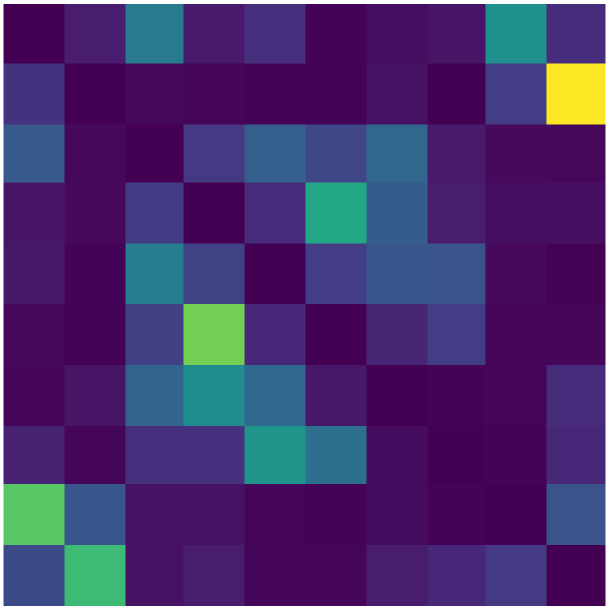


Adversarial Label Flips

Matthias Dellago & Maximilian Samsinger

June 17, 2021

Abstract

Given a neural network classifier and an untargeted evasion attack, in what class does the adversarial example fall post-attack? In the following, we will answer this question by evaluating some state of the art attacks on simple neural network classifiers trained on industry standard datasets. We discover that semantically similar classes are more likely to be confused with another, leading us to hypothesise that the convolutional neural networks recognise these similarities.

1 Introduction

In 2013 Szegedy et al. demonstrated that deep neural networks are susceptible to attacks [1]. These adversarial examples consist of a small perturbation applied to otherwise benign inputs, engineered to cause the neural network to misbehave.

We consider neural image classifiers, in particular convolutional neural networks [2] (CNN). That is, inputs are images and attacks apply small changes to said images, designed to cause the CNN to misclassify the target.

These attacks on classifiers come in two different variations: targeted and untargeted. Targeted attacks aim to cause a misclassification into a specific target class. For example, a cat image is to be misclassified as a dog. Untargeted attacks only try to evade correct classification. In this case, the cat image must only be misclassified as anything other than a cat, not a dog specifically. For a proper introduction to threat modelling in adversarial machine learning we refer to [3].

When considering untargeted attacks, the question what the adversarial image is classified as post-attack arises. This is what we will experimentally answer in this paper.

		are categorised as		
		Dog	Cat	Plane
Adversarial examples of a	Dog	2	6	2
	Cat	7	3	0
	Plane	1	2	7

Figure 1: An example of a confusion Matrix. Unsuccessful attacks (along the diagonal) are also included.

We will present our results in terms of confusion matrices. An simple example is presented in Figure 1. For large matrices numbers become more difficult to grasp, so we will display our results in heatmap-style images, as seen on the title page.

In our experiments (Section 3) we used the Foolbox framework [4], and simple CNNs trained on the MNIST [5], Fashion-MNIST [5] and CIFAR-10 [6] datasets. We applied three state of the art attacks: Projected Gradient Decent (PDG)[7], Carlini-Wagner [8] and Brendel-Bethge [9]. 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 are allotted a large perturbation budget, there exist certain attractor classes, which most of the adversarial images are classified as. (Section 4)

2 Background and related work

Existence of adversarial examples Since adversarial examples were first introduced in [1] a large body of literature has flourished on the topic. In the following we will briefly introduce the attacks we use.

2.1 Attacks

Fast gradient sign method Goodfellow et al. developed the fast gradient sign method (FGSM) [10], 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 perturbation are scaled to the same magnitude.

Projected gradient descent Projected gradient descent (PGD) was first shown in [7]. 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. Then iterated FGSM resumes. This attack leads to formidable results, especially using the L^∞-norm.

Carlini-Wagner attack Carlini and Wagner [8] invented a different style of attack, where the cost function of the classifier and the distance of the adversarial example to the original are wrapped into one function. They can then simultaneously optimize for both using the Adam stochastic optimizer [11].

