Making Neural Nets Robust Again

Jorio Cocola, Varun Sundar Rabindranath

Abstract

Despite successful applications in many fields, neural network still remain fragile under small adversarial perturbations of the input data. In this project we propose two methods to build classifiers that are more robust to such perturbations. The first method leverages generative models of the input data, the second method acts on the weight matrices of the network.

1 Introduction

In recent years deep neural networks are making stride in furthering the state of the art in many fields such as signal processing and visual recognition, and are successfully deployed in many real world applications. Recently it was discovered [6] that, even the state of the art networks could be easily fooled to misclassify an input image, when a small perturbation is applied to the input image. The perturbation is so small, that a human being, could not tell the original and perturbed images apart Figure 1. These perturbed images are called Adversarial Examples (Adv.Ex.).

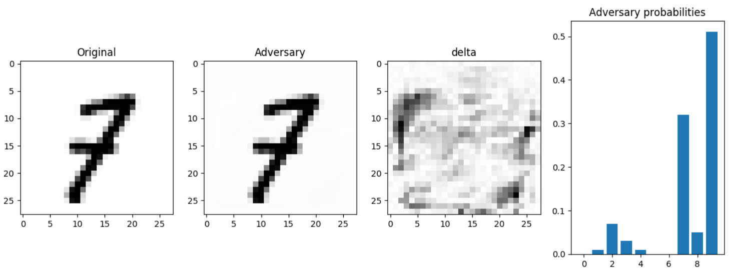


Figure 1: *On the left a correctly classified image $\bar{x}$, next the perturbed adversarial examples with the corresponding perturbation. On the far right the result of the attack on the unprotected classifier.*

This instability of current neural networks naturally rises concerns regarding their reliability and safe use in real world applications. For example, one can imagine the scenario where a self-driving car misclassifies a stop sign as a yield sign due to a small sticker attached to it, or where few small changes in a malicious email are enough to make it be classified as benign by a spam filter.

In this project, we propose and investigate two approaches to make classifiers more robust against adversarial attacks. In the first approach investigated, we use Autoencoder neural networks to learn a low-dimensional representation of the data and train classifiers in that latent space. In this approach we hope to remove adversarial perturbation by preprocessing the input data. In the second approach we train classifiers in picture space and then modify their weight matrices by operating on their corresponding SVD factorization, trying to directly address the weakness of the trained classifier.

1.1 Related work

The phenomenon of adversarial examples was originally observed in [6], where it was also noticed the so called "Universality of Adversarial Examples", that is an adversarial example generated for one classifier would most likely fool some other classifiers as well. We will use this observation to produce adversarial examples for the latent code classifiers (see section 3.5 for more details).

The idea of using a learned representation of the data to protect classifiers from adversarial examples has been proposed recently in [2, 5]. Different from these works we propose to directly classify the latent representation of the input signal instead of using classifiers defined on picture space. Finally while low-rank approximations have already been exploited for network compression [1, 9] and $L-2$ regularization has been used to make networks more robust [7] (even in absence of overfitting). To the best of our knowledge this is the first study employing SVD to increase robustness.

2 Technical Approach

In the following sections we will give details on the two approaches proposed and the methods used to produce adversarial examples.

2.1 Method 1: Robustness via Latent Representation

We considered Autoencoders and Variational Autoencoders for learning a low dimensional latent representation of the input data. Autoencoders are feedforward neural networks that are trained to approximate the identity map and contain a bottleneck layer whose output is latent representation of the input. The Autoencoder network $N : \mathbb{R}^n \to \mathbb{R}^n$ is realized by the composition of an *encoder* $E : \mathbb{R}^n \to \mathbb{R}^l$ and a decoder $D : \mathbb{R}^l \to R^n$, that is $N(x) = D \circ E(x)$. Variational Autoencoders are recently proposed Autoencoders [3], that are are trained as generative models.

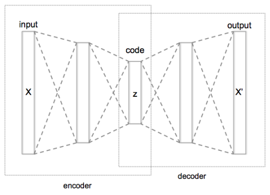


Figure 2: from *from https://en.wikipedia.org/wiki/File:Autoencoder$_s$tructure.png*

Once these networks are trained, we followed and compared two approaches to produce a latent code $z(x)$ for a given input image $x \in R^n$:

- **Encoding**: $z_E(x) = E(x)$
- **Projection**: $z_P(x) = \arg\min_{z \in \mathbb{R}^k} \|D(z) - x\|_p$ for $p = 1, 2$.

