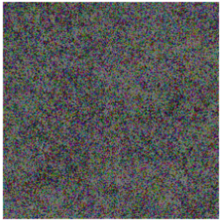
Adversarial Label Flips

Matthias Dellago & Maximilian Samsinger

A short recap

Adversarial attack

$+ \ \epsilon$



$=$

Husky	Noise (PGD-40)	Handkerchief
(42.82% confidence)	50x amplified	(99.999988% confidence)

Source: `ctf.codes`, circa 2021

What we want to do

Confusion Matrix

Adversarial Example of a (rows) / Categorised as (columns)

	Dog	Cat	Plane
Dog	0.0	?	?
Cat	?	0.0	?
Plane	?	?	0.0

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

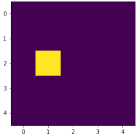
Some simple theory

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

Quantifying Changes

L^0-Norm

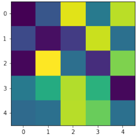
Number of
pixels changed



Perturb one
pixel maximally

L^1-Norm

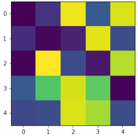
Sum of
all changes



Minimise sum

L^2-Norm

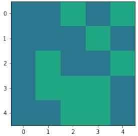
Sum of the *square*
of all changes



Minimise sum
of squares

L^∞-Norm

Maximum of
all changes



Perturb all
pixels equally

Two Different Approaches

small displacement and
big misclassification

maximise misclassification

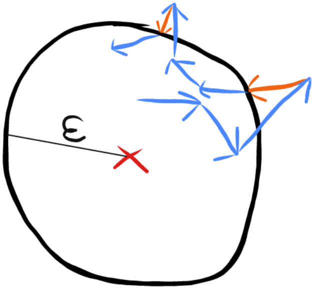
minimise displacement

small displacement

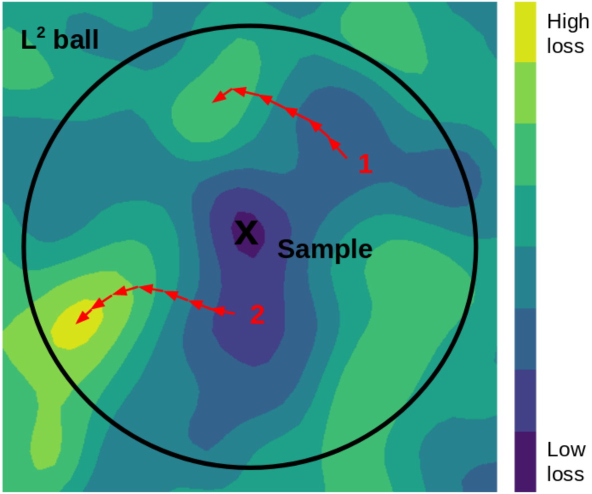
big misclassification

Projected Gradient Decent

1. Pick spot in epsilon ball
2. Iterate gradient decent
3. If leaving ball, project back onto surface.



Projected Gradient Decent



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

Carlini-Wagner-Attack

Original idea: minimise disance while always staying in "misclassification territory".
Problem: Nonlinearity of constraint makes for bad optimisation

Carlini-Wagner-Attack

Solution: Pack constraint into the function that is optimised.
=> minimise: distance + "how misclassified is x?"*
equiv. minimise distance while maximising misclassification.

*loss function

References I