MEASURING ROBUSTNESS OF IMAGE CLASSIFICATION MODELS

TECHNICAL REPORT

Viral Shanker
Department of Data Science
Stevens Institute of Technology
CPE 646
viralshanker@gmail.com

May 24, 2021

ABSTRACT

Computer vision problems have gained a lot of attention in the past decade or so, with Convolution based architectures like VGG16, Mobilenet and ResNet continually upping the bar with deeper and deeper networks, being able to outperform humans and offer fantastic classification accuracy. However there are relatively fewer studies done on their robustness - that is how do some of these architectures compare against one another, and even shallow networks, in terms of computer vision problems and what, if anything, can be done to alleviate them. I focus on variations of the ResNet model and a naive 1 hidden layer feed-forward neural network. Each of the models has two sets of training data - one taken as is, and the other transformed/infused with noise in various different ways. Finally, we reproduce this experiment on two classic datasets: the MNIST and CIFAR-10 datasets - both classic datasets in computer vision. We find that on the MNIST dataset, the Noisy Basic Classifier is far more robust than ResNet in terms of average change in pixels. In terms of number of pixels changed, the basic classifier outperforms the noisy basic classifier. However, the story is not clear cut since our robustness does not take into account believably. In the CIFAR dataset, normal feedforward networks perform extremely poorly, so instead the comparison was done between ResNet and a much deeper ResNet. Rather than injecting noise here, we perform transformations such as rotation, flips, shears, and zooms, just to name a few. Here we find it is much easier to attack the networks in general. Here, differences were far less pronounced than the MNIST setup, but the noisy Resnet and the non-noisy Deep Resnet performed the best. The difference, however, was not significant.

1 The Models and the Data

For each dataset I develop 2 networks. Each network has the same training data and is tested on the same testing data so that the comparison is as direct as possible. Each of the networks are trained on the normal training data and on noisy/transformed version of the data - so each network has two different sets of weights. We differentiate between them by calling one the noisy model.

All in all, each dataset has 4 associated models.

Note that for evaluation, we are using accuracy - which is a rather harsh metric with multiple classes as opposed to say, top-K accuracy which gives points for the correct answer being in one of the models top k guesses. Accuracy is 1 or 0 based on if the model gives the correct answer.

1.1 MNIST

Data and Transformations

The MNIST dataset is a set of hand written digits in black and white. Each image is 28x28 pixels, and we have 60000 of them total. 50000 are for training and 10000 are reserved for testing. For the noisy models, we add Gaussian noise to the data. Here is an example:

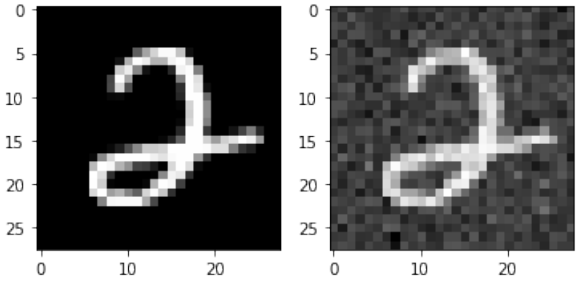

Figure 1: The effect of Gaussian Noise

It can be described as a sort of "static."

Note that the test data is identical for both - no static/noise is added to the testing data whatsoever. This noising trick is usually used in Autoencoders.

Please find full details in the TrainMNIST notebook.

Networks

The MNIST dataset involves two networks. The first is the basic feed-forward one, whose entire design you can see in Figure 2. Nothing too complicated here, with one 50 node hidden layer and a 10-fold output layer with softmax activation to classify digits (0-9).

The other network is a ResNet architecture [1]. Which let's us go extremely deep by having an identity module that alleviates the vanishing gradient problem. The scope of ResNet is not at all the focus of this paper, but in short, the network has 123 (!) layers total, with 16 skip connections (illustrated in Figure 3) and 20,402,188 trainable parameters. Note that ResNet is a convolution neural network so we heavily utilize convolution layers. [2].

