INDIVIDUAL FAIRNESS FOR ADVERSARIAL ROBUSTNESS

Valerio Pepe
Harvard SEAS
Cambridge, MA 01238
valeriopepe@college.harvard.edu

ABSTRACT

Defenses against adversarial attacks in the domain of image recognition are tools of great interest in computer vision settings but are often computationally expensive or require the re-training of the network they are to defend and/or access to its training data. In this paper, we define image classification as a problem of *individual fairness* [Dwork et al., 2011], and, in doing so, propose and implement an inference-time centroid-based adversarial defense method for images perturbed by the Fast Gradient Sign Method [Goodfellow et al., 2015]. We find the method to have the same asymptotic runtime as regular neural network inference and require minimal knowledge of the training data. We report increases of 15 to 20 percentage points in the $0.1 \leq \varepsilon \leq 0.2$ setting in both a fully-connected neural network baseline and a LeNet-5 baseline, both trained on MNIST and Fashion-MNIST [Lecun et al., 1998] [Xiao et al., 2017]. Finally, we find increased robustness up to $\varepsilon \leq 0.5$, with a trade-off in accuracy on clean examples ($\varepsilon \leq 0.1$).

1 Introduction

Adversarial attacks for image classification problems were first introduced in [Szegedy et al., 2014], defined as "perturbations found by optimizing the input to maximise prediction error" in the ImageNet database [Russakovsky et al., 2015], trained specifically on AlexNet [Krizhevsky et al., 2012], a Convolutional Neural Network-based architecture [LeCun et al., 1989], by far the best neural network architecture at the time for an image classification task. These were presumed to work due to some mechanism inherent to AlexNet's method of classification by which "small additive perturbations of the input [...] produce large perturbations of the output at the last layer [of the neural network]", moving the input across decision boundaries and causing its misclassification. With the birth of the study of adversarial attacks, the idea of defenses against them also soon came to be: this method of generating adversarial attacks was then formalized in [Goodfellow et al., 2015] as the Fast Gradient Sign Method, and Goodfellow was the first to propose 'adversarial training', placing the images generated by the perturbation method back into the training loop to increase the network's robustness to them. However, these methods fail to generalize their networks' robustness past the specific type of noise used to generate the images, so they are not very useful in practice [Carlini and Wagner, 2017]. Additionally, re-training models requires access to the model's original training data and its precise architecture and hyperparameters, as well as being incredibly computationally expensive as compared to inference on the model [Strubell et al., 2019], meaning that not everyone who is able to run a model may be able to harden it to adversarial attacks due to data or resource constraints. Even more recent methods, such as defensive distillation [Papernot et al., 2016] still require at least one re-training step, and therefore automatically disqualify individuals who do not have the necessary computational capabilities to train a network.

With these considerations in mind, we propose a computationally-efficient method for adversarial robustness that involves no re-training of the model at any stage. Instead, we leverage the characteristics of adversarial attacks – perturbations in the last layer (or the last few layers) of the neural network – through the lens of a set of individual fairness constraints [Dwork et al., 2011] enforced at inference-time. We find that this method achieves modestly better performance on adversarial examples than an undefended neural network (both fully-connected and convolutional, proving it to be architecture-agnostic) and asymptotically as computationally expensive as inference itself, suggesting that adversarial defenses that are much computationally cheaper than current techniques are possible and shedding light on how fairness-theoretical concerns may inform these.

2 Related Work

Individual Fairness. As presented in [Dwork et al., 2011], individual fairness is a definition of fairness that can be informally summarized by saying that 'similar individuals must be treated similarly'. Formally, it characterizes classification through a mapping M from individuals (inputs) to outcomes (outputs). It then defines two distance metrics d (on individuals) and D (on outcomes), and considers a system 'fair', if, for all individuals $x, y \in V$ and outcomes $Mx, My \in \Delta(A)$, we have

$$D(Mx, My) \leq d(x, y).$$

If a system satisfies this (D, d)-Lipschitz condition, it implies that the distance between the outcomes Mx, My of two individuals is at most the distance between the individuals x, y themselves, so we can certify that similar individuals are being treated similarly and our system is 'fair' with regards to any pair of two individuals in V. This definition of fairness avoids certain pitfalls of group fairness approaches like statistical parity (an approach that defines fairness as having a total variation distance lower than some ε between the distributions of outcomes for two main groups of individuals, therefore considering individuals in the aggregate and not individually) where small enough subsets of a certain group can be discriminated against without rendering the system unfair. However, it is still vulnerable to criticism regarding its practical feasibility with regards to the simultaneous optimization of a loss function along with achieving fairness, as well as the presence of the trivial classifier (everyone gets the same outcome) as a 'fair' one, and specfics regarding the construction of the task-specific metrics [Rothblum and Yona, 2018] [Fleisher, 2021].

Adversarial Attacks. The adversarial attacks we will consider in this paper are the ones hinted at in the introduction. We will therefore center our analysis around the Fast Gradient Sign Method (henceforth FGSM), introduced in [Goodfellow et al., 2015]. For some $0 \leq \varepsilon \leq 1$, FGSM returns an *$\varepsilon$-constrained perturbation* η calculated as follows:

$$\eta = \varepsilon \cdot \text{sign}(\nabla_x J(\theta, x, y))$$

Where x is the input to the neural network, y is its label, θ is the set of parameters of the model (its weights), and J is a loss function. This means $J(\theta, x, y)$ is the loss with respect to the current weights θ of the network, and some specific image x with label y; we then take the gradient ∇_x with respect to the image x, take the sign of the gradient, and multiply it by our perturbation budget ε. Finally, to perturb the image, we just let our perturbed image $x_p = x + \eta$, therefore adding the perturbation to the original image pixel-wise: intuitively, this attack takes the direction of the loss function with regards to the input image, and shifts every single pixel by a value of ε in that direction. This guarantees that we are moving the image orthogonally away from the decision boundary that would classify it as coming from label y, and thereby maximising the chance that the image is misclassified.

