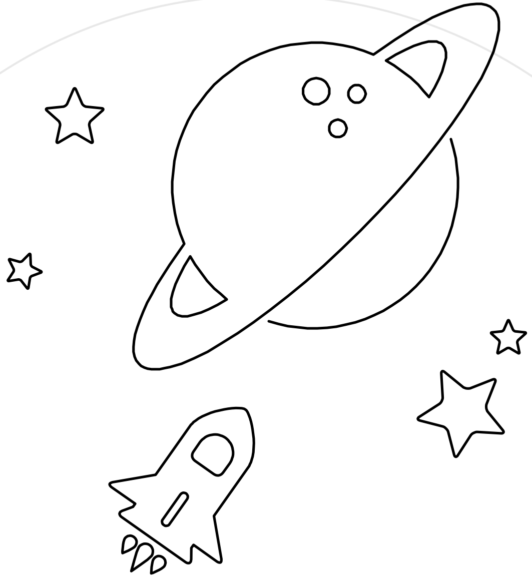


Defence Against the Dark Arts: Adversarial ML

I am
Amit Kushwaha

@yardstick17

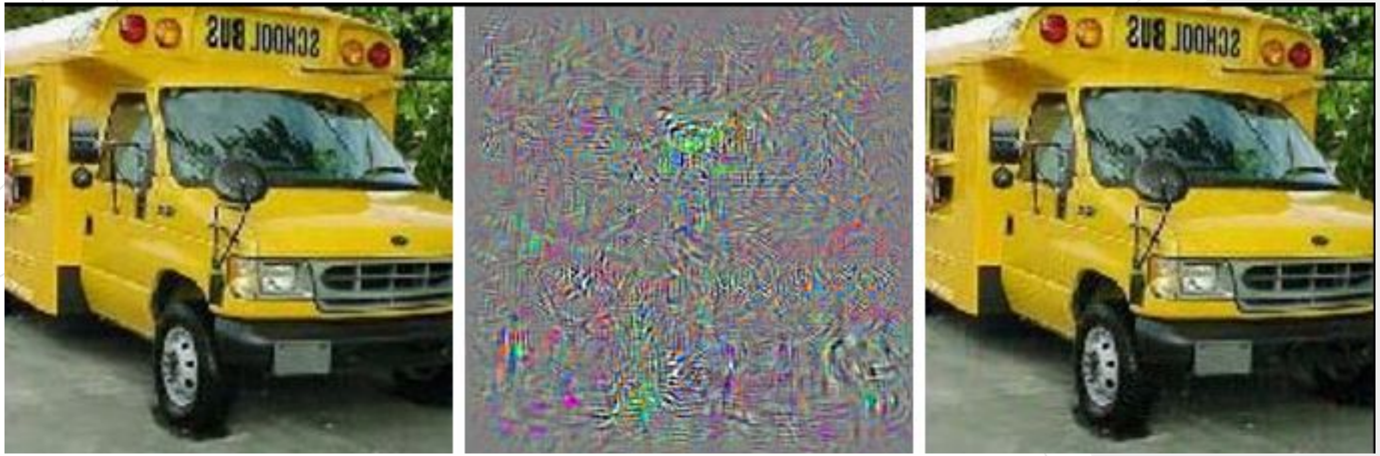
@imYardstick17

@yardstick17

Worked/Working as:

- Machine Learning Engineer: Zomato (India)
- Python Engineer: Zalando SE (Germany)
- Data Science Mentor: Redi School

