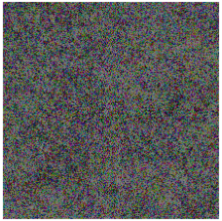
Adversarial Label Flips

Matthias Dellago & Maximilian Samsinger

A short recap

Adversarial attack

$+ \ \epsilon$



$=$

Husky
(42.82% confidence)

Noise (PGD-40)
50x amplified

Handkerchief
(99.999988% confidence)

Source: `ctf.codes`, circa 2021

What we want

Confusion Matrix

Categorised as

Adversarial Example of a		Dog	Cat	Plane
	Dog	0.0	?	?
	Cat	?	0.0	?
	Plane	?	?	0.0

How many modified dogs get classified as cats vs as planes? etc.

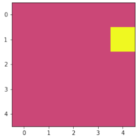
Some simple theory

We want similar images that are classified differently.
But what is "similar"?

Quantifying Difference (ϵ)

L^0-Norm

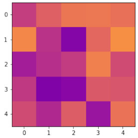
Number of
different pixels



Change very few
pixels maximally

L^1-Norm

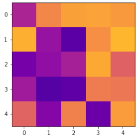
Sum of
all differences



Minimise sum

L^2-Norm

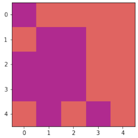
Root of the sum
of the squares
of all differences



Minimise sum
of squares

L^∞-Norm

Maximum of
all differences



Change all
pixels equally

Two Different Approaches

small difference and
big misclassification

maximise misclassification

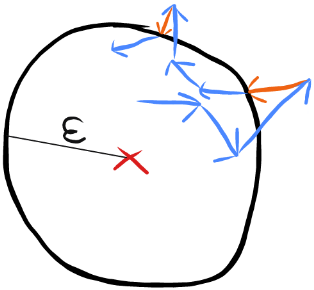
minimise difference

small difference

big misclassification

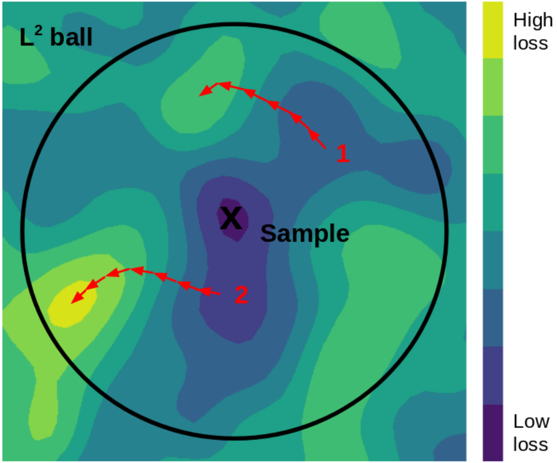
Projected Gradient Decent

1. Pick spot in epsilon ball around target
2. Iterate gradient decent
3. If leaving ball, project back onto surface
4. Repeat to convergence



Towards Deep Learning Models Resistant to Adversarial Attacks, Aleksander Madry et al., arXiv, 2019

Projected Gradient Decent



Know your enemy, Oscar Knagg, towardsdatascience.com, 2019

Carlini-Wagner-Attack

Original approach: minimise difference while always staying in "misclassification region".

Problem: Non-linearity of constraint makes optimimisation difficult.

Towards Evaluating the Robustness of Neural Networks, Nicholas Carlini and David Wagner, IEEE, 2017

Carlini-Wagner-Attack

Solution: Pack constraint into the function that is optimised.

$\rightarrow$ minimise: difference - "how misclassified is x?"*

i.e. minimise difference while maximising misclassification.

Apply Adam optimisation.

*loss function

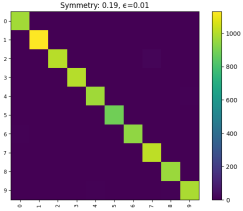
Towards Evaluating the Robustness of Neural Networks, Nicholas Carlini and David Wagner, IEEE, 2017

