Using Tensorflow to Study Effects of Adversarial Examples

Mustafa Yesilyurt
Dept. of Computer Science
San Jose State University
San Jose, California, USA
myesilyurt700@gmail.com

***Abstract* – The key goals of this project are 1) to find, within a given data set, a threshold for the factor of noise that must be injected into an image such that a machine learning model will misclassify that sample and 2) to observe how widespread the impact a given noise filter is across an entire data set. We utilize Python and the Tensorflow library to conduct these experiments on three data sets: MNIST, Fashion MNIST, and Kannada MNIST. The development environment used was Jupyter Notebook.**

Keywords – Adversarial example, attack, neural network, image, data, noise, Python, Tensorflow, confidence

I. INTRODUCTION

Neural networks are becoming increasingly widely used in today's industries, so it is only natural that some vulnerabilities would be found. Adversarial attacks exploit the data-dependent nature of machine learning techniques by modifying training data to induce misclassification of input but only with very subtle modifications that are indiscernible to the human eye, thus making them difficult for humans to detect and posing a serious threat to neural networks. The alteration is made through an injection of random values, or "noise" that takes effect via subsequent retraining of the classifier. It is common for this noise injection to have an associated factor that scales the amount of noise to be injected, so the goal is to find a minimum value for the noise factor such that misclassification is guaranteed. We hope that this value proves to be a kind of threshold that can apply to an entire data set, though this is highly unlikely. For that reason, we also want to measure the impact a given noise factor could have on a data set, that is we want see how many samples will end up being misclassified after the model is retrained. In Section III, we will dive deeper into the experiments that may help us find these results.

II. RELATED STUDY

A paper that was of particular help during the research step of this project was "Adversarial Patch" by Abadi et al. Specifically, it provides clear details about the dangers of adversarial attacks and gave examples of real-world applications where neural networks were fooled because of them. Such examples included the alteration of road signs, the defeat of facial recognition software using custom-made glasses, and more.

Another aspect I like about this paper is that it incorporates existing research into its premise; it acknowledges the research of small, frequent noise injections and ventures into finding ways to induce misclassification with large noise injections where undetectability is no longer a part of the attacker's objective. The paper, through its alternative approach, further illuminates exactly why adversarial examples are effective, specifically by touching upon the concept of saliency in image classification (saliency being how noticeable or easily observable something is). The entire concept of the eponymous "adversarial patch" employs this; the patch is a very colorful, smaller image that is transposed onto the training data sample such that it messes with the saliency of sample's true classification's qualities; if we put a colorful toaster oven in the corner of an image of a bird in flight, the learner will focus more on the toaster's many unique designs and confuse them with the bird's, thus restricting the bird's unique properties' saliency.

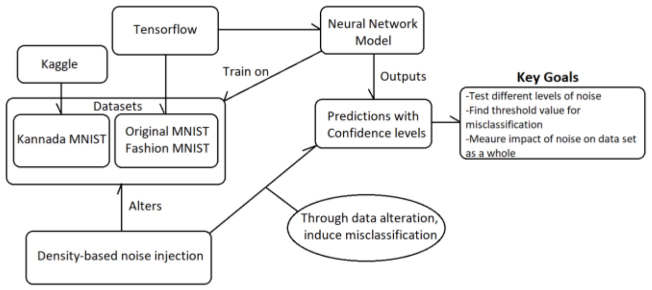
The problem I have with this paper, however, is that it seemingly takes for granted much of the core information of adversarial attacks; it revels in technical details that are ultimately tangential to the basic topic (even if they are relevant to the attack proposed by the paper), focusing more on the design and comparison of different attacks rather than examining the attacks' resultant effects on the data.

III. PROPOSED SOLUTION – PRELIMINARY STEPS

We must first decide on the kind of adversarial noise injection that we will replicate. There are two different flavors of this attack: targeted (where the input is misclassified as a specific class) and untargeted (where the resultant classification doesn't matter so long as it is incorrect). In this project, we only experimented with untargeted attacks.