Adversarial Example



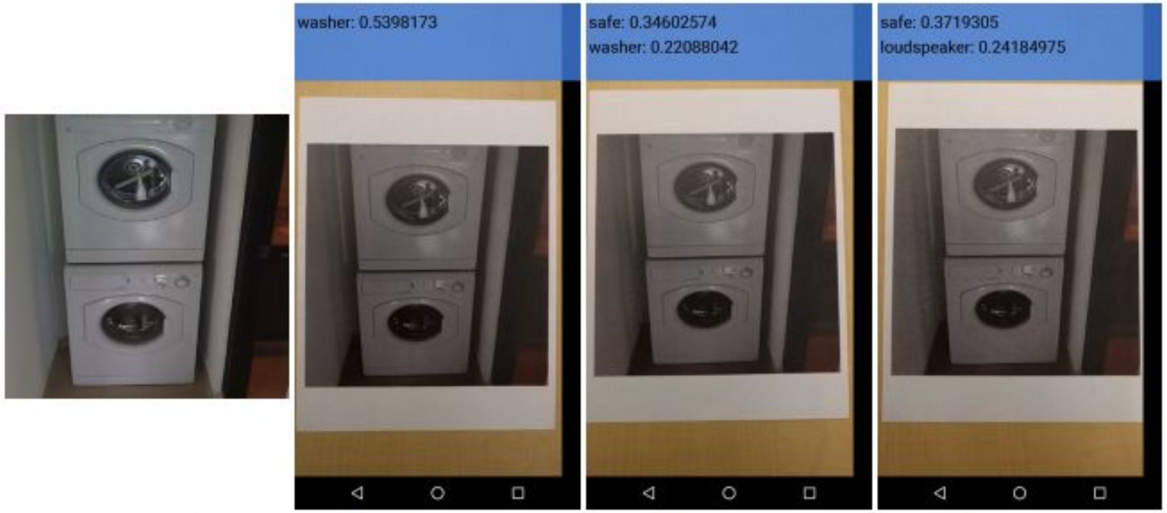
Truck (Correctly Classified) **Perturbation** **Ostrich** (**Incorrectly** Classified)

The image on the left is correctly classified as a truck by the neural network (AlexNet). The middle image represents the changes made to it, and the rightmost image is the result, labelled by the classifier as "ostrich". That's funny, right? But that's not all: it works in the physical world, too. Guess what a neural network sees in this picture.

Adversarial Example

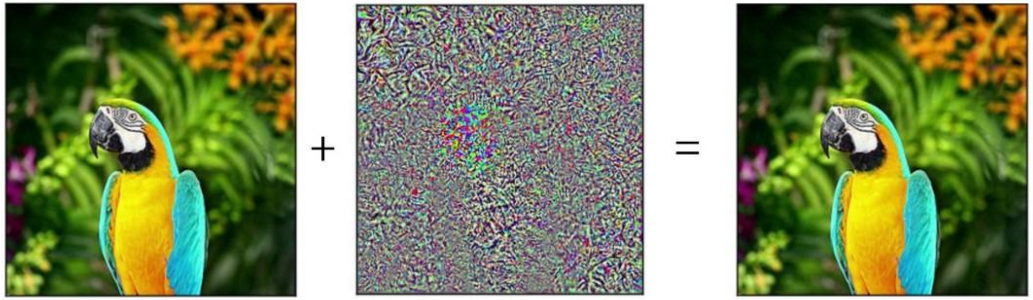
- Intentionally designed to cause the model to make a mistake

