Adversarial examples in deep learning

Grégory Châtel

Datategy
Intel Software Innovator

@rodgzilla
github.com/rodgzilla

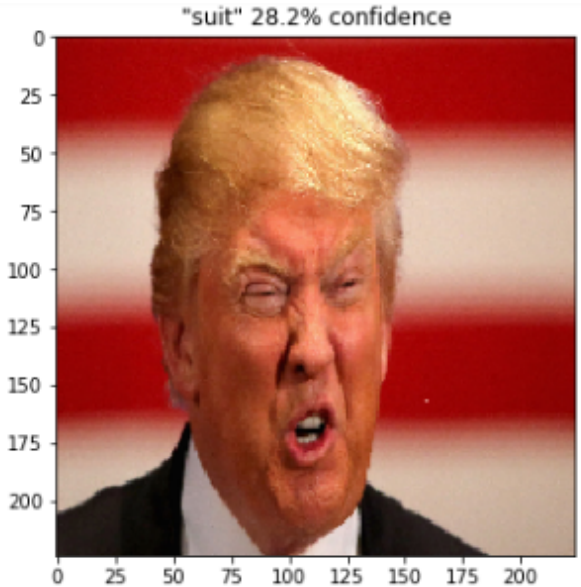
06/07/2017

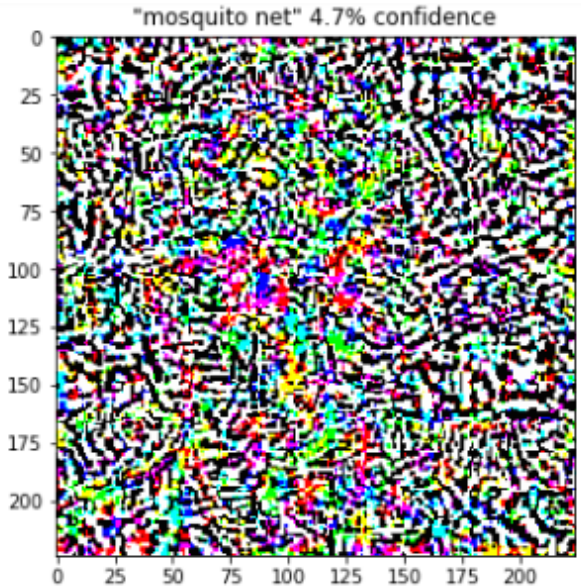
What is an adversarial example?

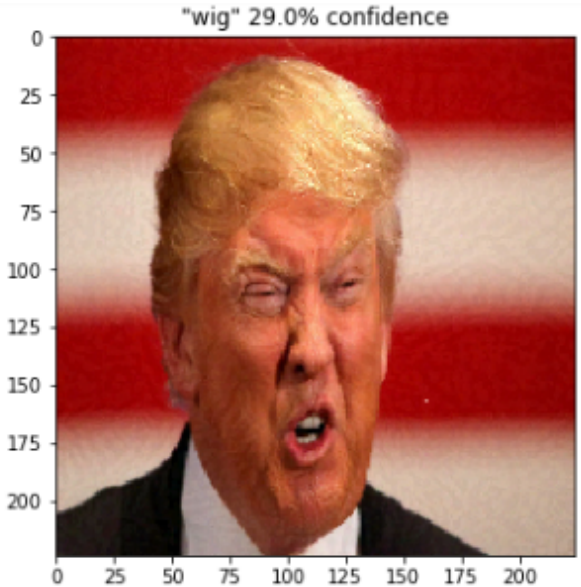
An *adversarial example* is a sample of input data which has been modified *very slightly* in a way that is intended to cause a machine learning classifier to misclassify it.

What is an adversarial example?

An *adversarial example* is a sample of input data which has been modified *very slightly* in a way that is intended to cause a machine learning classifier to misclassify it.

