Interpretable Gradients with Robust Training

Liam Collins, Gene Li, Walter Li

April 2019

1 Introduction

While deep learning models have achieved impressive performance on a wide range of machine learning tasks, they can easily be fooled by imperceptible perturbations on the input dataset. For example, one can slightly change the pixels of a picture representing a dog to cause an Imagenet to misclassify it as a "baboon". The term *adversarial examples* has been coined to describe such attacks at test time.

A large body of work both seeks to develop fast, effective attack methods, as well as develop methods to either detect adversarial examples or build robustness into models [1, 2, 3]. In this report, we focus on the recent work on adversarially robust machine learning by Tsipras et al. [4] and Schmidt et al. [5]. Specifically, in [4], they note a curious and unexpected phenomenon. When a CNN is trained in a standard fashion with the original image examples, the loss gradients of the image are noisy and do not form coherent patterns. However, when the same CNN architecture is trained with adversarial examples, the image gradients are extremely sparse and align with human perception. Essentially, they point out that the features generated by models trained for robustness are fundamentally different from those generated by models trained for standard accuracy. This report makes steps towards answering the question:

Why does adversarial training induce models with interpretable gradients?

Roadmap. We explore this question through both experimental and theoretical avenues.

In Section 2, we provide background on the adversarial robustness setup for classification, as well as the projected gradient descent (PGD) algorithm which is used to generate adversarial examples and train robust models. We provide experimental evidence of the interpretable gradients phenomenon for the MNIST dataset on robust CNNs trained with PGD, verifying the work in [4].

In Section 3, we extend theoretical results on *linear classification* in the Bernoulli model introduced in [5]. As Schmidt et al point out, the Bernoulli model is relevant to the analysis of MNIST because handwritten digit pixels are approximately binary. Our extension captures the varying correlation of a pixel with the class label (i.e. center pixels are more indicative of the label than the all black pixels on the border of the images). For linear classification tasks over the modified Bernoulli model, we show that standard training induces gradients that are directly proportional to the correlation of the feature with the output. We give some intuition for why robust training seems to zero out gradients for features which are weakly correlated.

Although the interpretable gradients phenomenon may hold for CNNs and linear models, we provide experimental evidence on MNIST that the phenomenon and the success of robust training are heavily dependent on architecture in Section 4. Compared to the CNN architecture, the gradients of fully connected NNs are significantly less interpretable and not sparse, In addition, robust training is generally more difficult for fully connected NNs, requiring significant overparameterization.

2 Adversarial Training

In this section, we provide background on the study of adversarial examples and robust training. We also verify the interpretable gradients phenomenon on the MNIST dataset.

Adversarial robustness setup. In the standard classification setting with no adversarial perturbations, we want to train models to have low expected loss $\mathcal{L}$:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}}[\mathcal{L}(x, y; \theta)] \tag{1}$$

whereas to make models robust we want to train models with low expected adversarial loss:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}}[\max_{\delta\in\Delta} \mathcal{L}(x+\delta, y; \theta)] \tag{2}$$

where Δ is the set of perturbations the adversary can carry out. Many results in literature consider the ℓ_∞ bounded perturbation, i.e. $\Delta = \{\delta : \|\delta\|_\infty \le \epsilon\}$ for some fixed $\epsilon > 0$.

Training via PGD attacks. Madry et al. [3] have advocated for using projected gradient descent (PGD) attacks to train robust models. Briefly, the algorithm can be described as follows (a full description can be found in Algorithm 1). We freeze the model parameters θ. Given an original datapoint $x^{(i)}$, we iteratively run PGD updates:

$$x_{t+1}^{(i)} = \mathcal{P}_A\left(x_t^{(i)} + \alpha\nabla_x\mathcal{L}(x, y, \theta)\right), \tag{3}$$

where $A := \{w : \|w - x^{(i)}\|_\infty \le \epsilon\}$. Typically, we run this for k rounds on datapoints which have been correctly classified.

