Adversarial Attacks used to Fool Image Recognition Software

Nicholas Bach
nbach@wustl.edu

Abstract

By studying adversarial images and how they are made we gain a much deeper understanding of how deep learning algorithms work. I assesed the adversarial methods, (1) fast gradient sign method (FGSM), (2) iterative least-likely class method (ILCM), and (3) a novel method, through three facets: visual analysis of FGSM, how effective the novel method is at reducing confidence levels, and fooling rate. Each analysis gave different insights into how each method works and how effective each method is. Though adversarial images are a threat to deep learning software, with more research into the topic it is likely that software will become more robust and defend against adversarial methods more successfully.

1 Introduction

When neural networks were first being developed, many looked at their inner workings as unknown black boxes. In regards to computer vision these algorithms learned their tasks very well, but people weren't exactly sure what features these networks were using to learn. The networks are certainly not without faults. Humans develop them, and sometimes human biases wind up manifesting in software that they don't belong in. Articles[3][15][6] are consistently coming out studying racial and demographic biases that have made there ways into facial recognition software, for instance. Even without human biases, image recognition can make errors where humans wouldn't. As shown in the following paper, adversarial algorithms can be used to create images that a human would always label correctly (given they know the object in the picture), but the software classifies the image as an incorrect label. By studying these adversarial images and how they are made, I believe we gain a much deeper understanding of how the computer vision software works. I believe it is also important to understand adversarial attacks for the purpose of defense, because while all adversaries created for the purpose of this paper were created through algorithms, they can also be produced in the real world. Eykholt et al. [4] calculated an adversarial design that, when graffitied onto a stop sign, fooled a classifier in 84.8% of the video frames captured from a moving vehicle. This could obviously be very dangerous if a computer vision driven car is fooled into thinking that a stop sign is not actually a stop sign. Adversarial attacks are not always insidious, however, and can sometimes be used for one's own protection. As The New York Times reported[10], for instance, facial recognition software has been used in China to track and control Uighur Muslims. While not used to target minorities in the U.S. (at least explicitely), police here have also used facial recognition software to find criminals[14], and that comes along with the demographic and racial biases alluded to earlier. I believe developing adversarial methods will be necessary to protect individual privacy as facial recognition software become more and more widespread.

2 Background & Related Work

2.1 First exploration into topic of adversarial images

Modern research into adversarial attacks began with a paper written by Szegedy et al. in 2014.[12] This paper made many discoveries about what it called "adversarial examples". It found that a perturbation, p, could be added to an image and cause image classification neural networks to classify the image completely differently. This can be written with the following equation:

$$\tilde{x} = x + \epsilon p \tag{1}$$

where x is the original image and $\tilde{x}$ is the misclassified image. The ϵ term is a multiplier applied to p, and if we minimize ϵ such that x does not equal $\tilde{x}$, we can get an adversarial image that is equivalent to the original to the human eye. What is even more interesting about the adversarial images created by Szegedy et al. is that they applied to a wide variety of image classification algorithms trained on different training sets. This led the authors of the paper to speculate that the nonlinearity of neural networks and overfitting were the reasons for their vulnerability to adversarial attacks.

2.2 Fast Gradient Sign Method (FGSM)

The next year, Ian Goodfellow et al.[5] argued that the primary cause for this vulnerability was actually neural

