
Adversarial MNIST Image Generation Analysis

Dayeol Choi

day3olchoi@gmail.com

Image Generation Procedure

Before producing adversarial images, we recall a typical procedure for finding optimal weights in a machine learning classifier. Consider an arbitrary machine learning classifier whose weights are given by θ , which we will use to classify MNIST images. We can use some form of gradient descent to find good values for θ . The training procedure is given by

$$\theta \leftarrow \theta - \epsilon \nabla_{\theta} [\text{CE}(\mathbf{f}_{\theta}(\mathbf{x}), \mathbf{y})],$$

where we denote the cross entropy with "CE", input by \mathbf{x} , the label by \mathbf{y} , and the step size by ϵ .

We can add a regularization term on the weights, in order to prevent them from getting too big. The modified training procedure with a regularization parameter λ will be given by

$$\theta \leftarrow \theta - \epsilon \nabla_{\theta} [\text{CE}(\mathbf{f}_{\theta}(\mathbf{x}), \mathbf{y}) + \lambda \|\theta\|_2^2].$$

In order to generate adversarial images, all we need to do is to hold the weights constant and update the images at each step instead.

$$\mathbf{x}_{\text{adv}} \leftarrow \mathbf{x}_{\text{adv}} - \epsilon \nabla_{\mathbf{x}_{\text{adv}}} [\text{CE}(\mathbf{f}_{\theta}(\mathbf{x}_{\text{adv}}), \mathbf{y}) + \lambda \|\mathbf{x} - \mathbf{x}_{\text{adv}}\|_2^2]$$

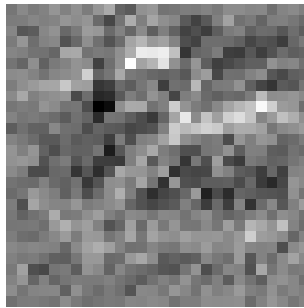
Note that at the first step of this optimization procedure, \mathbf{x}_{adv} will simply be \mathbf{x} , the original image. The l_2 regularization term encourages the generated images to stay close to the original image.

Generated Image Analysis

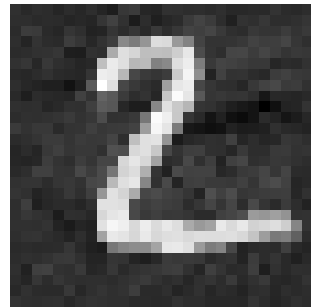
In the set of images below, we have an MNIST digit "2" that is used to create an adversarial image that is classified as a 6 by a trained Convolutional Neural Network provided at this link. The middle image shows the difference between the original and adversarial image. We see that even though the adversarial image clearly looks like a "2", it is classified as "6" by our trained ConvNet with 89.9% confidence. We note that the same ConvNet is capable of 99.2% accuracy on test MNIST digits and that our ConvNet has been trained with neither the original image shown nor the corresponding adversarial image.



(a) Original Image
Label:2; Confidence: 1.0


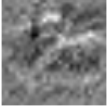


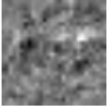
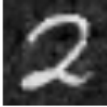




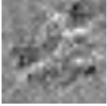
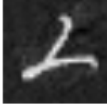





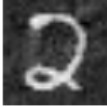

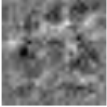


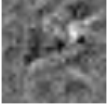





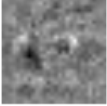



(b) Delta Image
Label:7; Confidence: 0.340



(c) Adversarial Image
Label:6; Confidence: 0.899

Appendix 1: Generated Images and Predictions

		
Label:2; Confidence: 1.0	Label:7; Confidence: 0.340	Label:6; Confidence: 0.899
		
Label:2; Confidence: 0.999	Label:5; Confidence: 0.442	Label:6; Confidence: 0.853
		
Label:2; Confidence: 0.999	Label:5; Confidence: 0.648	Label:6; Confidence: 0.909
		
Label:2; Confidence: 0.999	Label:1; Confidence: 0.252	Label:6; Confidence: 0.898
		
Label:2; Confidence: 1.0	Label:5; Confidence: 0.624	Label:6; Confidence: 0.886
		
Label:2; Confidence: 0.999	Label:5; Confidence: 0.301	Label:6; Confidence: 0.810
		
Label:2; Confidence: 0.999	Label:5; Confidence: 0.358	Label:6; Confidence: 0.838
		
Label:2; Confidence: 1.0	Label:7; Confidence: 0.490	Label:6; Confidence: 0.813
		
Label:2; Confidence: 0.999	Label:2; Confidence: 0.489	Label:6; Confidence: 0.822
		
Label:2; Confidence: 0.998	Label:5; Confidence: 0.505	Label:6; Confidence: 0.858

Appendix 2: Extensions

Dropout at Test Time

We can estimate the ConvNet's uncertainty with its prediction with a Monte Carlo estimate of the predictions, following [1]. We simply run the prediction multiple times with dropout. Then, we average the predictions to get the final prediction by the model. The variance of the predictions will give us an idea of the model's uncertainty.

With the original image as input, the ConvNet always classifies the image as "2", even with dropout, and even though the image was not in the training set of the model.

However, with the adversarial image as input, the ConvNet may not classify the image as "6". The answers differ due to the randomness from dropout, unlike when we input a real image. The results can be seen at this link.

One Class SVM

Alternatively, we can try and identify adversarial images directly. We train a one class SVM with real "6" digit images from the MNIST training data. Then, we test the model to see if it can identify the adversarial images. As the results at this link shows, the one class SVMs can identify adversarial images, but they also identify many real "6" digit images as non "6". Thus, those one class SVMs are not suitable for adversarial image detection on its own.

References

- [1] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *ICML*, 2016.