Algorithm 1 Robust Training via PGD

```
Input: Data batch $X \in \mathbb{R}^{n \times d}$ of $n$ examples in $d$ dimensions, correct (target) labels $y \in \mathbb{R}^n$,
    $\ell_\infty$-error margin $\epsilon$, number of PGD rounds $k$, PGD step-size $\alpha$, learning model $\mathcal{F}_\theta$
Output: Adversarial data batch $\tilde{X}$
  1. Compute indices of correctly labeled examples
  Compute initial label estimates: $\hat{y} = \mathcal{F}_\theta(X)$
  Set $\mathcal{S}$ be the set of indices with correct label estimates: $\mathcal{S} := \{i | y_i = \hat{y}_i\}$
  2. Randomly perturb each correctly-labeled element
  Let $\tilde{X} = X$
  for index $i \in \mathcal{S}$ do
      $\tilde{X}_i = \tilde{X}_i + \epsilon q$, where $q \in \mathbb{R}^d$ and each $q_j \overset{\text{iid}}{\sim}$ Unif[-1,1]
  end for
  3. Use PGD to find adversarial examples
  Let $L(\mathcal{F}_\theta(\tilde{X}), y)$ be the loss of the model on the data $\tilde{X}$ with correct labels $y$, and $\tilde{X}_\mathcal{S}$ be
  the examples indexed by $\mathcal{S}$
  for $t = 1, 2, ..., k$ do
      Projected gradient update:
          $$\tilde{X}_\mathcal{S} = \mathcal{P}_A\left(\tilde{X}_\mathcal{S} + \alpha \cdot \text{sign}\left[\nabla_{X_\mathcal{S}} L(\mathcal{F}_\theta(\tilde{X}), y)\right]\right) \quad (4)$$
      where $A := \{w : \|w - X_\mathcal{S}\|_\infty \leq \epsilon\}$
  end for
  return $\tilde{X}$
```

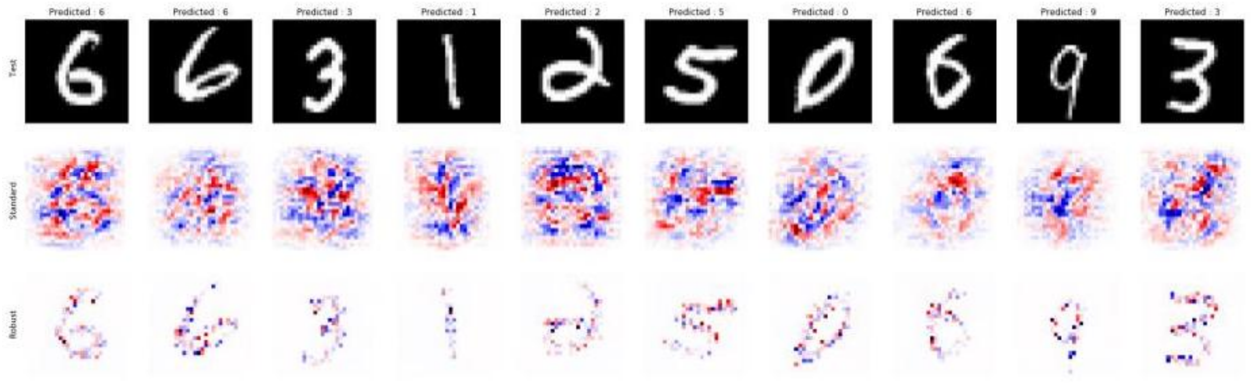
Adversarial training induces interpretable gradients. Recent results by Tsiparas et al. suggest that there may be an inherent trade-off between standard accuracy and robust accuracy of a model. Models which optimize for standard accuracy may be learning many features which are weakly correlated, while models optimized for robust accuracy only assign weight to features which are strongly correlated with the output.

Importantly, robust models seem to align better with human perception than standard models: loss gradients at the input level for robust networks heavily weight a small, interpretable set of features.[1] Tsiparas et al. suggest that humans pick out similar robust features as robust models do, while standard models may outperform humans because they pick out "brittle" features that are weakly correlated with the output.