Note that in the first case we use the Encoder of the network to produce the latent code z. In the second case instead we project the input image x on the range of the decoder D and then consider the latent code of the projected image. The optimization problem was solved by (sub)gradient method iterating for a large fixed number of iterations.

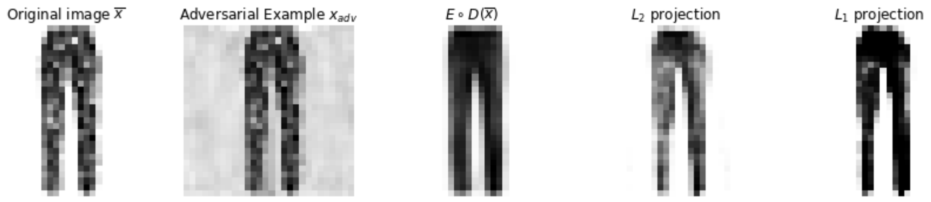

Original image $\overline{x}$
Adversarial Example x_{adv}
$E \circ D(\overline{x})$
L_2 projection
L_1 projection

Figure 3:

2.2 Method 2: Robustness via Low-Rank Approximation

We observed that in the MNIST and the FMNIST dataset, the content is laid on a white background and is at the center of the image, while the adversarial examples of these images had a considerable amount of noise on the white canvas (see Figure 1). We deduced that it is this background noise that makes it very easy to fool a classifier, while at the same time leaving the main features of the image mostly unaltered. We hypothesized therefore that with these slight modifications of the original image $\bar{x}$, the adversarial perturbation were mainly attacking the lowest singular values of the weight matrices.

In this method we therefore trained Neural Networks classifiers $\mathcal{C} : \mathbb{R}^n \to \{1, 2, \dots, L\}$ in input space. Let W_i for $i = 1, \dots K$ be the corresponding weight matrices and consider their SVD factorization:

$$W_i = U_i \Sigma_i V_i^T \tag{1}$$

We then obtained a new classifier $\mathcal{C}_R : \mathbb{R}^n \to \{1, 2, \dots, L\}$ using one of the following two methods:

- **Zeroing out** the bottom k singular values (low rank approximation);
- **Scaling** down the bottom k singular values by a factor of 10;

where k is a hyperparameter.

2.3 Classifiers

For classifier we considered:

- 1L : 1 linear hidden layer
- 2L : 2 linear hidden layers (over-parameterized 1L)
- LRL : 1 ReLU hidden layer, sandwiched between 2 Linear hidden layers

Note that the 2L model is effectively equivalent to the first 1L model. We decided to consider it explicitly, though, because: it was empirically found to train faster, especially considering we train all models for the same number of epochs (see section 3.2 below), and in order to test the effect of over-parameterization on adversarial robustness.

2.4 Adversarial Examples Generation

Given an input image $\bar{x} \in \mathbb{R}^n$ that is correctly classified by a classifier $C : \mathbb{R}^n \to \{1, \dots, K\}$, we produced a corresponding adversarial example x_{adv} in the class y_{goal} by considering the following constrained optimization problem:

$$\underset{x \in [0,1]^n}{\text{minimize}}\, L(x, \bar{x}) \triangleq \text{CrossEntropyLoss}(y_{\text{goal}}, \mathcal{C}(x)) + \lambda \|\bar{x} - x\|_2$$

This loss function is in general not convex, we empirically found though that gradient descent followed by clamping was successful in finding approximate solutions, that is adversarial examples for the input image $\bar{x}$ within small distance from it (see for example Figure 1).

3 Experimental Results

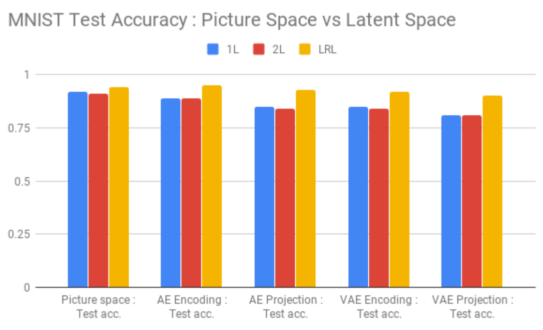
3.1 Dataset

We use the MNIST [4] and FashionMNIST datasets [8]. The MNIST dataset contains images of handwritten digits, while the FashionMNIST dataset contains images of clothing. In both datasets each image is tagged with labels 0-9, indicating a digit for MNIST and a type of clothing for FashionMnist. Both datasets consist of 60,000 training images and 10,000 testing images with 28×28 greyscale pictures. The only pre-processing performed was scaling all the pixels in the range $[0, 1]$.