In terms of accuracy, ResNet far outperforms the basic classifier even in this simple computer vision problem.

Table 1: MNIST Model Evaluation

Model	Accuracy
Basic NN	0.93
Noisy NN	0.8372
ResNet	0.9901
Noisy ResNet	0.9890

1.2 CIFAR-10

Data and Transformations

The CIFAR-10 dataset [3] is a set of 32x32 color images that falls into one of 10 classes: airplance, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. These images are from various angles and have no standard form - for instance a dog can be a dog taken from a distance or just the face of one. This problem is far more complex than MNIST. Transformation are self explanatory with rotation and zooming and such. This alleviates overfitting since each epoch is "new" data. Note once again that the test set is not at all touched.

Networks

[1] He, Kaiming, et al. "Deep Residual Learning for Image Recognition." ArXiv.org, 10 Dec. 2015, arxiv.org/abs/1512.03385.

[2] Y. LeCun, P. Haffner, L. Bottou and Y. Bengio: Object Recognition with Gradient-Based Learning, in Forsyth, D. (Eds), Feature Grouping, Springer, 1999

[3] https://www.cs.toronto.edu/ kriz/cifar.html

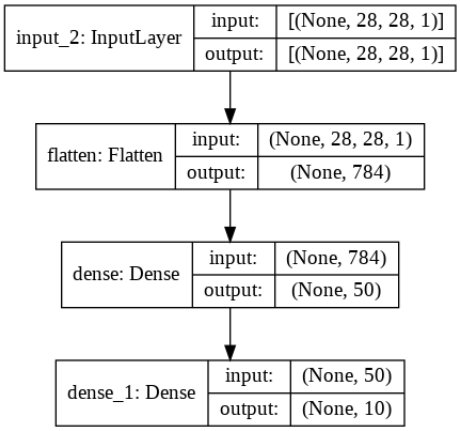

Figure 2: Simple Feed-Forward Network

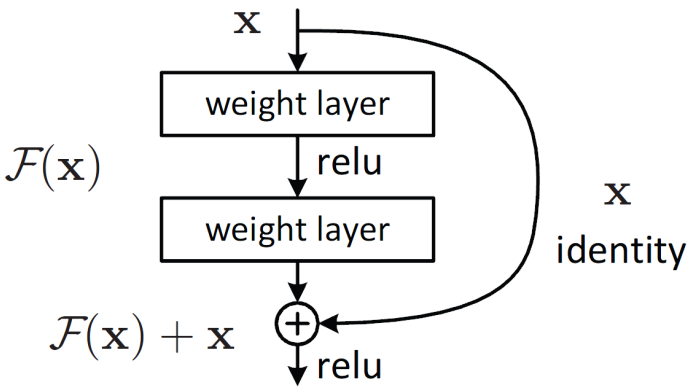

Figure 3: An Example of the Skip Connections that make up ResNet

A feedforward network performs terribly, to the point that it is not worth including, so we compare two different ResNets. One deeper than the other. Where one is the same depth as the MNIST (123 layers, 16 skip connections, 64 CNN filters, 20,402,188 trainable parameters) the deep ResNet is 151 layers, 20 skip connections, and 128 CNN filters. It is significantly more complex with 93,359,376 trainable parameters.

We can see the performances below. Note that basic ResNet actually performs rather poorly on this. Noisy Deep ResNet does worse than normal Deep ResNet - this may actually indicate an overfitting issue.

Table 2: CIFAR Model Evaluation

Model	Accuracy
ResNet	0.738
Noisy ResNet	0.884
Deep ResNet	0.8572
Noisy Deep ResNet	0.8245

Please find all details in TrainCIFAR notebook.

2 Attacking the Networks

This is the actual heart of the project, in which we attack the trained models in order to see how robust they are. This is in fact a form of adversarial network [4].

Let's say we have a model M which takes as input a vector X and returns a vector Y. Y is a probability distribution such that $\sum Y = 1$. The model M classifies vector X by choosing the index that contains the maximal element of Y.