- They're like optical illusions for machines

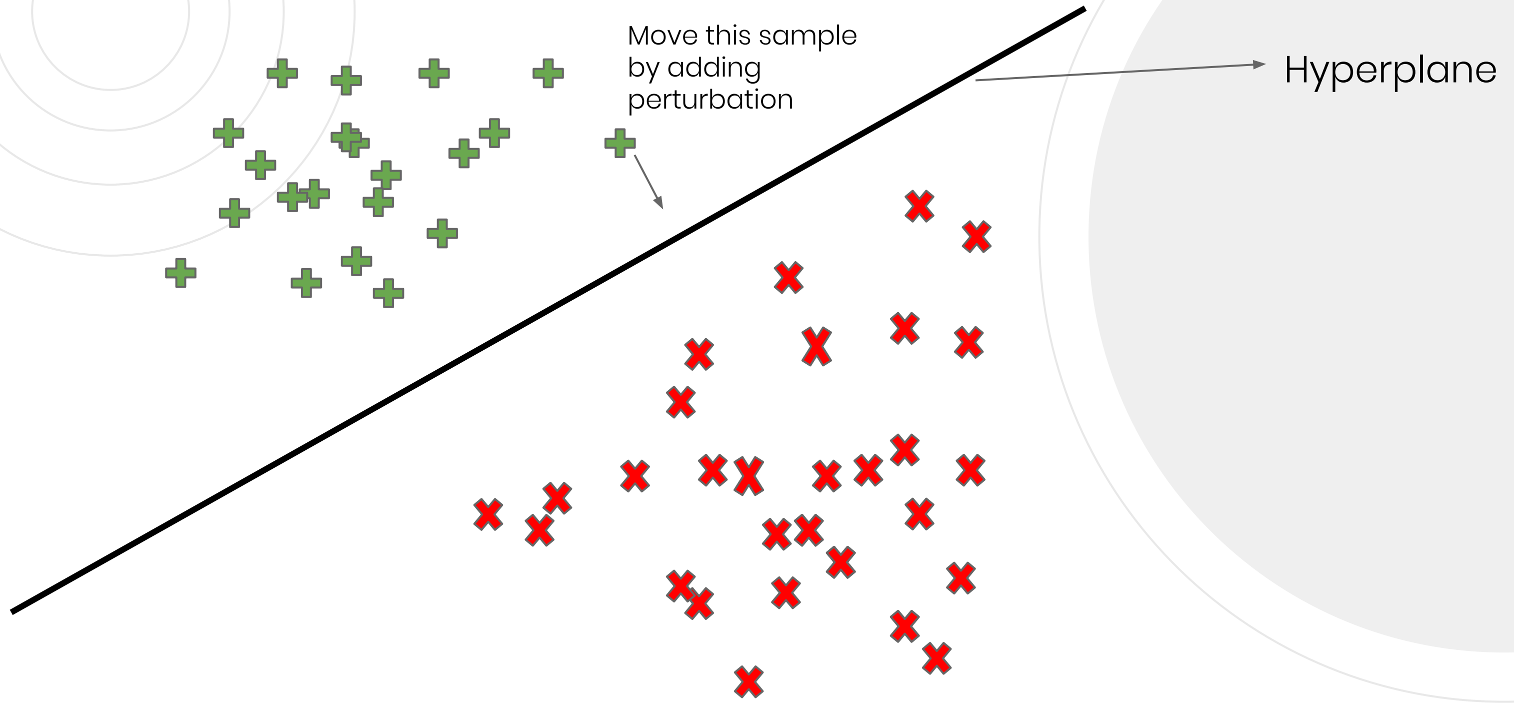


(a) Image from dataset (b) Clean image (c) Adv. image, $\epsilon = 4$ (d) Adv. image, $\epsilon = 8$

Crafting an Adversarial Example, viz. Attacking a Model


+
=
X
97.3% macaw
$\mathrm{sign}(\nabla_X J(\theta, X, Y))$
$X + \epsilon.\mathrm{sign}(\nabla_X J(\theta, X, Y))$
88.9% bookcase

Intuition Behind the Idea



Fast Gradient Sign Method- The Earliest technique

- Calculates the gradient of loss function w.r.t input.
- Adding these gradient to input will increase the loss and might switch the class prediction.

$$\boldsymbol{X}^{adv} = \boldsymbol{X} + \epsilon\,\mathrm{sign}\big(\nabla_X J(\boldsymbol{X}, y_{true})\big)$$

Deep Fool

- Iteratively “linearizes” the loss function at an input point.
- Applies the minimal perturbation that would be necessary to switch classes,

Carlini’s Attack

- Treats this as an Optimization Problem, and optimizes for having the minimal distance from the original example, under the constraint of having the example be misclassified by the original model.
- This is the current heavyweight champion of attacks

Cleverhans

It is a Python library to benchmark machine learning systems' vulnerability to adversarial examples

Demo

- Train a model:
  - accuracy on **clean** dataset
  - accuracy on **adversarial** dataset
- Generate Adversarial Examples
  - FSFM, Spatial Transformation Method etc.