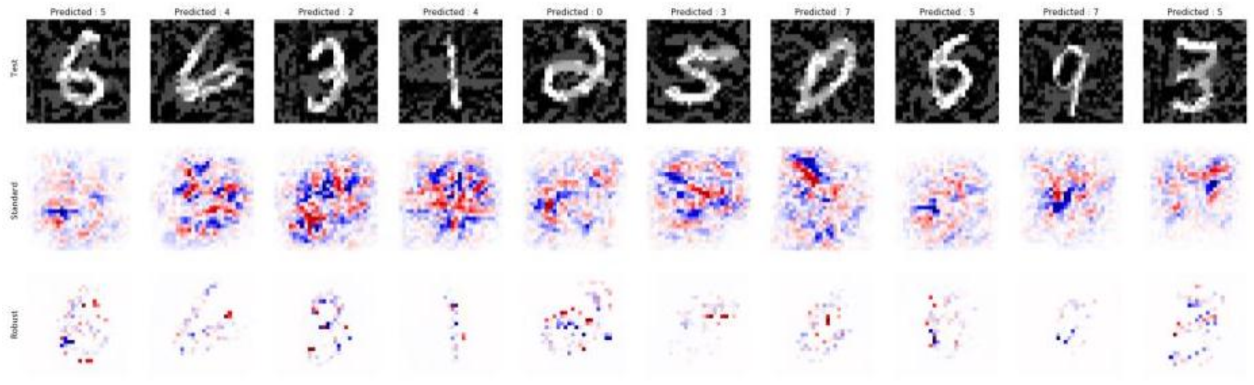
Here, we replicate the experiment from Tsiparas et al. (Fig. 2 and 10 in their paper) on MNIST, and verify the phenomenon that robust models yield much more interpretable gradients than standard models for the CNN architecture. Our results are shown in Figure 1. The experimental details can be found in Appendix B.1.

The gradients of the robust model on the original MNIST images are similarly as sparse and representative of the digits themselves as those in [4]. Likewise, the gradients of the standard model are similarly as dense and patternless as the results in [4]. We also observe this behavior on the adversarially-generated test examples, which is not shared in [4]. The test accuracies for this model are given in Appendix B.1.

[1] Tsiparas et al. give experimental evidence for this phenomenon on CNN models trained over MNIST, CIFAR-10, and a restricted ImageNet.



(a) Standard test examples



(b) Adversarial test examples

Figure 1: (a) Test outcomes on a standard dataset and (b) test outcomes on an adversarial dataset with $\epsilon = 0.3$. Within each subfigure, the top row is a sampling of test images, the middle row shows the loss gradients of the standard model on the test image belonging to the same column, and the bottom row shows the loss gradients of the robust model on the test image belonging to the same column. The gradients have been normalized such that white represents 0, red represents positive values and blue negative.

3 Linear Classifiers for the Bernoulli Model

In this section we analyze a binary classification problem with a linear classifier in an attempt to mathematically explain the observed interpretable gradients phenomenon in a simplified setting.

We first give some simple theoretical extensions to the Bernoulli model from [5]. Schmidt et al point out that the Bernoulli model is relevant to the MNIST dataset, because the individual pixels in handwritten digits are approximately binary (black or white). Our key insight is that certain features are more strongly correlated with the output than others (i.e. pixels near the center are more useful for digit classification than boundary pixels). We note several limitations: (1) this model is generally only appropriate for black/white images, (2) does not encode the translation invariance property which is desired for image classification. Nevertheless, even in the simple linear setting we observe a version of the "interpretable gradients" phenomenon.

3.1 Features with varying correlation

Definition 1 (Modified Bernoulli Model)**.** *Let $\tau \in \mathbb{R}^d$ be the vector of class bias parameters for each of the d features, where each $\tau_j \in [\epsilon, 0.5]$ for some $\epsilon > 0$. Let $\theta^* \in \{\pm 1\}^d$ be the per-class mean vector. Then the (θ^*, τ)-modified Bernoulli model is defined by the following distribution over $(x, y) \in \{\pm 1\}^d \times \{\pm 1\}$. First, draw a label y uniformly at random from $\{\pm 1\}$. Then, sample the data point $x \in \{\pm 1\}^d$ by sampling each coordinate x_j from the distribution*

$$x_j = \begin{cases} y \cdot \theta^*_j & w.p.\ \frac{1}{2} + \tau_j \\ -y \cdot \theta^*_j & w.p.\ \frac{1}{2} - \tau_j \end{cases} \tag{5}$$

In this definition, the vector τ controls the varying degree of correlation of each feature x_j with the output y. For example, if $\tau_j = 0.5$, then by solely looking at feature x_j, we are guaranteed to classify with 100% accuracy. If $\tau_j = 0$, then feature x_j gives us no information about the class y.

3.2 Standard Classification

We give an analogous result to Theorem 27 of [5] for linear classification of the modified Bernoulli model.