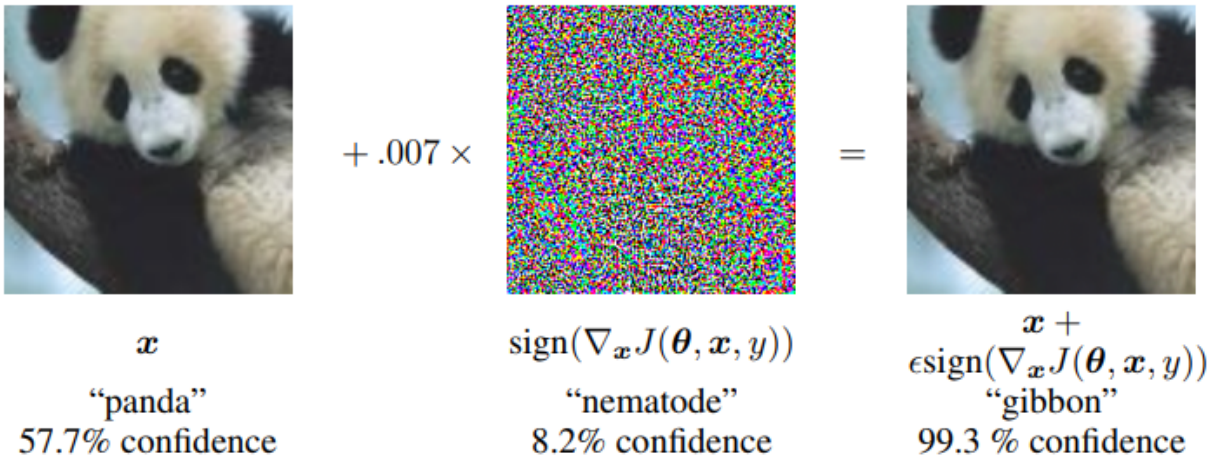

Figure 1: An example of a perturbation generated through FGSM applied to GoogLeNet on ImageNet. The ϵ of .007 in this example corresponds to the magnitude of the smallest bit after GoogLeNet's conversion to real numbers. This minimal addition, imperceptable to humans, is incredibly effective as it leads to the classifier misclassifying the image with over 99% confidence.[5]

networks' linearity as opposed to their nonlinearity. They showed this by explaining the following: consider the equation $\tilde{x} = x + \eta$, with the same variables as equation 1 but η substituted in for ϵp. Because pixel intensity precision is limited, if every element of η is smaller than that precision, a classifier should classify $\tilde{x}$ the same as x. Now consider the dot product between a weight vector w and $\tilde{x}$:

$$w^T \tilde{x} = w^T x + w^T \eta. \tag{2}$$

The activation now grows by $w^T \eta$, which is maximized if $\eta = sign(w)$. If w has n dimensions and an average magnitude of m, the activation will grow by $||\eta||_\infty mn$. This means that while η does not grow with dimensionality, the change in activation caused by η does, so if n is large enough many imperceptibly small changes to the image can result in one large change to its classification. Goodfellow et al. also came up with the first method of generating computationally cheap adversarial perturbations that should affect any linear model as well as neural networks designed in linear ways, such as LSTMs, ReLUs, and maxout networks. This method goes as follows: θ represents the parameters of a model, x is the input, y is the label associated with x, and $J(\theta, x, y)$ is the model's cost. η is calculated by taking the sign of the gradient with respect to x of the cost function and multiplying it by ϵ, i.e.

$$\eta = \epsilon sign(\nabla_x J(\theta, x, y)). \tag{3}$$

This is called the fast gradient sign method (FGSM), and can be done very quickly through backpropogation. By setting the perturbation to the sign of the gradient of the cost function, we are able to find a perturbation that is very effective at reducing the model's confidence in the current classification of an image. Figure 1 shows just how effective this algorithm can be, and even though this was one of the very first methods to generate adversarial images developed, it remains incredibly effective to this day. In 2018 Kurakin et al.[8] found FGSM has a fooling rate of about 63-69% for $\epsilon \in [2, 32]$ when tested on the very popular object recognition software ImageNet[2].

2.3 Fooling Rate

Fooling rate is the percentage of images that are expected to have their classification flipped by a certain adversarial method. This is also called the top-1 error rate, whereas the top-5 error rate (another popular metric) is defined as the percent of images that don't have the correct classification in their top 5 labels. As Mopuri et al.[9] note, this rate isn't the most effective way of describing an adversarial method's effectiveness in all cases. For instance, FGSM is very effective at changing the classification of an image, but in its simplest implementation it is untargeted so it has no control over what the classification changes to. If FGSM changes the ImageNet classification of a golden retriever to a saluki, as it did in my own implementation, that would count towards the fooling rate, but it wouldn't have a very large practical effect because both are similar looking dog breeds. This is usually dependent on how distinct a certain models classifications are, so when it comes

to ImageNet with 118 different dog classifications, its very likely that FGSM applied to a picture of any dog breed will result in it being classified as a different dog breed. Mopuri et al. go on to describe other metrics that may be more effective measurements which are tangential to the scope of this paper. That being said, it is important to keep in mind that while some methods may have lower fooling rates than their peers they may still have important practical applications.