There are also different kinds of noise that can be injected. The two that were used in this project are a uniform noise filter and a density-based method (we focus primarily on the latter). In the uniform filter, we use for-loops to iterate through every pixel in an image and, depending on the result of a coin flip, either add noise to or subtract noise from the pixel value. The density-based filter only alters pixels where the pixel value is above a certain threshold i.e. we only inject noise around the signal. There are three data sets used in this project, all of them rather simple but different enough from each other that we expect varied behavior across them. The samples themselves are all 28-by-28 monochrome images. Two of the sets (Original MNIST and Fashion MNIST) are included within the Keras library of Tensorflow itself, so they do not require any explicit downloading. The Kannada MNIST set, however, was retrieved from the data science competition website Kaggle.

A plethora of neural network model development tools is also included within Tensorflow. The model we used, like our data, is also quite simple: a sequential model with only two layers. Between the two data sets and model building capabilities, Tensorflow had already become an ideal library to use, but what really solidified it as central to our setup was its model prediction output style. When predictions are made by the Tensorflow model, it generates confidence levels for each of the classifications in the predictions array, making it invaluable in measuring the effects noise injection has on our model. Figure 1 is a visual representation of the how all these components and actions work with each other.



Tensorflow
Neural Network Model
Kaggle
Train on
Outputs
Key Goals
Datasets
Kannada MNIST
Original MNIST
Fashion MNIST
Predictions with Confidence levels
-Test different levels of noise
-Find threshold value for misclassification
-Meaure impact of noise on data set as a whole
Alters
Through data alteration, induce misclassification
Density-based noise injection

Figure 1. The pipeline diagram we followed in this project.

Considering the fact that each of these data sets contain 60,000 samples of training data, it became clear that only a representative few samples should be selected for experimenting purposes. Figure 2 showcases a few of the distinct input types. It is very important to note, though, that these types make up a very small percentage of the samples in the Fashion MNIST data set and are not present at all in Kannada MNIST and Original MNIST; instead, the overwhelming majority of samples across these sets are correctly classified and highly confident. It is for this reason that we primarily experiment with Fashion MNIST. We consider the other two data sets in order to see how our experiments perform on data that looks distinctly different.

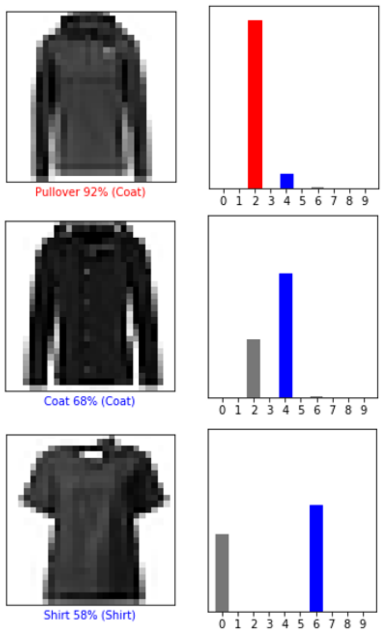


Figure 2. A representative spread of input types from Fashion MNIST pictured with no noise injection present. From left to right: confidently missed, middling confidence, low confidence.

For the following experiments, twelve specific samples are consistently used.

IV. PROPOSED SOLUTION – THRESHOLD EXPERIMENT

A. Setup

In this first experiment, we want to inject different levels of noise into the same group of samples so that we can determine if the noise factor has a threshold common to all the samples where misclassification starts. We conducted this process multiple times with different noise factors. Factors that were used include 0 (the raw image), 0.1, 0.5, 0.75, 1.0, and 1.25. For the uniform noise filter, we iterate across each pixel within each sample image, generate two random values (one that is scaled to the pixel data and then multiplied by the noise factor, the other that is the coin flip determining whether we add or subtract), and alter the pixel data. The noise injections do not propagate on one another; the pixel data is reset before every noise injection. The density-based filter is done in the same way except there is an if-condition dictating whether or not any noise is injected at all: we inject the noise only if the pixel data is above a threshold value. We tested threshold values of 10 and 40, but any value between 1 and 150 would have given the same result due to the signal and empty space of the images being of high contrast.

