images from *Geri's Game* (Pixar, 1997)

SIU2017

Adversarial Training

Attacks on Deep Networks and Generative Adversarial Networks

Erkut Erdem • Aykut Erdem • Levent Karacan

Computer Vision Lab, Hacettepe University

HACETTEPE UNIVERSITY COMPUTER VISION LAB

HACETTEPE UNIVERSITY

Outline

- **Part 1:** Attacks on Deep Networks

- **Part 2:** Generative Adversarial Networks (GANs)

10 Minutes Break

- **Part 3:** Image Editing with GANs

HACETTEPE UNIVERSITY COMPUTER VISION LAB

HACETTEPE UNIVERSITY

John Carpenter's The Thing (1982)

Part 1 – Attacks on Deep Networks

Erkut Erdem

Computer Vision Lab, Hacettepe University

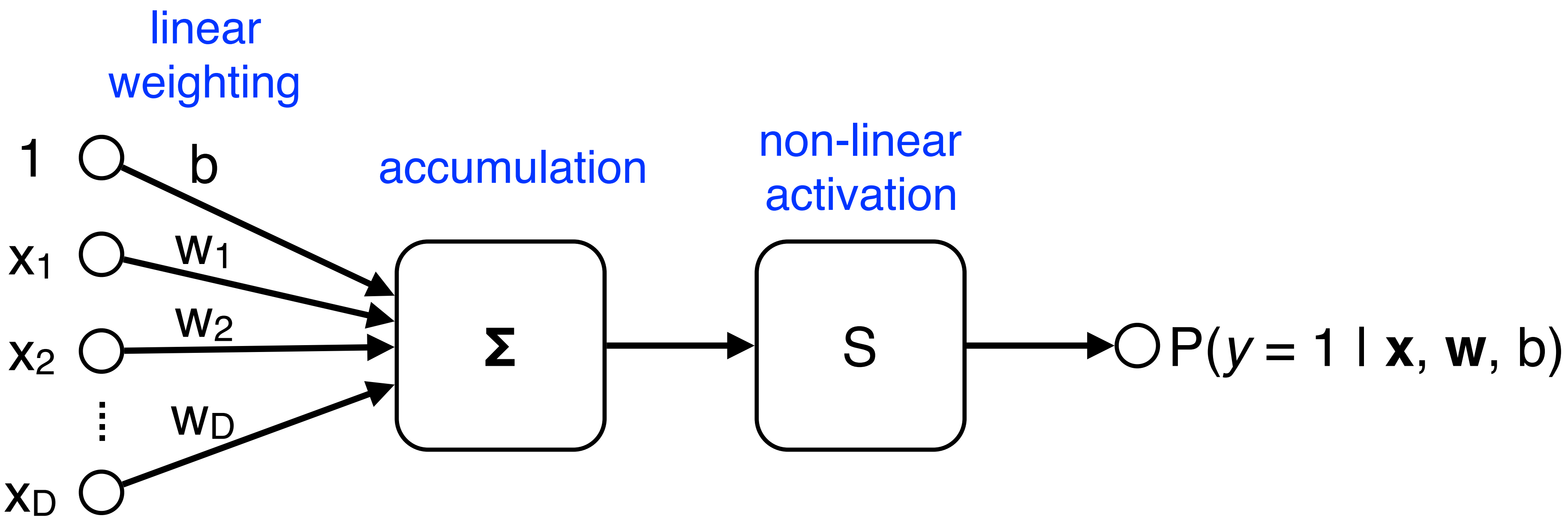
HACETTEPE UNIVERSITY COMPUTER VISION LAB

HACETTEPE UNIVERSITY

Deep Convolutional Networks
in 10 mins

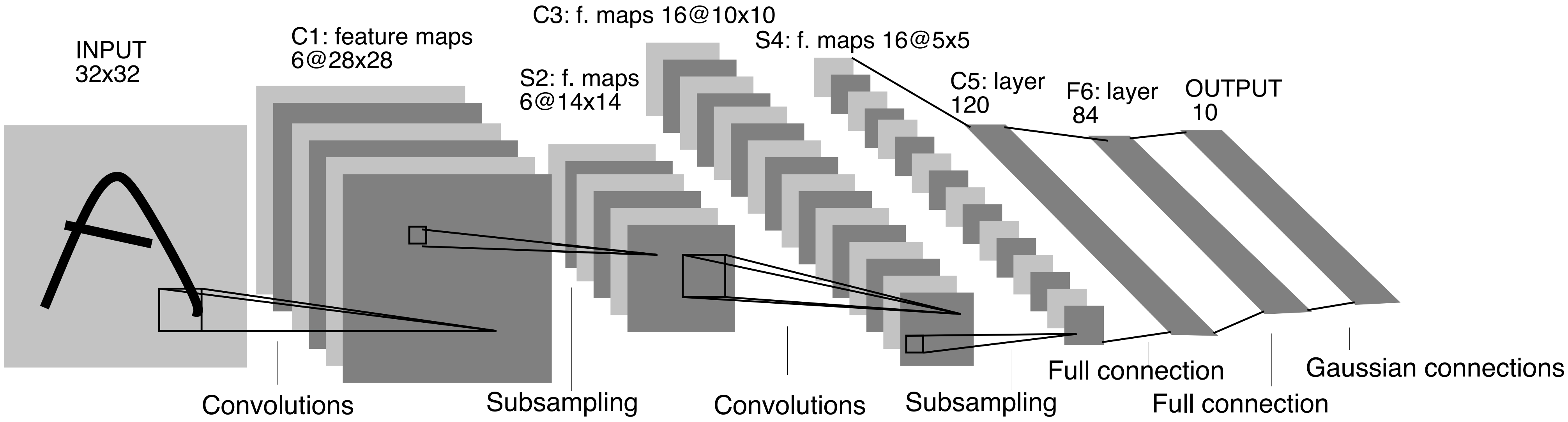
1^{st} Era (1940's-1960's): Invention

- Connectionism (Hebb 1940's): complex behaviors arise from interconnected networks of simple units
- Artificial neurons (Hebb, McCulloch and Pitts 1940's-1950's)
- Perceptron (Rosenblatt 1950's): Single layer with learning rule



2nd Era (1980's-1990's): Multi-layered Networks

- Back-propagation (Rumelhart, Hinton and Williams 1986 +others): effective way to train multi-layered networks
- Convolutional networks (LeCun et al. 1989): architecture adapted for images (inspired by Hubel and Wiesel's simple/complex cells)



Slide adapted from Rob Fergus

The Deep Learning Era (2011-present)

- Big gains in performance on perceptual tasks:
  - Vision
  - Speech understanding
  - Natural language processing

- Three ingredients:
  1. Deep neural network models (supervised training)
  2. Big labeled datasets
  3. Fast GPU computation

Slide credit: Rob Fergus

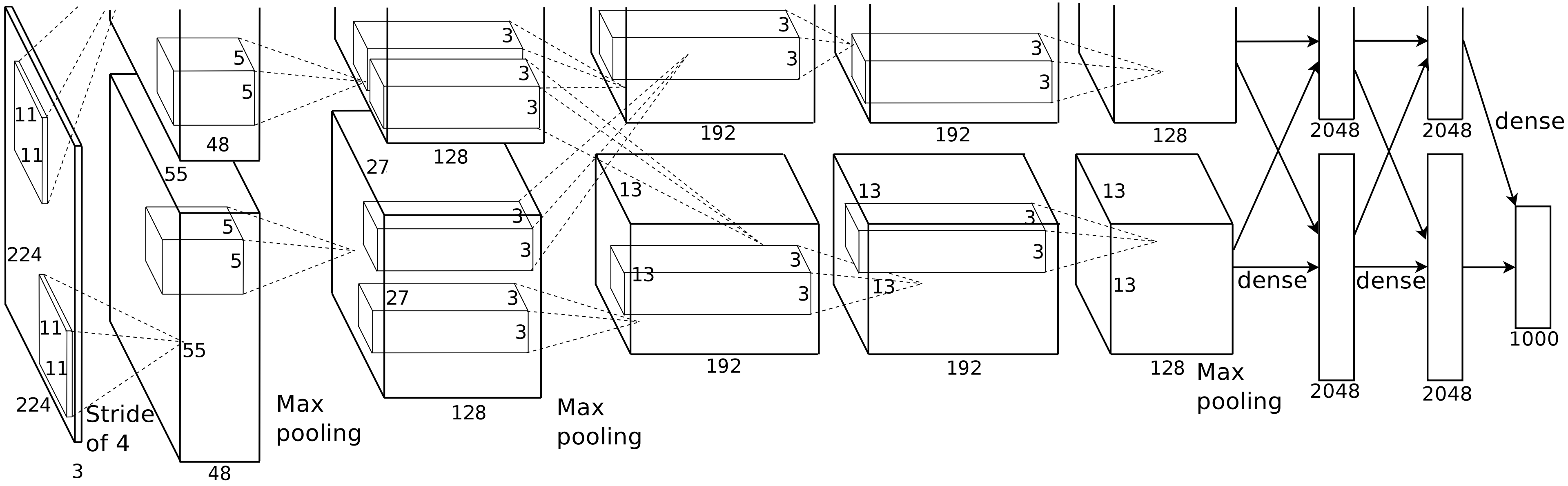
Powerful Hardware

- Deep neural nets highly amenable to implementation on Graphics Processing Units (GPUs)
  - Matrix multiplication
  - 2D convolution