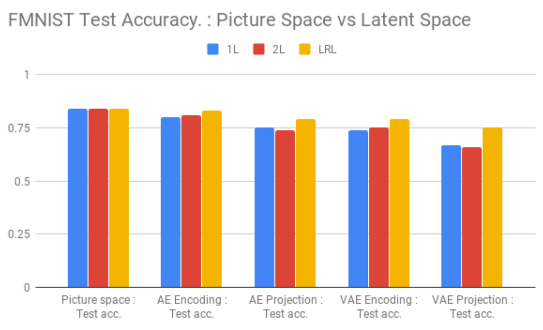
3.2 Autoencoders and Projections

We implemented and trained Autoencoders and Variational Autoencoders on the above mentioned datasets, using the Pytorch framework. The Autoencoders had 3 layers with leaky-ReLU in the hidden one (32 nodes) and sigmoid at the output. The Variational Autoencoder had 5 layers (150 and 32 nodes) with leaky-ReLU and linear activation functions in the hidden layers . At the we used output sigmoid function.

Once the models were trained we obtained latent codes following one of the mentioned approaches. In particular we observed that gradient descent for the projection on the range of the decoder worked well enough with one random initialization and 500 iterations. Despite giving good reconstruction error the ℓ_1 projection gave not satisfactory visual results so we only experimented with latent codes classification obtained via ℓ_2 projection.



(a) MNIST



(b) FMNIST

Figure 4: Classifiers test accuracy

3.2.1 Latent Dataset

We generate latent codes for the MNIST and FashionMNIST via the "Encoding" and the "Projection" approaches discussed above. For each dataset, we generate,

- latent codes using the "Encoding" approach on a Autoencoder (MNIST-AE-ENCODING / FMNIST-AE-ENCODING)
- latent codes using the "Projection" approach on a Autoencoder (MNIST-AE-PROJECTION / FMNIST-AE-PROJECTION)
- latent codes using the "Encoding" approach on a Variational Autoencoder (MNIST-VAE-ENCODING / FMNIST-VAE-ENCODING)
- latent codes using the "Projection" approach on a Variational Autoencoder (MNIST-VAE-PROJECTION / FMNIST-VAE-ENCODING)

All latent codes are in R^{32}.

3.3 Classifiers

We trained 3 classifiers for all the datasets. The latent space classifiers were trained for 150 epochs. We limit the training to 150 epochs on all models, as we wanted the evaluation to be a function of model complexity only. The image space classifiers were trained for 50 epochs, as we observed the test/train accuracy plateaued around 50 epochs. All classifiers were trained with a train batch size of 1000 and with Adam optimizer. From figure 4, we make the following observations,

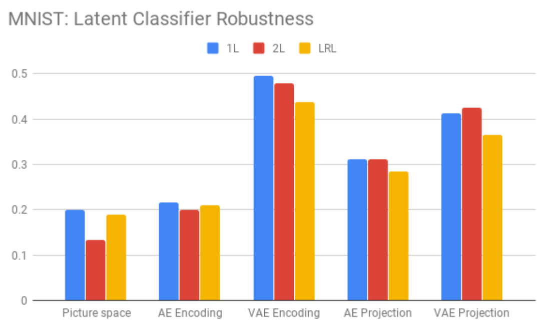
- Image space classifiers achieve the highest test accuracy.
- Among the latent classifiers, the AE-ENCODING+LRL classifier performs best
- Accuracy of the 1L and 2L classifiers are always less than the LRL classifier We believe this is because the expressivity/capability of the LRL classifier is higher
- We see that the accuracy of models on the MNIST dataset is higher as compared to the accuracy of the models on the FMNIST dataset. We believe this is because the MNIST dataset is relatively simple and has less number content-creating pixels in the image.