Theorem 1. *Suppose n samples $\{(x^{(i)}, y^{(i)})\}_{i \in [n]}$ are drawn from a (θ^*, τ)-modified Bernoulli model, where each $\tau_j \in [\epsilon, 0.5]$, for some $\epsilon > 0$. Let $z := \frac{1}{n}\sum_i y_i x_i$ and $\hat{w} := z/\|z\|$.*[2] *Then the linear classifier $f_{\hat{w}}(x) = \operatorname{sign}(\hat{w} \cdot x)$ has classification error that is bounded as*

$$\mathbb{P}_{(x,y)\sim\mathcal{D}}[f_{\hat{w}}(x) \neq y] \leq \exp\left(\frac{-2\epsilon^2 (\sum_j \tau_j)^2}{d}\right) \tag{6}$$

[2]For simplicity, we consider the linear classifier which is an unweighted average of the samples. Results on logistic regression or SVM classifiers are left to future work.

with probability at least $1 - \exp(-\frac{(\sum_j \tau_j)^2 n}{2d})$.

Proof. Proof deferred to Appendix A. $\square$

This result shows that to achieve an arbitrarily small upper bound on the classification error, either the minimum element in τ must be far from zero or the average element of τ must be large.

We pause to note that in standard classification, *the magnitude of the gradients is proportional to how much the feature correlates with the output.* We give an informal rationale for why this is true. Recall that the gradient of the linear classifier is simply the parameter vector $\hat{w}$. Over many samples, we expect $\hat{w} \approx \mathbb{E}[\hat{w}] = 2\tau\theta^*$ due to concentration in high dimension. Therefore, our linear classifier has gradient for the feature j proportional to τ_j!

3.3 Robust Classification

In adversarial training, we would also expect that the magnitude of the gradients is concentrated on the features j with τ_j sufficiently large, and is 0 otherwise. Here we give informal justification for why this is so, with conjecture regarding our experimental results shown later.

Suppose we adversarially train the modified Bernoulli model using the 0-1 Loss function. Then we want to train the model to have low expected adversarial loss as follows:

$$\min_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}}[\max_{\delta\in\Delta} \mathbf{1}\{\text{sign}(w^T(x+\delta)) \neq y\}]$$

$$= \min_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}}[\max_{\delta\in\Delta} \mathbf{1}\{\text{sign}(w^T(x+\delta)y) \leq 0\}]$$

Though this saddle point problem is difficult to explicitly solve, note that for features j weakly correlated with the output, small perturbations δ may cause $w^T(x+\delta)y$ to switch from positive to negative thus causing error. The robust classifier would want to assign $w_j = 0$ to this feature (since this feature is not important) to minimize the probability of misclassification. Note that we cannot simply take the modified Bernoulli model and perturb each term by an additive factor of ϵ, because the perturbations are computed from iterations of PGD in the direction of w^T, which in this robust classification scenario may not take the form $z/\|z\|$ where $z = \frac{1}{n}\sum x_i y_i$.

So because the classifier wants to assign weights of 0 to sufficiently weakly features, this may induce a thresholding effect, something that we observed in our experimentation. There exists τ^* such that for all $\tau_j > \tau^*$ the assigned weights by the robust classifier increase linearly with τ_j. We believe that this threshold τ^* is a function of d, ϵ.

3.4 Experiments

We empirically examine the effect of robust training for a logistic regression classifier on samples drawn from the modified Bernoulli model, trained via gradient descent. Further experimental details can be found in Appendix B.2. We provide evidence for (1) the thresholding phenomenon for weights of robust classifiers, and (2) the tradeoff between standard and robust accuracy.

Thresholding phenomenon. Robust training induces a thresholding phenomenon, as seen in Figure 2. There exists a threshold value τ^* such that when $\tau_j \leq \tau^*$, the robust classifier assigns weight $w_j \approx 0$. Above this threshold, the weights increase linearly with τ_j. In contrast, the standard classifier assigns weight values *roughly proportional* to τ_j, which is shown by our analysis from Section 3.2 and backed up by experimentation.