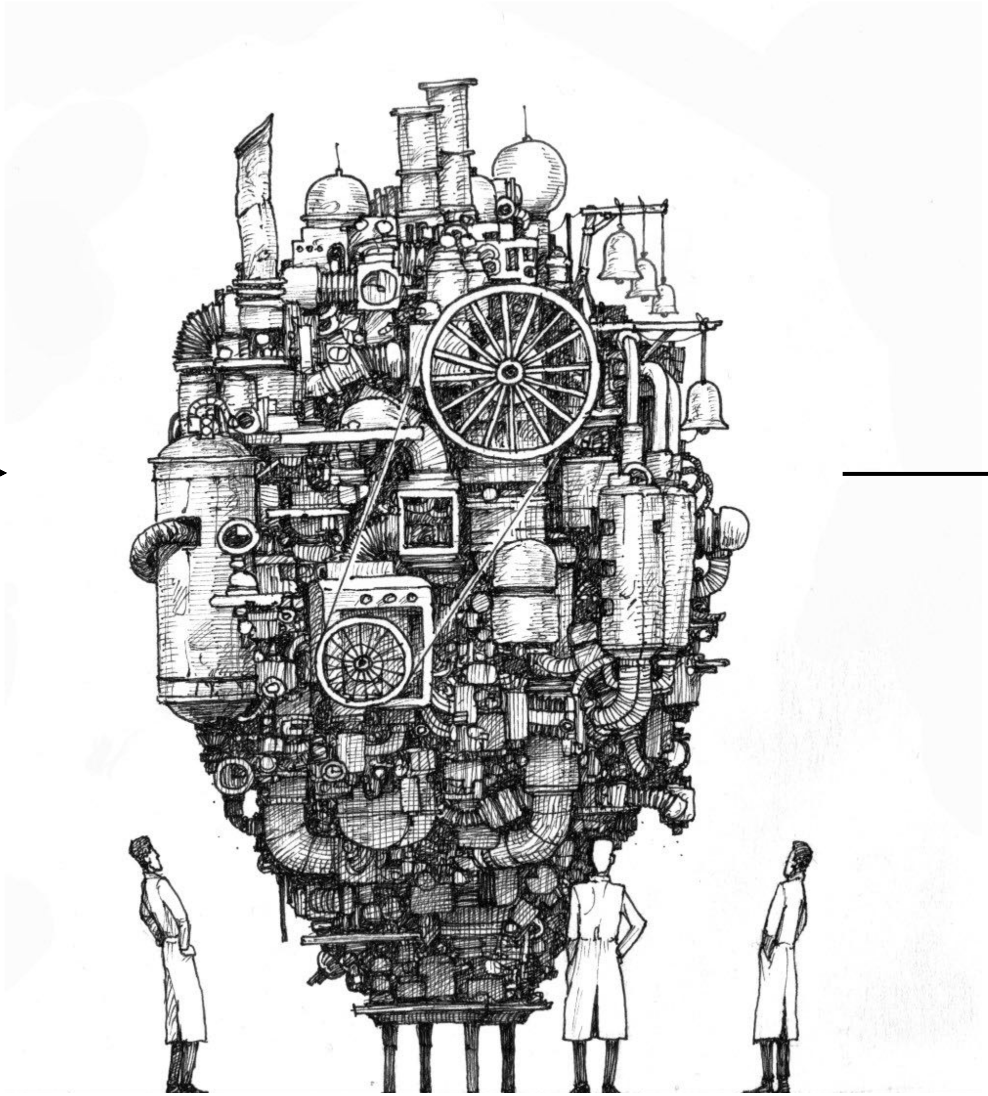
- Latest generation nVidia GPUs (Pascal) deliver 10 Tflops
  - Faster than fastest computer in the world in 2000
  - 10 million times faster than 1980's Sun workstation

AlexNet: The Model That Changed The History

- Krizhevsky, Sutskever and Hinton (2012)
  - 8 layer Convolutional network model [LeCun et al. 1989]
  - 7 hidden layers, 650,000 neurons, ~60,000,000 parameters
  - Trained on 1.2 million ImageNet images (with labels)
  - GPU implementation (50x speedup over CPU)
  - Training time: 1 week on pair of GPUs



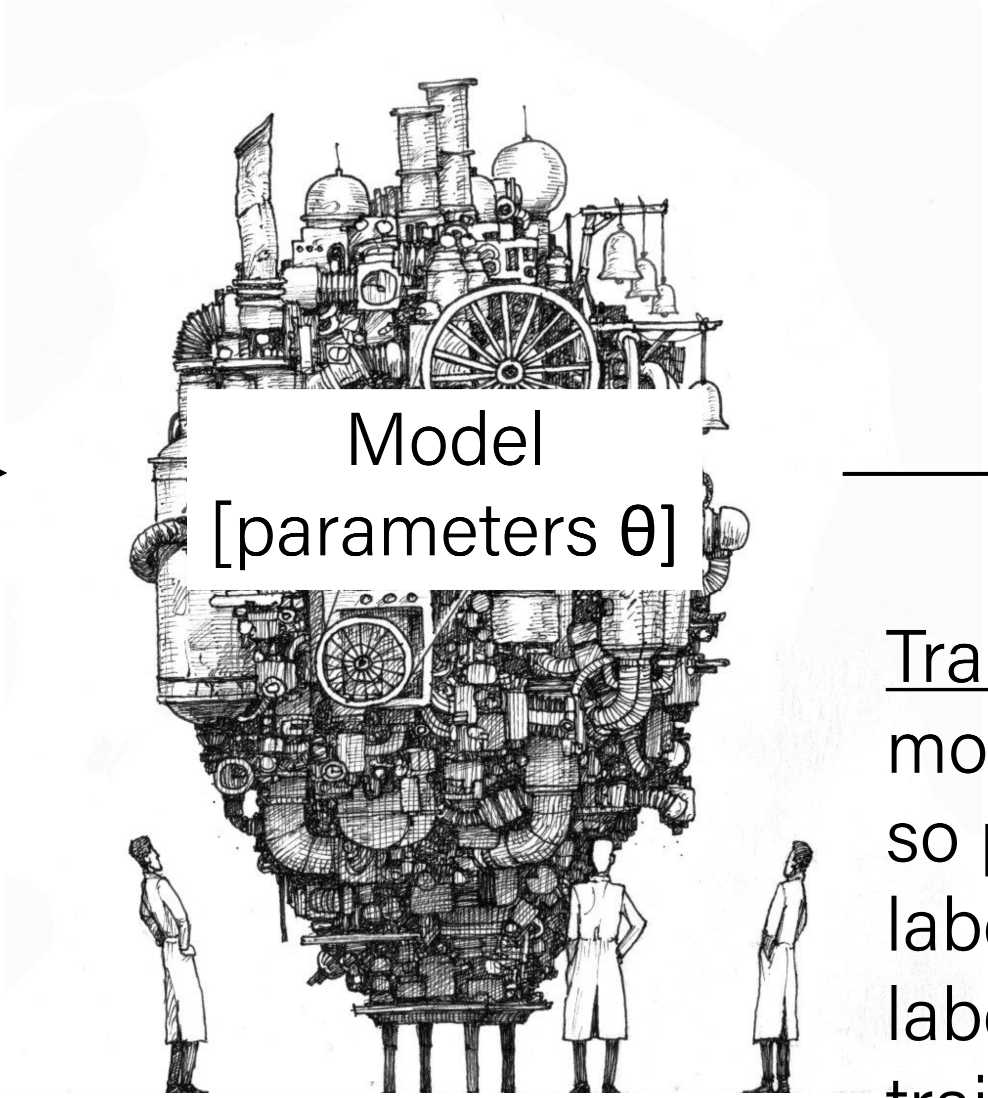
Supervised Learning: Image Classification



Joshua Drewe

"Cat"

Supervised Learning: Image Classification

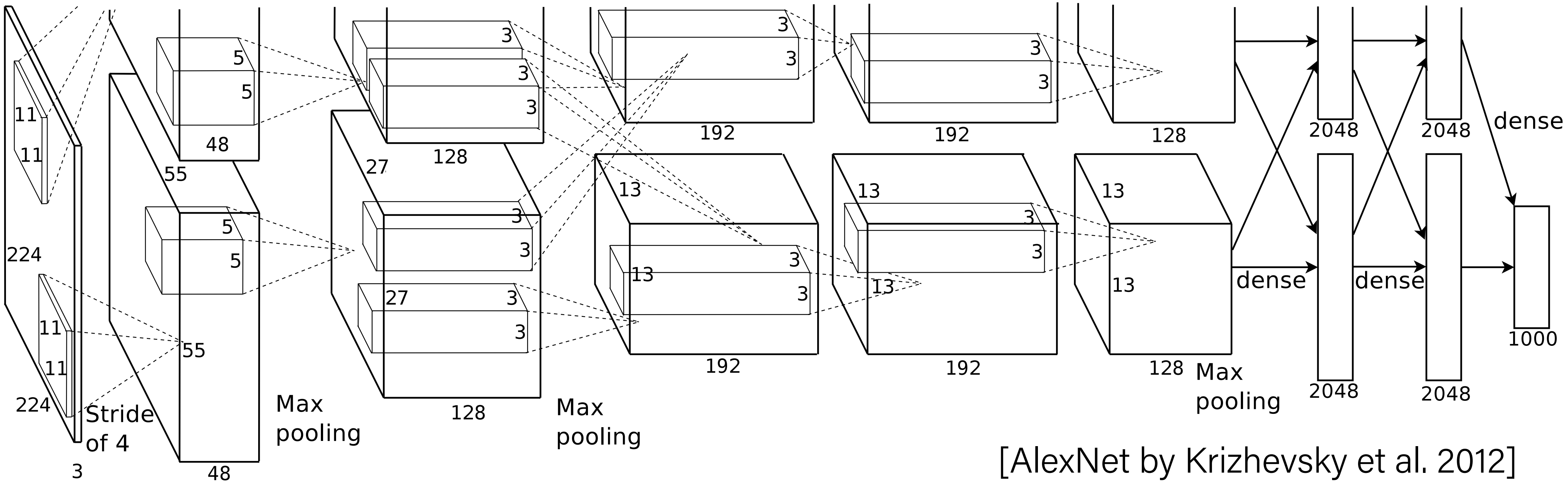


Joshua Drewe

"Cat"

<u>Training:</u> Adjust model parameters θ so predicted labels match true labels across training set

Modern Convolutional Nets



[AlexNet by Krizhevsky et al. 2012]

Excellent **performance** in most image understanding tasks

Learn a sequence of **general-purpose representations**

Millions of parameters learned from data

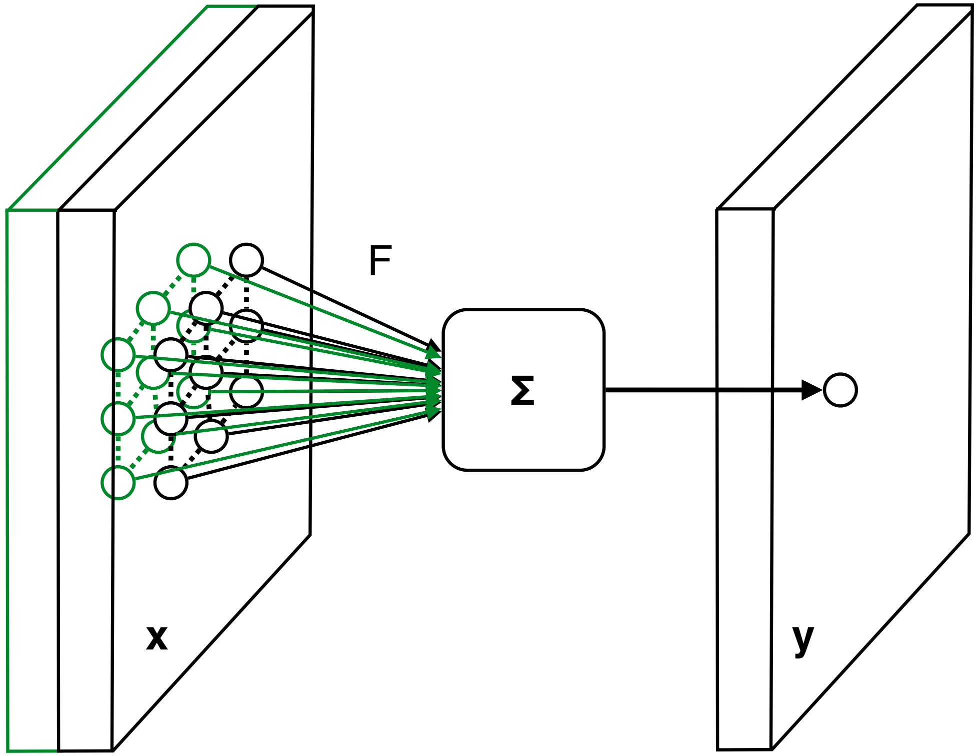
The "**meaning**" of the representation is unclear

