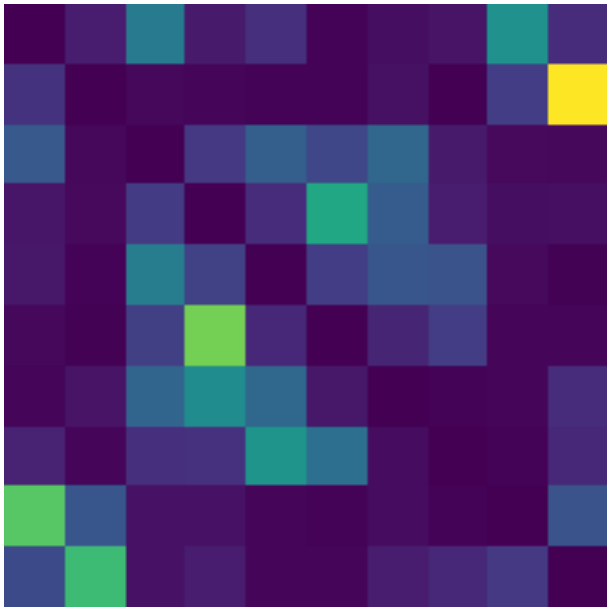
universität
innsbruck



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 a simple neural network classifiers trained on industry standard datasets. We discover that this intuitively similar classes are more likely to be confused with another, leading us to hypothesise that the NN recognises 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 a otherwise benign input, engineered to cause the neural net to misbehave.

We consider the case of neural image classifiers, in particular convolutional neural networks (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. In targeted attacks the attacker aims to have the adversarial example identified as a specific class by the CNN. (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.) For a more in-depth discussion of threat modelling in adversarial machine learning we refer to [2].

When considering untargeted attacks, the question what the adversarial image is classified as post-attack naturally 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. In figure 1 you can see an example. In larger 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 [3], and simple CNNs trained on the MNIST [4], Fashion-MNIST [4] and CIFAR-10 [5] datasets. We applied three state of the art attacks: Projected Gradient Decent (PDG)[6], Carlini-Wagner [7] and Brendel-Bethge[8].

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 being first demonstrated [1] a large body of literature has flourished around adversarial examples. We in the following we will briefly introduce the attacks we use.

2.1 Attacks

Fast gradient sign method Goodfellow et al. seminally developed the fast gradient sign method (FGSM) [9], 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 [6]. 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 [7] 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 [10].

Brendel-Bethge attack Brendel and Bethe invented a quite different approach [8]. 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 convenient python toolbox for generating adversarial examples [3]. 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 (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:

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

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

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

Below are 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 [4], Fashion-MNIST [11] and CIFAR-10 [5] 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).

3.1 Experimental setup

All experiments are conducted in Python 3.8.5 [12] using the PyTorch 1.8.1 [13] library on a Windows machine. Adversarial attacks are computed using FoolBox 3.3.1 [3]. 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.

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

3.2 Architectures

We consider the MNIST Model and CIFAR Model, two architectures which are described in [7], Table 1 and 2. Both architectures consist of two blocks of Convolutional-Convolutional-MaxPooling layers followed by three fully connected layers. All but the last layer use ReLU [14] as its activation function, whereas the final layer uses the softmax function.
One author of [7] provides a reference implementation 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

Here we present all the confusion matrices we calculated. We add our symmetry score as well as the epsilon value, if applicable.

4.1 CIFAR-10

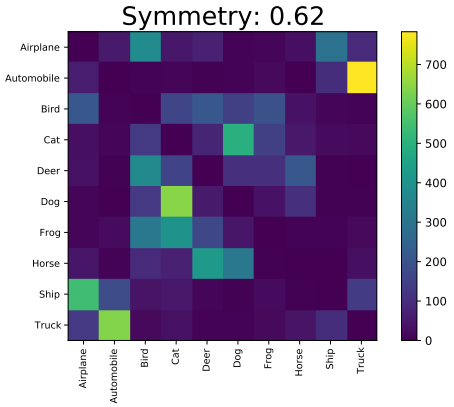
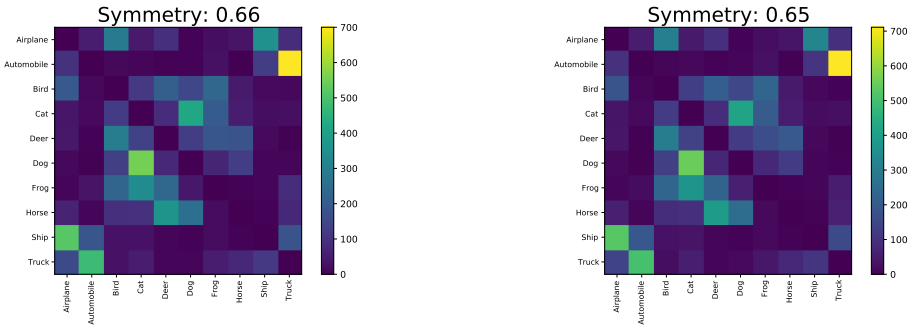