2.4 Iterative Lest-Likely Class Method (ILCM)

Similar in premise to FGSM, the iterative least-likely class method was developed in part to combat the fact that FGSM led to very similar classifications. Created by Kurakin et. al.[7] in 2017, the ILCM uses the gradient with respect to x of the cost function, but in this case it's the cost function with input image x and a desired label y as opposed to the true label y, as follows:

$$X_0^{adv} = X,$$
$$X_{N+1}^{adv} = Clip_{X,\epsilon}\{X_N^{adv} - \alpha sign(\nabla_x J(X_N^{adv}, y_{LL}))\}. \quad (4)$$

The adversarial image is initialized to the original image, and each iteration adds the sign of the gradient with respect to x of the current image and the desired label until the number of iterations times alpha equals epsilon. In this way the resulting adversarial image differs from the original by the same metric as FGSM (ϵ). α is usually set to one, meaning pixels can only change by one intensity value during each iteration. y_{LL} could be any desired label, but Kurakin et. al. used the least likely label of the original image to get the most interesting results. Note that while FGSM adds the sign of the gradient the original image, ILCM subtracts the sign of the gradient because it aims to pull the image towards a desired label as opposed to pushing it away from the current one.

3 Proposed Approach

3.1 FGSM

I first wanted to use FGSM to develop some working examples of adversarial attacks. Following a tutorial[1], I created a program to run FGSM on any input image and to classify it with ImageNet classifiers with TensorFlow's MobileNetV2 architecture. I then wanted to test the performance of the calculated perturbations compared to random perturbations. I generated random perturbations by using random labels to calculate the gradient of the cost function instead of the correct labels. Figure 2 shows some examples using the correct labels, and figure 3 shows the same examples using random labels. I then wanted to explore the actual layers of the model to find where the adversarial examples began looking different from the original image. I could not access the individual layers of the MobileNetV2 model, so this required me to train my own model. Adapting the code provided by the author of [13], I created and trained a model to parse handwritten numbers and label them as digits zero through nine. I then took inspiration from the author of [11] to access each layer of the model I had created. Because the model acepts 28 by 28 pixel images as inputs, by the second convolution the individual elements of the layer were 4 by 4 pixel images. This means that by the second layer, the visualization already becomes relatively meaningless to humans, as shown in figure 4. Figures 7 and 8 show the first layer (not including the input as a layer) with a handwritten five as the input, before and after the perturbation was added respectively. In this case, the classifier correctly identified the image as a five before the perturbation was added but misclassified the image as a three after the perturbation was added. Figures 9 and 10, on the other hand, show the first layer before and after a perturbation was added to a handwritten zero, but in this case the classifier correctly identified the zero before and after the perturbation was added.

3.2 ILCM

I first adapted the FGSM code from section 3.1[1] in accordance with the math from Kurakin et al.'s paper [7] to achieve a functioning iterative least-likely class method. Using this code, as shown in figure 5, the classifier was fooled into thinking an image of a golden retriever was an image of a llama and an image of a panda was an image of a conch, both with high confidence. I wanted to compare this to a novel method with a very slight difference to ILCM, where instead of setting each iteration's label to the least-likely classification of the overall input image, I set each iteration's label to the least-likely classification of that specific iteration's input. I did this because I wanted to reduce the classifier's confidence in general, instead of pushing it towards a specific label. I figured from a defense standpoint, there could be tell-tail signs of ILCM that would not exist in the novel one. For instance, after a certain amount of iterations Kurakin's algorithm resulted in confidence levels of over 99% in a certain label, but I never saw confidence levels this high using real-world images. This could lead someone to immediately suspect data with extremely high confidence levels because those levels rarely happen in real-world examples but are easy to achieve through manipulation. By changing the label each iteration, however, I predicted that confidences levels for all labels would go down. If confidence levels all decrease evenly, a classifier might still classify the image correctly.