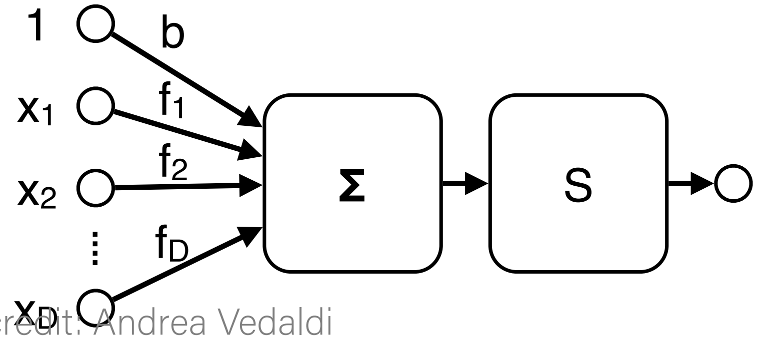
Convolutions with Filters

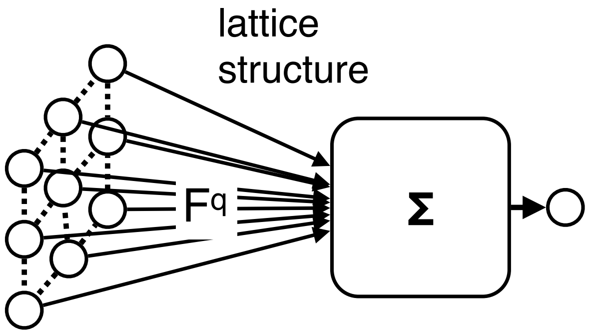
- Each filter acts on multiple input channel

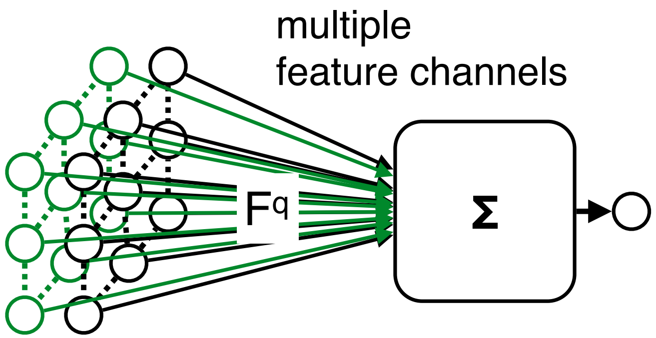
  – **Convolution is local**
  Filters look locally
  Parameter sharing

  – **Translation invariant**
  Filters act the same everywhere



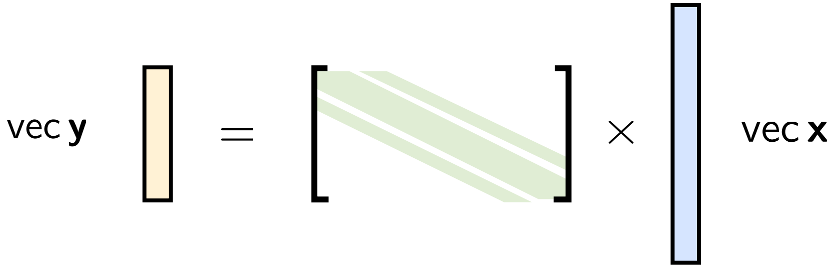






Slide credit: Andrea Vedaldi

Convolution



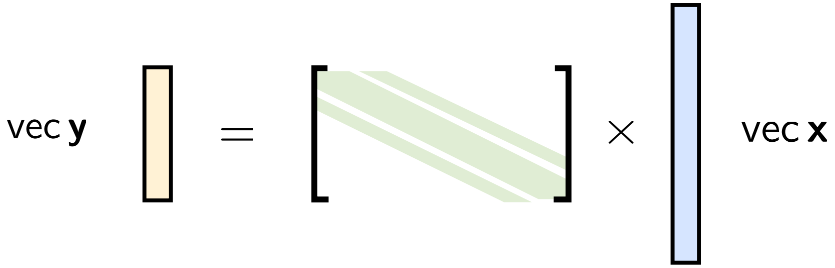
- Convolution = Spatial filtering

$$(a \star b)[i,j] = \sum_{i',j'} a[i',j']b[i-i',j-j']$$

- Different filters (weights) reveal a different characteristics of the input.

$$* \, 1/8 \begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Convolution



- Convolution = Spatial filtering

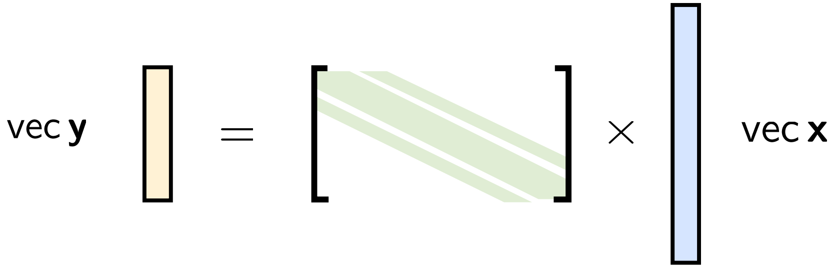
$$(a \star b)[i, j] = \sum_{i', j'} a[i', j'] b[i - i', j - j']$$

- Different filters (weights) reveal a different characteristics of the input.

$*$

0	-1	0
-1	4	-1
0	-1	0

Convolution



- Convolution = Spatial filtering

$$(a \star b)[i,j] = \sum_{i',j'} a[i',j']b[i-i',j-j']$$

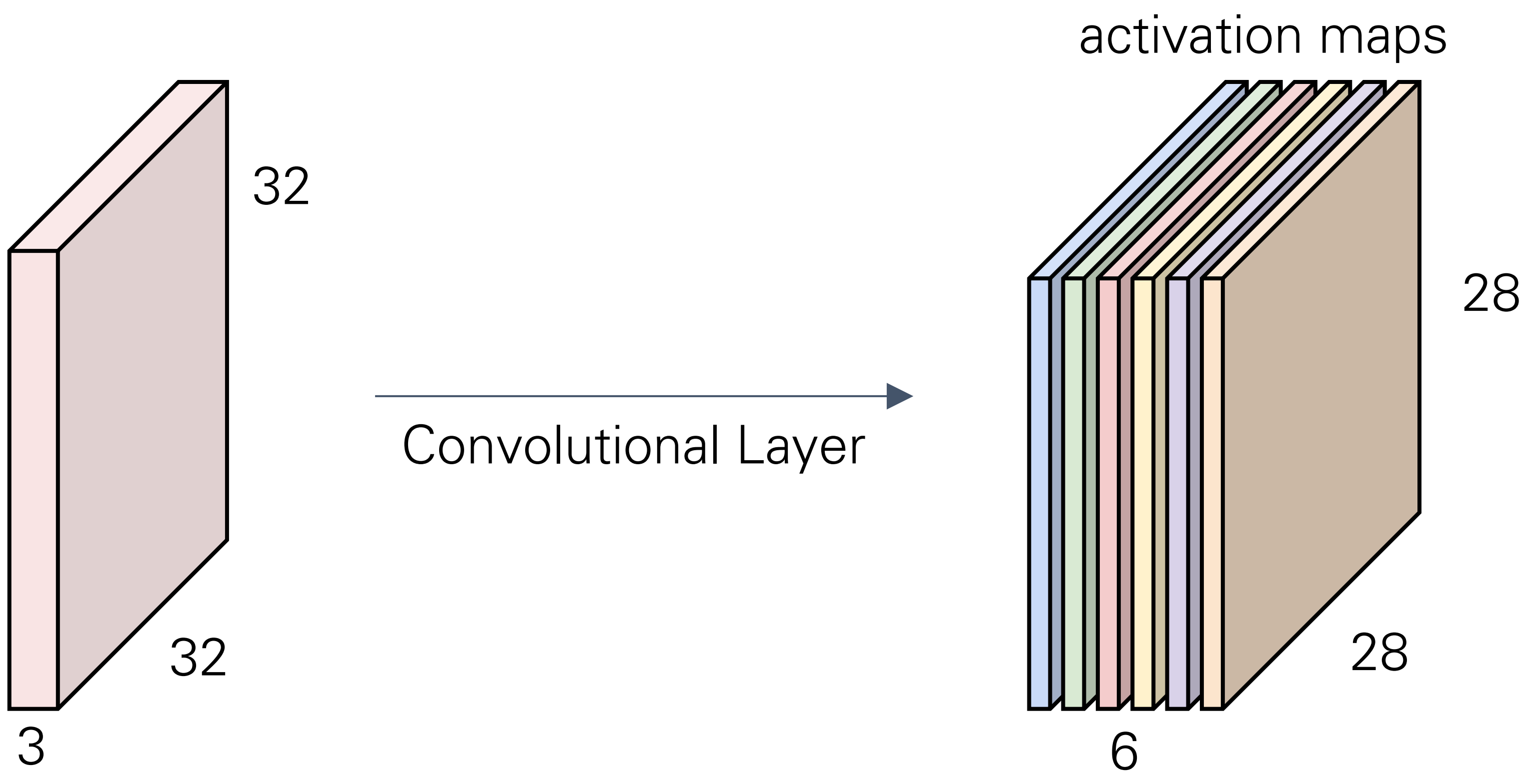
- Different filters (weights) reveal a different characteristics of the input.

