Seminar thesis

Adversarial Label Flips (sexier title?)

Matthias Dellago & Maximilian Samsinger

June 16, 2021

Abstract

Given a neural network (NN) trained to classify and a untargeted evasion attack [], in what class does the adversarial example fall? In the following, we will answer this question by evaluating some state of the art attacks, on a simple NN trained on industry standard datasets [**?**, **?**, **?**, **?**]. We discover that intuitively similar classes are more likely to be confused with another.

1 Introduction

Adversarial examples exist. Targeted and untargeted. What does that look like? We use confusion matrices. example. how did we do it? -¿ NN + foolbox + mnist cifar (section methods) what did we get out of it? -¿ symmetric matrices and interesting insight about noise. (section results)

In 2013 Szegedy et al. demonstrated that attacking deep neural networks (NN) are susceptible to attacks [1]. These adversarial examples consist of a small perturbation applied to a otherwise innocuous input, engineered to cause the NN to misbehave.

In our case, the inputs will be images and the attack will be small changes to said image, designed to cause the classifier NN to misclassify the target.

These attacks on classifiers come in two different variations: targeted and untargeted. In targeted attacks[] the attacker aims to have the adversarial example identified as a specific class by the NN. (Say, misclassify a dog as a cat.) An untargeted attack meanwhile, only tries to evade correct classification. (Make this dog appear as anything, apart from a dog.)

Now, when considering untargeted attacks, the question as what the adversarial image *actually* is then classified, naturally arises. This is what we will experimentally answer in this paper.

We will present our results in terms of confusion matrices. In figure !! you can see an example. In larger matrices numbers become more difficult to process, so we will display our results in heatmap-style images (fig. !!). (Section results)

In our experiments we used the foolbox framework [], and simple NNs trained on the MNIST, FashionMNIST, and CIFAR-10 datasets []. We applied three state of the art attacks: Projected Gradient Decent (PDG)[], Carlini-Wagner [] and ...-Bethge[].

We show that, for the CIFAR-10 dataset the confusion matrices are surprisingly symmetric, and intuitively similar classes are often confused with each other. Furthermore we observe that for attacks which can choose large pertubations, there exist certain attractor classes, which most of the adversarial images are classified as. (Section Results.)

2 Background and related work

Existence of adversarial examples Since being first demonstrated [1] a large body of literature has flourished around adversarial examples. Blablabla...

Notions of similarity As mentioned previously we want to fool the NN with a small perturbation, that is to say a image which is very similar. But what do we precisely mean by "small pertubation" and "similar"? We could argue that an image with only one pixel changed is very similar, but also that an image in which each pixel was altered only very slightly is similar.

These different intuitive notions of distance and similarity are captured by norms.

The distance of two images (x,y) (i.e. vectors) is described by a metric D. This metric can be defined via different norms, most commonly the L^∞-, L^1- and L^2-norms.(The L^0-"norm" is also frequently used, though not a norm in the strict sense.) The distance between these the two images, is then defined as the L^p-norm of their difference, for a given p:

$$D(x, y) := \|y - x\|_p \tag{1}$$

$$D(x, y) := \|y - x\|_p \tag{2}$$

... Hmmm probably not that important, maybe if there is time and space after everything else in finished.

2.1 Attacks

Fast gradient sign method Goodfellow et al. famously developed the fast gradient sign method (FGSM) [2], making attacks fast and easy. Its key insight was that the backpropagation commonly used to update the weights and biases can be applied all the way back to the input data itself to yields the gradient of the cost function. They then apply gradient decent to find an adversarial example. Since they optimise for the L^∞-norm, all entries of the perturbance are scaled to the same magnitude.

Projected gradient descent Projected gradient descent (PGD) was first shown in [3]. Conceptually it is very similar to iterating FGSM until converging in a local misclassification optimum. The "projected" part of the name, derives from the fact that upon leaving a ball of radius ϵ instead of continuing iteration, they project back onto said ball. From iterated FGSM resumes. This attack leads to formidable results, especially using the L^∞-norm.

Carlini-Wagner attack Carlini and Wagner [4] invented a different style of attack, where the cost function of the classifier and the distance of the adversarial example are wrapped into one function. They can then simultaneously optimise for both using the Adam stochastic optimiser [5].