B. Results

For the uniform filter, the results, seen in Figure 3, were far from ideal. Not only was the learner's confidence in the classification of samples that were highly confident before injection not shaken in the slightest, the more pressing problem is the fact that the images before and after injection are highly visually discernable; it is blatantly obvious that the image was modified, making it easily identifiable.

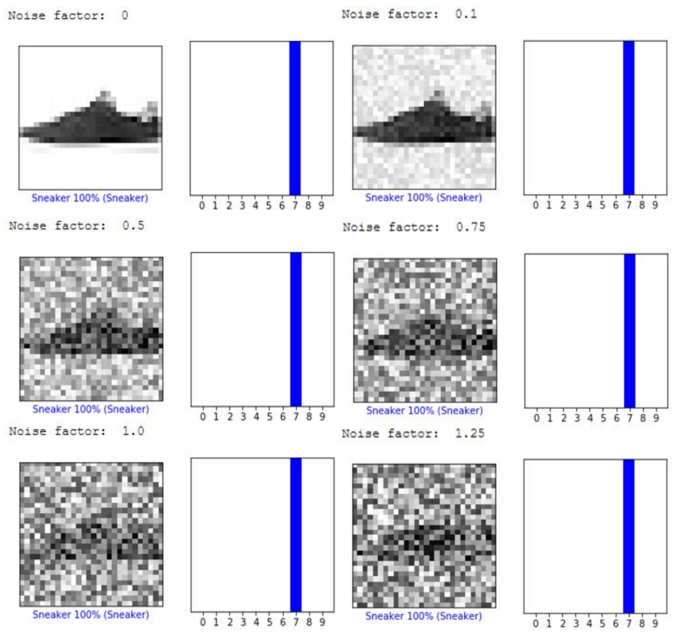


Figure 3. Uniform filter: a Fashion MNIST sample that starts and ends with a high confidence in the correct class.

Because of this, we immediately know that a uniform filter is not a sensible path to follow in pursuit of a good adversarial attack, but we will see in the next experiment how it can actually provide better insight than the density-based filter. That filter, unfortunately, only performed marginally better than the uniform one for a similarly high-confidence sample in this threshold experiment. We observe in Figure 4 a slight reduction in confidence level for a few noise factors only for it to revert to higher confidence for even higher noise factors. What makes the density-based filter much more preferable to its uniform counterpart, though, is that the images remain visually indiscernible even with a high noise factor.

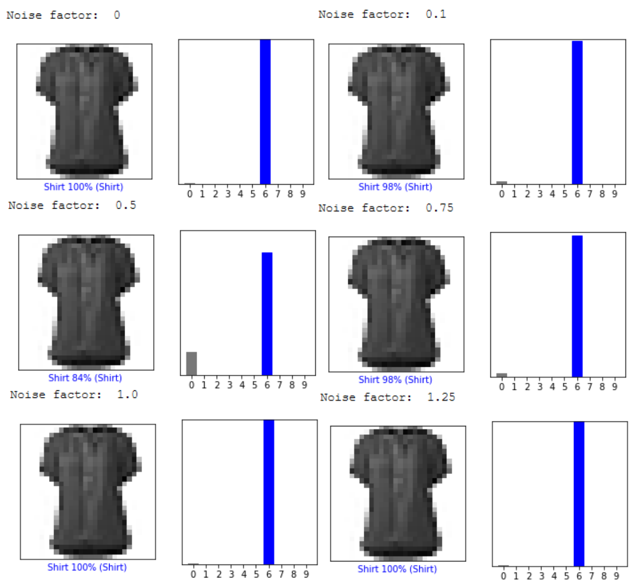


Figure 4. Density-based filter: there is some confidence variation, but the Fashion MNIST sample reverts back to high-confidence even with more noise.

Upon proceeding to test the other data sets with the density-based filter, we observe the same visual indiscernibility but an unfortunate lack of any change in the confidence levels, as seen in Figure 5. This leads us to conclude that, despite the differences from the Fashion MNIST set present in them, the Original MNIST and Kannada MNIST data sets will not be of particular help.