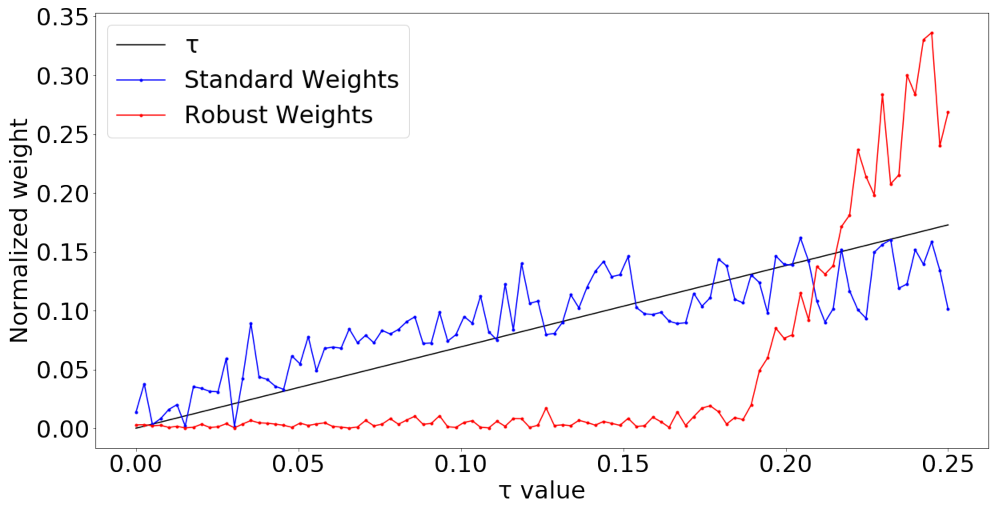


Figure 2: Standard and robust classifiers for the modified Bernoulli model are trained ($\epsilon = 0.5$). All learned weights are normalized to the unit ℓ_2 ball. While the learned standard weights (blue) increase roughly proportional to the τ_j value of the feature, the learned robust weights (red) exhibit a sharp threshold at $\tau^* \approx 0.18$.

Tradeoff between standard and robust accuracy. We show that in the modified Bernoulli model, there exists a tradeoff between standard and robust accuracy. This can be seen in Table 1. This can be explained by the thresholding phenomenon. The robust classifier assigns disproportionately high weight to the feature indices which are more strongly correlated with the output, and assigns no weight to feature indices which are weakly correlated with the output. Therefore, the robust classifier suffers on standard classification because it is not taking advantage of the information in the "weak features".

4 Fully connected NNs train differently than CNNs

Our experimentation on MNIST uncovered an unexplained result: the behavior of robust training on MNIST when using a fully connected neural net (FC) is extremely different when using a CNN (see Figure 3):

1. FCs do not exhibit the "interpretable gradients" phenomenon. The loss gradients for FCs consistently have a nonzero value for the background of the image, and they are

ϵ	Standard Training		Robust Training	
	Std. Test	Rob. Test	Std. Test	Rob. Test
0	98.9%	-	-	-
0.1	98.9%	92.4%	98.9%	94.0%
0.2	98.9%	68.5%	98.7%	79.7%
0.3	98.9%	32.3%	97.8%	60.5%
0.4	98.9%	0.08%	97.0%	38.8%
0.5	98.9%	0.01%	95.2%	24.0%

Table 1: Robust accuracy comes at (slight) cost in standard accuracy for the modified Bernoulli model. Robust training and testing is done with the same ϵ.

not as sparse as the gradients for CNNs.

2. FCs are harder to train robustly. For standard training, it seems like 2 layer FCs can approach the accuracy of CNNs (i.e. $\approx$ 98%). However, FCs perform very poorly in robust training. For a 2 layer FC, robust accuracy converges to $\sim$ 47%, while for a CNN, we can obtain robust accuracies of $\sim$ 93%. We noticed that by significantly overparameterizing the FC via increasing width and depth, we were able to obtain slightly higher robust accuracy, but we could not obtain the CNN robust accuracy.

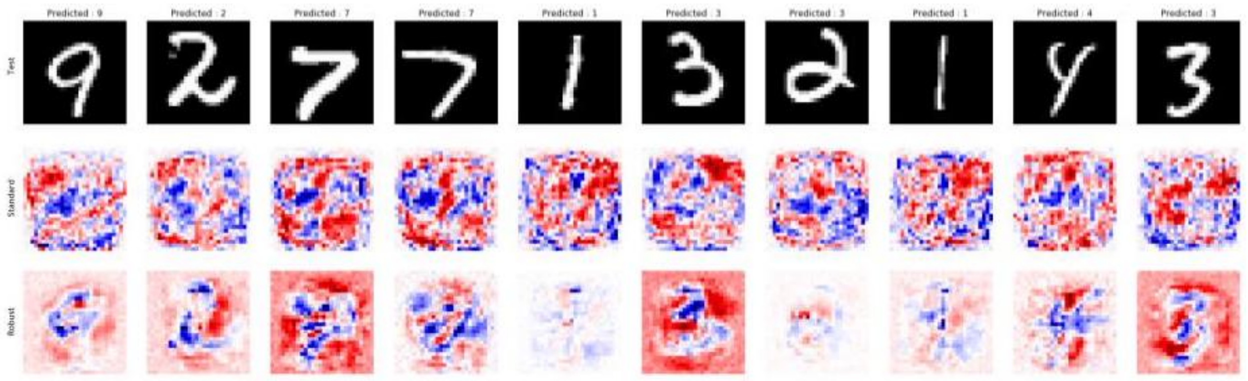
We do not have a good explanation for why this occurs and leave this to future work.