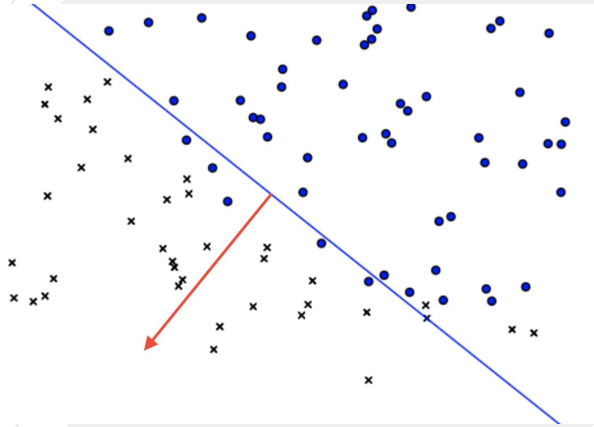
Culprit behind this Madness

- Deep models are **too** linear.
  - For an adversarial example, $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\eta}$

    Dot product between the weights and input

$$\boldsymbol{w}^\top \tilde{\boldsymbol{x}} = \boldsymbol{w}^\top \boldsymbol{x} + \boldsymbol{w}^\top \boldsymbol{\eta}.$$

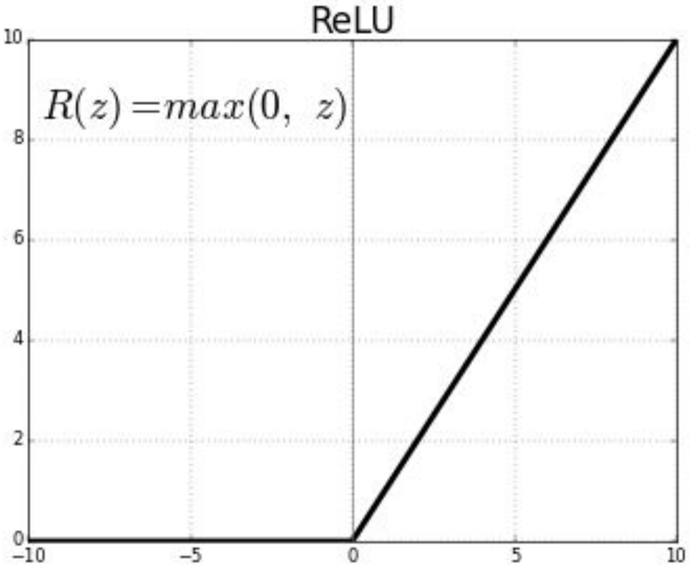
- It is correct that deep networks need to be nonlinear in order to work. But, within the space of possible nonlinear activation functions, modern deep nets have actually settled on one that is very close to linear: **the ReLu**



Culprit behind the Madness

- **Linear Activation Function**

  The fault is not in our stars, but in our activation functions, for they are linear.



Why Adversarial threat is real?

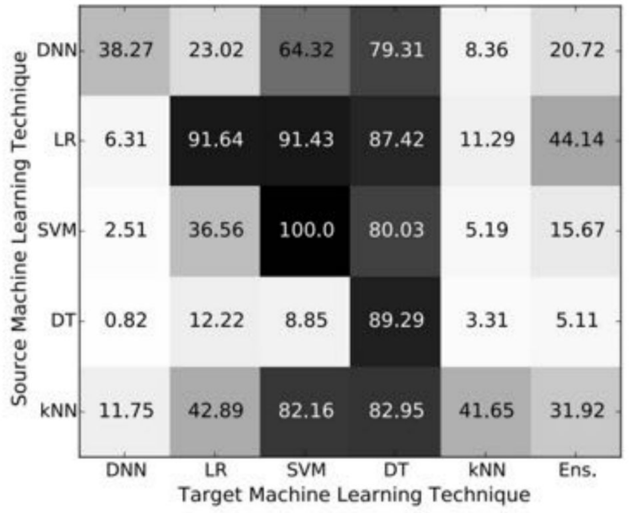
As crafting an adversarial example **requires access to model parameters** i.e. the gradient of the underlying deep neural network.