Additionally, a brief note on the 'philosophy' of adversarial attacks. Since we normalize the image's pixel values to be between 0 and 1 to prepare them for perturbation (since ε itself is between 0 and 1), it is trivial to see that by simply setting $\varepsilon = 1$, the image that will result from an application of FGSM will be overwhelmed by the 'image' produced by the gradient, and therefore be completely unintelligible, resulting in a misclassification. We will not be considering these adversarial attacks: they adhere to Szegedy and Goodfellow's initial definition of the term, but the information about the original class was destroyed in the perturbation, so we cannot expect a neural network to be able classify it (a human would not be able to, either). The adversarial robustness literature also has no standard on what constitutes a reasonable ε value, because it may be affected by the setting in which a system is deployed, with more safety critical applications (e.g. self-driving vehicles) requiring much stronger robustness than other applications (e.g. automatic bird recognition in ornithologists' images). To this end, we will only consider perturbations $0 \leq \varepsilon < 0.5$, since after $\varepsilon = 0.5$, FGSM can 'output a uniformly gray image for any pixel value, thus fooling any classifier' [Madry et al., 2019].

Datasets. For our evaluation, we employ two standard datasets in computer vision and adversarial robustness tasks. The first is MNIST [Lecun et al., 1998], a collection of 28×28-pixel, grayscale images representing handwritten digits taken from a mixture of data from the American Census Bureau and American high school students. This is an incredibly common dataset in computer vision due to its simplicity (the images are very low resolution, so it is not computationally expensive to train on them as opposed to other, more complex datasets) and the interesting semantic distinctiveness of each of its classes; every number is clearly distinct from others (4 and 2 look quite different for example), but there are also interesting gray areas (1 and 7, 8 and 0) that inject some noise into the classes.

We also use Fashion-MNIST, a more recent database produced by Zalando Research in [Xiao et al., 2017] in response to the lack of challenge and the lack of real-world ambiguity that the regular MNIST poses as a computer vision model. Fashion-MNIST is a collection of 10 types of items of clothing (t-shirts, trousers, pullovers, dresses, coats, sandals, shirts, sneakers, bags, and ankle boots) scraped from Zalando's website and resized to the same 28×28 grayscale format as the original MNIST dataset. This is a more interesting (albeit much less famous) dataset than MNIST since it includes more axes of variation for the items (e.g. material: one can have a fuzzy or a leather coat and need to know that they are both 'coats', but we will never encounter a fuzzy '8' as opposed to a leather '8'), and the shapes of the

items themselves are much more complex and varied, requiring a better model to solve it than MNIST would. It also has the advantage of testing whether neural network models can encode the 'shape bias', a cognitive inductive bias in humans where the shape of an object is taken to determine its membership in some class of objects much more than its material or its color [Kemp et al., 2007].

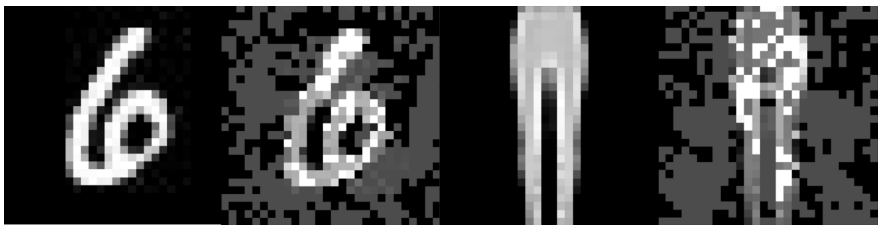


Figure 1: Example images from MNIST (left) and Fashion-MNIST (right), original and perturbed by an FGSM with $\varepsilon = 0.2$

CNNs and LeNet. Finally, aside from a regular, fully-connected neural network (of which the architecture will be explained later), we will also be using a convolutional neural network baseline model [LeCun et al., 1989]. Specifically, we will be using LeNet, a simple, 5-layer convolutional neural network developed by Yann Lecun in 1998 for the purpose of solving the classification of the MNIST dataset. We use this model because it is ubiquitous in the computer vision literature (along with AlexNet) and therefore provides a well-known baseline to test our defense against to make sure it's not the architecture of the specific fully-connected neural network that allows our defense to work.

3 Methods

3.1 Threat Modeling

Before mentioning the algorithm and theoretical insights behind the adversarial defense, we will elaborate on what scenario we are targeting in terms of resources on both the attacker and the defender's side. We assume both the attacker and the defender have white-box access to the same neural network classifier (but cannot modify it), and have enough computational resources to run inference on it and calculate gradients with respect to the loss function (which the attacker will need to run the FGSM perturbations). Additionally, we assume that neither the defender nor the attacker have access to the full training set of the network, but they both know what the input image format is, what the classes represent, and both have access to one image from each class.