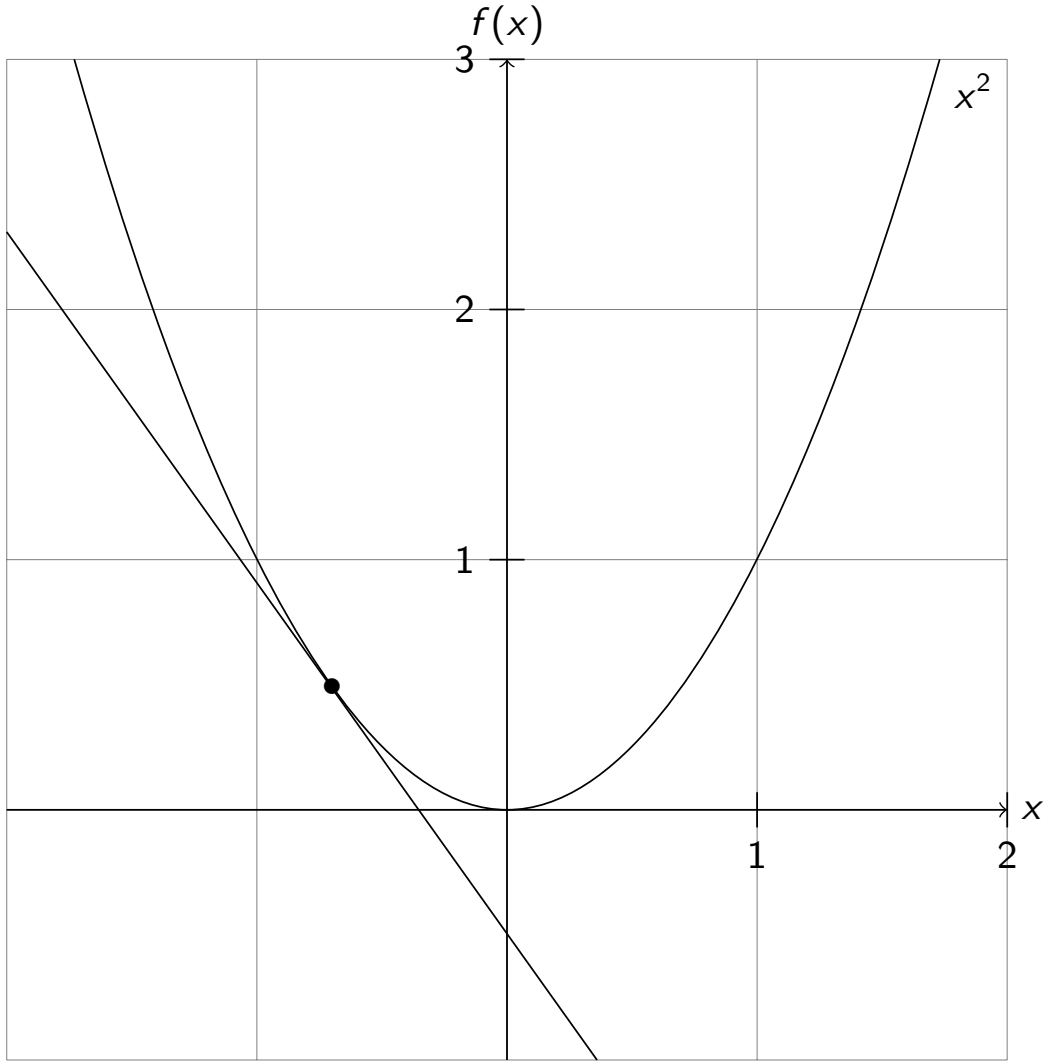






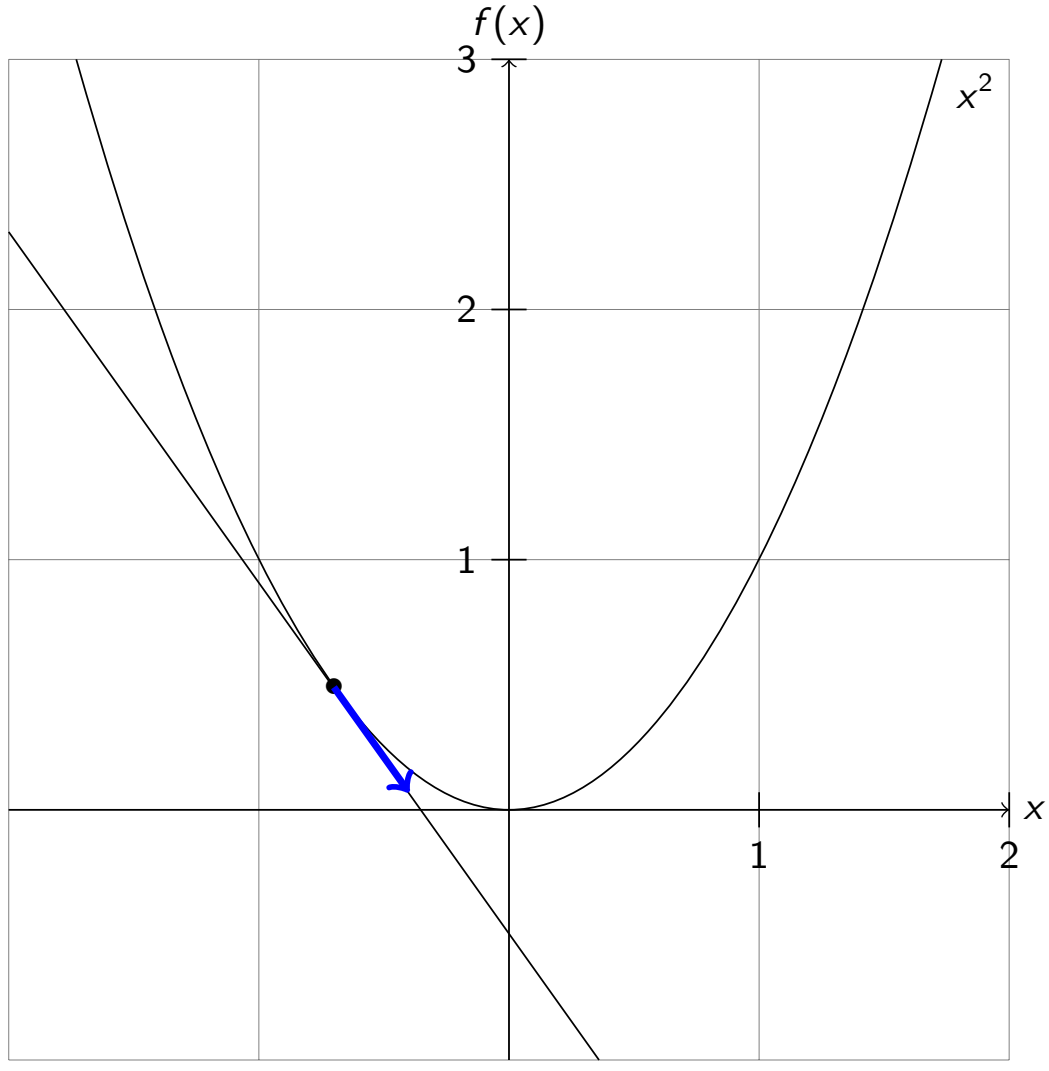
Gradient descent

Basic concept



Gradient descent

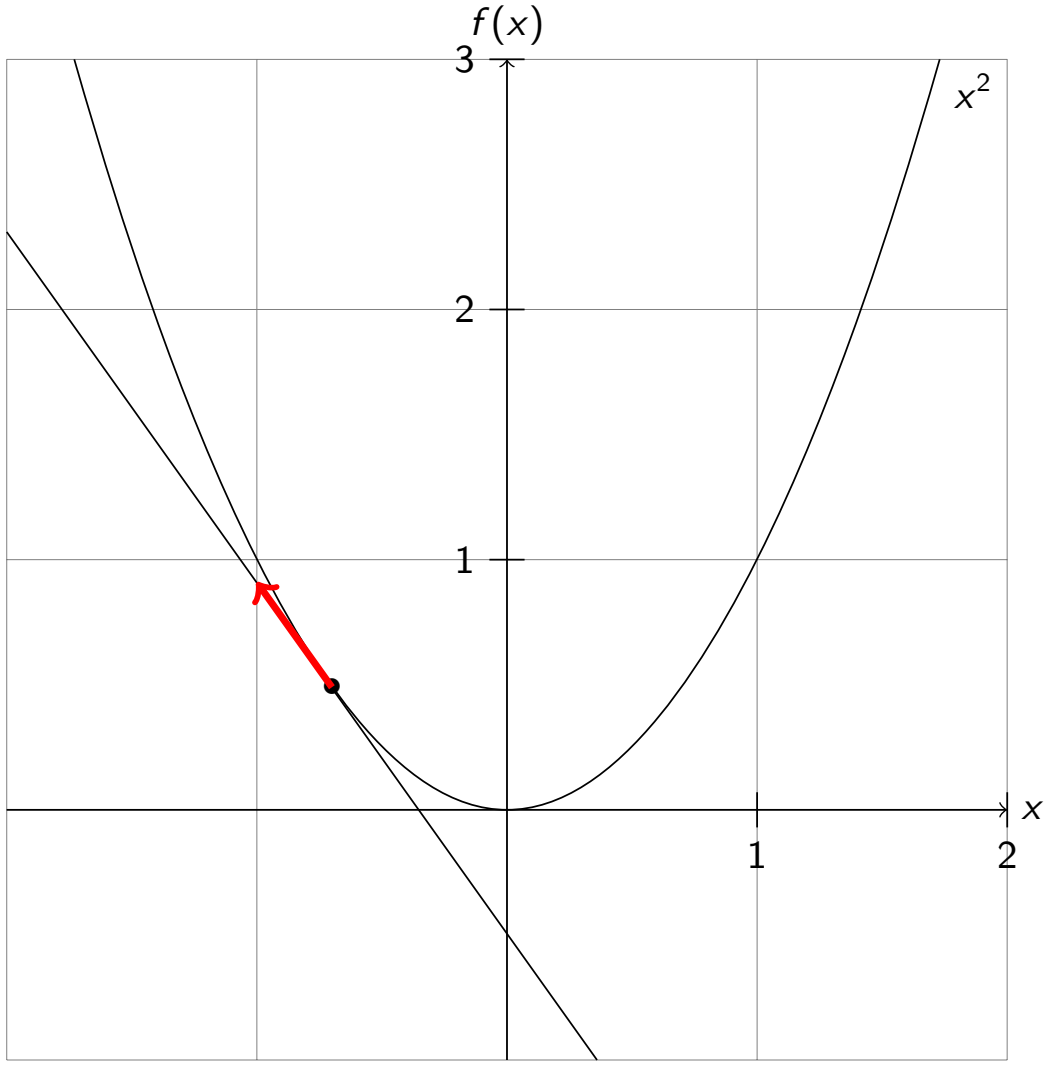
Basic concept



Optimization

Gradient descent

Basic concept



De-optimization

Neural networks

To train and evaluate neural networks, we use a *loss function* $L(\theta, x, y)$ with θ the parameters of the models, x an input and y the real value corresponding to x. This function measures *how good* is the prediction of the model on a specific example.

Neural networks