3.4 Adversarial Robustness

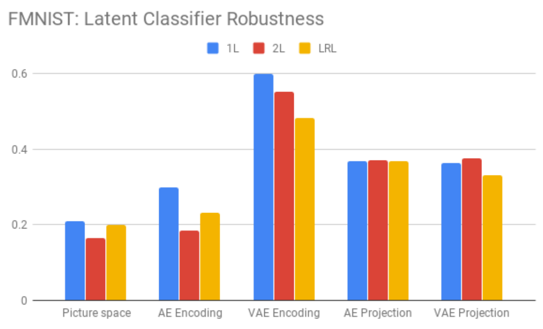
We define adversarial robustness, as the average perturbation required on the input for the classifier to misclassify the input with 95% confidence. For every trained classifier, we generated 180 such adversarial examples, and calculated the average perturbation as:

$$\frac{\|x_{adv} - x\|_2}{\|x\|_2}$$

Where the denominator serves as a normalization factor, so the average input perturbation of the image and latent space classifiers could be reliably compared. From figure 5, we make the following observations,

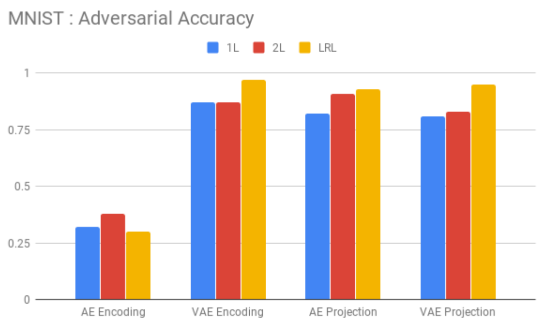


(a) MNIST

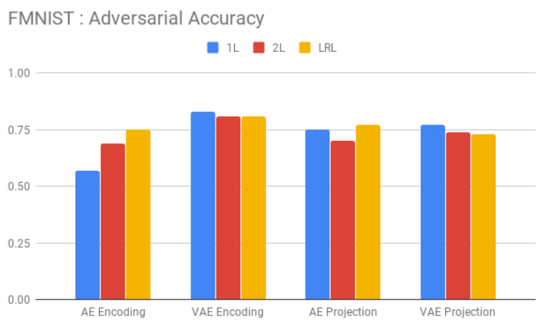


(b) FMNIST

Figure 5: Adversarial robustness of latent classifiers



(a) MNIST



(b) FMNIST

Figure 6: Adversarial accuracy of latent classifiers

- VAE-ENCODING space models are the most robust irrespective of the dataset.
- AE-ENCODING and image space models, which enjoyed high test accuracy are the least robust. We infer that these spaces are very insecure that it is easy to fool classifiers operating on these spaces.

3.5 Adversarial Accuracy: Universality of Adversarial Examples

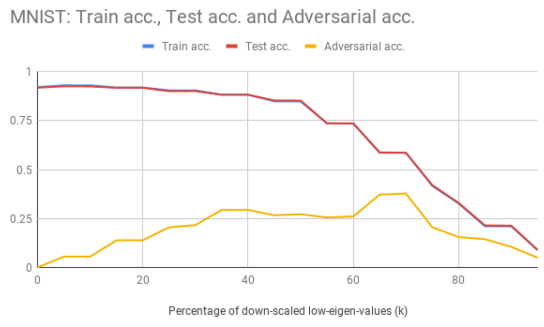
In this section we investigate how the latent space classifiers behave when supplied with adversaries of image space classifiers.

For every adversarial example produced by the image space classifiers, we generated 4 latent codes (AE-ENCODING, AE-PROJECTION, VAE-ENCODING, VAE-PROJECTION), and test if the low dimensional representation could fool the corresponding classifiers.

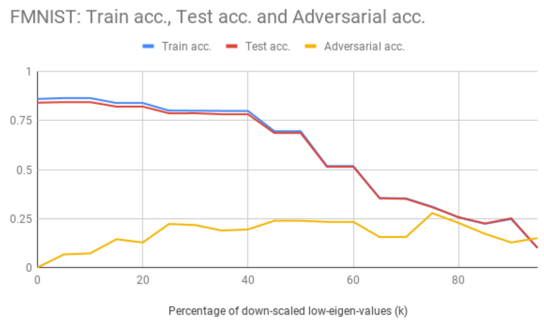
From figure 6, we make the following observations,

- VAE-ENCODING models have the highest average adversarial accuracy between the datasets
- AE-ENCODING models have the least average adversarial accuracy between the datasets

We observe VAE-ENCODING models to be a clear winner in terms of adversarial accuracy and robustness and on examining figure 4, we see that these models are on-par with the best models for the classfication tasks on MNIST and FMNIST datasets. We hypothesize that the superiority of the Variational Encoding approach is due to the following reasons. Contrary to the projection approach the stochastic embedding of the input image to a latent code seemed to prevent to fit also the adversarial perturbation, on the other hand since the network is trained as a generative model we speculated that this forced it to have a better behaved latent space contrary to the Autoencoder case that only attempts to reconstruct the input.