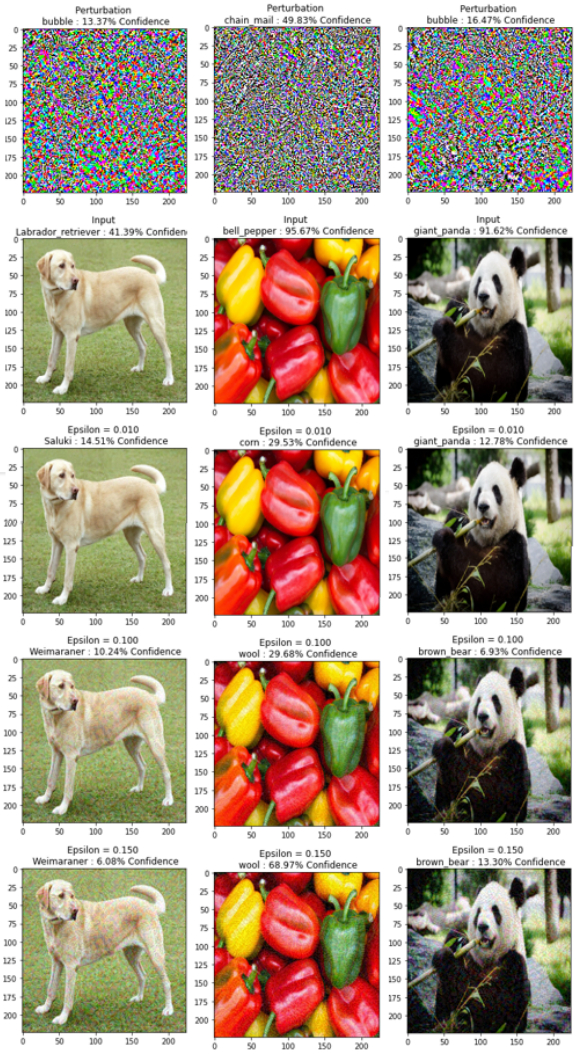

Figure 2: Examples of FGSM being used at different epsilon values and their respective affects on the classification of the image.

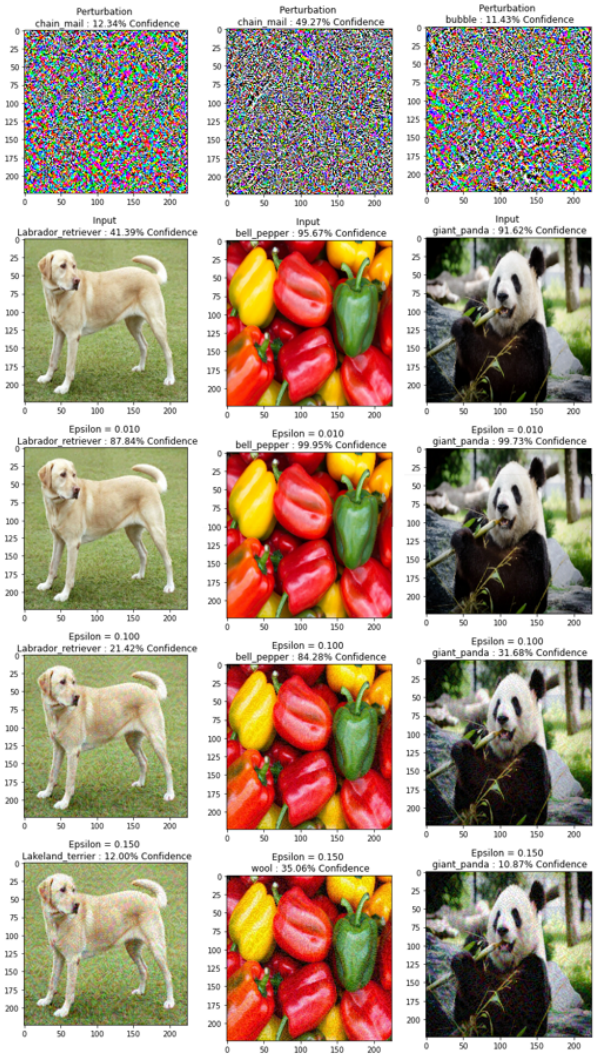

Figure 3: Examples of a random perturbation being used at different epsilon values and their respective affects on the classification of the image.