Brendel-Bethge attack Brendel and Bethe invented a quite different method, which they aptly named Brendel & Bethge attacks [6]. **zu spicy? :D** Their method works by starting from a image deep inside the misclassification region

and then preforming binary search between it and the benign, target image, to find the decision boundary. Once there, they move along the boundary to minimise the distance to the benign image, yielding a powerful adversarial example.

Foolbox Foolbox is a python toolbox for generating adversarial examples[7]. It supports a large collection of attacks, including all of the above.

2.2 Quantifying Symmetry

Since we found a surprising amount of symmetry in the confusion matrices (**??**), we wanted to somehow quantify exactly *how* symmetric our matrices were. Upon a review of the literature we could not find any such metric, and so we decided to improvise our own.

Give the confusion matrix A we first replace the diagonal values with zero, since these only represent failed attacks to get A_0. We then split A_0 into its symmetric and anti-symmetric constituent matrices:

$$A_{sym} = \frac{A_0 + A_0^T}{2}, \qquad A_{anti} = \frac{A_0 - A_0^T}{2} \tag{3}$$

We then define our measure for symmetry s in these terms:

$$s = \frac{\|A_{sym}\|_1 - \|A_{anti}\|_1}{\|A_{sym}\|_1 + \|A_{anti}\|_1} \tag{4}$$

The 1-norm is not special, and could be replaced with any other p-norm.

For confusion matrices (i.e. all values greater zero) s can take values from $[0, 1]$. $s = 1$ represents maximally symmetric matrices, and $s = 0$ a maximally skewed matrix.

Below some examples, to aid our dear reader in building an intuition.

$$s\left(\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\right) = 0, \qquad s\left(\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\right) = 1, \qquad s\left(\begin{bmatrix} 0 & 1 \\ 0.5 & 0 \end{bmatrix}\right) = 0.5$$

3 Experiments

In this Section we use the adversarial attacks introduced in Section **??** to compute adversarial examples on the MNIST [8], Fashion-MNIST [8] and CIFAR-10 [9] datasets. Each dataset is split into a training and test set and all adversarial attacks are computed and evaluated on the test set. Given a dataset and an attack we generate a pair of labels containing the original class and the predicted class for each adversarial example. We use i to denote the i-th original class and j to denote the j-th predicted class. Since all three datasets contain 10 classes each we obtain 10×10 confusion matrices

$$A = (a_{ij})_{\substack{1 \le i \le 10 \\ 1 \le j \le 10}},$$

where $a_{ij} \in \mathbb{N}$ corresponds to the total number of occurrences of each pair (i, j).

3.1 Experimental setup

All experiments are conducted in Python 3.8.5 [10] using the PyTorch 1.8.1 [11] library on a Windows machine. Adversarial attacks are computed using FoolBox 3.3.1 [7]. Each attack is instantiated using the FoolBox default parameters. All minimization attacks, i.e. `L0BrendelBethgeAttack`, `L1BrendelBethgeAttack`, `L2CarliniWagnerAttack` have their perturbation budget `epsilons` set to `None`. This prevents any early termination; a fixed number of iterations is used to compute the adversarial example with minimal perturbation size. For `LinfPGD` we set `epsilons` to one of the values in $[0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1]$ to cover a wide range of perturbation budgets.

Reference to figures.
Reference to our github.

3.2 Architectures

`https://arxiv.org/pdf/1608.04644.pdf` ¿¿¿¿¿¿¿ fc30d912d4fbae6a812c94b67d31111a632f83e6 ¡¡¡¡¡¡¡ HEAD

4 Results

Below we present all the confusion matrices we calculated. We add our symmetry score and the epsilon value, if applicable.