Brendel-Bethge attack Brendel and Bethe invented a quite different approach [9]. Their method works by starting from a image deep inside the misclassification region and then preforming binary search between it and the original image, to find the decision boundary. Once there, they move along the boundary to minimize the distance to the original, yielding a powerful adversarial example.

Foolbox Foolbox is a convenient Python [12] toolbox for generating adversarial examples [4]. 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 in Section 4, 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.

Given 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:

$$A_{sym} = \frac{A_0 + A_0^T}{2}, \qquad A_{anti} = \frac{A_0 - A_0^T}{2} \tag{1}$$

We then define our measure for symmetry s:

$$s = \frac{\|A_{sym}\|_1 - \|A_{anti}\|_1}{\|A_{sym}\|_1 + \|A_{anti}\|_1} \tag{2}$$

For confusion matrices (where all values a_{ij} are greater zero) s can take values from $[0, 1]$. $s = 1$ represents maximally symmetric matrices, and $s = 0$ a maximally anti-symmetric matrix.

Note that s is well-defined for confusion matrices since any reordering of classes labels, that is permutations of the rows and columns of A, does not change s. Furthermore the 1-norm is not special, and could be replaced with any other p-norm, since it is well-defined for all of them.

Below are some examples to aid our esteemed 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 2.1 to compute adversarial examples on the MNIST [5], Fashion-MNIST [13] and CIFAR-10 [6] 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

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

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

Reproducability All our code, Figures and the models we trained are available on GitHub [1].

[1] `https://github.com/MXSMCI/AdversarialLabelFlips`

3.1 Experimental setup

All experiments are conducted in Python 3.8.5 [12] using the PyTorch 1.8.1 [14] library on a Windows machine. Adversarial attacks are computed using Foolbox 3.3.1 [4]. 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 $[0.01, 0.02, 0.05, 0.1, 0.2, 0.5]$ to cover a wide range of perturbation budgets.

3.2 Architectures

We consider the MNIST Model and CIFAR Model, two architectures which are described in [8], Table 1 and 2. Both architectures consist of two blocks of Convolutional-Convolutional-MaxPooling [15] layers followed by three fully connected layers. All but the last layer use ReLU [16] as its activation function, whereas the final layer uses the softmax function.
Carlini, an author of [8], provides a reference implementation, which is available on GitHub [2]. We achieved similar results in terms of accuracy with our PyTorch reimplementation and can therefore confirm the validity of their results.
For our experiments on the MNIST and Fashion-MNIST dataset we use the MNIST Model. For the CIFAR-10 dataset we use the CIFAR Model.

4 Results

We present the confusion matrices for all adversarial attacks we considered. Figure 2,4, and 6 show the results for the Carlini-Wagner attack and Figure 3,5, and 7 show the results for all PGD attacks. Since the results for both Brendel-Bethge attacks are similar to the Carlini-Wagner attack, we moved them to the appendix A. We add our symmetry score as well as the perturbation budget used ϵ to the Figure title, if applicable.

[2] `https://github.com/carlini/nn_robust_attacks`

4.1 CIFAR-10

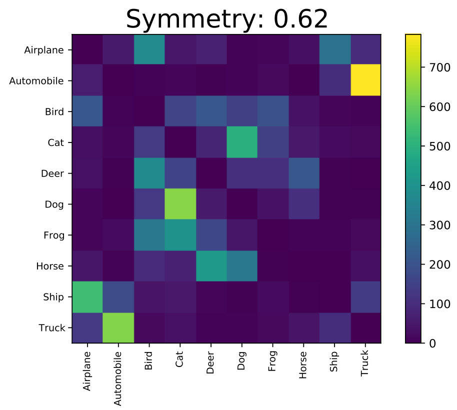


Figure 2: Confusion matrix for the L^2-Carlini-Wagner attack on CIFAR-10. Attacks were computed using the Foolbox default parameters with no early termination.