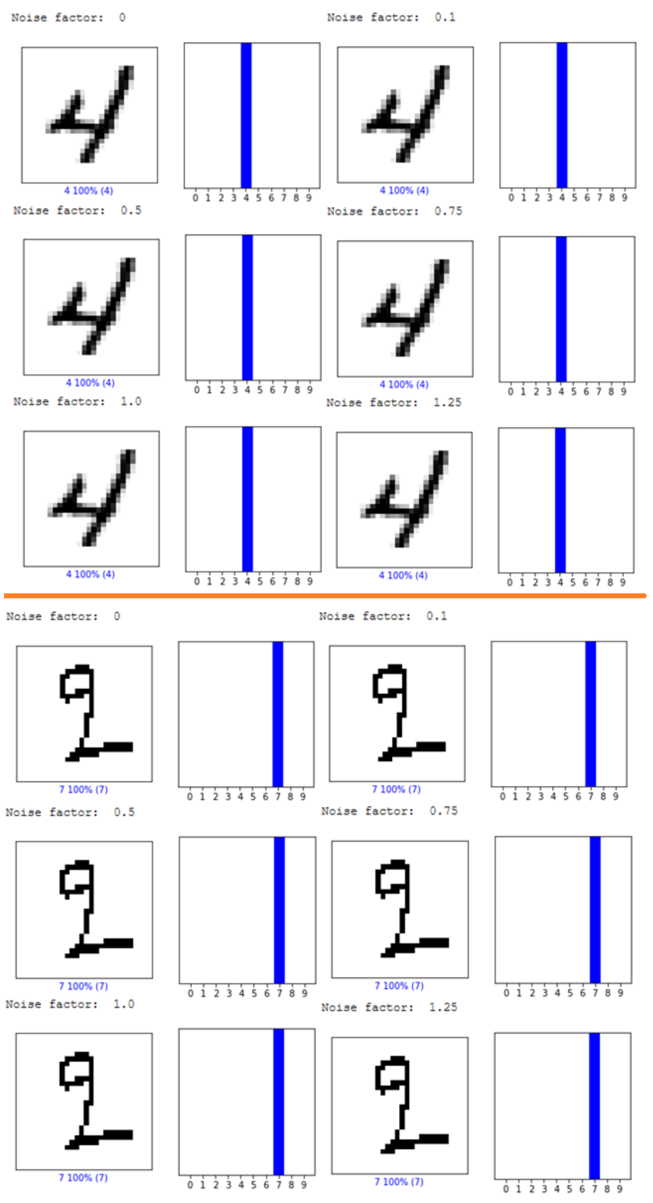


Figure 5. Density-based filter: the original MNIST (top) and Kannada MNIST (bottom) data samples remain entirely unchanged in confidence.

As mentioned earlier, however, unlike the other sets, the Fashion MNIST set has samples with varying confidence levels (recall Figure 2). Running the same threshold experiment on these samples, we observe that there is no clear threshold for either the uniform or the density-based filter. While most of the tested samples only had one factor where misclassification was observed, some samples showcased two; an example of both using the uniform noise filter is found in Figure 6. In this figure, the decisive noise factor(s) that induce misclassification are not shared across the samples: the left one reads 1.25 whereas the sample to the right has two reading 0.75 and 1.5. Also, note that, in the right sample, the two noisy images are not misclassified to the same class (surefire proof that this is not a targeted attack).