These assumptions imply neither actor can retrain or otherwise tamper with the classifier, both because it's explicitly disallowed to modify it, but also because they lack the training data and the computational resources needed to retrain it, even for a single step. These are reasonable assumptions when one considers the current state of open-source, state-of-the-art machine learning models: for example, Meta AI's LLaMa family of models' weights are public, and the model's inference costs are low enough such that its smaller models (e.g. the 7b-parameter sub-family) can be run on a single GPU [Touvron et al., 2023]. However, the model is still much larger than would be feasible to re-train given the high costs of training over inference, its training data is proprietary, and the specific way in which it was trained is also unknown to the public, rendering re-training impossible.

3.2 Adversarial Defense

Having cleared the assumptions with regards to what resources both the attacker and the defender have, we can now describe the proposed defense. We will first outline how we consider image classification to be an individual fairness problem, and then elaborate on the specific algorithm we implement.

3.2.1 Individual Fairness Formalization

To formalize the problem of image classification through the lens of individual fairness, we must first define our mapping M; doing so will also define what our sets of outcomes and individuals are, and fully specify the remainder of the formalization.

Therefore, we define M as a 'generative process', a mapping that takes *image classes* and maps them onto *images*: an agent creating an image does not think of an image first and then arbitrarily assigns it a class second, but does the opposite, thinking of the class they want to instantiate first, and then producing an image based on that. This can also be seen from the point of view of an image classification task: in image classification, our central object of interest is the

class, not the image, and we can just consider images some noisy instantiations of the classes we want to characterize, as per the mapping above.

Our individuals for this formalization will thus be the classes c_i that the neural network can sort images into, as well as a class p for all of the input classes. We will denote an outcome from a class i, Mc_i, a 'centroid for class i', and outcomes specifically for class p, Mp, 'input images'.

For it to be fair for some input image Mp to be classified as some class c_i, the following $(D, d)-$Lipschitz condition must therefore be met:

$$D(Mc_i, Mp) \leq d(c_i, p)$$

Given that Mc_i and Mp are images, $D(x, y)$ should be some metric that computes the similarity between two vectors. The task-specific metric for classes, $d(x, y)$, is more difficult to give an exact formulation of, but it should intuitively capture a level of noise, with $d(c_i, p) = \alpha_i$ indicating the upper-bound for how distant a centroid can be from an input image in the same class (this captures the idea that the input class p is a noisy version of one of the c_i's, so the distance between the two should be the level of that noise). This should be a value scaled in the same way as $D(x, y)$, and should reflect the structure of the classes at hand.

For example, in MNIST digit recognition, we should have $d(c_1, p) \geq d(c_8, p)$, i.e. the distance between the class of all input images and the class of ones should be higher than the distance between the distance between the class of eights and the class of all input images. This is because there is more than one way to draw a '1' but only one way to draw an '8', so we should allow for more noise between the centroid of ones and an unknown input than the centroid of eights and an unknown input. Similarly, in Fashion-MNIST, we should have $d(c_{\text{Coats}}, p) \geq d(c_{\text{T-shirts}}, p)$, since coats are more different between them than T-shirts are (they can have different shapes or materials, while T-shirts are similarly-shaped by definition and are all made of cotton).

Alternatively, if intrinsic differences within classes are hard to characterize or seem opaque for image classification purposes, a cognitively-informed approach could be used, similar to the one proposed in [Xu and Tenenbaum, 2007], which formalizes 'distances' between concepts in the human mind based on visual similarity.

Putting all of these together, we can therefore say we classify an input image Mp as a particular class c_i if the Lipschitz condition $D(Mc_i, Mp) \leq d(c_i, p)$ is satisfied, and we have reformulated image classification as an individual fairness problem, under a similar mantra of 'Similar inputs (with regard to a class's centroid) are treated similarly'.

3.2.2 Algorithm

We will first describe the algorithm informally. Informally, as an input image passes through the neural network, at every layer we take the cosine distance of its activation vectors against the activation vectors of each class' centroid, and note the classes for which the fairness constraint is satisfied through this distance metric. Once the image has passed through every layer, we classify the image as the class that has satisfied the fairness constraint the most times. A formal description follows.

Let our input image be denoted by $x \in \mathcal{X}$, our label for the image be some $y \in \mathcal{Y}$ and the output of our neural network of some n layers be denoted by $NN_n(x)$, where for all $1 \leq j \leq n$, $NN_j(x)$ indicates the output of the input image x at the jth layer (so $NN_1(x)$ is the intermediate activation of the network after one layer, and $NN_n(x)$ is the final set of activation vectors).

For every class j, define some centroid $k_j \in \mathcal{X}$, an image of which both the attacker and the defender know the true label. Also define some acceptable noise threshold α_j denoting the maximum allowable difference between two images in the same class, as explained in the previous section. This will define our task-specific metric for individuals, and we'll have $d(c_j, p) = \alpha_j$ for all classes j.

Now, to define our task-specific metric for outcomes D, recall the notion of the cosine similarity between two vectors. For two vectors $u, v \in \mathbb{R}^d$, where $d > 0$ is some integer, the *cosine similarity* of u and v is the value

$$C(u, v) = \frac{u \cdot v}{||u|| \cdot ||v||}.$$

This is the value of the cosine of the angle, derived from the formula of the geometric intepretation of the dot product, where, in the following equation, we solve for $\cos\theta$:

$$u \cdot v = ||u|| \cdot ||v|| \cos\theta$$

Intuitively, note that this captures the difference in the directions u and v are pointing in: a cosine value of 1 means the two are pointing in the same direction, and a cosine value of -1 means they are pointing in exactly opposite directions. Therefore, we can use the cosine similarity as a metric of how 'close' two vectors are [1].