This would have less practical meaning, however, because the difference in its confidence in that label compared to any other label would decrease as well.

3.3 Fooling Rates

Different studies have calculated fooling rates for these methods, but I wanted to calculate my own fooling rate as well. To do this, I tried to be as scientific as possible by using a random number generator to find 20 different ImageNet labels, then I googled these labels and took the first image from google images. I then tested all 20 images and replaced the ones that my classifier misclassified through the same method. I did this using FGSM, ILCM, and I used a random perturbation as the control. In each case I set ϵ to 0.05 in order to keep the algorithms comparable. This graph is shown in figure 6. I would have liked to calculate fooling rate for the novel method described in section 3.2, but I had no way of computationally calculating the index of a specific ImageNet label so I would have had to manually track down and input hundreds of labels.

4 Experimental Results

4.1 Qualitative Layer Analysis

Starting with figure 7, a five is still discernible in most of the images. Some of them emphasize different aspects of

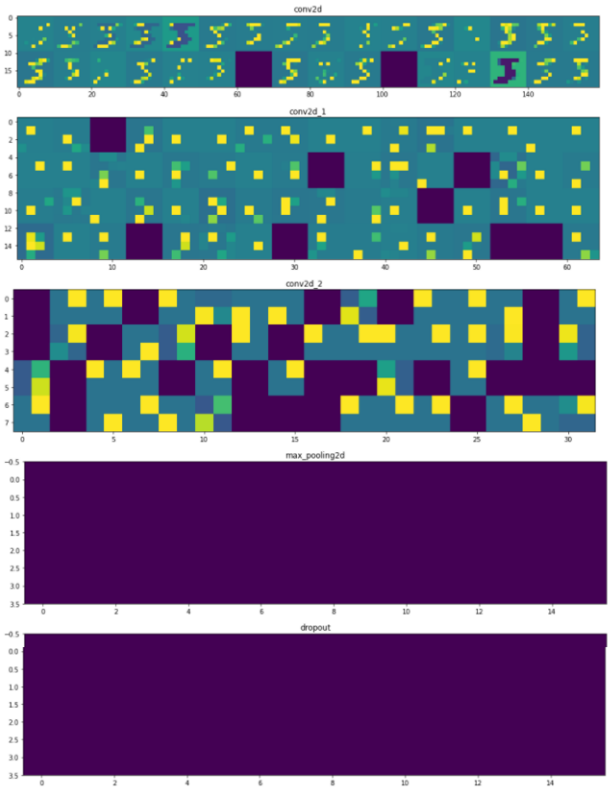

Figure 4: Visualization of every layer of the model. The top layer looks like the original image, but from the second layer on this visualization largely becomes meaningless.

Figure 5: Examples of ILCM being used on two target images. After only four iterations the classifier has a 99.92% confidence in the image of a golden retriever actually being an image of a llama. After eight iterations, the classifier has a similar 99.96% confidence in the imade of a panda actually being an image of a conch.

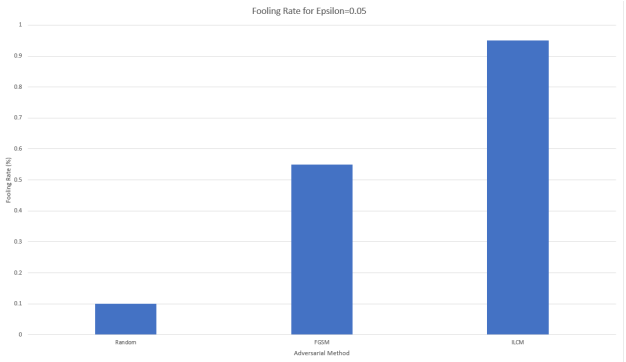

Figure 6: Fooling rates of FGSM, ILCM, and a random perturbation. Out of the twenty images tested, the random perturbation successfully fooled the classifier two times, FGSM was successful eleven times, and ILCM was successful nineteen times.