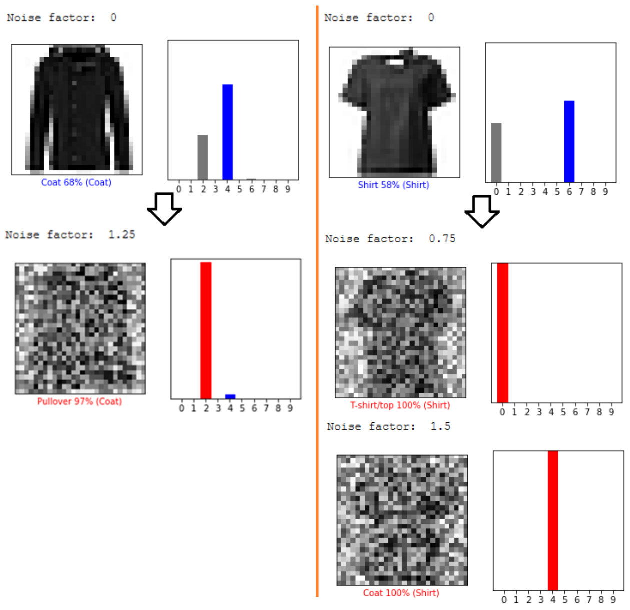


Figure 6. Two samples (separated by the orange line) of the Fashion MNIST data set, both with middling confidence levels in the raw image (top of each column), and misclassified with the uniform noise filter.

The density-based filter also has no clear threshold. Figure 7 demonstrates a raw misclassified sample (the leftmost one) has its confidence in the wrong class further increased with a noise factor of 1.0 whereas the rightmost sample is misclassified in, again, for two different, consecutive levels starting at 0.75. Interestingly, however, across both of these specific samples, the 1.0 noise factor induces misclassification, but the threshold theory does not hold because one of them misclassifies at a lower factor. The density-based filter also shows slight signs of progression as more noise is added; in most of the samples tested, the misclassification reverted to correct classification with even higher noise levels (or sometimes even corrected a misclassified sample, an occurrence also observed with the uniform filter), but these two samples were chosen in part because they showcase how confidence in the wrong class is actually strengthened as noise increases. This behavior of showing misclassification for one or more levels of noise but demonstrating inconsistency and a lack of any clear, noise-related reason was unfortunately quite common across our experiments, more so for the uniform filter as opposed to the density-based one. In Section V, we will take a closer look at what the reasons for this behavior may be.

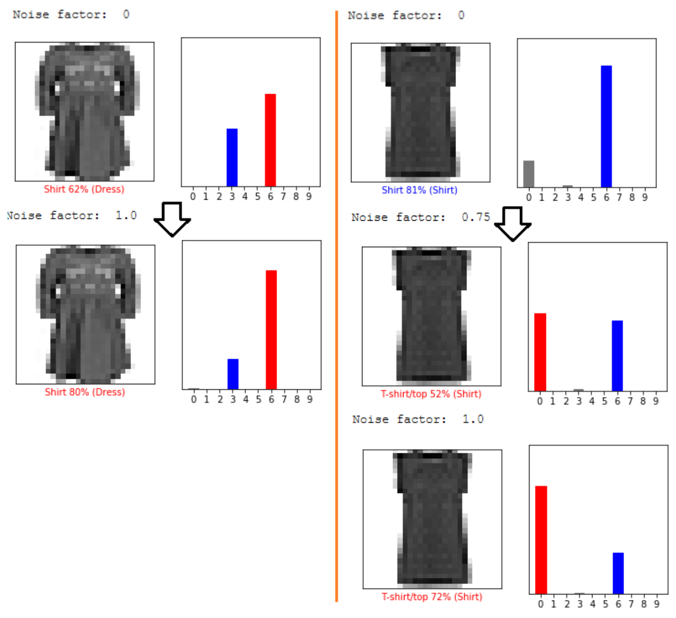


Figure 7. Two samples (separated by the orange line) of the Fashion MNIST data set misclassified with the density-based filter. The noisy images remain visually indistinct.

IV. PROPOSED SOLUTION – IMPACT EXPERIMENT

A. Setup

The impact experiment was heavily based on Section III's threshold experiment, but required some specific changes. Seeing as the objective is to observe changes within a large number of samples for a common noise filter, we ended up doing away with the iteration over a collection of noise factors in favor of a filter generator step that preceded the iteration through the twelve chosen samples. In this step, we generate and store random values to act as the scaled additive/subtractive factor for an entire 28 by 28 array i.e. the random value used to modify each pixel remains the same across all samples. These values are also modified with the tested noise factor. While generating the random values, we also generate coin-flip results for each pixel. Note that many of these values (both the pixel-altering and coin flip ones) would remain unused in the density-based filter due to its conditional modification of the images.