From cosine similarity, we can define a notion of distance, *cosine distance*, as $C_D(u, v) = 1 - C(u, v)$: two vectors with a cosine similarity of -1 will have a cosine distance of 2, and a cosine similarity of 1 will result in a cosine distance of 0. For simplicity, in this paper we will be defining our '(rescaled) cosine distance' as the following equation, which is just a cosine distance rescaled between 0 and 1 by dividing the result by 2. This will act as our distance metric between outcomes for the individual fairness of the problem.

$$D(u, v) = \frac{1 - C(u, v)}{2}$$

Having defined all relevant terms and distance metrics, we can summarize the algorithm as follows:

Algorithm 1: Implementation of the Adversarial Defense

Input : Input Image x, Classes $c_1, \ldots, c_l$, Centroids $k_1, \ldots, k_l$, n-layer Neural Network NN
Output : Class c_j

$T = \{c_i = 0$ **for** i in $\{1, \ldots, l\}\}$
`#Makes a dictionary containing all the classes and a tally of how many times they satisfy the Lipschitz condition`
for i *in* $\{1, \ldots, n\}$ **do**
  $a_i = NN_i(x)$
  for j *in* $\{1, \ldots, l\}$ **do**
    $b_i = NN_i(k_j)$
    if $D(a_i, b_i) < d(c_j, x)$ **then**
      $T[c_j] = T[c_j] + 1$
      `#If a class satisfies the Lipschitz condition, augment its tally by 1`
return $\underset{c_j}{\text{argmax}}\ T[c_j]$ `#Returns the class` c_j `that satisfied the condition most often`

Intuitively, this should work as a defense against FGSM: as mentioned above, the attack only perturbs the image in terms of the last layer's gradient (and therefore decision boundary). In contrast, this algorithm uses information present at every layer, therefore being able to leverage features about the true class that were not perturbed by FGSM in order to recover it. Given specifically our notion of distance as cosine distance, since cosine distance measures the difference in the directions of its vectors, the defense can be thought of as preventing images that drift too far out from some class from being classified as that class, regardless of what the neural network believes to be the most accurate prediction.

3.3 Neural Network Architectures

To test the performance of the algorithm described above, we provide two baselines, a fully-connected neural network and a convolutional neural network.

The fully-connected neural network is composed of 7 fully-connected linear layers. The first layer compresses the 28×28 input into 64 neurons. Five 64-neuron linear layers follow, each with a ReLU [Fukushima, 1969] as its non-linear activation function. The final layer takes the 64-dimensional input down to the 10 classes (either digits or articles of clothing), and is activated by a log-softmax nonlinearity to output a final label.

The convolutional neural network is an instantiation of the LeNet architecture, an architecture proposed for digit recognition in MNIST in [Lecun et al., 1998]. It is composed of 5 layers, two of which are convolutional, and three of which are linear layers. Layers 1 and 2 are the fully-connected ones: these both generate 5×5 kernels, with Layer 1 training 6 and Layer 2 training 16. There is a 2×2 maxpool non-linearity between the two convolutional layers. These 16 channels then get flattened, and passed through three linear layers, of sizes 120, 84, and finally 10, all with ReLU activation functions. The final activation function is, once again, a log-softmax.

[1] Cosine similarity ignores any differences in magnitude: this is ideal for our purposes, since our 'magnitudes' are not necessarily relevant to image classification as they are mainly affected by pixel brightness values, and, for example, we would like a 'line detector' circuit within the neural network to function well with regards to all lines, not just bright or dark ones.

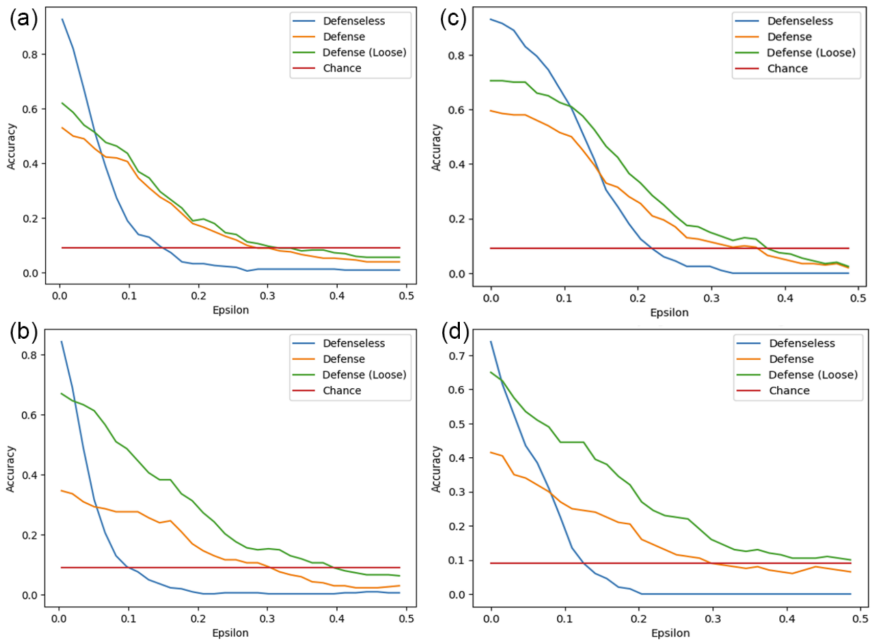
The hyperparameters and training used to train both of these models were the same: they were trained on 55000 random images from either MNIST or Fashion-MNIST, through Stochastic Gradient Descent [2] with a learning rate of $\ell = 10^{-2}$ and a batch size of 32. Training was stopped after either 95% accuracy was reached on a validation set of 5000 images or 20 epochs, whichever came first.