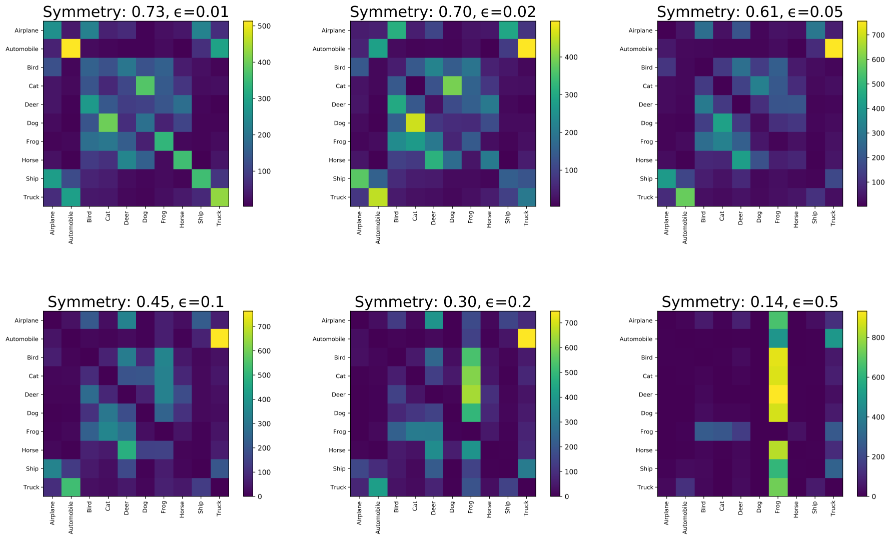


Figure 3: Confusion matrices for the L^∞-PGD attack on CIFAR-10 for varying maximal perturbation sizes per image pixel $\epsilon \in [0, 1]$. Larger ϵ corresponds to stronger attacks. Attacks were computed using the Foolbox default parameters.

4.2 FashionMNIST

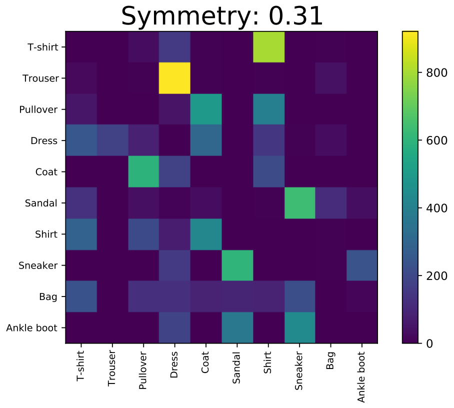
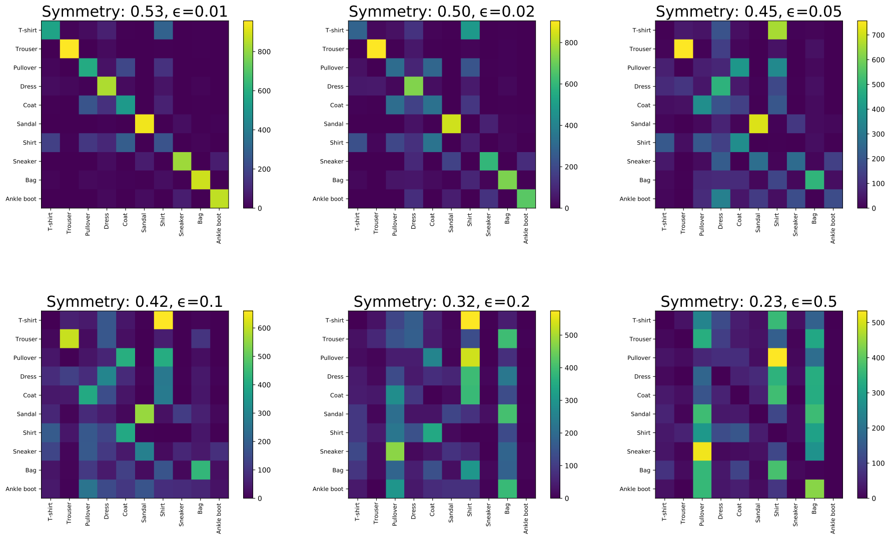

Symmetry: 0.31

Figure 4: Confusion matrix for the L^2-Carlini-Wagner attack on FashionMNIST. Attacks were computed using the Foolbox default parameters with no early termination.