B. Results

Figure 8 shows a comprehensive look at the raw images of all twelve samples used in this experiment. The visible results for the density-based and uniform filters are seen in Figures 9 and 10, respectively.

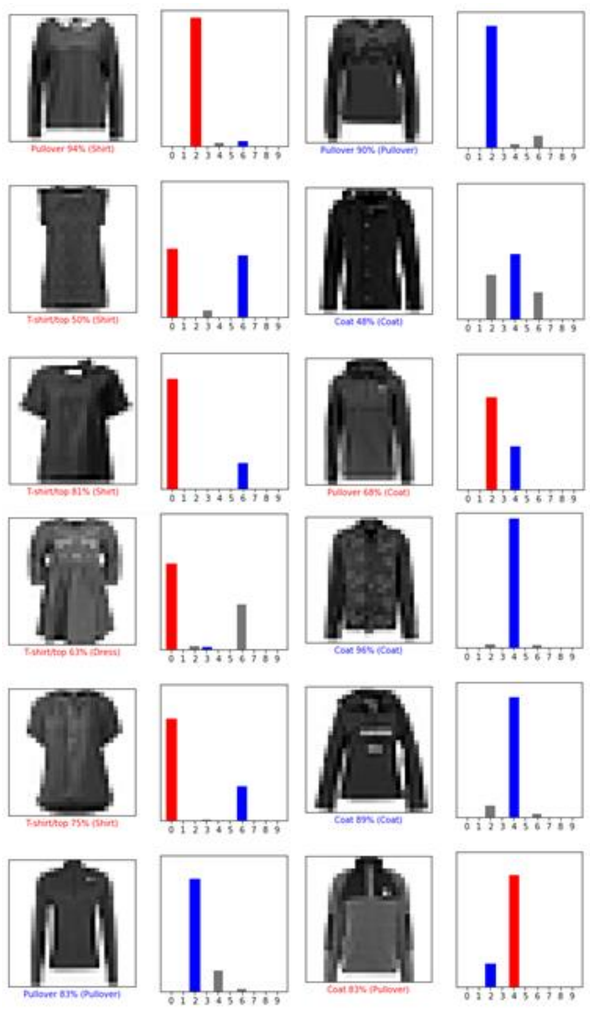


Figure 8. The raw Fashion MNIST samples used to test the impact experiment.

Starting with the density-based approach, we come to the unfortunate outcome of the noise injection having little effect. In the last section, we explored what amounted to just about the only cases where misclassification was induced with the density-based approach. In Figure 9, we see just how ineffective this method of noise injection was; for all the samples, the confidence levels in the correct classification were not weakened, but in fact strengthened to the point that only one of the samples was misclassified in the example of the noise factor being equal to 1.0. Larger noise factors were just as ineffective in inducing misclassification.