The objective becomes to change vector X such that the classification - that is, the index of the maximal element of the output Y - is different than the original, correct prediction.

Our adversary has reward: $\sum M(X) - M(X')$ where X' is the edited image.

We take it a step further and impose that $maxM(X') \geq 0.9$ and $maxM(X') \neq maxM(X)$. This means that the model must be over 90 percent confident of the incorrect guess.

To maximize reward, we take the derivative of the model with respect to X' this gives us a d dimensional gradient where d is the number of elements in X. We then use gradient ascent to step toward the maximal step, and repeat either 1000 times, or until the model incorrectly classifies the modified image, though in practice, the latter happens almost every time in all but one model.

[4]Goodfellow, Ian J., et al. "Explaining and Harnessing Adversarial Examples." ArXiv.org, 20 Mar. 2015, arxiv.org/abs/1412.6572.

The one other ending condition is defaulting to a random state. This occurs only in one particular case: the noisy feedforward network, which, when it sees any random assortment, actually returns a particular configuration that remains constant. So, if our loss remains static for 20 consecutive epochs, and the prediction on the image is the same as the prediction on randomly generated static - we end.

Note that instead of using the actual gradient we use the signs of the gradient (or, alternatively, the normalized gradient) to determine our step direction. The issue was that occasionally, we would have the vanishing gradient issue, with the gradient being zero. Hence we have added a bit of randomness to help us break out of the saddle points if need be.

It is important to note that model is frozen. We do not change the model at all, only the image. The goal is to change the image as little as possible to force an erroneous classification. So models which require more changes to image to misclassify will be considered more "robust."

This method is know as the Fast Gradient Signed Method. [5].

2.1 Measuring Robustness

As mentioned above robustness is somehow the amount an image can be changed before it is misclassified. The larger this amount, the more robust the model.

I use 2 different measures to measure "change"

- Average Per Pixel Change: This is self explanatory, we measure the absolute difference between every single pixel in the original image X and the modified image X'. In the case of RGB pixels like in CIFAR, we sum the RGB differences together. This averaged across many images X gives us average per pixel change.
- Number of Pixels Changed: This is also fairly self explanatory, we count the number of pixels that saw change. The more pixels changed, the more the image has changed. However, this has the issue of treating all changes as equal. The human eye is incapable of distinguishing very fine changes, so I considered a subcategory of this in which we count the number of pixels that changed by some threshold amount. For MNIST this was 10 percent, and for CIFAR this was 1 percent.

2.2 Attacking MNIST

Please see CPE-MNIST-ATTACK notebook for full details and many more images.

Attacking these was fairly difficult (at least relatively, as we will find out). Requiring lots of iterations of the method, generally. Let's see some examples:

Figure 4: 7 After Processing. Each Network believes with at least 90 percent accuracy that the number is not 7 (with the title containing their belief

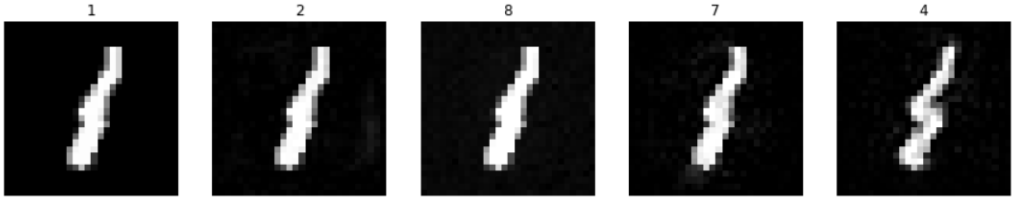

Figure 5: 1 After Processing. Changes Much more pronounced here. Models correspond to the same columns, with the title being their guesses.

[5] Goodfellow, Ian J., et al. "Explaining and Harnessing Adversarial Examples." ArXiv.org, 20 Mar. 2015, arxiv.org/abs/1412.6572.