$*$

1	0	-1
2	0	-2
1	0	-1

Convolutional Layer

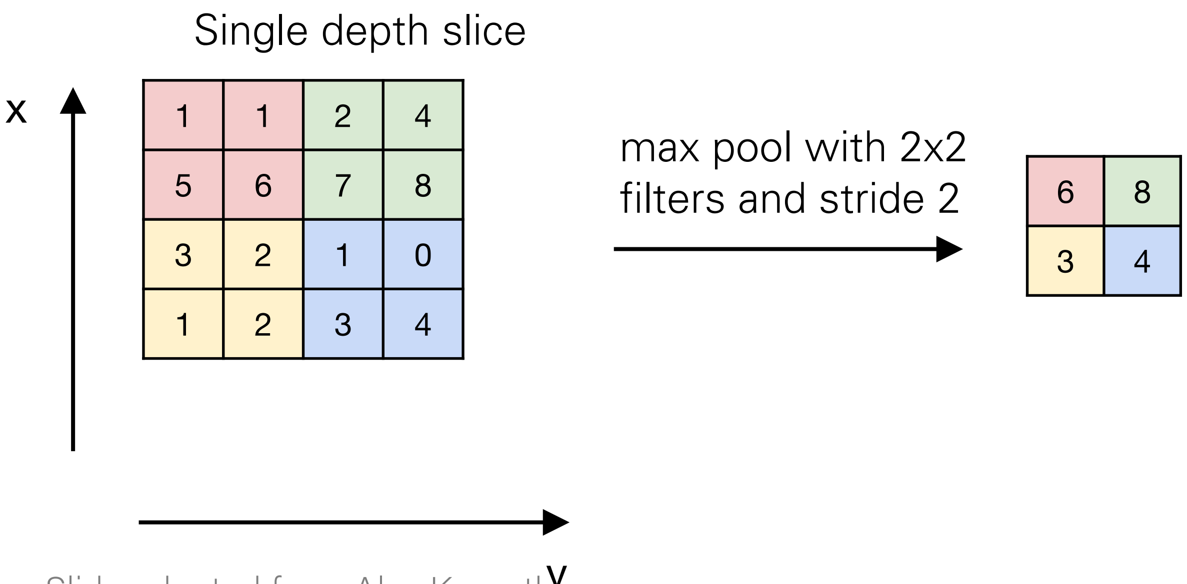
- Multiple filters produce multiple output channels
- For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

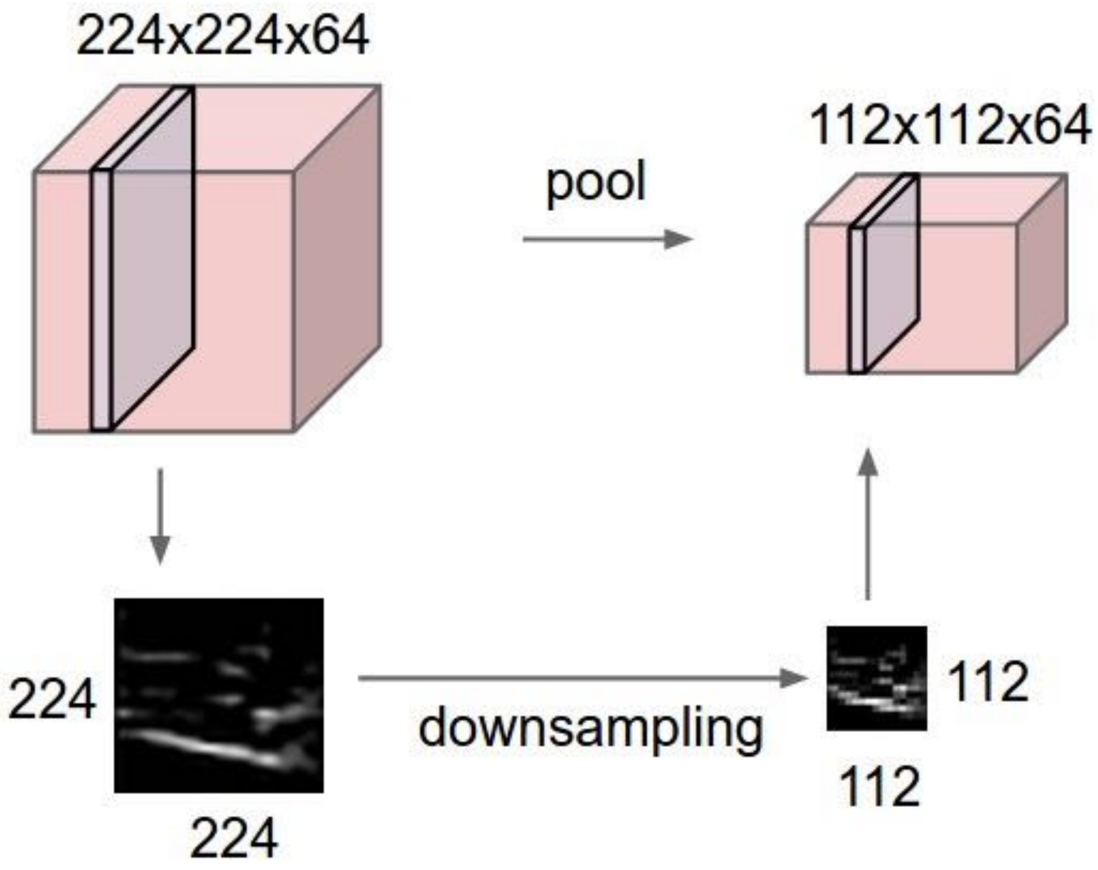


We stack these up to get an output of size 28x28x6.

Pooling Layer

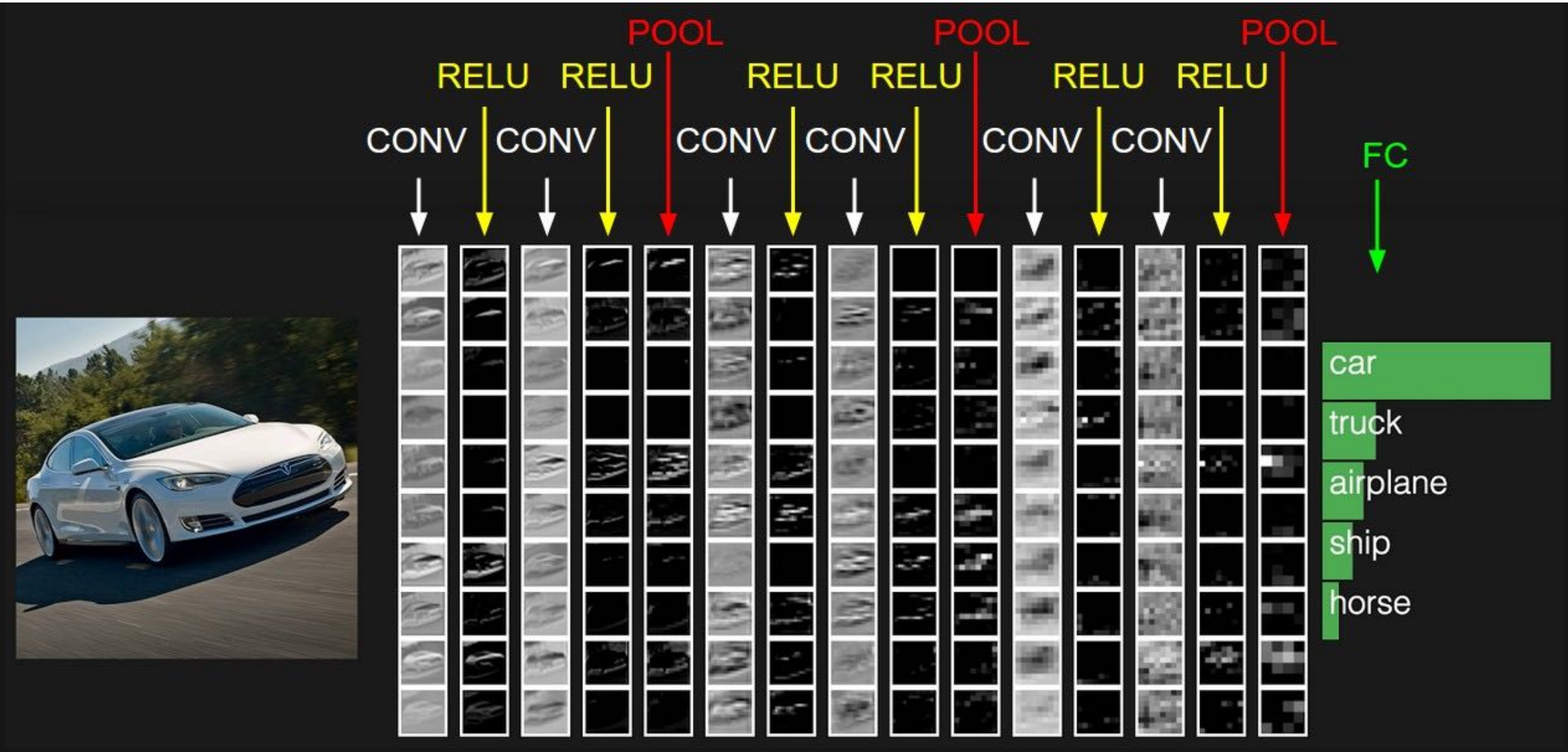
- makes the representations smaller and more manageable
- operates over each activation map independently:
- Max pooling, average pooling, etc.





Fully Connected Layer

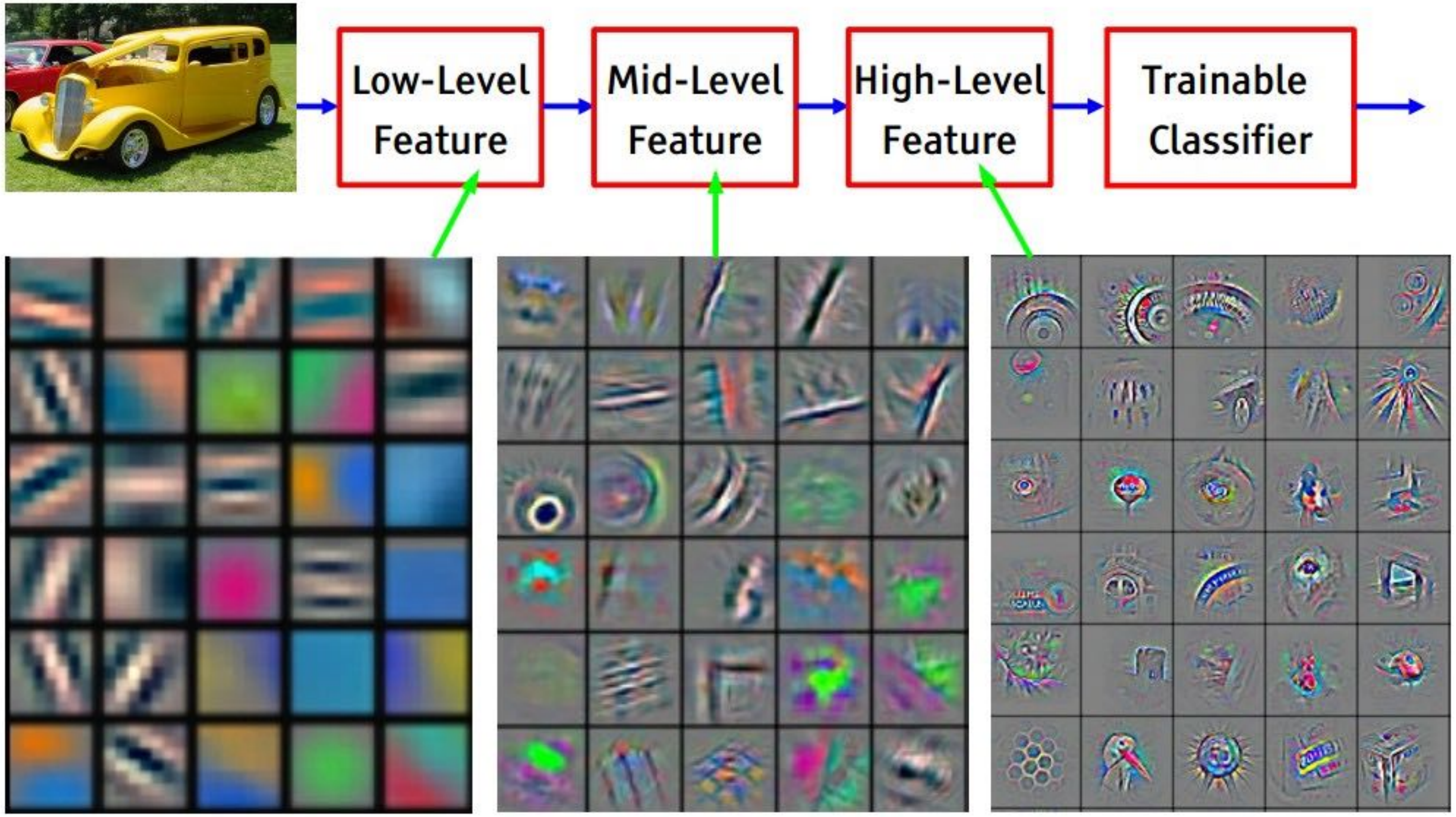
- contains neurons that connect to the entire input volume, as in ordinary Neural Networks