5 Concluding Remarks and Future Directions

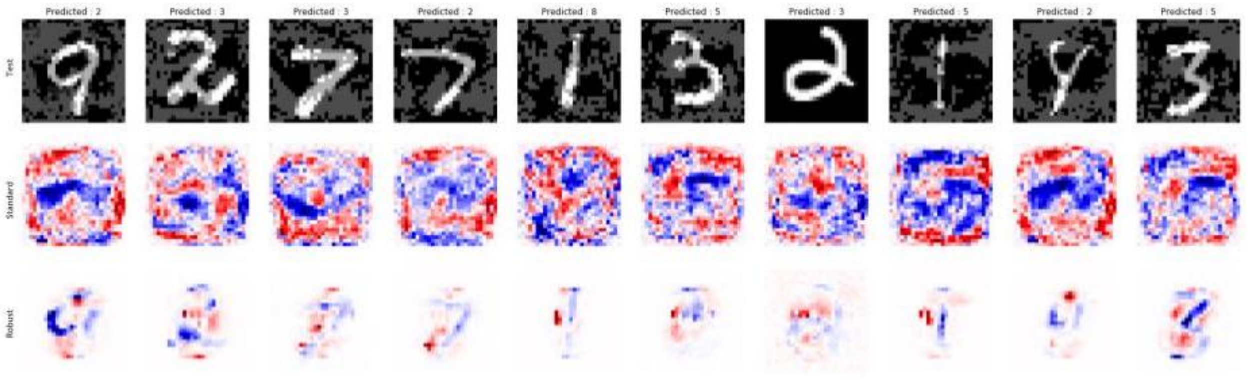
In this report, we have investigated the "interpretable gradients" phenomenon that occurs when one trains robust classifiers. This phenomenon is observed in Tsipras et al. for CNNs trained on image datasets. Here, we show that a simpler version of interpretable gradients occurs in linear classifiers for the modified Bernoulli model, i.e. robustly trained classifiers assign no weight to features whose correlation with the output label falls below a certain threshold. This result is backed by theoretical investigation. We show why standard classification seems to yield a parameter vector whose components scale linearly with the correlation, and give some intuition for the behavior of robust classifiers.

Future Directions. There are several directions for future work. One immediate question from our experimentation is why robust classification with the modified Bernoulli model exhibits this thresholding effect - something our brief analysis did not manage to explicitly uncover. We conjecture that the exact threshold is probably a function of the adversarial perturbation ϵ and d.

Ideally, we would like to show theoretical results on the modified Bernoulli model *from an optimization perspective.* Our theoretical results are stated in terms of the unweighted linear classifier $z := \sum_i y_i w_i$. We would like to understand why (stochastic) gradient descent on logistic or hinge loss yields robust solutions, i.e. Figure 2. Is it possible that the gradient descent algorithm has implicit bias towards certain (robust) solutions (e.g. [6])? Can we provably show that robust training via PGD produces robust classifiers?



(a) Standard test examples



(b) Adversarial test examples

Figure 3: (a) Test outcomes on a standard dataset and (b) test outcomes on an adversarial dataset in the same format as Figure 1, but for FCs instead of CNNs.

Lastly, we would like to further investigate the differences between fully connected NNs vs CNNs in the context of robust training - why are CNNs so much easier to robustly train than FCs? Perhaps explaining why the intepretable gradients phenomenon holds for CNNs but not FCs will lead to the answer to this question.

References

[1] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[2] E. Wong and J. Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," *arXiv preprint arXiv:1711.00851*, 2017.

[3] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial examples," 2017.

[4] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," 2018.

[5] L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, and A. Madry, "Adversarially robust generalization requires more data," 2018.

[6] D. Soudry, E. Hoffer, M. S. Nacson, S. Gunasekar, and N. Srebro, "The implicit bias of gradient descent on separable data," *The Journal of Machine Learning Research*, vol. 19, no. 1, pp. 2822–2878, 2018.