Some of the images change a lot more than others, though note that all of these examples have in fact achieved the objective as outlined above. To really see this in aggregate we cannot cherry pick examples. We must look over the test set.

Now we aggregate and see how each model performs over many images on average:

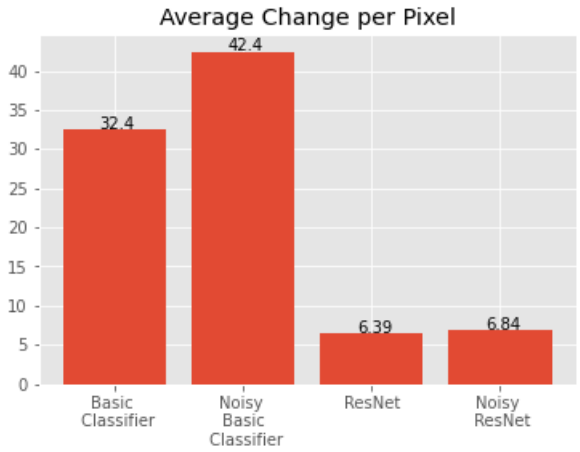

Figure 6: Average Change Per Pixel

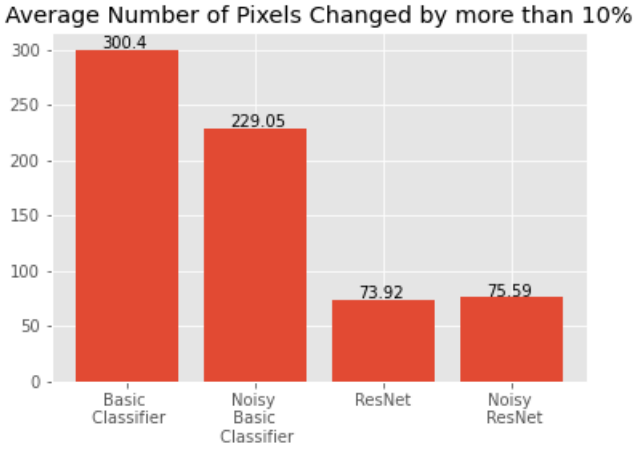

Figure 7: Number of pixels that changed more than 10 percent. 784 pixels total

Clearly the classifiers, despite having much lower accuracy are far, far more robust than their deeper counterparts by a huge margin. Note that the noisy classifier is more robust in that each pixel on average needs to be changed by more than the non-noisy counterpart, while the basic classifier needs more pixels changed. Both ResNets do equally terrible in terms of robustness. Note that Noisy feedforward is NOT achieving it's goal. It is defaulting to the "random" state after a certain amount of iterations. So on one hand, an attack can be easily identified - but it is not robust in terms of accuracy. The changed image looks very similar to the original.

And this is where we see the drawback to our metric.

Caveats in Evaluating Robustness

Our metric in this case is not perfect. Turning our attention back to figures 4 and 5, note the actual changes. The Noisy Resnet, while modifying few pixels by less, actually change the image to look like another number, rather than random noise. The 7 almost looks like a 3. The 1 looks like a 4, with that bend. This phenomenon is consistent in all the images that I generated. Running the the code will show this. But it is hard to measure this numerically. So the above bar charts should be taken with a grain of salt.

2.3 Attacking CIFAR

Please see CPE-CIFAR-ATTACK notebook for full details and many more images.

This much more complex dataset was very easily tricked. With far lesser iterations being required to achieve said objectives. The deeper ResNet did not make too big a difference overall, but first, let's see some examaples. We also look at the differences, since the image changes are much more subtle.