Figure 2: L^2-Carlini-Wagner-Attack on CIFAR-10

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



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

Figure 3: Brendel-Bethge-Attacks on CIFAR-10

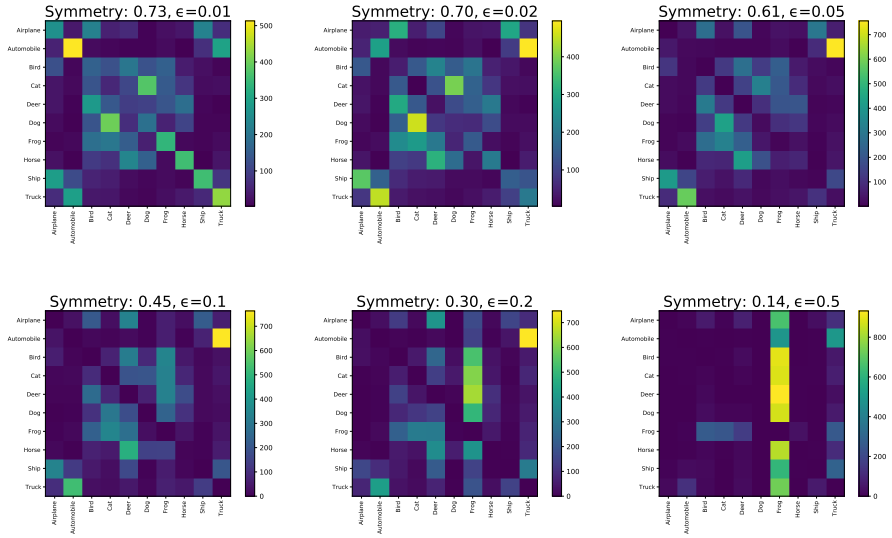


Figure 4: L^∞-PGD-Attack on CIFAR-10

4.2 FashionMNIST

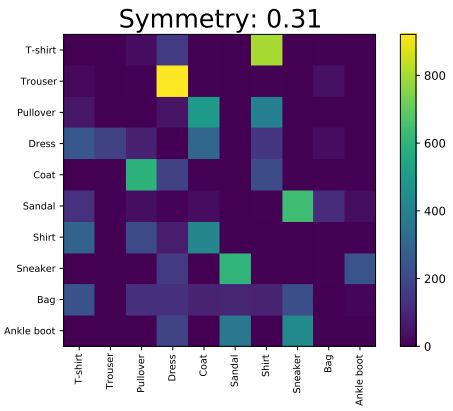
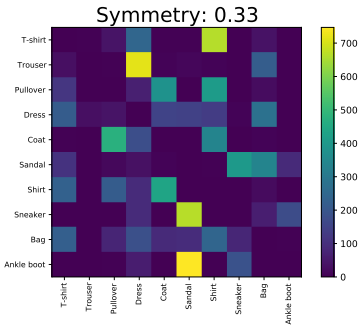
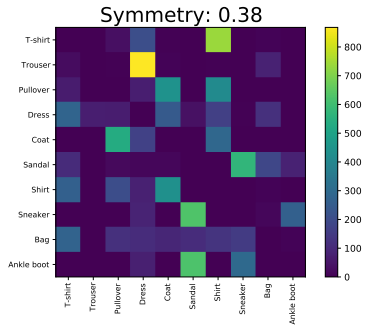