the five, such as the top flat part or the concave bottom, but I can make out a five in about twenty out of the thirty-two total images of the layer. Now compare that to figure 8, which is the same layer of the image once the perturbation has been added to it. Fives are still discernible in some of the images, but far less than the previous example. I would call about eleven of these fives, a decrease of almost half. What's interesting is that none of them look like threes, even though I know the final label of this image is a three. I can point to certain parts of some of the images as three-like, but before I knew it would be labeled as a three I could not figure out the final misclassification based on this layer. Compare the previous two figures to figures 9 and 10, which depict a adversarial image with the same ϵ value but that was unsuccessful in fooling the classifier. The majority of the images created by the original input are distinctly zeros, and even after the perturbation that remains true. Just as figure 8, images appear after the perturbation is added that are extremely noisy. In contrast to figure 8, however, most of the images in figure 10, including some of these noisy ones, retain their round edges and shape, aspects that would still lead the classifier to the label of zero.

4.2 Novel Method

Figure 11 shows the confidence the classifier has in the original input after each iteration, as well as the difference between that confidence and each of the second through fourth other top labels. The lines are all incredibly close because these confidence levels all change at nearly the same rate. The top label never changed in this case, and the classifier always classified the image correctly, but by

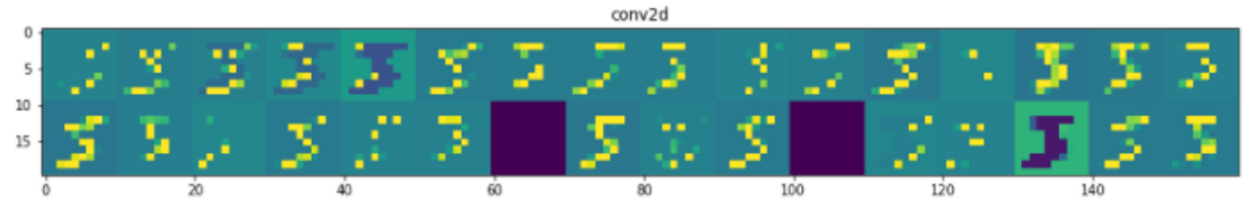

Figure 7: Visualization of the top layer of the model. The input in this case was an unperturbed image of a handwritten five.

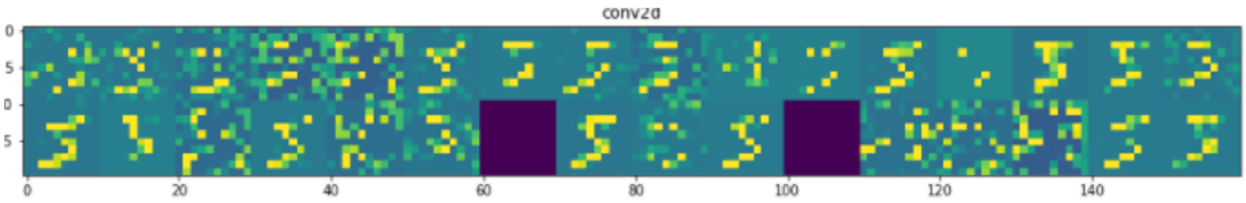

Figure 8: Visualization of the top layer of the model. The input in this case was an image of a handwritten five with an FGSM claculated perturbation added. The classifier misclassified this image as a three.

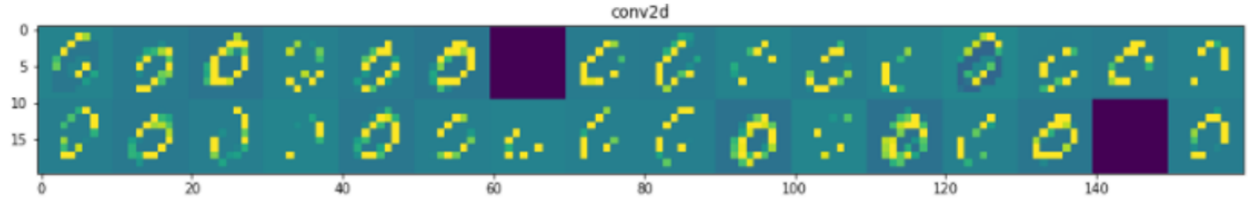

Figure 9: Visualization of the top layer of the model. The input in this case was an unperturbed image of a handwritten zero.