4.1 CIFAR-10

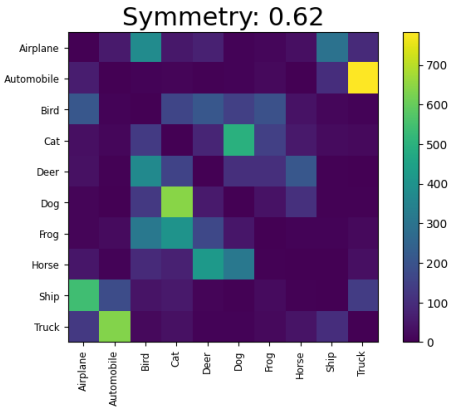
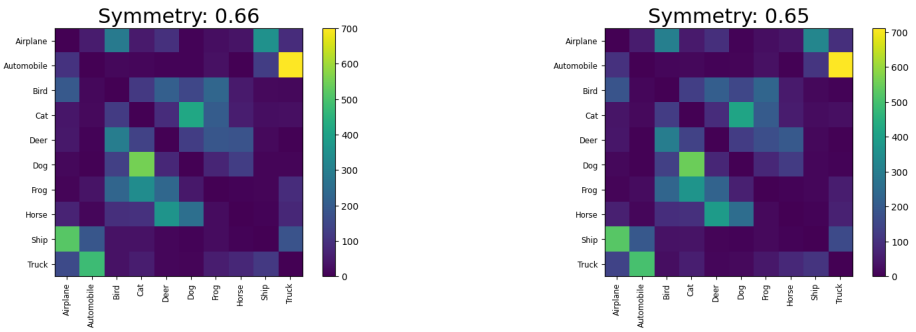


Figure 1: L^2-Carlini-Wagner-Attack on CIFAR-10



(a) L^0 (b) L^1

Figure 2: Brendel-Bethge-Attacks on CIFAR-10

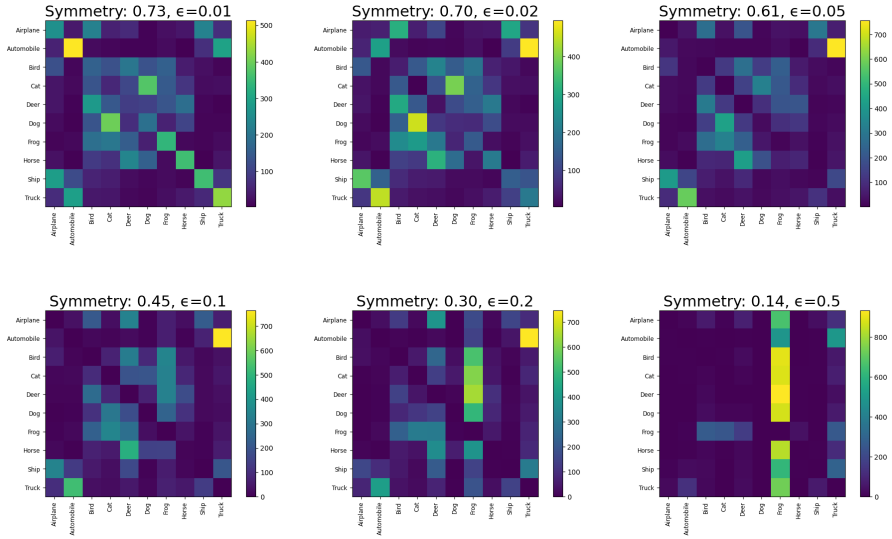


Figure 3: L^∞-PGD-Attack on CIFAR-10

4.2 FashionMNIST

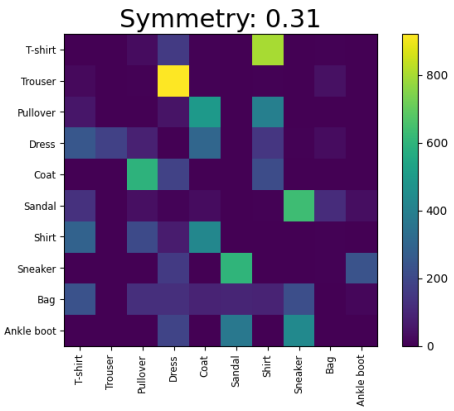
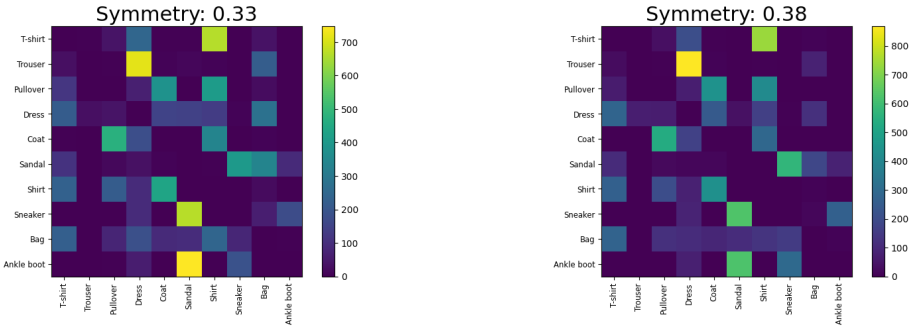