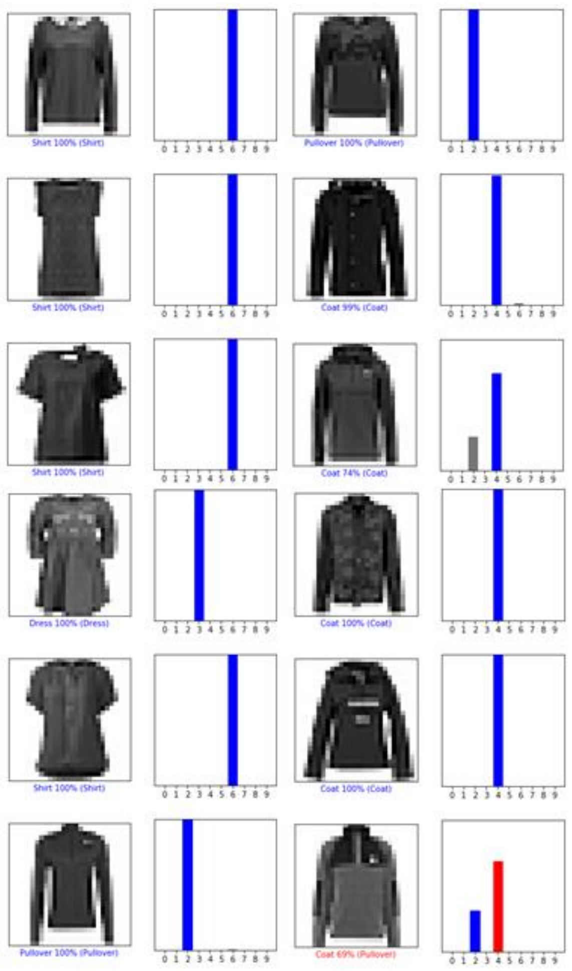


Figure 9. The same samples from Figure 8 but modified with the density-based filter using a noise factor of 1.0.

The uniform filter proved to suffer from similar problems of generally strengthening confidence levels in the correct classes. There are, however, some notable exceptions to be observed in Figure 10. Specifically, the rightmost indicated image demonstrates a mostly-effective adversarial attack while a couple of other images showcase a decrease in confidence, even when starting from a high confidence level. Unfortunately, though, most of the images were merely strengthened by the noise, and the leftmost indicated image even provides an example of a misclassified image being classified correctly after the noise injection.

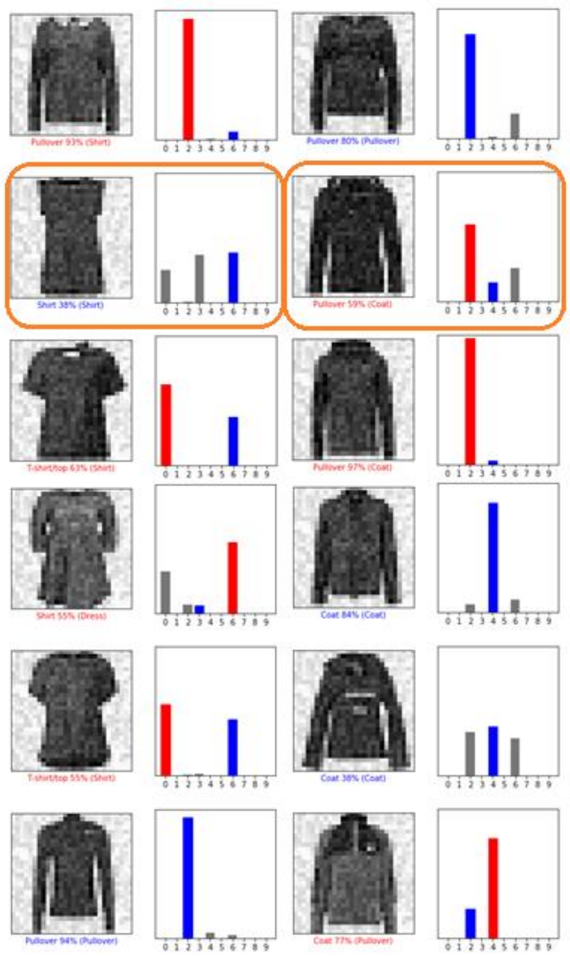


Figure 10. The same samples from Figure 8 but modified with the uniform filter using a noise factor of 0.1. Take note of the two indicated samples and how their confidence levels compare to those of the raw images seen in Figure 8.

V. ANALYSIS AND INTERPRETATION

Having completed our experimentation and after reflecting upon our results, we believe the impact experiment to be most comprehensively telling, providing clear evidence that any kind of noise injection cannot be expected to have the same effect across all samples of a data set. Figure 10 showed this very well, allowing us to observe that with the exact same noise filter applied, some samples demonstrated reduced confidence while others showed a higher confidence in the correct class, both as a direct result of the noise. Furthermore, Figures 3, 4, and 5 all demonstrated the most common occurrence: even with high noise factors and across different noise application methods, the overwhelming majority of the samples had confidence levels that remained mostly unchanged. It is clear, then only samples with lacking confidence can be considered more susceptible to adversarial attacks, but even that much is not guaranteed; Figure 9 did an excellent job of showing that even with samples holding low confidence, the noise that is injected can have only a strengthening effect.