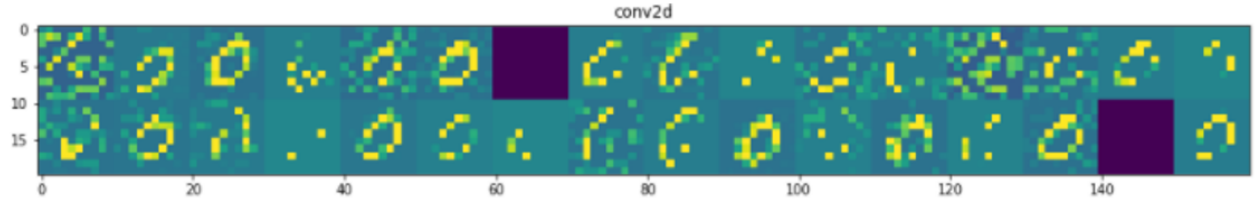

Figure 10: Visualization of the top layer of the model. The input in this case was an image of a handwritten zero with an FGSM claculated perturbation added. The classifier classified this image correctly as a zero even with the perturbation added.

the third iteration the top label had a sub-1% confidence associated with it. I believe this method could be most effective when used in tandem with another method. For instance, it could be used before a targeted method to increase the effectiveness of that targeted method. If a classifier has very low confidence in every label, it will be much easier to convince the classifier of a certain label when the targeted method is used to push the classifier towards that label.

4.3 Fooling Rates

Looking at figure 6, we can see the FGSM performs very well compared to a random perturbation. I obtained a fooling rate of 55%, which is lower that the rate estimated by Kurakin et al.[8] in 2018 of about 63-69%. I believe this can be attributed to the method I used to choose my photos. I noticed while running tests that FGSM had a harder time confusing the classifier if the image being used was an object in front of a white backdrop. I think in these cases the classifier has very high confidence in the correct image label and very low confidence in all incorect image labels. When the FGSM calculated perturbation is added, the classifier loses confidence in the correct label, but its confidence in every other label is so low it still classifies the image correctly. ILCM performed incredibly well, only failing to fool the classifier once out of the twenty times I tested it. This exemplifies how susceptible image recognition software can be to targeted attacks, however it is important to note that in five of the nineteen successful cases the classifier did not label the image as the targeted least-likely label. The classifier still misclassified these, however, so they count towards fooling rate as it's defined. I was unable to test the fooling rate of the novel method, but I would have expected the novel method to have a fooling rate similar to the fooling rate of random perturbations. This is because the goal of the novel method is to reduce confidence in all labels. As shown in section 4.2, it ends up reducing all confidences relatively constantly, so often the classifier will be most confident in the correct classification by the end even if its confidence in that label is below 1%.

5 Conclusion

This article presented some of the earliest methods used to create adversarial images. By understanding why adversarial images exist and how they are formed, we are better able to understand the underlying neural networks, and we are better able to defend these neural networks against such attacks. It is clear that adversarial attacks can cause a great practical threat to deep learning, and could be detrimental to the computer vision powered future of self-driving cars. Hopefully, with further research into the area, deep learning networks can become more robust. This article also develops a novel method which has practical applications that are different from the closely related FGSM and ILCM. More specifically, this novel method could be used in tandem with other trageted adversarial software to make them more effective.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, 2015.

[2] N. Akhtar and A. Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *CoRR*, abs/1801.00553, 2018.

[3] J. G. Cavazos, P. J. Phillips, C. D. Castillo, and A. J. O'Toole. Accuracy comparison across face recognition algorithms: Where are we on measuring race bias? *IEEE Transactions on Biometrics, Behavior, and Identity Science*, pages 1–1, 2020.

[4] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song. Robust physical-world attacks on deep learning models, 2018.

[5] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. abs/1412.6572, 2015.