Symmetry: 0.53, ϵ=0.01

Symmetry: 0.50, ϵ=0.02

Symmetry: 0.45, ϵ=0.05

Symmetry: 0.42, ϵ=0.1

Symmetry: 0.32, ϵ=0.2

Symmetry: 0.23, ϵ=0.5

Figure 5: Confusion matrices for the L^∞-PGD attack on FashionMNIST for varying maximal perturbation sizes per image pixel $\epsilon \in [0, 1]$. Larger ϵ corresponds to stronger attacks. Attacks were computed using the Foolbox default parameters.

4.3 MNIST

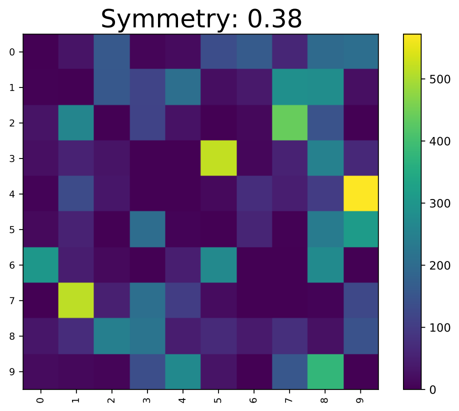


Figure 6: Confusion matrix for the L^2-Carlini-Wagner attack on MNIST. Attacks were computed using the Foolbox default parameters with no early termination.

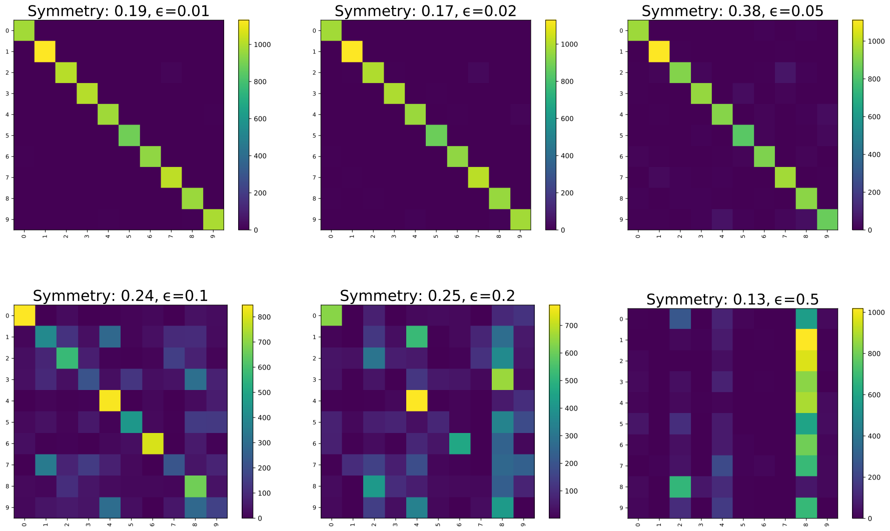


Figure 7: Confusion matrices for the L^∞-PGD attack on MNIST for varying maximal perturbation sizes per image pixel $\epsilon \in [0, 1]$. Larger ϵ corresponds to stronger attacks. Attacks were computed using the Foolbox default parameters.