Figure 4: L^2-Carlini-Wagner-Attack on FashionMNIST



(a) L^0 (b) L^1

Figure 5: Brendel-Bethge-Attacks on FashionMNIST

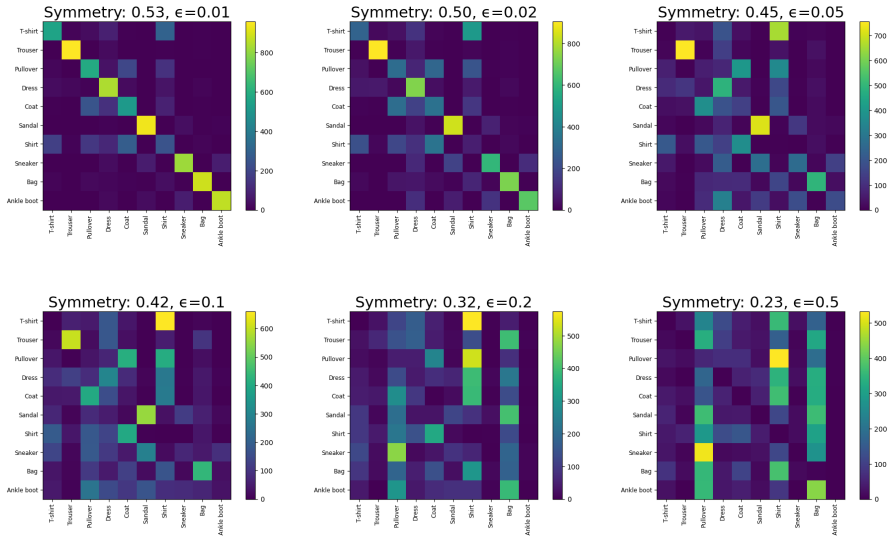


Figure 6: L^{∞}-PGD-Attack on FashionMNIST

4.3 MNIST

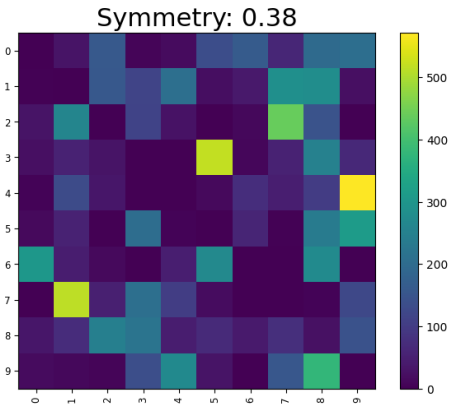
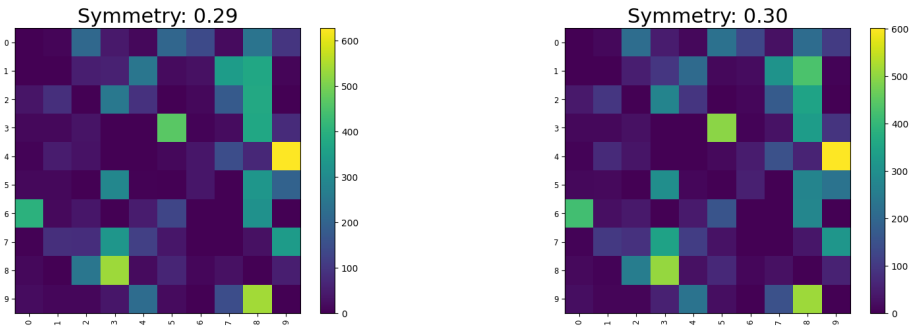