To train and evaluate neural networks, we use a *loss function* $L(\theta, x, y)$ with θ the parameters of the models, x an input and y the real value corresponding to x. This function measures *how good* is the prediction of the model on a specific example.

To train a neural network we compute the derivative of L according to θ and use the result to update θ in order to decrease the loss value.

Neural networks

To train and evaluate neural networks, we use a *loss function* $L(\theta, x, y)$ with θ the parameters of the models, x an input and y the real value corresponding to x. This function measures *how good* is the prediction of the model on a specific example.

To train a neural network we compute the derivative of L according to θ and use the result to update θ in order to decrease the loss value.

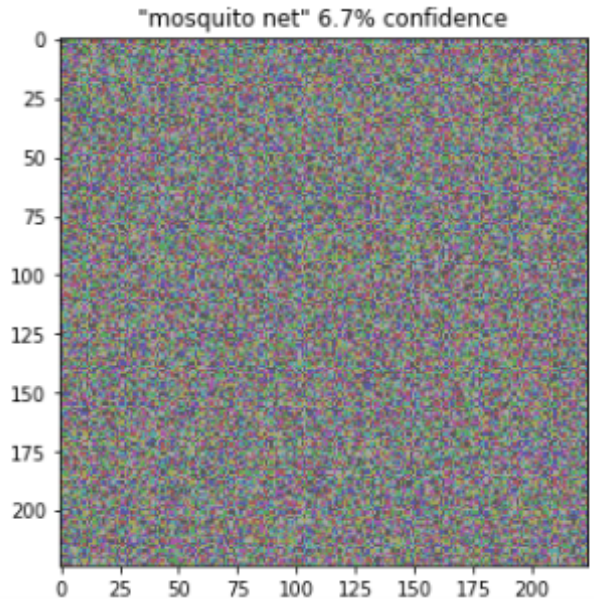
To create an adversarial sample, we compute the derivative of L according to x and use the result to update the pixel values of x in order to increase the loss value. It happens that such modifications of x often cause the network to misclassify it.

Random noise perturbation

Do we really need to do that?

Random noise perturbation

Yep



Fast Gradient Sign Method [2015]

Let x be the original image, θ the parameters of the model, y the target associated with x and $L(\theta, x, y)$ the loss function.