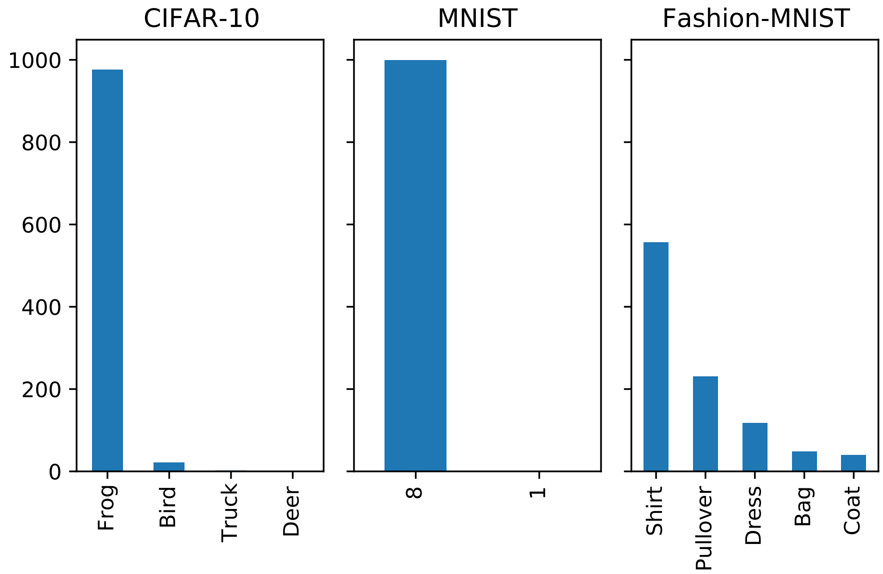


Figure 8: Bar chart of the top 5 most common predictions of 1000 random uniform white noise images on CIFAR-10, MNIST and Fashion-MNIST respectively. On CIFAR-10 and MNIST only four and two different classes where predicted respectively.

Symmetry Probably the most notable feature of our results is the high degree of symmetry of the CIFAR-10 confusion matrices (Figures 2, 9, 3). This means that the CNN is about as likely to mistake adversarial examples of class i for j as the other way around (j for i). For example adversarial examples of cats are about as likely to be classified as ”Dogs” as vice versa.

Upon taking a closer look at the classes that are likely to be mistaken for each other, 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 an understandable mistake; these two categories are in fact similar in appearance. The other matrix entries follow a similar pattern: Animals are likely to be mistaken for other animals, and vehicles for other vehicles. Confusions of animals and vehicles are significantly rarer.

It therefore seems that the CNN captures some notion of similarity, that is quite close to what a human would intuit.

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

In Figures 4 and 10 the classifier appears prone to confusing the pairs ”Sandals”-”Sneakers” and ”Coat”-”Pullover”, but adversarial examples of Dresses are far more likely to be classified as ”Trouser” than vice versa. The PGD-

Matrices (Fig. 5) have quite high symmetry scores, but nothing particularly captures the eye.

The MNIST confusion matrices yield comparatively low symmetry scores (Fig. 6, 11, 7). We propose that this is due to a overpowered CNN and resulting overfitting.

Catch-all Classes In Figures 3 and 7 one can observe that adversarial examples computed with large perturbation budgets ϵ are most often misclassified as "frog", and "8" for MNIST and CIFAR-10 respectively. For FashionMNIST 5, there are multiple high probability classes. 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 8 shows that these randomly generated images are also, most commonly, classified as "frog" and "8" respectively. More complex behaviour emerges for the Fashion-MNIST dataset. These results suggests that the neural networks in question have one or multiple default outputs for low probability images. This in turn affects adversarial examples computed with large perturbation budgets.

5 Conclusion

We explored confusion matrices generated by strong, well-studied untargeted attacks, and discovered two patterns.

- Adversarial images with small perturbation sizes (e.g. small ϵ or computed with minimization attacks) 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 classifier has learned that some classes are more closely related than others. This idea is corroborated by the fact that, in these cases, semantically similar classes are misclassified in a rather intuitive fashion; for instance confusing "Cat" with "Dog", and "Automobile" with "Truck".

- Attacks which are given a more generous ϵ tend to cluster into one or multiple specific class ("Frog" and "8"). We hypothesized that this is due the CNN using these classes as a catch-all for all images it has difficulty classifying. We sampled and predicted the class of white noise images to confirm this hypothesis.

6 Contribution Statement

This is joint work from Maximilian Samsinger and Matthias Dellago. Max contributed most of the code, which included training and evaluating the neural networks. Matthias wrote most parts of the this thesis. Both contributed equally to the presentations. We thank Alexander Schlögl for the research idea and all members of the seminar for their valuable feedback, which lead to, among other revisions, the inclusion of Figure 8.

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] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer, 1999.

[3] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

[4] 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.

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

[6] Alex Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, Univ. of Toronto, 2009.

[7] 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.

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

[9] Wieland Brendel, Jonas Rauber, Matthias Kümmerer, Ivan Ustyuzhaninov, and Matthias Bethge. Accurate, reliable and fast robustness evaluation. *arXiv preprint arXiv:1907.01003*, 2019.

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

[11] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2017.