Slide credit: Alex Karpathy

Feature Learning

- Hierarchical layer structure allows to learn hierarchical filters (features).



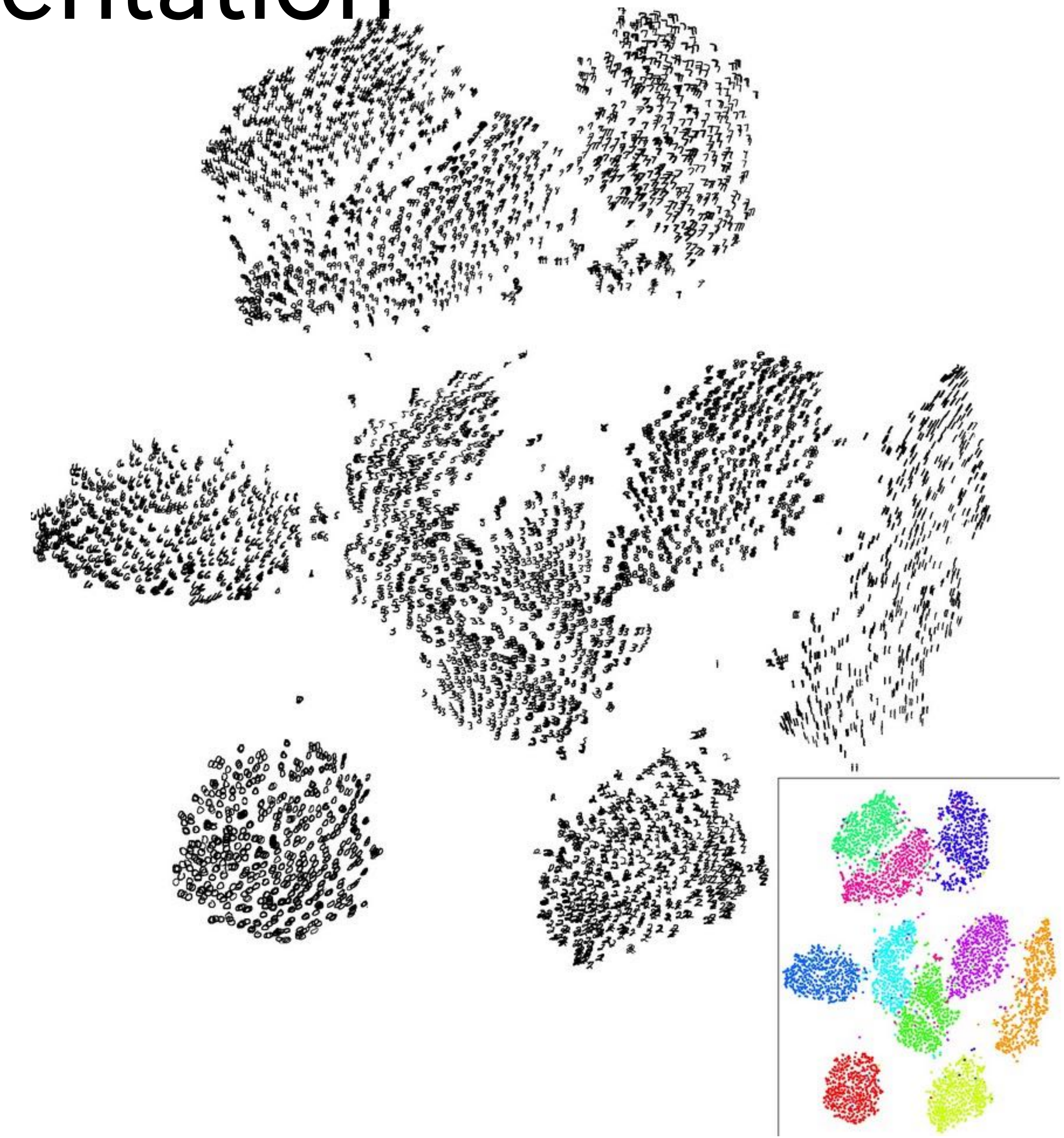
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Slide credit: Yann LeCun

Visualizing The Representation

t-SNE visualization
(van der Maaten & Hinton)

- Embed high-dimensional points so that locally, pairwise distances are conserved
- i.e. similar things end up in similar places. dissimilar things end up wherever
- **Right**: Example embedding of MNIST digits (0-9) in 2D

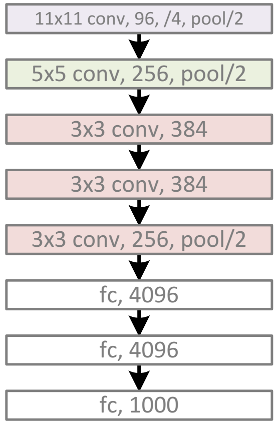


Slide credit: Alex Karpathy

Three Years of Progress

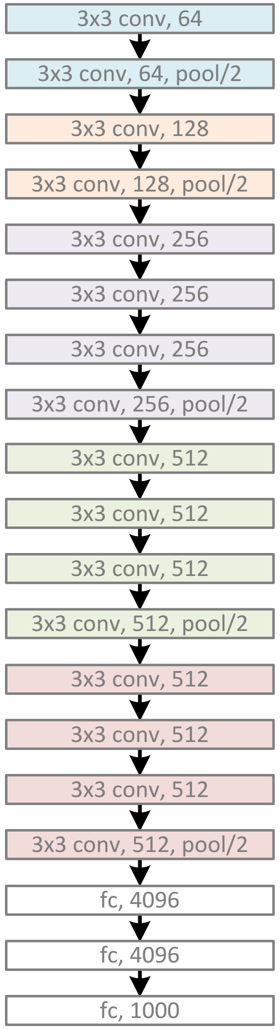
AlexNet, 8 layers

(ILSVRC 2012)



VGG, 19 layers

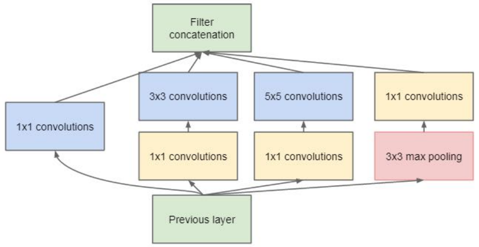
(ILSVRC 2014)

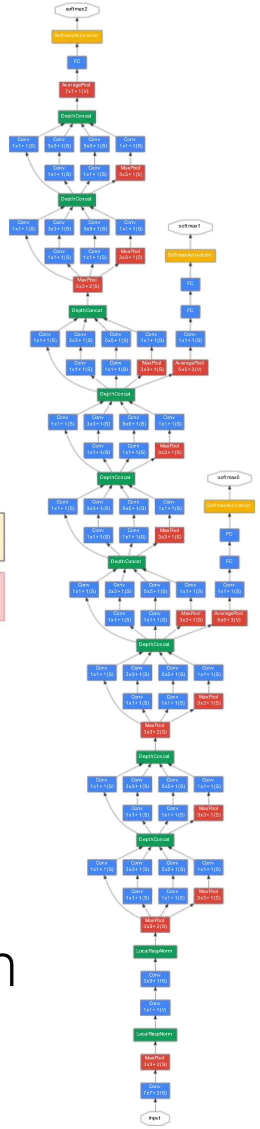


- Very deep
- Simply deep

GoogLeNet,
22 layers

(ILSVRC 2014)



