 1.53$ | $\mathbf{34.01 \pm 6.38}$ | $\mathbf{37.24 \pm 1.89}$ |
| Union Attack | $30.29 \pm 4.24$ | $28.07 \pm 1.48$ | $\mathbf{31.61 \pm 6.07}$ | $\mathbf{34.63 \pm 2.02}$ |
| Nat. Acc. | $78.56 \pm 1.46$ | $\mathbf{81.80 \pm 0.38}$ | $79.64 \pm 0.51$ | $\mathbf{80.66 \pm 0.89}$ |

Table 7. Comparison of the adversarial accuracies achieved on the CIFAR-10 dataset by the greedy algorithm, the round robin algorithm and our proposed algorithm in Figure 3.

| | Greedy | Round Robin | Mult. Weights ($h = 1$) | Mult. Weights ($h = 3$) |
|---|---|---|---|---|
| Pixel ($\ell_\infty$) | $53.50 \pm 24.05$ | $59.60 \pm 24.55$ | $\mathbf{60.88 \pm 17}$ | $\mathbf{60.01 \pm 16.61}$ |
| Pixel ($\ell_2$) | $7.50 \pm 4.48$ | $6.04 \pm 4.27$ | $\mathbf{63.32 \pm 8.25}$ | $\mathbf{63.47 \pm 7.72}$ |
| Pixel ($\ell_1$) | $63.60 \pm 30.36$ | $41.98 \pm 22.87$ | $\mathbf{72.23 \pm 10.54}$ | $\mathbf{72.51 \pm 9.68}$ |
| DCT ($\ell_\infty$) | $53.80 \pm 24.53$ | $53.24 \pm 22.41$ | $\mathbf{55.29 \pm 12.16}$ | $\mathbf{54.81 \pm 13.09}$ |
| DCT ($\ell_2$) | $9.18 \pm 5.19$ | $5.90 \pm 5.71$ | $\mathbf{79.97 \pm 5.92}$ | $\mathbf{79.50 \pm 6.50}$ |
| DCT ($\ell_1$) | $65.45 \pm 31.95$ | $36.81 \pm 21.10$ | $\mathbf{71.74 \pm 8.49}$ | $\mathbf{72.51 \pm 7.33}$ |
| Min. Accuracy | $7.31 \pm 4.38$ | $5.19 \pm 4.72$ | $\mathbf{52.53 \pm 11.31}$ | $\mathbf{53.29 \pm 12.73}$ |
| Union Attack | $3.37 \pm 3.10$ | $1.97 \pm 3.03$ | $\mathbf{27.65 \pm 7.06}$ | $\mathbf{28.09 \pm 9.38}$ |
| Nat. Acc. | $77.16 \pm 37.64$ | $60.93 \pm 32.31$ | $\mathbf{91.00 \pm 10.50}$ | $\mathbf{91.43 \pm 9.61}$ |

Table 8. Comparison of the adversarial accuracies achieved on the MNIST dataset by the greedy algorithm, the round robin algorithm and our proposed algorithm in Figure 3.

dataset and Table 8 shows the results for the MNIST dataset. Furthermore, Table 9 shows the results for the Tiny ImageNet dataset. Similar to the results presented in Section 6, the multiplicative weights method significantly outperforms the baselines on both the minimum accuracy metric and accuracy against a union attack. Finally, in Table 10 we present the performance of our algorithm for against both pixel based $\ell_p$ perturbations and perturbations that correspond to small image rotations (up to $30^\circ$).

| | Greedy | Round Robin | Mult. Weights ($h = 1$) | Mult. Weights ($h = 3$) |
|---|---|---|---|---|
| Pixel ($\ell_\infty$) | $13.93 \pm 3.92$ | $12.54 \pm 0.2$ | $\mathbf{14.8 \pm 0.26}$ | $\mathbf{14.97 \pm 0.25}$ |
| Pixel ($\ell_2$) | $14.65 \pm 2.59$ | $16.36 \pm 0.34$ | $\mathbf{17.55 \pm 0.33}$ | $\mathbf{17.84 \pm 0.32}$ |
| Pixel ($\ell_1$) | $22.10 \pm 0.88$ | $\mathbf{26.66 \pm 0.30}$ | $26.15 \pm 0.36$ | $26.45 \pm 0.31$ |
| DCT ($\ell_\infty$) | $17.17 \pm 4.47$ | $16.87 \pm 0.38$ | $\mathbf{17.60 \pm 0.43}$ | $\mathbf{17.96 \pm 0.31}$ |
| DCT ($\ell_2$) | $14.65 \pm 2.59$ | $16.36 \pm 0.31$ | $\mathbf{17.53 \pm 0.35}$ | $\mathbf{17.84 \pm 0.35}$ |
| DCT ($\ell_1$) | $16.74 \pm 4.61$ | $18.71 \pm 0.23$ | $\mathbf{20.19 \pm 0.19}$ | $\mathbf{20.27 \pm 0.20}$ |
| Min. Accuracy | $13.42 \pm 3.63$ | $12.54 \pm 0.20$ | $\mathbf{14.80 \pm 0.26}$ | $\mathbf{14.97 \pm 0.25}$ |
| Union Attack | $11.90 \pm 3.05$ | $12.10 \pm 0.27$ | $\mathbf{14.04 \pm 0.24}$ | $\mathbf{14.20 \pm 0.18}$ |
| Nat. Acc. | $44.37 \pm 1.46$ | $\mathbf{46.75 \pm 0.23}$ | $45.48 \pm 0.29$ | $45.71 \pm 0.48$ |

Table 9. Comparison of the adversarial accuracies achieved on the Tiny ImageNet dataset by the greedy algorithm, the round robin algorithm and our proposed algorithm in Figure 3.

| | Greedy | Round Robin | Mult. Weights ($h = 1$) | Mult. Weights ($h = 3$) |
|---|---|---|---|---|
| Pixel ($\ell_\infty$) | $24.90 \pm 22.28$ | $23.10 \pm 16.65$ | $36.53 \pm 4.97$ | $\mathbf{38.43 \pm 4.77}$ |
| Pixel ($\ell_2$) | $26.82 \pm 23.67$ | $26.23 \pm 18.40$ | $39.38 \pm 5.03$ | $\mathbf{41.08 \pm 4.60}$ |
| Rotation | $\mathbf{61.26 \pm 20.77}$ | $27.40 \pm 3.07$ | $34.57 \pm 3.06$ | $33.85 \pm 2.00$ |
| Min. Accuracy | $24.14 \pm 21.55$ | $19.71 \pm 13.59$ | $\mathbf{33.96 \pm 3.96}$ | $\mathbf{32.94 \pm 2.53}$ |
| Union Attack | $17.79 \pm 16.03$ | $11.97 \pm 8.42$ | $\mathbf{20.80 \pm 3.39}$ | $\mathbf{21.04 \pm 2.04}$ |
| Nat. Acc. | $\mathbf{84.81 \pm 5.64}$ | $72.73 \pm 4.92$ | $78.52 \pm 1.97$ | $79.03 \pm 0.91$ |

Table 10. Comparison of the adversarial accuracies achieved on the CIFAR-10 dataset by the greedy algorithm, the round robin algorithm and our proposed algorithm in Figure 3.