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

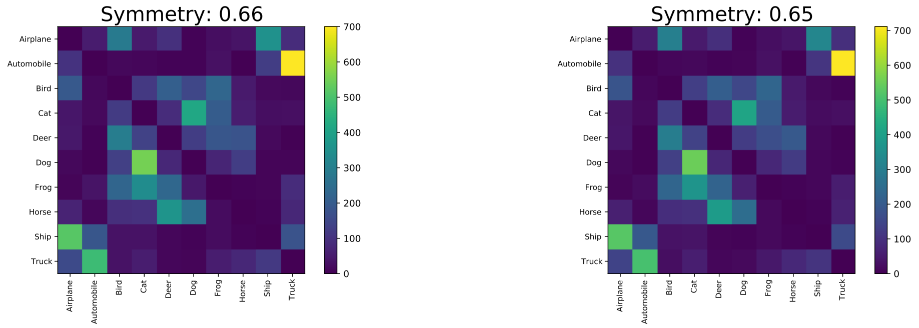
[13] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[14] 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 *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019.

[15] Jawad Nagi, Frederick Ducatelle, Gianni A Di Caro, Dan Cireşan, Ueli Meier, Alessandro Giusti, Farrukh Nagi, Jürgen Schmidhuber, and Luca Maria Gambardella. Max-pooling convolutional neural networks for vision-based hand gesture recognition. In *2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, pages 342–347. IEEE, 2011.

[16] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Citeseer, 2013.

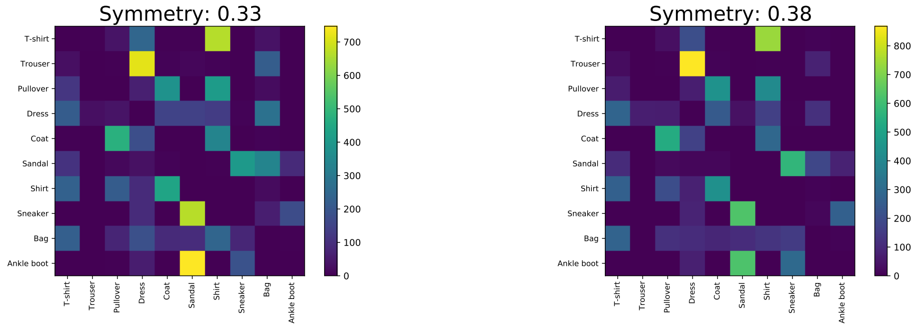
A Figures for Brendel-Bethge attacks



(a) L^0

(b) L^1

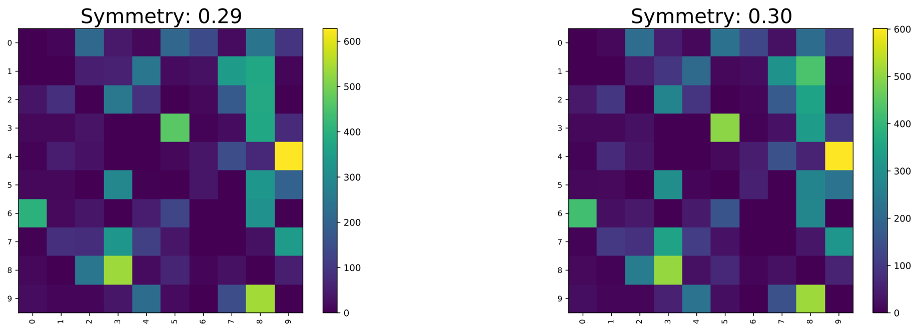
Figure 9: Confusion matrix for the Brendel-Bethge attack on CIFAR-10. The left and right plot shows the L^0 and L^1 variant of the attack respectively. Both attacks were computed using the Foolbox default parameters with no early termination.



(a) L^0

(b) L^1

Figure 10: Confusion matrix for the Brendel-Bethge attack on FashionMNIST. The left and right plot shows the L^0 and L^1 variant of the attack respectively. Both attacks were computed using the Foolbox default parameters with no early termination.



(a) L^0

(b) L^1

Figure 11: Confusion matrix for the Brendel-Bethge attack on MNIST. The left and right plot shows the L^0 and L^1 variant of the attack respectively. Both attacks were computed using the Foolbox default parameters with no early termination.