[6] B. F. Klare, M. J. Burge, J. C. Klontz, R. W. Vorder Bruegge, and A. K. Jain. Face recognition performance: Role of demographic information. *IEEE Transactions on Information Forensics and Security*, 7(6):1789–1801, 2012.

[7] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world, 2017.

[8] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial machine learning at scale. abs/1611.01236, 2017.

[9] K. R. Mopuri, V. Shaj, and R. V. Babu. Adversarial fooling beyond "flipping the label", 2020.

[10] P. Mozur. One month, 500,000 face scans: How china is using a.i. to profile a minority. *The New York Times*, 2019.

[11] G. Pierobon, 2018.

[12] C. Szegegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. abs/1312.6199, 2014.

[13] S. Theiler, 2019.

[14] J. Valentino-DeVries. How the police use facial recognition, and where it falls short. *The New York Times*, 2020.

[15] M. Wang, W. Deng, J. Hu, X. Tao, and Y. Huang. Racial faces in the wild: Reducing racial bias by information maximization adaptation network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

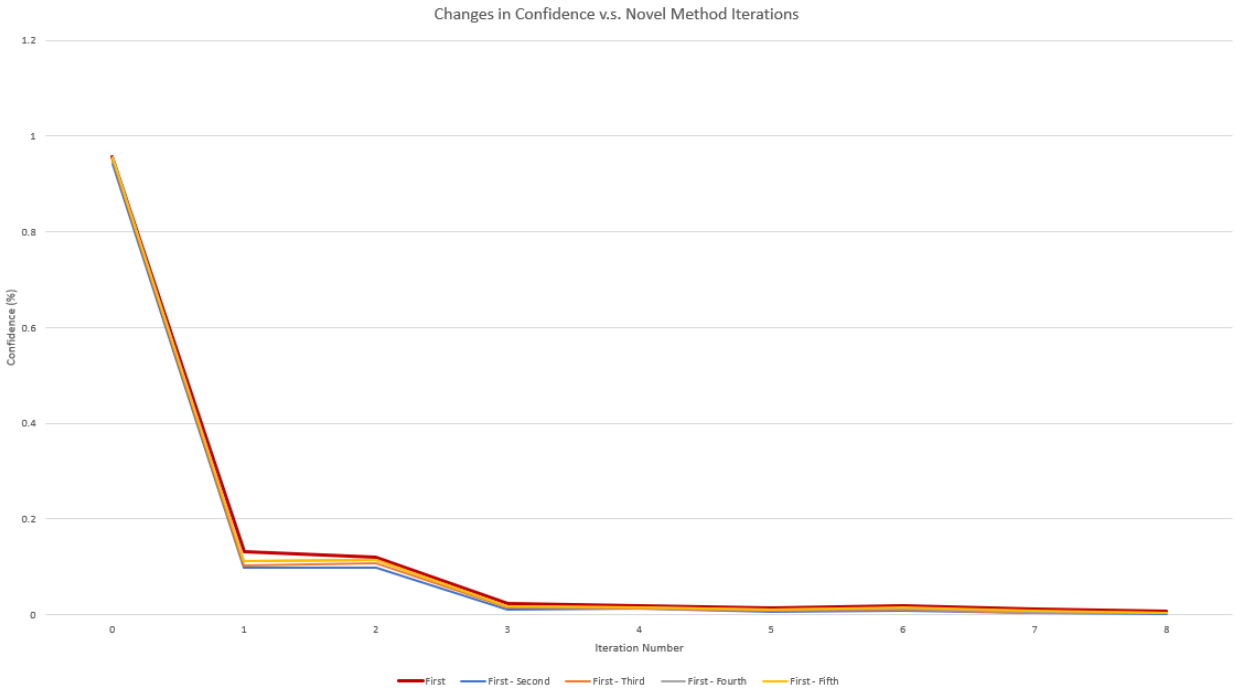

Figure 11: Plot of the confidence levels of the input and after each of eight iterations of the novel method. Though all five lines are extremely close together, the lines represent the confidence in the correct label and the differences between the correct label and each of the other top four labels. After three iterations' the method has reduced the confidence level of every label, including the correct one,tso under 1%.