Code

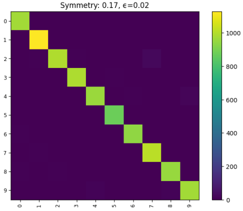
Results

MNIST, L^∞-PGD

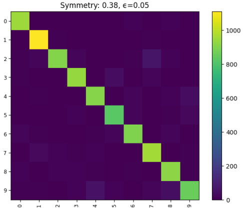
$\epsilon = 0.01$



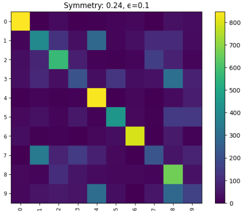
$\epsilon = 0.02$



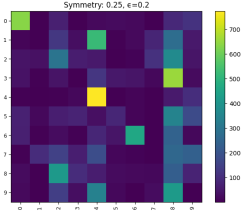
$\epsilon = 0.05$



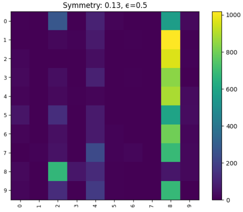
$\epsilon = 0.1$



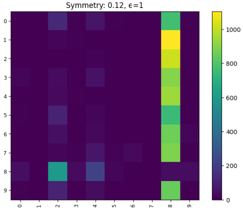
$\epsilon = 0.2$



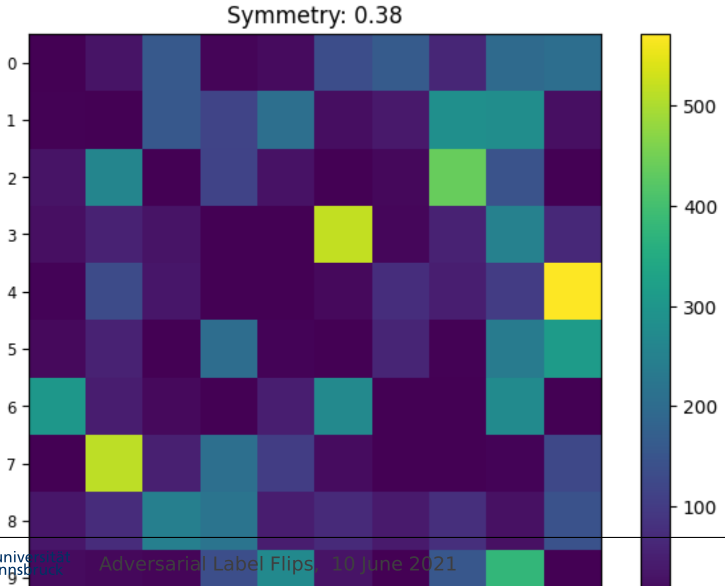
$\epsilon = 0.5$



$\epsilon = 1$

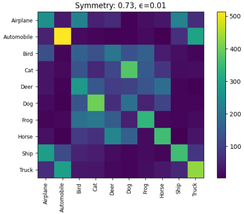


MNIST, L^2-Carlini-Wagner-Attack

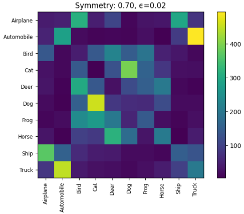


CIFAR-10, L^{∞}-PGD

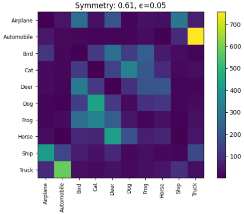
$\epsilon = 0.01$



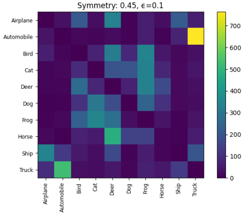
$\epsilon = 0.02$



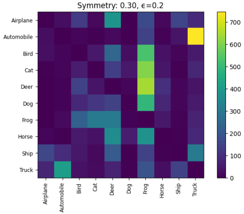
$\epsilon = 0.05$



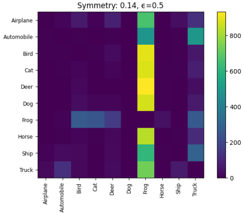
$\epsilon = 0.1$



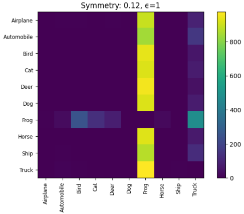
$\epsilon = 0.2$



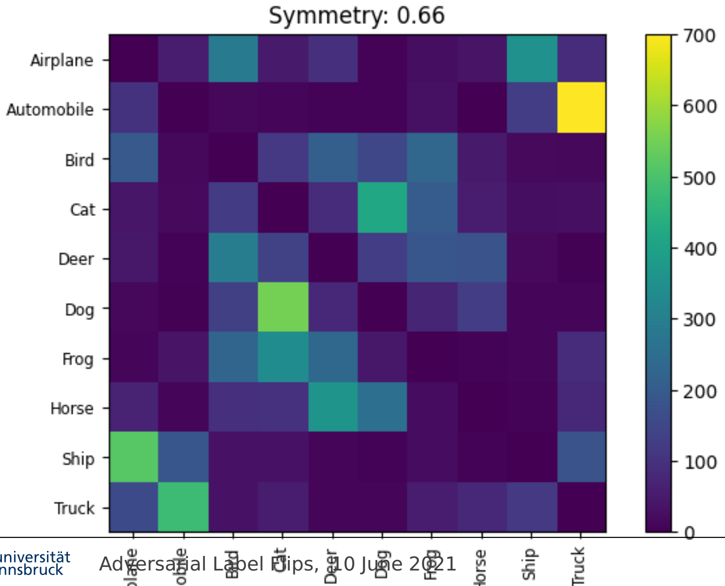
$\epsilon = 0.5$



$\epsilon = 1$



CIFAR-10, L^0-Brendel-Bethge-Attack



Tentative Findings

Small $\epsilon \rightarrow$ symmetric confusion matrix

Large $\epsilon \rightarrow$ strong attractor classes ("8" and "frog")

References I