What if this classification is exposed just only as service?

Why Adversarial threat is real?

Deep Networks show the property of **Transferability**.

Cross-technique transferability



Source Machine Learning Technique \ Target Machine Learning Technique	DNN	LR	SVM	DT	kNN	Ens.
DNN	38.27	23.02	64.32	79.31	8.36	20.72
LR	6.31	91.64	91.43	87.42	11.29	44.14
SVM	2.51	36.56	100.0	80.03	5.19	15.67
DT	0.82	12.22	8.85	89.29	3.31	5.11
kNN	11.75	42.89	82.16	82.95	41.65	31.92

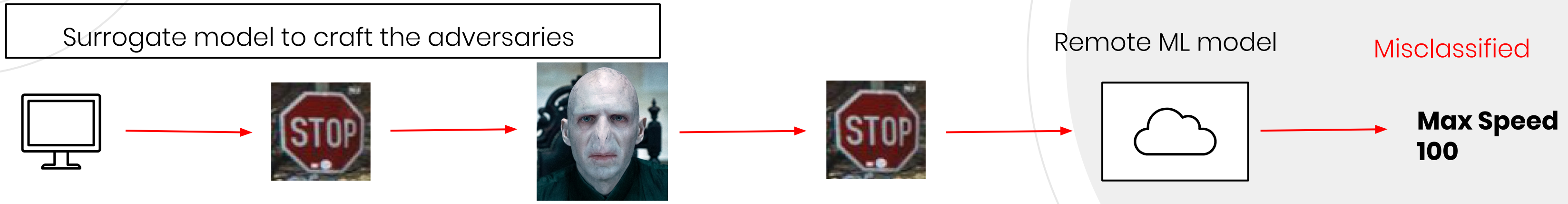
(Papernot 2016)

(Goodfellow 2

Why Adversarial threat is real?

Deep Networks show the property of **Transferability**.

- Example crafted to mislead a model A are likely to mislead a model B



- Adversarial examples often transfer between entirely different kinds of machine learning algorithms, such as SVMs and Decision Trees.

Defence Against the Dark Arts

Adversarial training:

- Generate adversarial examples
- And explicitly train the model not to be fooled by each of them

$$\tilde{J}(\boldsymbol{\theta}, \boldsymbol{x}, y) = \alpha J(\boldsymbol{\theta}, \boldsymbol{x}, y) + (1 - \alpha) J(\boldsymbol{\theta}, \boldsymbol{x} + \epsilon \operatorname{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y)))$$

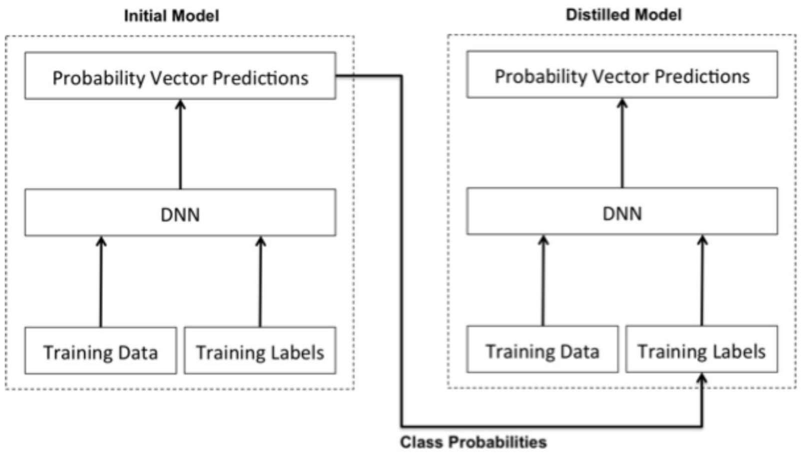
Demo:

- Adversarial Training

Defence Against the Dark Arts

Defensive Distillation:

- Train a model and use the class probability output rather than hard class labels to train another model.
- Smoothes out the surface which will typically be used by adversary to change the input.

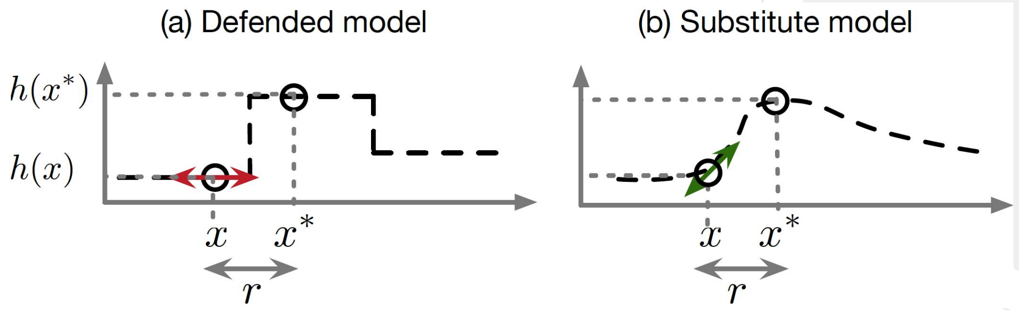


And Many more. Most of them like above, are based on the principle of **Gradient Masking.**

Gradient Masking: A failed defence

- In simple words, defence based on gradient masking made the model gradient in-significant which **made the adversarial example generation difficult.**

- But, it doesn't show robustness against the adversarial example. This example can be generated by the fact that models show the property of **Transferability.**



In summary, most of the defences tries flatten out the model and remove the gradient rather than learning to classify more point correctly..

Future Directions

Indirect Methods Do not just optimize the performance measure exactly

- Best methods so far
  - Logit pairing (non-adversarial)
    - Logits from two images to be similar to each other. Adversarial logit pairing consists of minimizing the loss:

$$J(\mathbb{M}, \boldsymbol{\theta}) + \lambda \frac{1}{m} \sum_{i=1}^{m} L\left(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), f(\tilde{\boldsymbol{x}}^{(i)}; \boldsymbol{\theta})\right).$$

  - Label smoothing (paper under review for ICLR 2019)
  - Logit squeezing (paper under review for ICLR 2019)

- Can we perform a lot better with other methods that are similarly indirect?

Cybersecurity Perspective

Three golden rules:

- **Know your adversary**: Know your system exploit points, vulnerabilities
- **Be proactive**: Don't wait for your system to be attacked. Proactively look how the system can be exploited.
-
- **Protect yourself:** Example could be Iterative training against with adversarial examples.

Cybersecurity Perspective

For two models with the same error volume, for reasons of security we prefer the model:

- With lower confidence on mistakes
- Whose mistakes are harder to find
- That does not repeatedly make the same mistake on the same input
- Whose mistakes are less valuable to the attacker / costly to the defender
- That is harder to reverse engineer with probes
- That is less prone to transfer from related model

References

- Cleverhans
- Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks
- Defense Against the Dark Arts: An overview of adversarial example security research and future research directions
- Know Your Adversary: Understanding Adversarial Examples
- EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES
- Practical Black-Box Attacks against Machine Learning
- Towards Evaluating the Robustness of Neural Networks
- Adversarial Logit Pairing

- And Countless papers

Thanks a lot.

@yardstick17

@imYardstick17

@yardstick17