We compute the gradient of the loss function according to the input pixels.

$$\nabla_x L(\theta, x, y)$$

The perturbation is the signs of these derivatives multiplied by a small number ε.

$$\eta = \varepsilon \ \text{sign}(\nabla_x L(\theta, x, y))$$

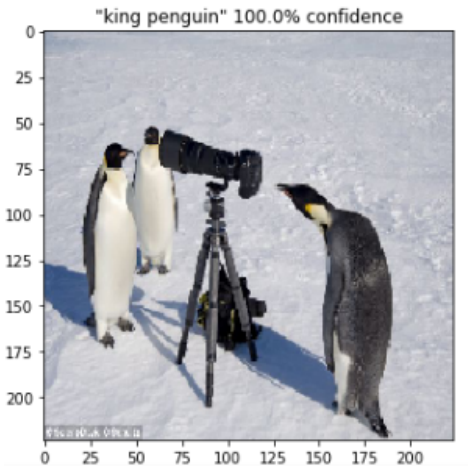
The final adversarial sample is the sum of the original image and the perturbation.

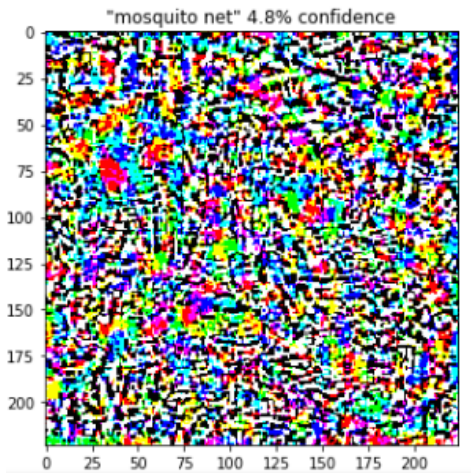
$$x_{adv} = x + \eta$$

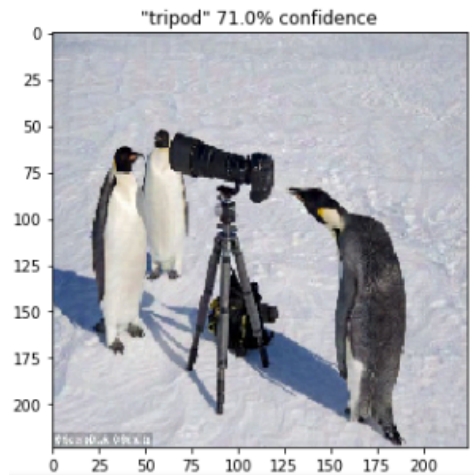
Fast Gradient Sign Method

VGG16 network

$$x + \varepsilon \operatorname{sign}(\nabla_x L(\theta, x, y)) = x_{\text{adv}}$$







Black-box attack [2016]

or good luck getting gradients out of your self-driving car

Black-box attack [2016]

Transferability of adversarial samples

We can train a new model M' to solve the same classification task as the target model M.

Black-box attack [2016]

Transferability of adversarial samples

We can train a new model M' to solve the same classification task as the target model M.

Once trained, we can create an adversarial sample x'_{adv} for the M' model and experiences have shown that x'_{adv} will also fool M very often.

Black-box attack [2016]

Transferability of adversarial samples

We can train a new model M' to solve the same classification task as the target model M.

Once trained, we can create an adversarial sample x'_{adv} for the M' model and experiences have shown that x'_{adv} will also fool M very often.

What if we do not have a training set for the target network? Well... build one using M predictions.

Black-box attack [2016]

Transferability of adversarial samples

We can train a new model M' to solve the same classification task as the target model M.

Once trained, we can create an adversarial sample x'_{adv} for the M' model and experiences have shown that x'_{adv} will also fool M very often.

What if we do not have a training set for the target network? Well... build one using M predictions.

"After labeling 6,400 synthetic inputs to train our substitute (an order of magnitude smaller than the training set used by MetaMind) we find that their DNN misclassifies adversarial examples crafted with our substitute at a rate of 84.24%"

- Papernot et al., about their attack on the MetaMind deep neural network.