Figure 7: L^2-Carlini-Wagner-Attack on MNIST



(a) L^0 (b) L^1

Figure 8: Brendel-Bethge-Attacks on MNIST

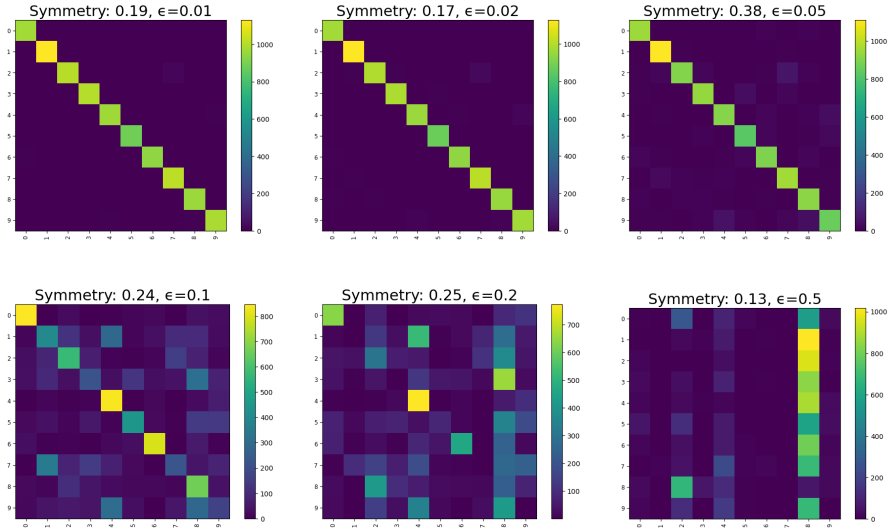


Figure 9: L^∞-PGD-Attack on MNIST

5 Discussion

Symmetry Probably the most notable feature of our results is the high degree of symmetry of the CIFAR-10 confusion matrices (Figures 1, 2, 3). This means that, the NN that learned to classify CIFAR-10 is equally likely to mistake class i as class j as the other way around (j as i).

Upon taking a closer look at the classes that are likely to be mistaken for another, our reader may notice a pattern. Across the board, the two most common causes of confusion are the pairs "Automobile"-"Truck", and "Dog"-"Cat". From a human perspective, this is a understandable mistake; these two

categories are in fact similar in appearance. The other matrix entries follow a similar pattern, with confusions amongst animals and vehicles, being significantly more common than across these two categories.

The MNIST, and FashionMNIST dataset do not appear to match this hypothesis too well.

In Figures 4 and 5 the FashionMNIST-NN appears prone to confusing the pairs "Sandals"-"Sneakers2 and "Coat"-"Pullover", but adversarial examples of Dresses are far more likely to be classified as "Trouser" than vice versa. The PGD-Matrices (Fig. 6) have quite high symmetry scores, but nothing captures the eye. **Erklärung?**

The MNIST confusion matrices yield comparatively low symmetry scores (Fig. 7, 8, 9). We propose that this is due to a overpowered NN and resulting overfitting. **Passt das von dir aus Max?**

Catch-all Class In Figures 3, 6, 9, one can observe that adversarial examples computed with large perturbation budgets ϵ are misclassified as "8", "TODO" and "frog" for MNIST, Fashion-MNIST and CIFAR-10 respectively. In order to shed light onto this phenomenon we generate and classify 10000 white noise images sampled from a uniform distribution on the input domain. Figure A shows that these randomly generated images are also, most commonly, classified as "8", "TODO" and "frog" respectively. This result suggests that the neural networks in question have a default output for low probability images with respect to distribution of the input domain, which in turn affects adversarial examples computed with large perturbation budgets.

6 Conclusion

We explored confusion matrices generated by many common untargeted attacks, and discovered interesting structure. They are decidedly non-random in at least two ways:

- Adversarial images which come very close to the original image (e.g. small ϵ) can yield surprisingly symmetric confusion matrices. This symmetry means that class i is confused with class j about as often as vice versa. To us this indicates, that the NN has difficulty telling these two classes apart. This idea is corroborated by the fact that, in these cases the NN confuses semantically similar images in a rather intuitive fashion; for instance confusing "horses" with "deer", and "automobiles" with "trucks".

- Attacks which are given a more generous ϵ tend to cluster into one specific class ("frog" and "8"). We hypothesised that this is due the NN using this class as a catch-all for all images it has difficulty classifying. Using a large batch of noise images we were able to confirm this hypothesis.

7 Contribution Statement

This is joint work from Maximilian Samsinger and Matthias Dellago. Max wrote the code... We thank Alexander Schlögl for the research idea.

References

[1] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

[2] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[3] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[4] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.

[5] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[6] Wieland Brendel, Jonas Rauber, Matthias Kümmerer, Ivan Ustyuzhaninov, and Matthias Bethge. Accurate, reliable and fast robustness evaluation, 2019.

[7] Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131*, 2017.

[8] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[9] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[10] Guido Van Rossum and Fred L Drake Jr. *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

[11] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative

style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.