A Proof of Theorem 1

Proof. Without loss of generality, let the Bernoulli model parameter θ^* be the vector of all ones in $\mathbb{R}^d$, denoted $\mathbf{1}$ (note that this is because later in our analysis we consider inner products of the form $\langle z, \theta^* \rangle = \sum_{j=1}^d \pm y^2(\theta^*)^2)$. We will bound the generalization error of the linear classifier given by $\hat{w} := z/\|z\|_2$, where $z := \frac{1}{n}\sum_i y_i x_i$. Note that we can write the classification error $f_{\hat{w}}(x) = \text{sign}(\hat{w}^\top x)$ as follows:

$$\begin{aligned}
\mathbb{P}_{(x,y)\sim\mathcal{D}}[f_{\hat{w}}(x) \neq y] &= \mathbb{P}[\langle \hat{w}, yx \rangle \leq 0] && (7)\\
&= \mathbb{P}[\langle \hat{w}, 2\tau + g \rangle \leq 0] \text{ (let } xy := 2\tau + g) && (8)\\
&= \mathbb{P}[\langle \hat{w}, g \rangle \leq -\langle \hat{w}, 2\tau \rangle] && (9)
\end{aligned}$$

From here note that g is a vector of mean-zero sub-Gaussian random variables, each with parameter 1 because they are bounded by intervals of length 2. Therefore we invoke Hoeffding's inequality to get:

$$\mathbb{P}[\langle \hat{w}, g \rangle \leq -\langle \hat{w}, 2\tau \rangle] \leq \exp\left(-\frac{\langle \hat{w}, 2\tau \rangle^2}{2\|\hat{w}\|}\right) = \exp\left(-2\langle \hat{w}, \tau \rangle^2\right). \quad (10)$$

Next, we will show that the inner product $\langle \hat{w}, \theta^* \rangle$ is large, i.e. our estimate of the linear classifier is close to the ground truth. First we compute a tail bound for the un-normalized z. To do so we rewrite $\langle z, \mathbf{1} \rangle$ in terms of the random variables $\{(y_i x_i)_j\}$. Let $\mathcal{S}$ be the set of all such random variables $\{(y_i x_i)_j\}_{1 \le i \le n, 1 \le j \le d}$, and define the function $h : \mathcal{S} \to \mathbb{R}$ such that

$$h(\mathcal{S}) := \sum_{j=1}^{d} \sum_{i=1}^{n} \tfrac{1}{n}(y_i x_i)_j \tag{11}$$

Note that $h(\mathcal{S}) = \langle z, \mathbf{1} \rangle$, and $\mathbb{E}[h(\mathcal{S})] = \mathbb{E}[\langle z, \mathbf{1} \rangle] = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}[\langle y_i x_i, \mathbf{1} \rangle] = \sum_{j=1}^{d} 2\tau_j$, where the expectation is over the random choice of examples (x_i, y_i). If we perturb any of the random variables $(y_i x_i)_j$ in $\mathcal{S}$ to form the set $\mathcal{S}'$, $h(\mathcal{S}')$ differs from $h(\mathcal{S})$ by at most $\frac{2}{n}$. Then by McDiarmid's Inequality, we have that for any $t > 0$,

$$\mathbb{P}[h(\mathcal{S}) \le \mathbb{E}[h(\mathcal{S})] - t] \le \exp\Big(- \frac{2t^2}{\sum_j \sum_i (\frac{2}{n})^2} \Big) \tag{12}$$

After simplifying the RHS, writing $h(\mathcal{S})$ in terms of z, and evaluating $\mathbb{E}[h(\mathcal{S})]$, we have

$$\mathbb{P}[\langle z, \mathbf{1} \rangle \le 2\sum_j \tau_j - t] \le \exp\Big(- \frac{nt^2}{2d} \Big) \tag{13}$$

Therefore letting $\delta := \exp\Big(-\frac{nt^2}{2d}\Big)$, we have:

$$\mathbb{P}\Big[\langle z, \mathbf{1} \rangle \le 2\sum_j \tau_j - \sqrt{\frac{2d \log 1/\delta}{n}}\Big] \le \delta. \tag{14}$$