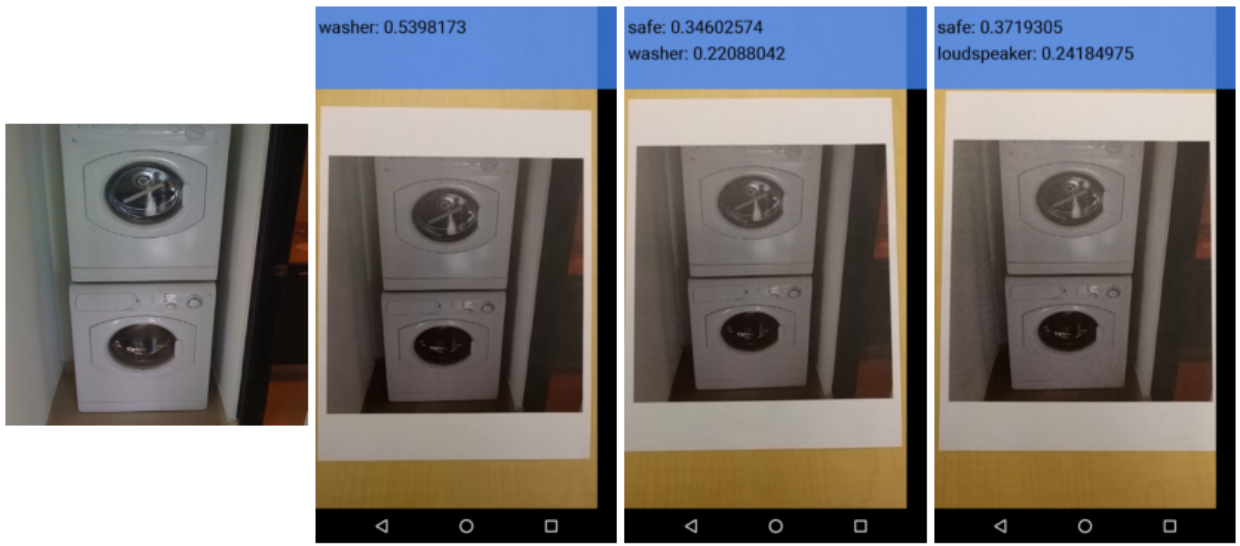
Adversarial examples in the physical world [2017]

or good luck attacking a self-driving car with your USB flash drive

Adversarial examples in the physical world [2017]

In real world scenarios, the target network does not take our image files as input. It acquires the data by the network's system (*e.g.* a camera).

It also works, for free.



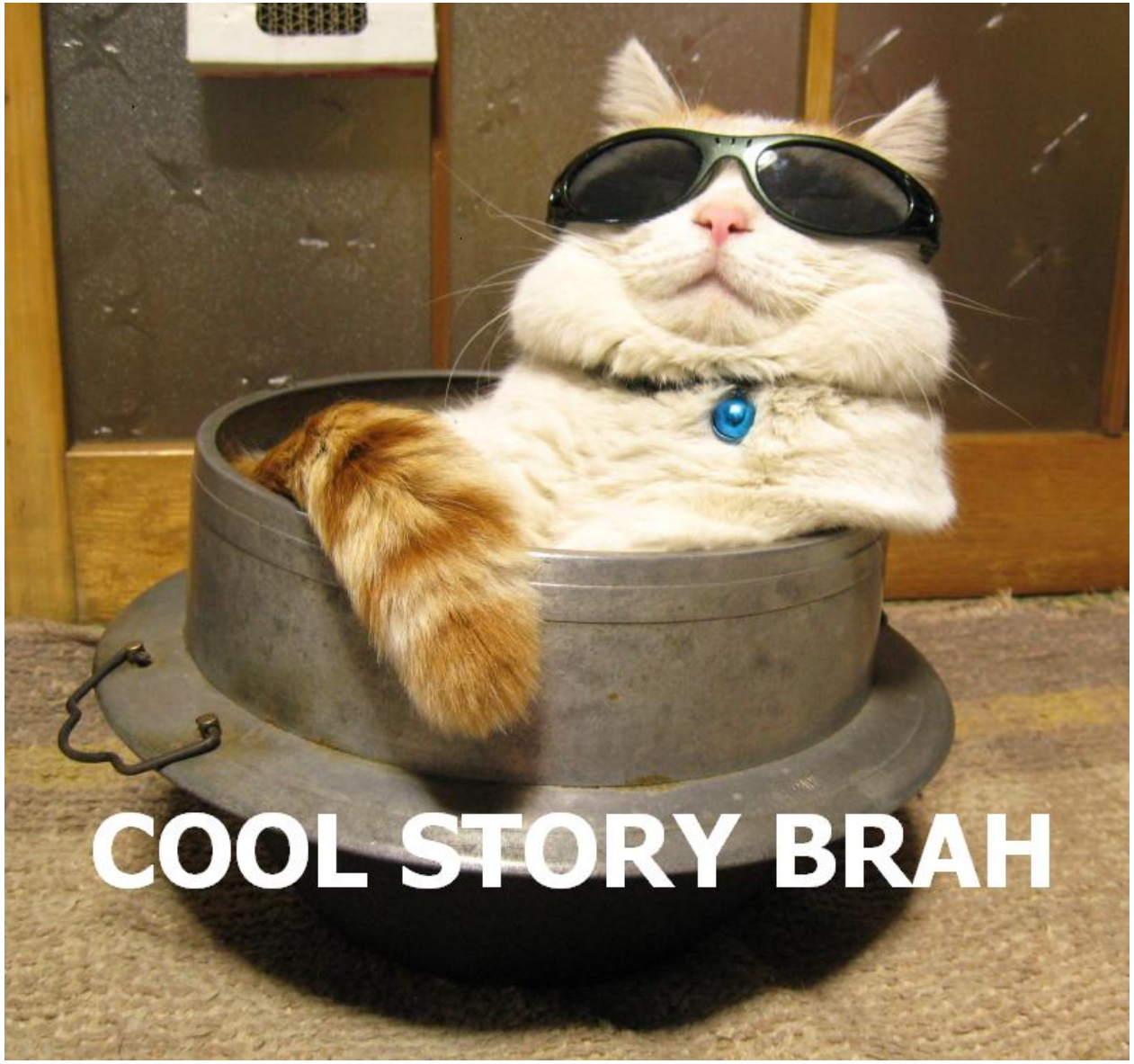
(a) Image from dataset (b) Clean image (c) Adv. image, $\epsilon = 4$ (d) Adv. image, $\epsilon = 8$