These hyperparameters and architectures quite standard, with LeNet being a staple of computer vision research, so there is no further comment to be made.

4 Results

4.1 Baselines

The results for the experiments listed above, adversarial defense performance on both the fully-connected and convolutional neural networks, across both MNIST and Fashion-MNIST, are shown below in Figure 2. The neural network's performance before the defense is applied is shown as the 'defenseless' curve, while the adversarial defense's performance is represented by the 'defense' curve. A 'loose defense' (i.e. if the true class was in the top 3 most fair classes [3]) baseline is also shown, as are chance levels of performance. Tabular data at 0.1-epsilon intervals, as well as the specific values of α_j used for each class in each dataset are given in Appendix A.



7

Figure 2: Main Experiment Results. Panel (a): Fully-Connected Neural Network Performance on MNIST, (b): Fully-Connected Neural Network Performance on Fashion-MNIST, (c): CNN Performance on MNIST, (d): CNN Performance on Fashion-MNIST

We also propose three natural tweaks to the experiments above. Firstly, changing the α_j values, first to be all equal (to see if our heuristic that some classes should allow for more noise than others holds on these datasets), and also test whether generally, allowing more or less noise yields better results. Secondly, we analyse the effects of more sets of centroids being used in the defense, from only one set to 2 and 5. Thirdly, we bias the tallying procedure towards earlier or later layers: instead of just flatly adding 1 to a class' counter every time it is considered fair, we add some $e^{-\ell}$ or e^{ℓ} (where ℓ is the number of the layer in the neural network) to bias the counting procedure towards either the earlier or the later layers. Figures describing the results of these experiments can be found in Appendix B.

[2]There is no commonly agreed upon citation for SGD, but [Robbins, 1951] is probably the closest thing to the first paper to describe an SGD-like method for optimization.

[3]Note that there are 10 classes in both datasets, so the true class is guaranteed to appear in the top 10 most fair classes, which would be an uninteresting result. Therefore, we limit ourselves to the top 3, which we find to be a good balance between highlighting the network's actual performance and the increases given naturally by loosening the defense.

4.2 Theoretical Results

Proposition 1. For a neural network with n layers and d neurons, assuming all arithmetic operations are $O(1)$, both regular inference and inference augmented by our adversarial defense have runtime $O(nd^2)$.

Proof.

We will assume a neural network is composed of some n layers, each of which has d neurons. Additionally, we will assume scalar addition and multiplication are $O(1)$, for two reasons: firstly, because that is the assumption that underlies most matrix multiplication algorithms, so since this algorithm makes heavy use of both scalar and matrix multiplications, we need to keep our assumptions constant, and if they are $O(1)$ in the matrix multiplication algorithm, they must be $O(1)$ in scalar multiplication, and secondly because, in practice, all of this arithmetic will be 32- or 64-bit floating point arithmetic, which is $O(1)$ given the input size is constant at 32/64 bits.

Recall the formalism of a forward-pass in machine learning: at some layer, a forward pass calculates some $a = f(z)$, where f is a function known as an 'activation function', applied element-wise to z, and $z \in \mathbb{R}^{1\times d}$ is some 'activation vector'. For some fixed weights $\theta \in \mathbb{R}^{d\times d}$, and an input $x \in \mathbb{R}^{1\times d}$, $z = \theta \times x$, where $\times$ indicates a matrix multiplication. Thus, at every forward pass, we are computing $a = f(\theta \times x)$, performing one $d \times d$ matrix multiplication and d element-wise applications of f.

We will not be assuming any specific matrix multiplication algorithm; however, $d \times d$ matrix multiplication must take some time at least $\Theta(d^2)$, even in theory, since it takes at least $\Theta(d^2)$ time to read all the entries in the matrices being multiplied, so we will be using this bound. We will assume our activation function will take some time $O(1)$, because it is also performing fixed-bit floating point arithmetic; however, this does not really matter, because our defense doesn't call the activation function any more times than regular inference does, so this will not contribute to the difference of one over the other.

Therefore, one forward-pass at a single layer must take at least time $O(d^2 + d(1)) = O(d^2)$. Across all n layers, inference thus takes $O(nd^2)$ time.

Our augmented defense takes the cosine similarity at every layer: a dot product of two d-dimensional vectors, the magnitudes of two d-dimensional vectors, and divides one by the other. A dot product of two d-dimensional vectors takes d multiplications and $d-1$ additions, which take time $O(d(1) + (d-1)(1)) = O(d)$. Taking the magnitude of a d-dimensional vector can be done by taking the dot product of it with itself and taking the square root of the result. Through the Newton-Raphson method we can reduce taking a square root to multiplication, so this entire procedure takes d multiplications and $d-1$ additions, plus an additional multiplication for the square root, for time $O((d+1)(1)+d(1)) = O(d)$. We do this twice and then multiply the two results, for a total time of $O(2d+1) = O(d)$, once more. Finally, we assumed arithmetic operations take time $O(1)$, so dividing the two is also constant-time, and the total time to calculate the cosine distance, up to constant factors, is $O(d + d) = O(d)$.

Therefore, repeating this n times we obtain an increase in runtime of an additive factor of $O(nd)$ as compared to regular inference. However, note that $O(nd^2 + nd) = O(n(d^2 + d)) = O(nd^2)$, so the time taken for regular inference still dominates the runtime of this algorithm, and, asymptotically, we are not increasing the runtime by running our defense. QED.