Now we note that by the triangle inequality, $\|z\| \le \sqrt{d}$. Therefore:

$$\begin{aligned}
& \mathbb{P}\left[\langle \hat{w}, \mathbf{1} \rangle \le \frac{1}{\sqrt{d}} \left(2\sum_j \tau_j - \sqrt{\frac{2d \log 1/\delta}{n}} \right) \right] && (15)\\
\le\ & \mathbb{P}\left[\langle \hat{w}, \mathbf{1} \rangle \le \frac{1}{\|z\|} \left(2\sum_j \tau_j - \sqrt{\frac{2d \log 1/\delta}{n}} \right) \right] && (16)\\
=\ & \mathbb{P}\left[\langle z, \mathbf{1} \rangle \le 2\sum_i \tau_i - \sqrt{\frac{2d \log 1/\delta}{n}} \right] && (17)\\
\le\ & \delta. && (18)
\end{aligned}$$

By setting $\delta = \exp(-\frac{(\sum \tau_j)^2 n}{2d})$, we have that with probability at least $1 - \exp(-\frac{(\sum \tau_j)^2 n}{2d})$,

$$\begin{aligned}
\epsilon \frac{\sum \tau_j}{\sqrt{d}} &\le \epsilon \langle \hat{w}, \mathbf{1} \rangle && (19)\\
&\le \langle \hat{w}, \tau \rangle && (20)
\end{aligned}$$

where the last inequality follows from the fact that $\tau_j \geq \epsilon \quad \forall j$. Plugging this result into Equation 10 and making use of Equations 7-9, we see that with probability at least $1 - \exp(-\frac{(\sum \tau_j)^2 n}{2d})$, the generalization error obeys the following upper bound:

$$\mathbb{P}_{(x,y)\sim\mathcal{D}}[f_{\hat{w}}(x) \neq y] \leq \exp\left(\frac{-2\epsilon^2(\sum_j \tau_j)^2}{d}\right) \tag{21}$$

$\square$

B Experimental Details

B.1 MNIST

We implement the model discussed in Section 2 in PyTorch using the tutorial `https://www.tensorflow.org/tutorials/estimators/cnn`. The model uses an Adam optimzer, and a learning rate of 0.001. We trained for 10 epochs with a batch size of 64. To find adversarial examples, we used $k = 100$ iterations of PGD with a step size of $\alpha = 0.01$. The standard and robust test accuracies for the standard and robustly trained models are given in Table 2.

ϵ	Standard Training		Robust Training	
	Std. Test	Rob. Test	Std. Test	Rob. Test
0	99.1%	-	-	-
0.1	99.1%	59.4%	99.5%	96.7%
0.2	99.1%	0.06%	99.1%	95.2%
0.3	99.1%	0.00%	99.0%	93.0%

Table 2: Test accuracies for convolutional NN discussed in Section 2.

We also replicate our experiments with a fully connected neural network. The FC was a 2 hidden layer, 1024 node wide network. Again, the optimizer used was an Adam optimizer, and the learning rate was 0.001, and we trained for 10 epochs with a batch size of 64. During training, we run PGD for 100 iterations. The accuracies for varying ϵ are shown in Table 3.

ϵ	Standard Training		Robust Training	
	Std. Test	Rob. Test	Std. Test	Rob. Test
0	97.8%	-	-	-
0.1	97.8%	10.4%	98.7%	90.2%
0.2	97.8%	0.00%	98.7%	81.6%
0.3	97.8%	0.00%	97.1%	48.1%

Table 3: Test accuracies for 2 layer fully connected NN discussed in Section 2.

B.2 Modified Bernoulli Model

We verify our theoretical results with experimental results on synthetic data, which is generated in the following manner.

Bernoulli Model. We initialize a Bernoulli model with dimensionality $d = 100$. We generate $\theta^* \in \mathbb{R}^d$ by setting each entry $\theta_i^* = \pm 1$ with equal probability. We generate $\tau \in \mathbb{R}^d$ by setting each entry linearly spaced from 0 to 0.25. The datapoints are draw according to Definition 1.

Training Details. We draw $N = 10000$ datapoints from the Bernoulli model, and train linear classifiers via GD, using the logistic loss function:

$$\mathcal{L}(x, y, w) = \sum_{i=1}^{n} \log(1 + e^{-y_i w^\top x_i}). \tag{22}$$

For both standard and robust classification, we run GD updates on w for 100 iterations, using Adam with an initial learning rate of 0.01. For robust classification, we compute PGD attacks on *every other iteration.* We find that this performs better than using only adversarial examples on every iteration, however we have no justification for this. Our PGD attack parameters are $k = 1$ rounds, $\alpha = 1$ learning rate, and we vary $\epsilon \in [0, 0.5]$ (One can note that this is essentially the simpler Fast Sign Gradient attack [1]). We do not use the initial random perturbation.