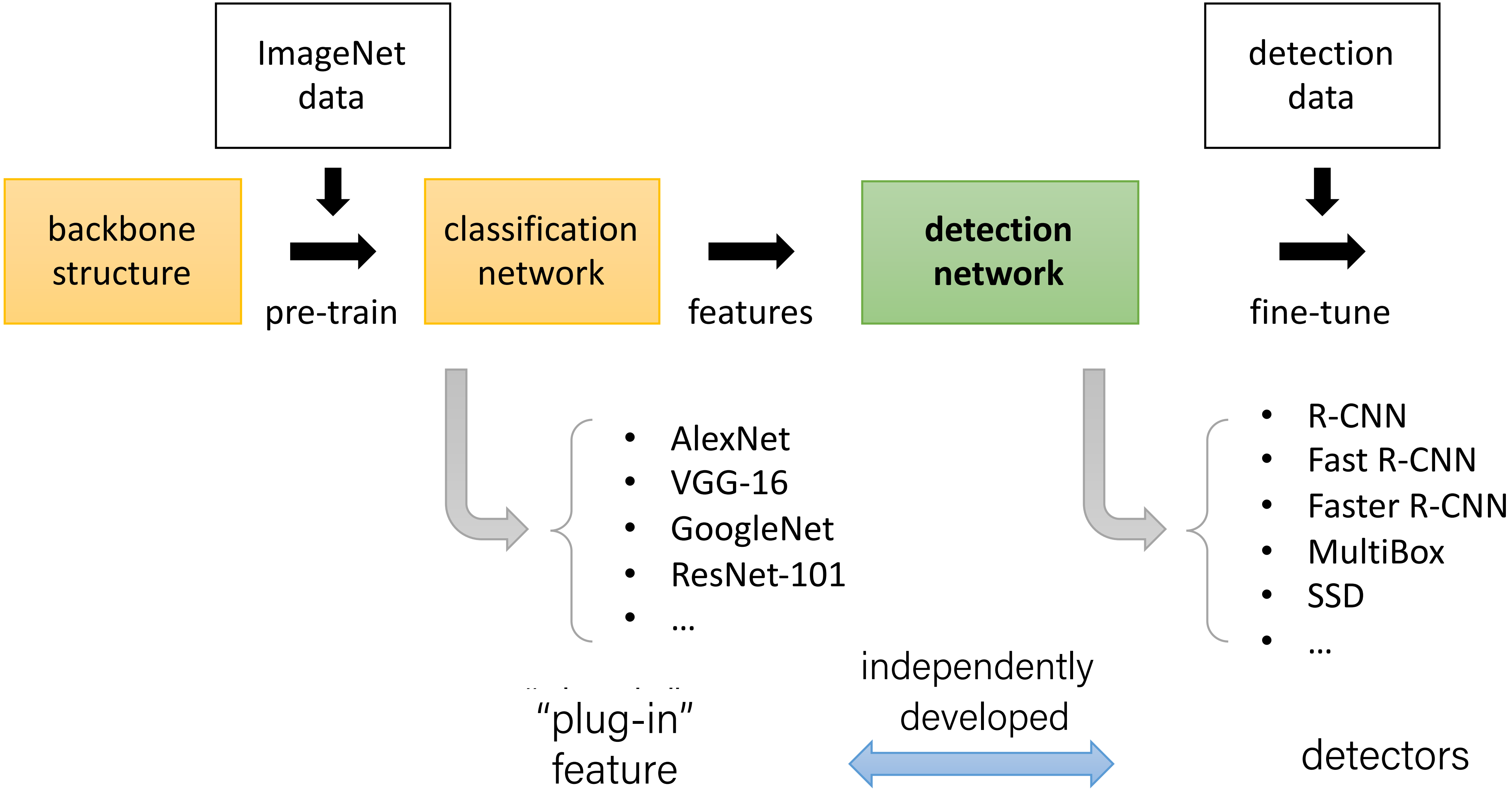


- Branching
- Bottleneck
- Skip connection

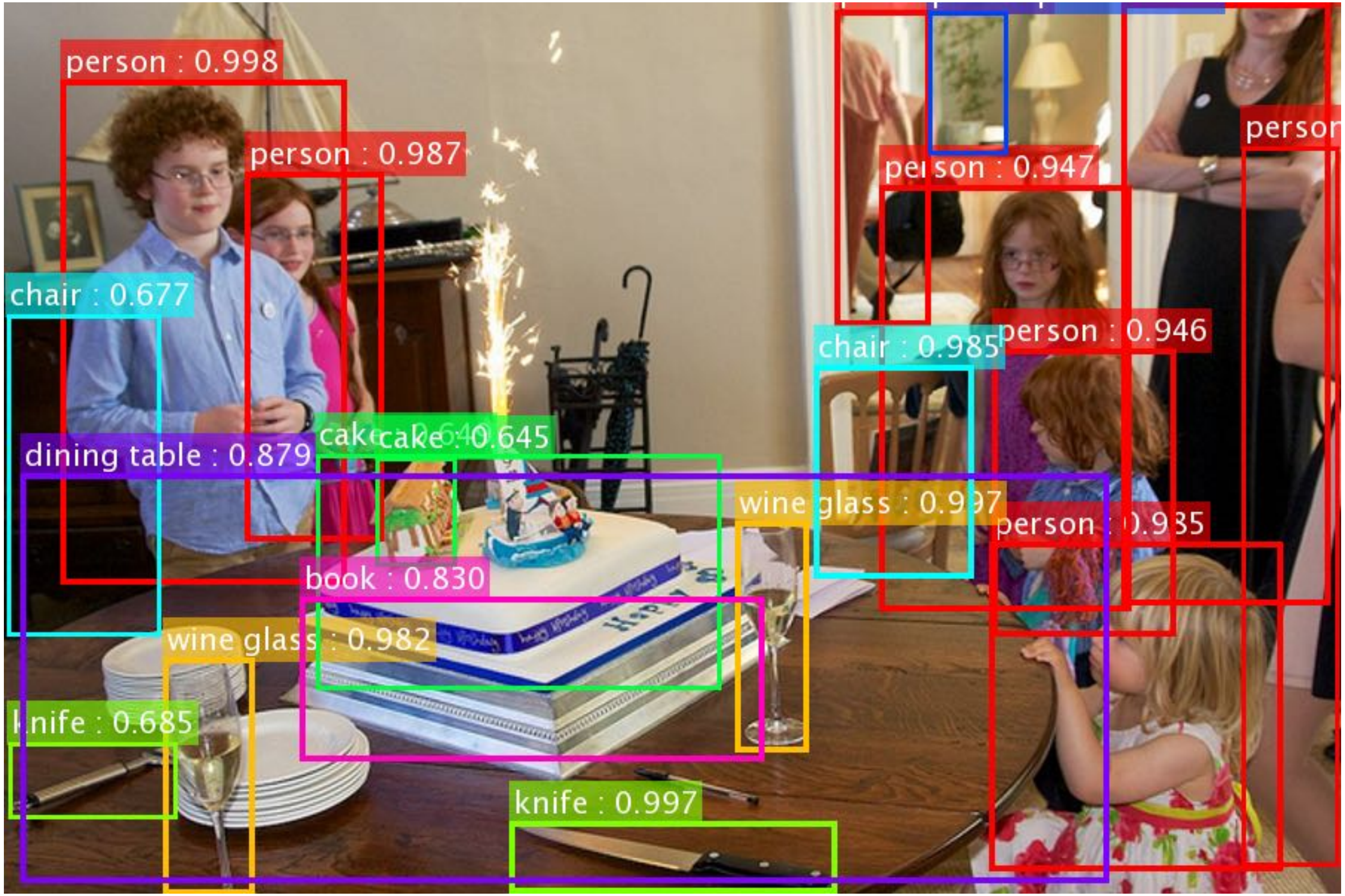
Training Deep Neural Networks

- The network is trained by stochastic gradient descent.
- Backpropagation is used similarly as in a fully connected network.
- Pass gradients through element-wise activation function.
- We also need to pass gradients through the convolution operation and the pooling operation.

Object Detection Networks



ResNet's Object Detection Results on COCO

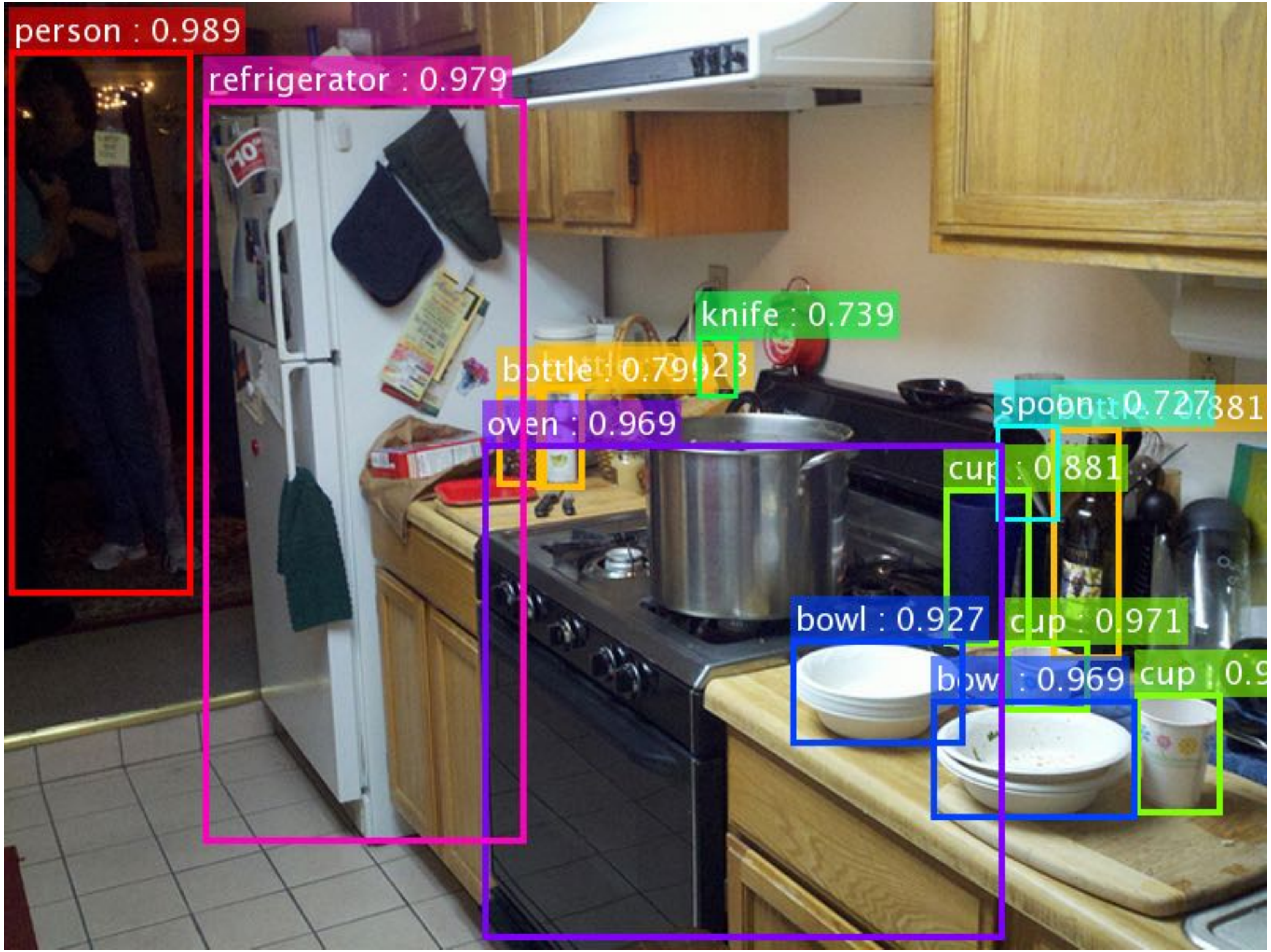


Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. Deep Residual Learning for Image Recognition. CVPR 2016.

Shaoqing Ren, Kaiming He, Ross Girshick, & Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NIPS 2015.

Slide credit: Kaiming He

ResNet's Object Detection Results on COCO



Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. Deep Residual Learning for Image Recognition. CVPR 2016.