5 Discussion

As can be seen in Figure 2, the defense offers significant increases in accuracy under attacks by FGSM for both the fully-connected and the convolutional neural network baselines, across both MNIST and Fashion-MNIST. The strongest effect can be seen on Fashion-MNIST, where for both baselines, around $0.1 \leq \varepsilon \leq 0.2$ we get accuracy increases of about 0.2; the defenseless models also fall to 0 accuracy at $\varepsilon \approx 0.2$, while the defended models never do, and only cross below chance accuracy at $\varepsilon \approx 0.3$. The loose defense is even better (expectedly), and, at the same point, is more accurate by another 15 percentage points, resulting in ≈ 0.4 accuracy across the same range.

Even though the defense is stronger on Fashion-MNIST, regular MNIST is also defended well, with similar 0.2 increases in accuracy at $\varepsilon \approx 0.1$ for the fully-connected neural network and 0.1 increases in accuracy at $\varepsilon \approx 0.2$ for the convolutional neural network baseline. These increases are not ubiquitous across all ε, though, as all defended models tended to do significantly worse than their undefended counterparts in low-epsilon ($\varepsilon < 0.1$) environments. This is likely a form of trade-off between overall robustness and accuracy on clean inputs which is standard in adversarial defense environments, and is therefore not worrying. Future work may explore if it is possible to eliminate it, or

decrease it significantly: in our testing, all the aforementioned extensions tried worsened the gap, though, so we do not believe this is an issue that can be easily remedied.

That being said, the fact that the defense works better on the harder dataset of the two is encouraging, since it means that it is more likely to succeed on noisier, real-world datasets than it would have been if it had only succeeded on MNIST. Coupled with the previous remarks on the tradeoff, it is likely that this defense could see applications in very noisy environments such as TinyML, where cameras and other sensors are limited in their resolution by their size and the amount of power they are allowed to draw (especially since, as proven by Proposition 1, the runtime cost of this intervention is minimal).

Similarly, it is positive that there was comparable performance across the two different types of architectures, especially CNNs. Even with the recent more widespread use of vision transformers, CNNs remain the dominant architecture for computer vision tasks and research, so it is imperative that proposed adversarial defenses work on them. Interestingly, however, in CNNs we also see an increased range in which the defense is worse than the defenseless networks (especially on the MNIST dataset): this is likely due to CNNs generalizing better in the first place due to their inbuilt location invariance, so they are intrinsically more adversarially robust than their fully-connected counterparts.

In terms of the results of the main experiment's extensions, in Figure 3 (figures $3, 4, 5$ can be found in Appendix B) we find that modifying the α_j's has a drastic effect on the accuracy if they are loosened by 0.1, dropping it dramatically even at lower ε (likely because we allow more noise than there actually is in the dataset, leading to the misclassification of many more elements than would be misclassified previously). In contrast, increasing them by 0.1 slightly increases performance in lower-epsilon environments, but makes the defended network drop below chance levels of accuracy sooner (at $\varepsilon \approx 0.25$ instead of $\varepsilon \approx 0.3$). Setting all the α_j's to the same value seems to have a similar effect. These are both likely due to the fact that stricter noise constraints work well in low-noise (and thus low-ε) environments, but then work much worse when more noise is introduced. Future work may explore tuning the α_j's to both the dataset and the expected ε that could increase the accuracy of the models and finetune them for a specific task.

As can be seen in Figure 4, increasing the centroids increases the tradeoff between accuracy in low-epsilon environments and robustness. This leads the models to have approximately chance levels of accuracy at all $\varepsilon \leq 0.5$, but also leads it to remain stable, therefore beating both the undefended network and a network defended by a single set of centroids by significant margins in high-epsilon environments. Intuitively, this makes sense by considering this trade-off: the more diverse examples of a Future work here may concentrate on better centroid selection (or eliminate the requirement that the defender has access to the centroids, and have them generate and optimize for synthetic centroids).

Finally, layer bias. Figure 5 shows that changing the layer bias either way (to favor earlier or later layers) has minimal effects. This is an interesting result in and of itself, because it suggests that unperturbed information which can be used to recover the true class is present relatively uniformly across the network, since favoring neither favoring information at the beginning or the end of the network increases its accuracy. Further work can use this insight to see if we can reduce the (non-asymptotic) time complexity further by excluding some fraction of the layers from the defense and seeing if the result still holds: this goes against the concept of this defense if done en masse, but it would not be unreasonable to believe that cutting some layers out (the most high-dimensional ones, for instance) may lead to reductions in the defense's empirical runtime while also maintaining the gains in accuracy.

In conclusion, in reformulating image classification as an individual fairness problem, we have presented an inference-time adversarial defense for classification, which brings modest gains in accuracy across a range of ε values for the same asymptotic runtime as regular inference. We have also shown this defense to work across two datasets of differing difficulty and across two different neural network architectures, suggesting that the defense may be applicable to a broad range of low-resource computer vision applications and research. Weaknesses that further work may tend to are a more systematic formulation of the metric for fairness between individuals d that allows for less heuristic picking of the α_j's as well as trying to decrease the tradeoff between accuracy on relatively clean examples and robustness in high-epsilon environments. The work may also be improved by considering some of the extensions we have described (changes in the value of the metric, different methods of centroid picking, and different layer biases) and seeing how they may empirically lead to better runtime and/or accuracy, as well as trying to implement the defense on vision transformers to see how well it interfaces itself with self-attention mechanisms.