(a) MNIST 2L



(b) FMNIST 2L

Figure 7: Effect of SVD on the image space 2L classifier

3.6 SVDs defense

In this section, we study if the scaling down of the bottom singular values helps in increasing a classifier's adversarial accuracy. We measure this by asking the following question, "with x% of lowest singular values scaled down from all weight matrices, how many of its original adversaries can the classifier detect ?" Note that We report here only the results for the scaling since we found they were matching those for the zeroing out approach.

Let us concentrate on the effect of this approach in a MNIST classifier. We find the behaviour of the 2L model to be the most interesting and make the following observations from figure 7:

- We see that, with 40% of bottom singular-values scaled down from all layers, there is a high boost in adversarial accuracy with a very weak decrease in test/train accuracy.
- We also observe a slight boost in train/test accuracy when only 5%-15% of the bottom singular-values were scaled down by 10%.

We observe similar trends in the FMNIST 2L classifier as well.

4 Conclusions and Future Work

We proposed two methods to make classifiers more robust to adversarial attacks.

We generated latent codes for the MNIST and FashionMNIST datasets using 4 approaches and trained simple neural network classifiers that work on the image and latent spaces. We evaluated these classifiers on grounds of adversarial accuracy and adversarial robustness. Further, We studied the effect of SVD on the weigth matrices of these classifiers. We found that the classifiers trained on the latent codes generated by Variational Autoencoders using the "Encoding" approach to be the most robust to adversarial examples. We also found that scaling-down or zeroing out the bottom singular values of the weight matrices of a classifier increases the robustness of the classifier without much drop in train and test accuracy.

We found these approaches very interesting and motivate interest in pursuing these preliminary studies. In particular woudl be interesting to check if VAE-ENCODING models perform the best also on other complex datasets such as CIFAR-10, Celeb-A etc. It gives a fast way (compared with projection) to protect against adversarial attacks. On the other hand We found very interesting that the model's test accuracy as well as robustness could increase when zeroing out or scaling down the low-singular-value . This observation seems to point in the direction of safe, with respect to generalization and adversarial attacks, compression of neural networks. Moreover the effect of SVD on classifiers could potentially lead to a better understanding of the problem adversarial vulnerability and generalization in general.

5 Participants Contribution

Jorio Cocola: Was primarily responsible for setting up the project, literature review and constructing and training generative networks

- Implemented (Variational) Autoencoders neural networks with pytorch (also Gan networks that we in the end decided to discards)
- Implemented projections steps
- Produced and processed the various latent code datasets

Varun Sundar Rabindranath: Varun was primarily responsible for constructing the image and latent space classifiers and training them on all the datasets.

- Implemented neural network models with pytorch and trained classifiers on the all the datasets
- Implemented the generation of adversarial examples in the image space
- Implemented the SVD of weight matrices of a classifier's linear layers

Both Jorio and Varun, worked on both the datasets and contributed equally in generating the results, interpreting them and producing the final report.

References

[1] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, *Exploiting linear structure within convolutional networks for efficient evaluation*, in Advances in neural information processing systems, 2014, pp. 1269–1277.

[2] A. Ilyas, A. Jalal, E. Asteri, C. Daskalakis, and A. G. Dimakis, *The robust manifold defense: Adversarial training using generative models*, arXiv preprint arXiv:1712.09196, (2017).

[3] D. P. Kingma and M. Welling, *Auto-encoding variational bayes*, arXiv preprint arXiv:1312.6114, (2013).

[4] Y. LeCun, C. Cortes, and C. Burges, *Mnist handwritten digit database*, AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist, 2 (2010).

[5] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, *Pixeldefend: Leveraging generative models to understand and defend against adversarial examples*, arXiv preprint arXiv:1710.10766, (2017).

[6] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, *Intriguing properties of neural networks*, arXiv preprint arXiv:1312.6199, (2013).

[7] T. Tanay and L. D. Griffin, *A new angle on l2 regularization*, arXiv preprint arXiv:1806.11186, (2018).

[8] H. Xiao, K. Rasul, and R. Vollgraf, *Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms*, arXiv preprint arXiv:1708.07747, (2017).

[9] J. Xue, J. Li, and Y. Gong, *Restructuring of deep neural network acoustic models with singular value decomposition.*, in Interspeech, 2013, pp. 2365–2369.