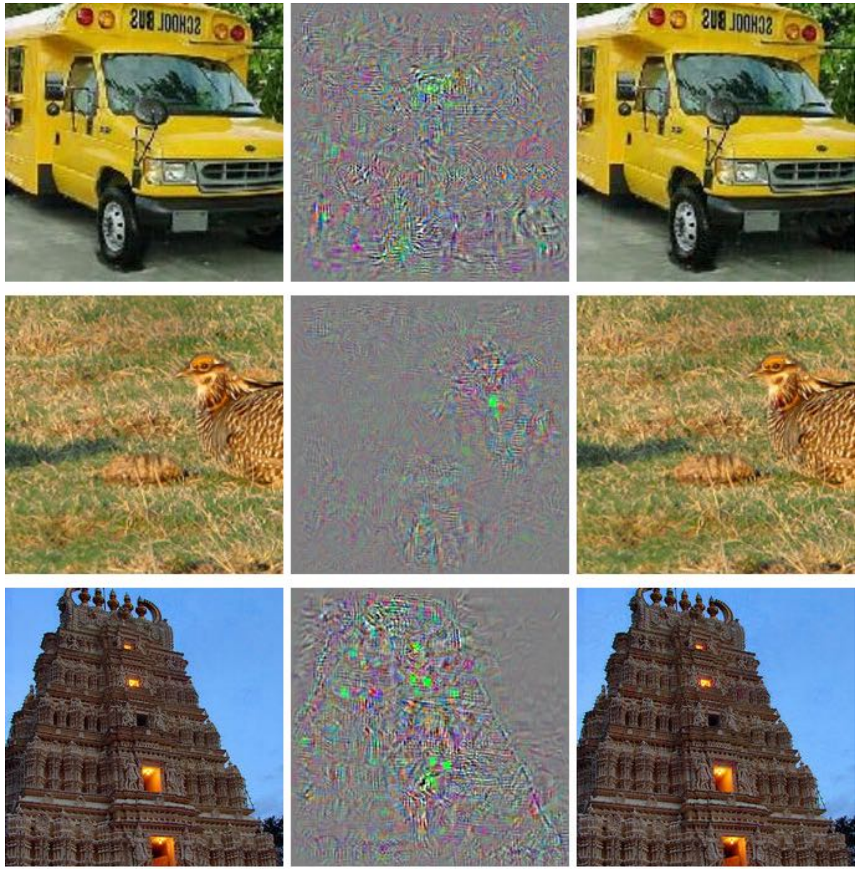
Shaoqing Ren, Kaiming He, Ross Girshick, & Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NIPS 2015.

Slide credit: Kaiming He

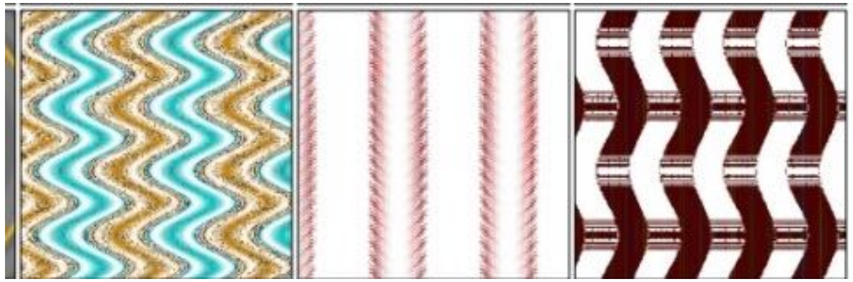
Story isn't over yet!

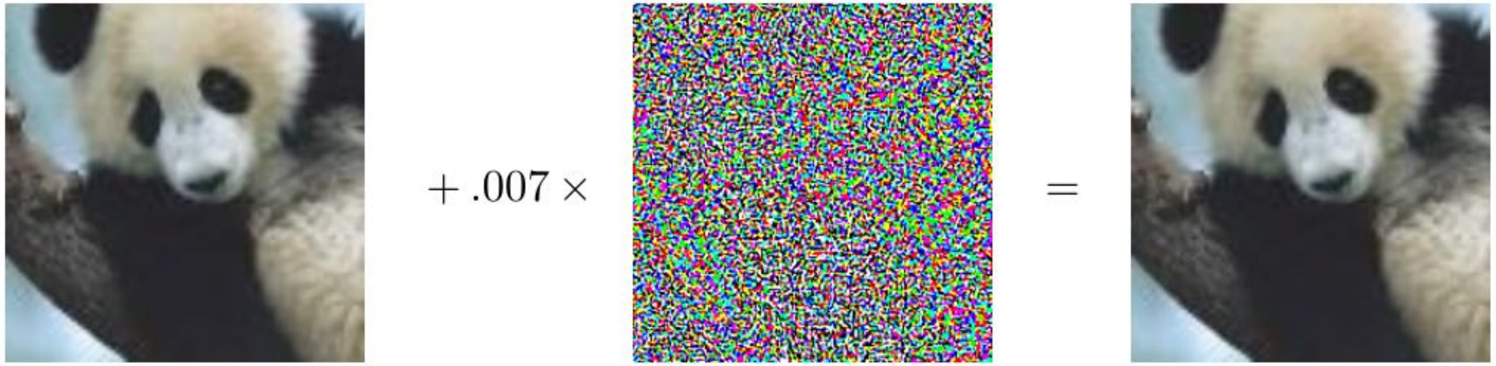
Story isn't over yet!

... we have reached
the point where ML works,
but let’s see how it can be
easily fooled.



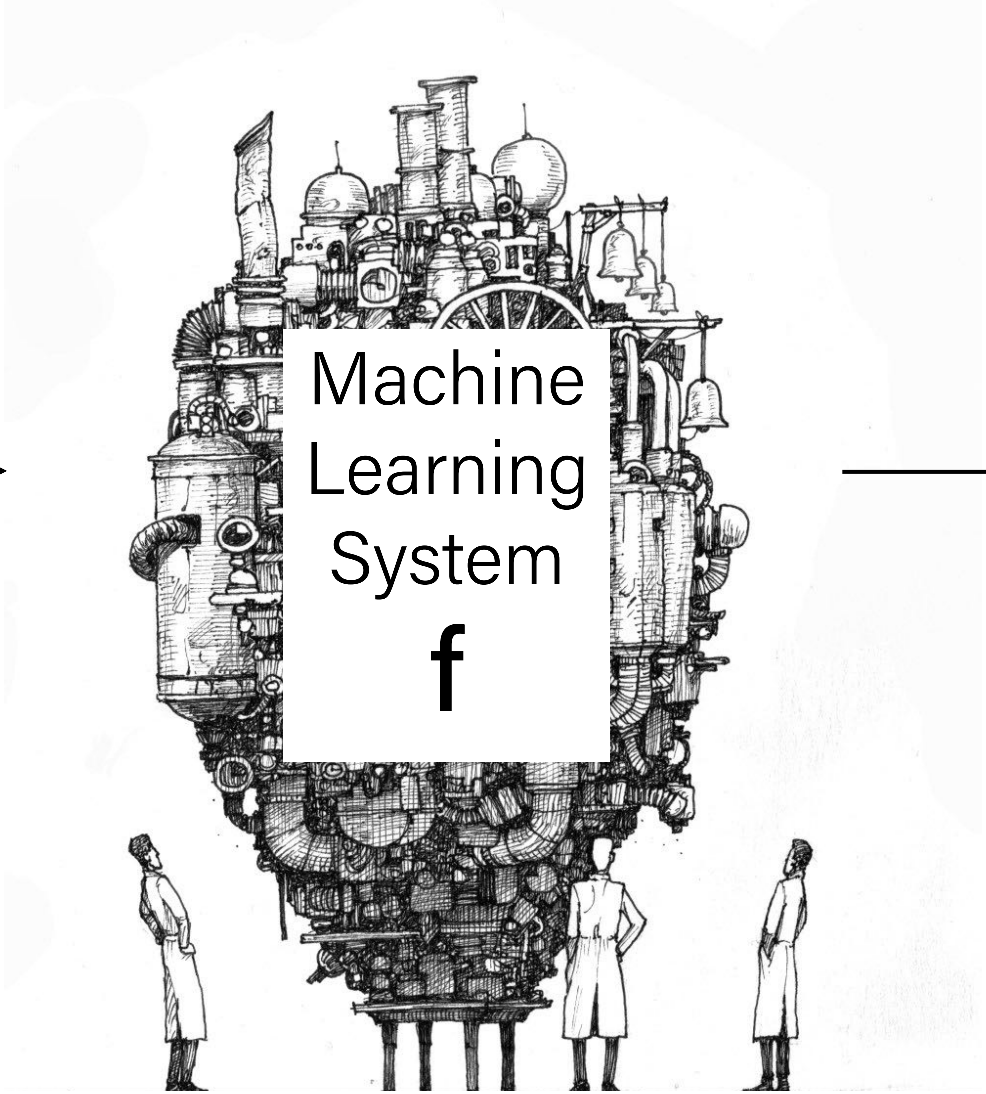
Adversarial Examples





Machine Learning System

Sample x



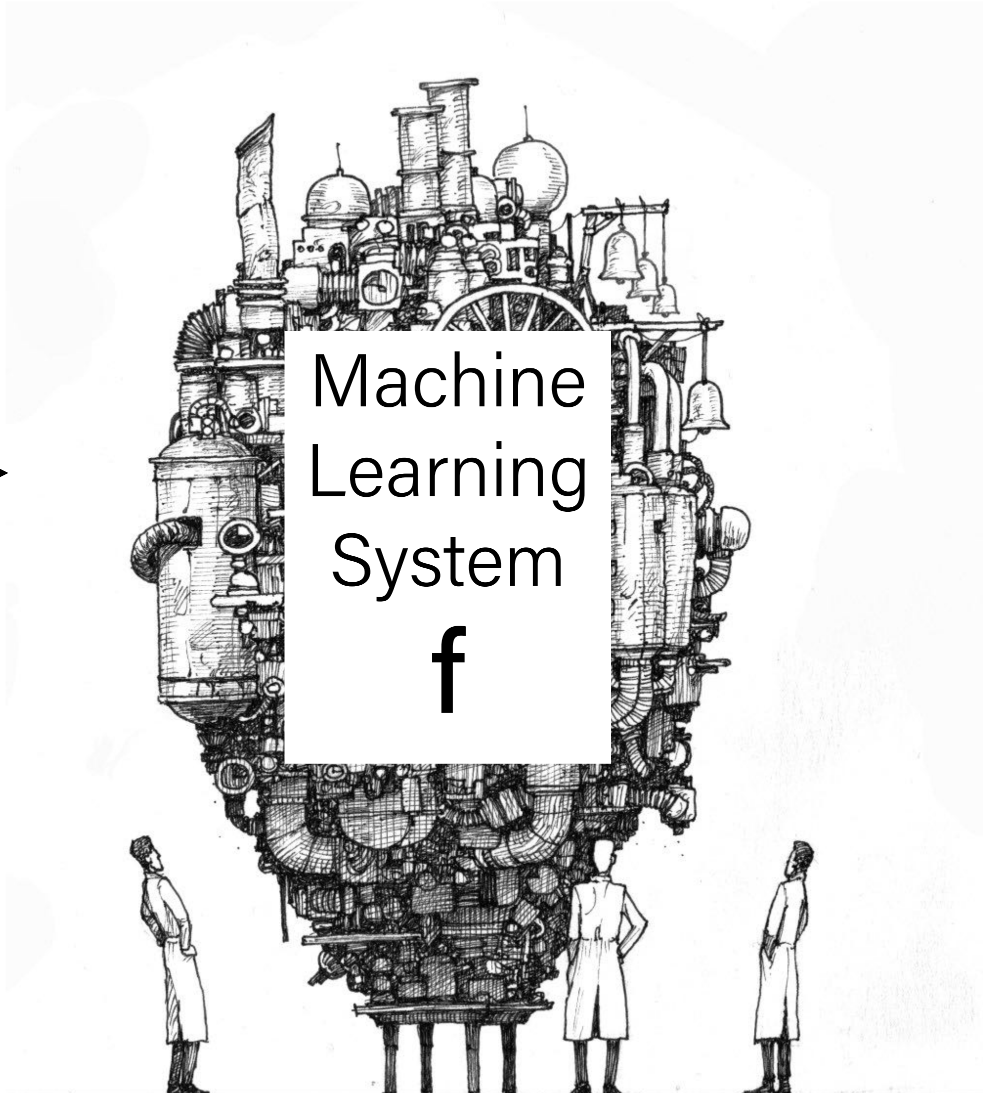
Joshua Drewe

“Cat”