Acknowledgements & Data Availability

I would like to thank Prof. Dwork for the helpful conversations about theoretical grounding for the project and for the encouragement to pursue this over a simpler, more straightforward project. Additionally, I would like to thank all the TFs for their help throughout the semester and with sporadic questions about the project. Thank you all for a fantastic class! All of the code and centroids used can be found in the following GitHub repo: vpepe/CS226r Project.

Appendix A

5.1 Distance Metric Parameters

For the results above, the following distance metrics were used for MNIST and Fashion-MNIST:

MNIST	Fashion-MNIST
$d(c_0, x) = 0.3$	$d(c_{T-shirt}, x) = 0.2$
$d(c_1, x) = 0.4$	$d(c_{Trouser}, x) = 0.2$
$d(c_2, x) = 0.35$	$d(c_{Pullover}, x) = 0.3$
$d(c_3, x) = 0.3$	$d(c_{Dress}, x) = 0.2$
$d(c_4, x) = 0.35$	$d(c_{Coat}, x) = 0.35$
$d(c_5, x) = 0.3$	$d(c_{Sandal}, x) = 0.3$
$d(c_6, x) = 0.3$	$d(c_{Shirt}, x) = 0.3$
$d(c_7, x) = 0.4$	$d(c_{Sneaker}, x) = 0.25$
$d(c_8, x) = 0.3$	$d(c_{Bag}, x) = 0.2$
$d(c_9, x) = 0.3$	$d(c_{AnkleBoot}, x) = 0.2$

5.2 Tabular Data

ε	Defenseless Accuracy	Defended Accuracy	Defended (Loose) Accuracy
0.0	0.95	0.53	0.61
0.1	0.18	0.42	0.45
0.2	0.05	0.18	0.20
0.3	0.02	0.11	0.11
0.4	0.01	0.07	0.08
0.5	0.00	0.06	0.07

Table 1: Tabular Results for the FC NN on MNIST

ε	Defenseless Accuracy	Defended Accuracy	Defended (Loose) Accuracy
0.0	0.84	0.36	0.66
0.1	0.11	0.28	0.47
0.2	0.00	0.18	0.28
0.3	0.00	0.11	0.18
0.4	0.01	0.03	0.11
0.5	0.01	0.03	0.07

Table 2: Tabular Results for the FC NN on Fashion-MNIST

ε	Defenseless Accuracy	Defended Accuracy	Defended (Loose) Accuracy
0.0	0.95	0.60	0.71
0.1	0.62	0.51	0.62
0.2	0.14	0.23	0.34
0.3	0.04	0.12	0.17
0.4	0.00	0.08	0.09
0.5	0.00	0.05	0.05

Table 3: Tabular Results for LeNet-5 on MNIST

ε	Defenseless Accuracy	Defended Accuracy	Defended (Loose) Accuracy
0.0	0.74	0.41	0.65
0.1	0.17	0.28	0.44
0.2	0.00	0.17	0.26
0.3	0.00	0.10	0.16
0.4	0.00	0.08	0.12
0.5	0.00	0.08	0.10

Table 4: Tabular Results for LeNet-5 on Fashion-MNIST

Appendix B

5.2.1 Different Metric Values

In this section, we tested the performance of the defense across three different new scenarios: setting all the α_j's to the same value, 0.3, and shifting the values seen above up by 0.1 (resulting in 0.2, 0.25 and 0.3 being used) or down by 0.1 (resulting in 0.4, 0.45 and 0.5 being used).

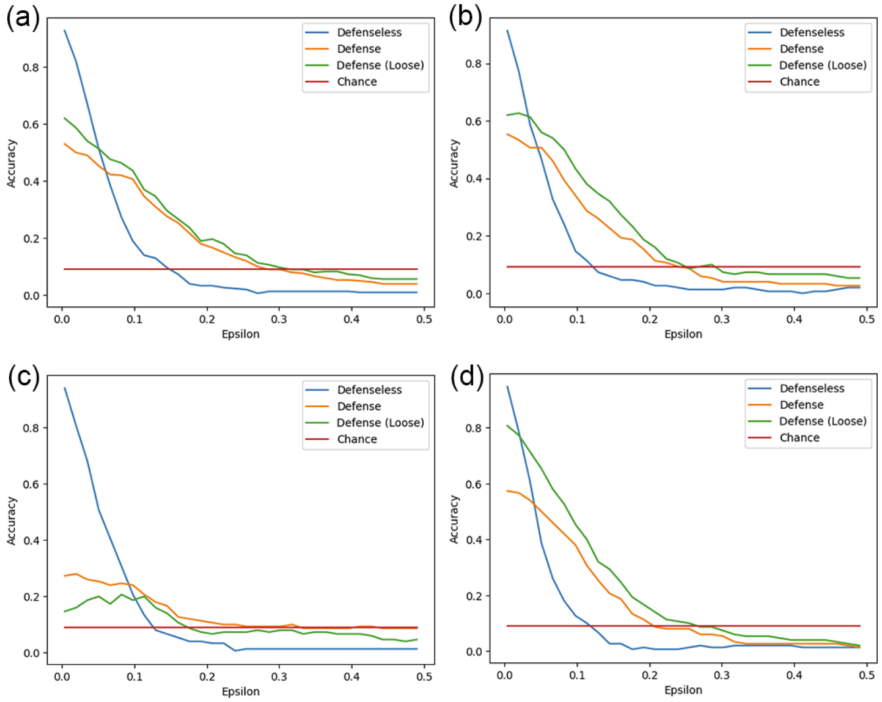


Figure 3: Adversarial defense accuracy with different α_j. Panel (a): Baseline (see Figure 1). Panel (b): All α_j set to 0.3. Panel (c): All α_j in the baseline increased by 0.1. Panel (d): All α_j in the baseline decreased by 0.1.