Figure 5: L^2-Carlini-Wagner-Attack on FashionMNIST



(a) L^0



(b) L^1

Figure 6: Brendel-Bethge-Attacks on FashionMNIST

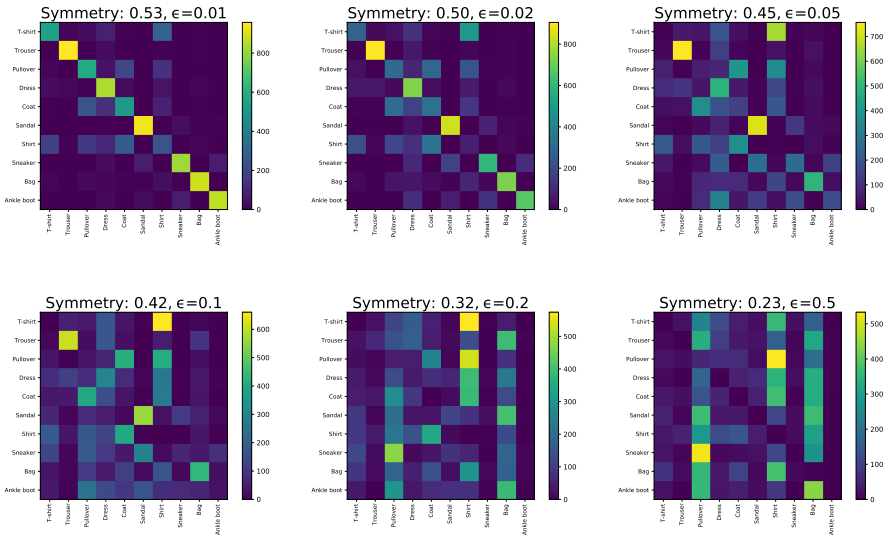


Figure 7: L^∞-PGD-Attack on FashionMNIST

4.3 MNIST

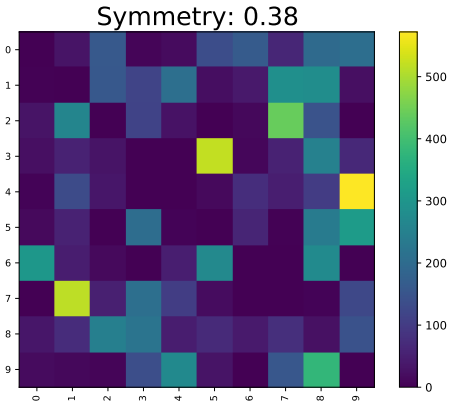
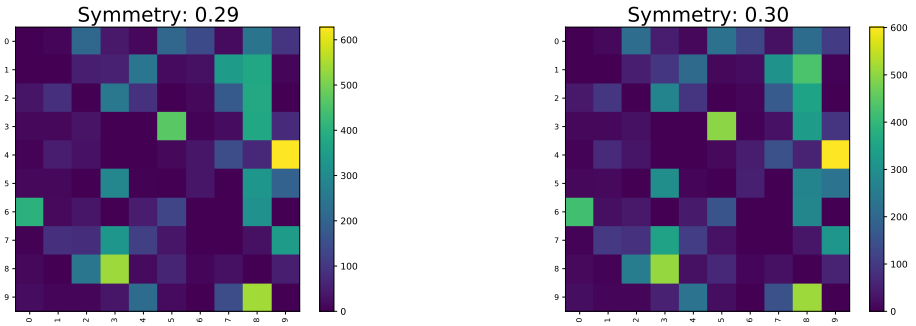


Figure 8: L^2-Carlini-Wagner-Attack on MNIST



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

Figure 9: Brendel-Bethge-Attacks on MNIST

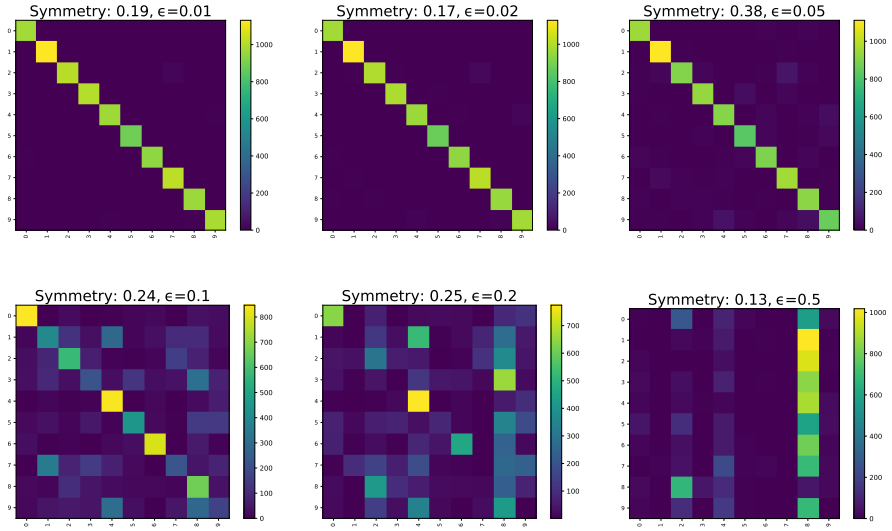


Figure 10: L^∞-PGD-Attack on MNIST

5 Discussion

Symmetry Probably the most notable feature of our results is the high degree of symmetry of the CIFAR-10 confusion matrices (Figures 2, 3, 4). This means that, the CNN 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.

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 5 and 6 the FashionMNIST-CNN 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. 7) have quite high symmetry scores, but nothing particularly captures the eye.

The MNIST confusion matrices yield comparatively low symmetry scores (Fig. 8, 9, 10). We propose that this is due to a overpowered CNN and resulting overfitting. **Passt das von dir aus Max?**

Catch-all Class In Figures 4, 7, 10, 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 CNN has learned that some classes are more closely related than others. This idea is corroborated by the fact that, in these cases the CNN 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 CNN 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 trained the CNN and evaluated the attacks, Matthias wrote most of the paper. **Passt das so? Soll noch was rein?** 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] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

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

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

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

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

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

[13] 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.

[14] 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