"We used images taken from a cell-phone camera as a input to an Inception v3 image classification neural network. We showed that in such a set-up, a significant fraction of adversarial images crafted using the original network are misclassified even when fed to the classifier through the camera."

- Kurakin et al.

Robust Physical-World Attacks on Machine Learning Models [2017]

or good luck perturbating the background in real life



Robust Physical-World Attacks on Machine Learning Models [2017]

Perturbations can also be constrained to mimic vandalism or art in order to go unreported by casual observers.

This concept allows the algorithm to create perturbations with much bigger magnitudes since we do not care about being perceived anymore

White-box defense

Adversarial training

We can generate adversarial samples and train the network to produce the correct classification on these new data points.

- Works against FGSM
- Expensive
- Does not defend subtler white-box attacks (*e.g.* I-FGSM or RAND+FGSM)
- Does not defend against black-box attack

Black-box defense

Ensemble adversarial training [Tramèr et al. May 2017]

Augment training data with adversarial inputs from a number of fixed pre-trained models.

- Best defense so far against black-box adversary
- Does not defend subtle white-box attacks

Error rate from 15.5% to 3.9% between adversarial trained and ensemble adversarial trained model on MNIST for a black-box attack.

Defending machine learning

Wide open problem

"Most defenses against adversarial examples that have been proposed so far just do not work very well at all, but the ones that do work are not adaptive. This means it is like they are playing a game of whack-a-mole: they close some vulnerabilities, but leave others open."

- Ian Goodfellow, Nicolas Papernot, February 2017

References

Research papers:

1. Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.
2. Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., & Swami, A. (2016). Practical black-box attacks against deep learning systems using adversarial examples. arXiv preprint arXiv:1602.02697.
3. Kurakin, A., Goodfellow, I., & Bengio, S. (2016). Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533.
4. Tramèr, F., Kurakin, A., Papernot, N., Boneh, D., & McDaniel, P. (2017). Ensemble Adversarial Training: Attacks and Defenses. arXiv preprint arXiv:1705.07204.
5. Tramèr, F., Papernot, N., Goodfellow, I., Boneh, D., & McDaniel, P. (2017). The Space of Transferable Adversarial Examples. arXiv preprint arXiv:1704.03453.
6. Evtimov, I., Eykholt, K., Fernandes, E., Kohno, T., Li, B., Prakash, A., Rahmati A. & Song, D. (2017). Robust Physical-World Attacks on Machine Learning Models. arXiv preprint arXiv:1707.08945.