Figure 8: What was originally a dog, and all networks modified image and guess

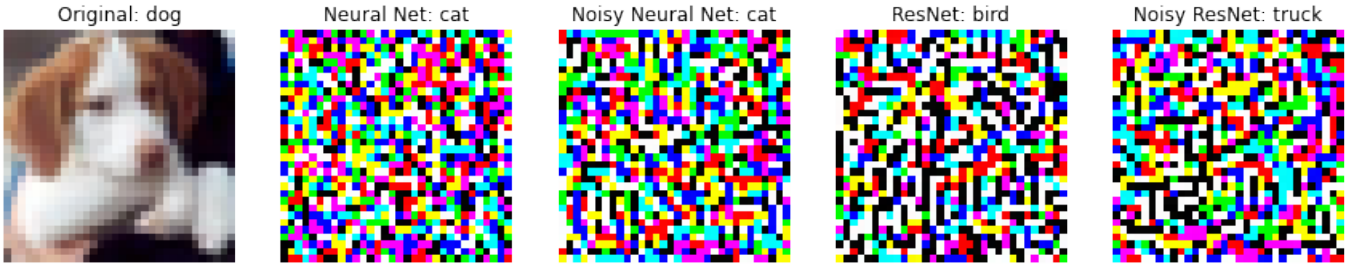

Figure 9: The added pixel amount to each vector

Figure 10: What is originally a ship, and each network's image and guess

Figure 11: The added pixel amount to each vector

Finally, we look at robustness just as we did with MNIST

The differences are much less across models, though note that the Deep ResNet requires the most number of pixels changed. Note also, relative to MNIST, the lack of robustness. The pixels have to be changed by an order of magnitude less. and the number of pixels needed to be changed is also much lower. (see Figure 12 and 13 at the end)

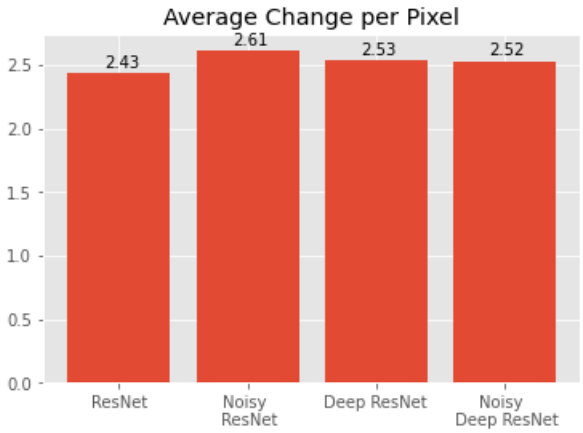

Figure 12: Average Change Per Pixel

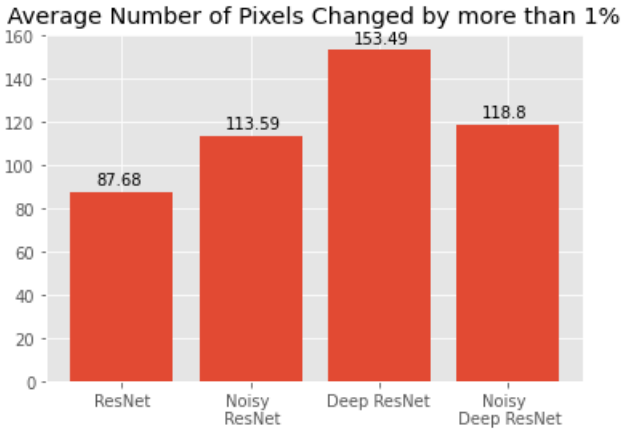

Figure 13: Number of pixels that changed more than 1 percent. 1024 pixels total

3 Conclusion

Adversarial attacks of this kind are extremely effective, clearly, especially in more complex cases. The implications of such attacks, in a world dominated by computer vision problems - including self driving cars - are great. While adding noise does seem to help in some cases, it appears a more targeted approach may be required to make a network resistant to this kind of attack.

And indeed, this attack can be made even more malicious by targeting. This attack was untargeted - we only cared to make the network guess wrong, but a targeted attack may be one that directly wants the network to classify ships as , say cars. Or 3s as 9s. The procedure is identical, simply the loss function is slightly different.

While deep networks offer high accuracy and seemingly better results, the complex architecture comes at a price - with simpler networks (i.e. feed forward instead of CNN) being more robust, but for complex problems, they are simply not an option.