5.2.2 More Centroids

In this experiment, we relaxed the condition in Section 3.1 that both the attacker and the defender only have access to one set of centroids, and extended it to 2 or 5 sets of centroids to see how using more than one set at the same time would affect the defense.

5.2.3 Layer Bias

In this section, we biased the fairness constraints towards different parts of the neural network. To do this, instead of adding 1 to a class' counter every time that class was considered fair, for some layer number ℓ we added e^{ℓ} when biasing towards the later layers in the network, or $e^{-\ell}$ when biasing towards earlier layers. These are exponential increase or decay functions, making each layer's contribution exponentially more or less important than the last.

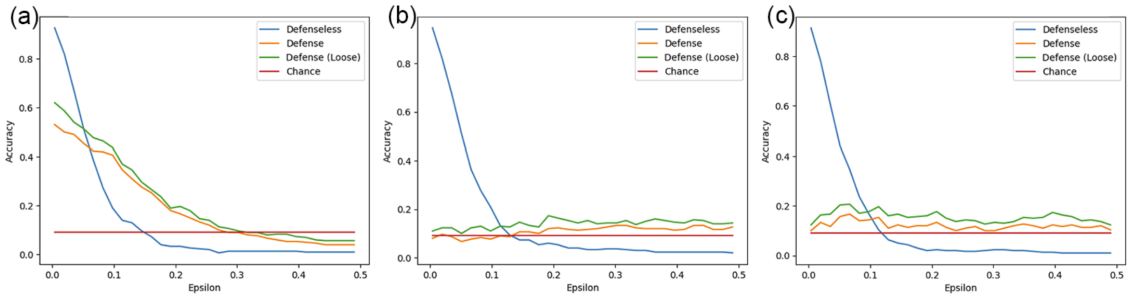


Figure 4: Adversarial Defense Performance with regards to different numbers of sets of centroids. Panel (a): Baseline (1 set of centroids). Panel (b): 2 sets of centroids. Panel (c): 5 sets of centroids.

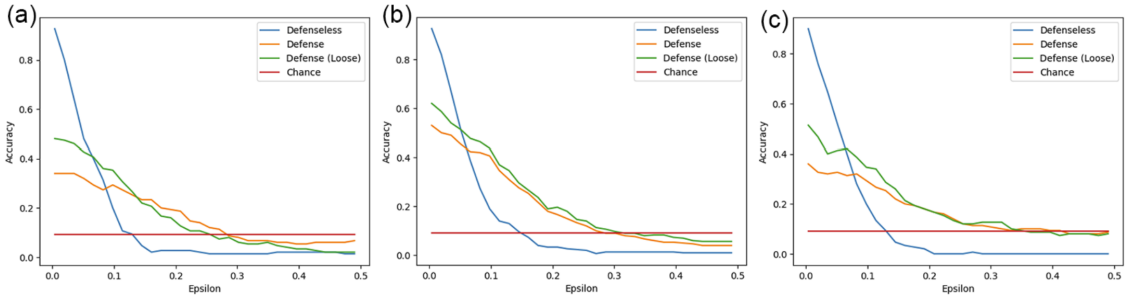


Figure 5: Adversarial defense performance with differing layer biases. Panel (a): Early layer bias ($e^{-\ell}$). Panel (b): Baseline (no bias) Panel (c): Late layer bias (e^{ℓ}).

References

[Carlini and Wagner, 2017] Carlini, N. and Wagner, D. (2017). Towards evaluating the robustness of neural networks.

[Dwork et al., 2011] Dwork, C., Hardt, M., Pitassi, T., Reingold, O., and Zemel, R. (2011). Fairness through awareness.

[Fleisher, 2021] Fleisher, W. (2021). What's fair about individual fairness? In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '21. ACM.

[Fukushima, 1969] Fukushima, K. (1969). Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Transactions on Systems Science and Cybernetics*, 5(4):322–333.

[Goodfellow et al., 2015] Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples.

[Kemp et al., 2007] Kemp, C., Perfors, A., and Tenenbaum, J. B. (2007). Learning overhypotheses with hierarchical bayesian models. *Developmental Science*, 10(3):307–321.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

[LeCun et al., 1989] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551.

[Lecun et al., 1998] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[Madry et al., 2019] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2019). Towards deep learning models resistant to adversarial attacks.

[Papernot et al., 2016] Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. (2016). Distillation as a defense to adversarial perturbations against deep neural networks.

[Robbins, 1951] Robbins, H. E. (1951). A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407.

[Rothblum and Yona, 2018] Rothblum, G. N. and Yona, G. (2018). Probably approximately metric-fair learning.

[Russakovsky et al., 2015] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge.

[Strubell et al., 2019] Strubell, E., Ganesh, A., and McCallum, A. (2019). Energy and policy considerations for deep learning in nlp.

[Szegedy et al., 2014] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing properties of neural networks.

[Touvron et al., 2023] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. (2023). Llama: Open and efficient foundation language models.

[Xiao et al., 2017] Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

[Xu and Tenenbaum, 2007] Xu, F. and Tenenbaum, J. B. (2007). Word learning as bayesian inference. *Psychological Review*, 114(2):245–272.