Figure 9 also showed how poorly the density-based noise filter performed. This result leads us to ask questions regarding where exactly within images we should be injecting the noise. We can consider the Figure 3 sample of a sandal and compare it to the Figure 4 sample of a shirt, itself very similar to other classes such as dresses, T-shirts/tops, pullovers, and coats. If we want the sandal to not look like a sandal, then maybe we would want to put noise in places that would make it look more like those other classes. With this style of noise injection as the objective, it may then be easier to in fact implement a successful targeted attack rather than the untargeted attack with which we ultimately went.

We unfortunately also observed in Figure 5 just how unhelpful two of three data sets used in this project really were; the Original MNIST and Kannada MNIST data sets were utilized to demonstrate the noise filters' effects on data that looked different from the Fashion MNIST set, but the classes within them turned out to be perhaps too visually distinct (a far cry from the frequently similar-looking classes of Fashion MNIST), and this distinction is likely a key factor in why there are such consistently high confidence levels in every image that was used. Furthermore, this similarity between classes in Fashion MNIST is of particular note as it may just explain why, in examples like Figure 6, different levels of noise caused misclassifications that were highly confident but extremely varying with regard to which class the learner thought the given sample belonged. We noticed that in cases where these widely varying misclassifications took place, the classes in question were all of the coat, pullover, dress, shirt, or T-shirt/top classes, never sneakers, sandals, or anything else.

VI. CONCLUSION AND FUTURE WORK

With all of these results in mind, it can be safely said that different samples respond to noise differently. We therefore have to conclude that a threshold noise factor value that holds for an entire set is ultimately unrealistic, or at the very least quite improbable. In order to maximize impact for each level of noise, it may be required to alter where the noise is injected, as explained in the previous section. However, altering areas of an image where there is no signal was attempted in the uniform noise filter and it broke the cardinal rule of adversarial attacks by not remaining visually indiscernible from the raw images.

Of course, we can look at that statement differently; the problem may instead lie with data that is used rather than the implemented noise injection method. Essentially, all of the data used in this project having some kind of unused whitespace in the image can be viewed as what is problematic, and that is definitely true within the context of this study, but is likely also true outside of it as well; we lose real-world applicability by utilizing overly simplistic data sets. The decision to use such simple data was made early on in the project and was borne primarily out of inexperience with the subject matter and (to be a bit harsher on ourselves) an overall naïveté. It was meant to be used as more of a slow introduction to implementing adversarial attacks, and we unfortunately did not begin looking to more complex and higher resolution color images until much closer to the deadline. Clearly, these experiments should be further tested but with more complicated data sets, and we have already started working with a data set of colorful, 128 by 128 images of flowers found on Kaggle. Due to these images being of the RGBA rather than simply monochrome, there are still some logistical matters to sort through, but the prospects of using a more complicated data set are exciting. More complex models may also be worth using, if only to stay truer to real-world applications, and this may also apply the kind of noise filters we design.

Lastly, based off of what we learned using Fashion MNIST, an interesting conclusion can be drawn from its classes' similarities that we think very much applies to more complicated data as well. Ultimately, we believe the varying nature of the misclassifications across different noise filters for the same samples can be attributed to a combination of the aforementioned similarity between classes and the over-simplistic nature of the data (being very low resolution and monochrome). This leads us to a somewhat complicated conclusion; in theory, it may be easier to more widely impact the data set with a given noise filter if that data operates within a much smaller range of values. Such an impact was expected of the density-based filter as the high-contrast nature of the simplistic data focusing less on the subtle changes to the image signal and more on the uniform filter's wanton noise covering empty space. Unfortunately, this was not observed of the density-based filter, likely because the images themselves are simply too small to really accommodate the scale necessary for the learner to pay attention to a larger wave of subtler changes in and around the signal. The density-based filter was ineffective because it was not able to alter the image enough while staying hidden.

REFERENCES

[1] Tom B. Brown, Martín Abadi, et al, "Adversarial Patch". *arxiv.org/abs/1712.09665*. 2017.