Implementations:

1. github.com/tensorflow/cleverhans
2. github.com/rodgzilla/machine_learning_adversarial_examples

Targeted perturbation

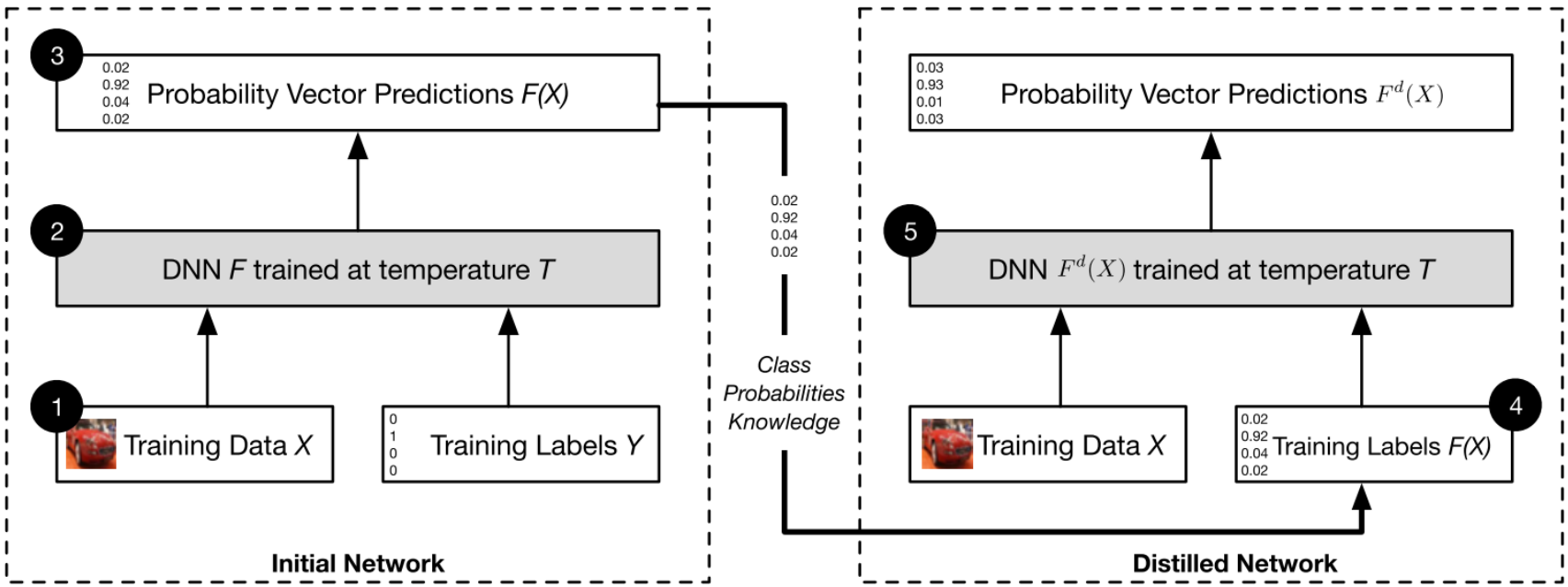
Papernot algorithm

This algorithm iteratively computes the adversarial saliency value $S(x,t)[i]$ of pixel i of the image x according to the class t.

$$S(x,t)[i] = \begin{cases} 0 \quad \text{if} \quad \frac{\mathrm{d}L_t}{\mathrm{d}x_i}(x) < 0 \quad \text{or} \quad \sum_{j \neq t} \frac{\mathrm{d}L_j}{\mathrm{d}x_i}(x) > 0 \\ \\ \frac{\mathrm{d}L_t}{\mathrm{d}x_i}(x) \left| \sum_{j \neq t} \frac{\mathrm{d}L_j}{\mathrm{d}x_i}(x) \right| \text{ otherwise.} \end{cases}$$

and use it iteratively to produce x_{adv} classified as t by the network.

Defensive distillation



Training a network with explicit relative information about classes prevents models from fitting too tightly to the data.

Papernot, N., McDaniel, P., Wu, X., Jha, S., & Swami, A. (2016). Distillation as a defense to adversarial perturbations against deep neural networks. In Security and Privacy (SP), 2016 IEEE Symposium on (pp. 582-597). IEEE.