$f(x)=y_{true}$

Adversarial Examples

Adversarial example a
(indistinguishable from x)

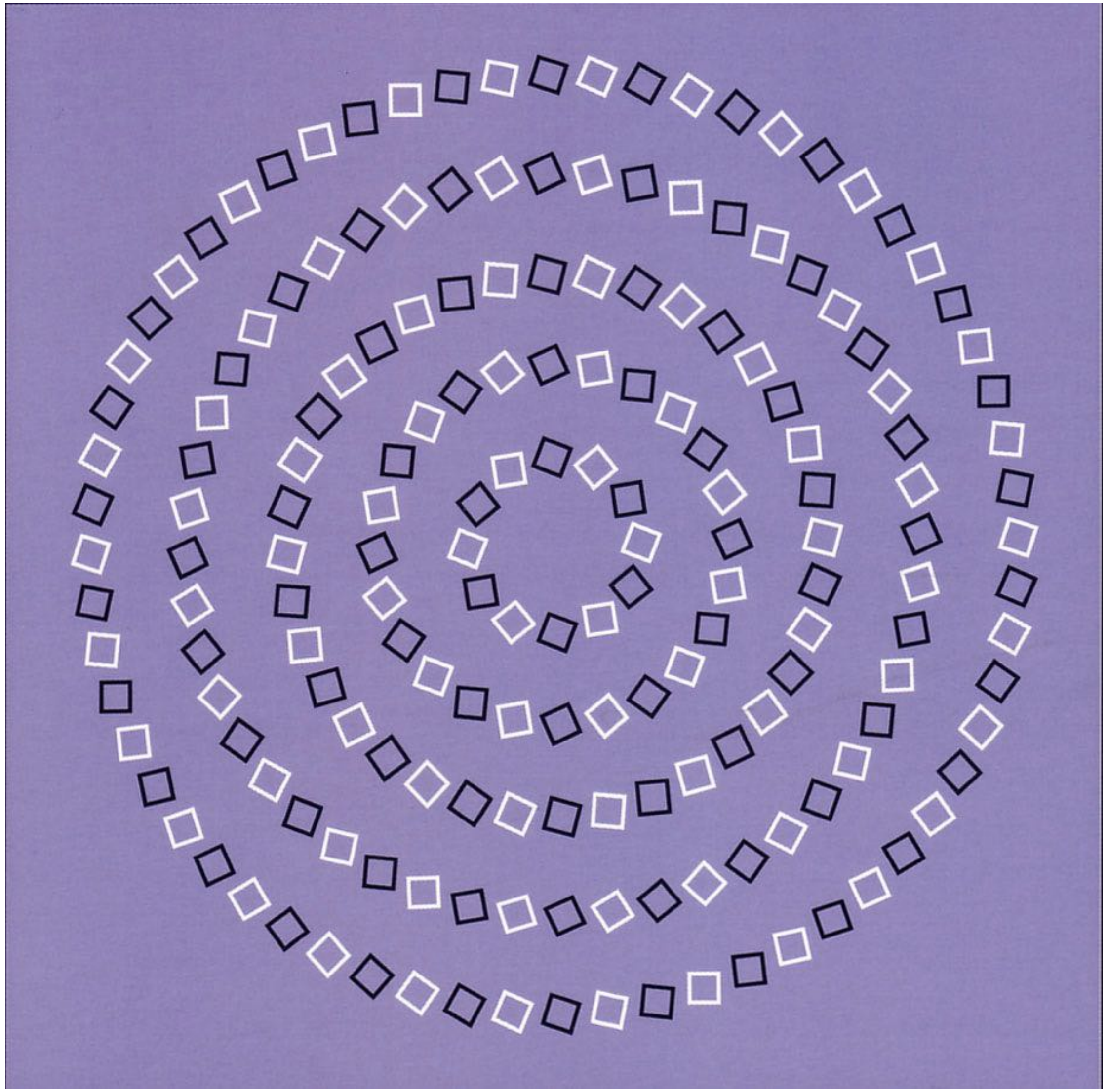


Joshua Drewe

→ "Dog"

f(a)≠y_{true}

Adversarial Examples in The Human Brain

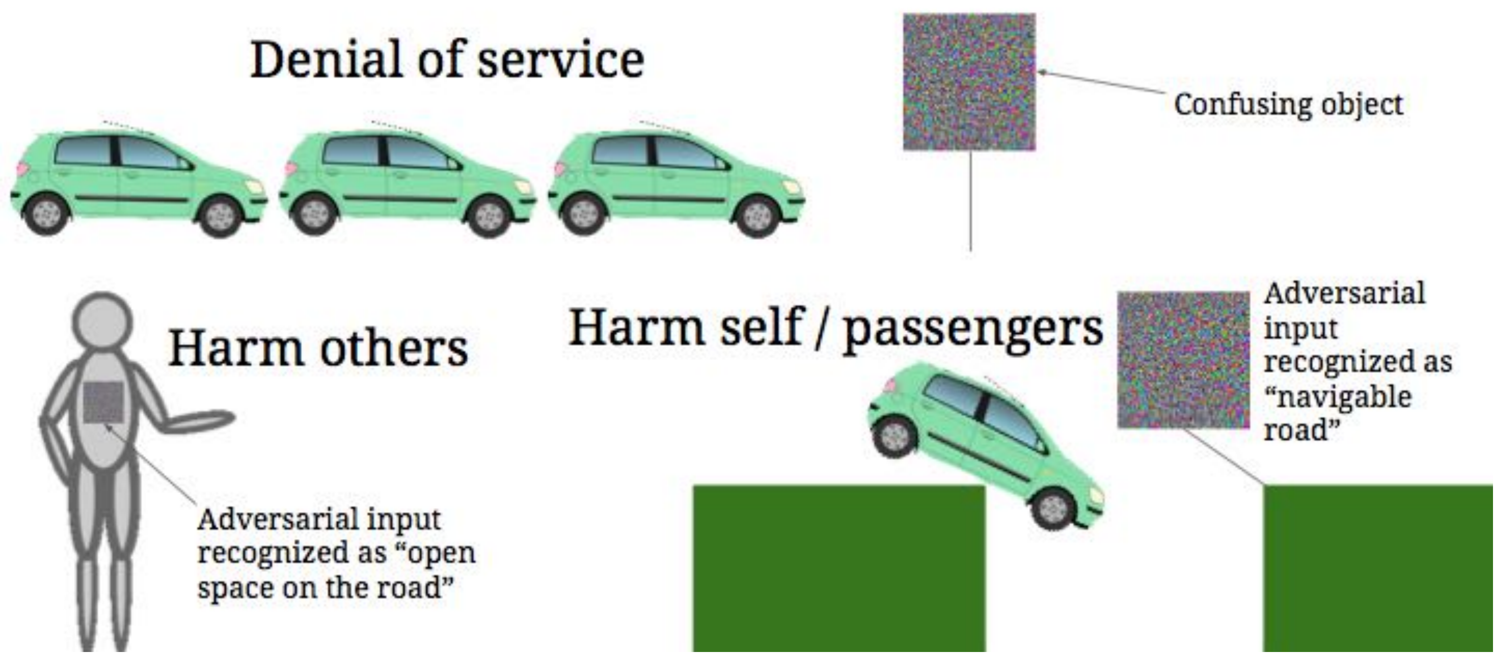


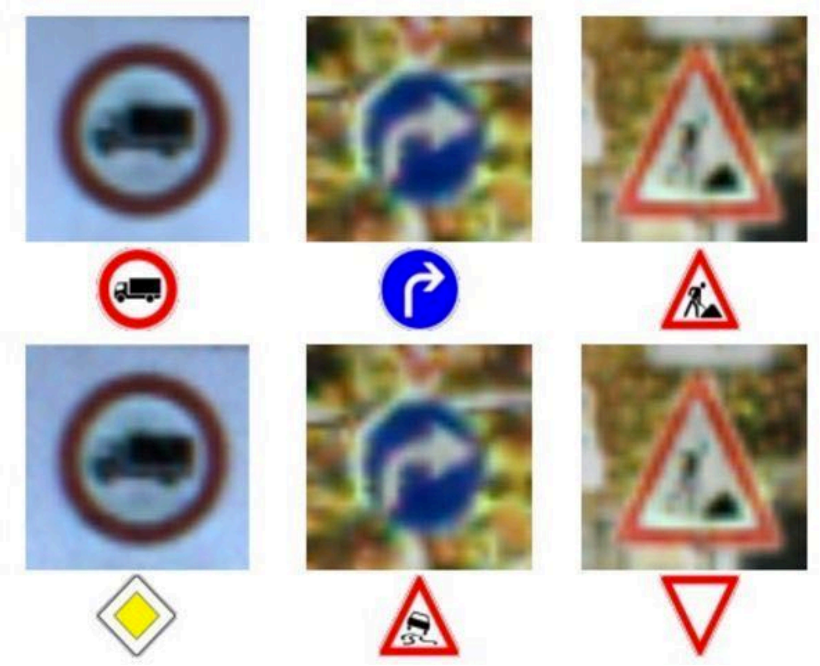
(Pinna and Gregory, 2002)

These are concentric circles, not intertwined spirals.

Adversarial Examples

- Adversarial examples pose potential security threats for practical machine learning systems.
- e.g., hypothetical attacks on autonomous vehicles





Slide adapted from Ian Goodfellow

Adversarial Examples

- Two types of adversaries (Papernot and Goodfellow 2016):

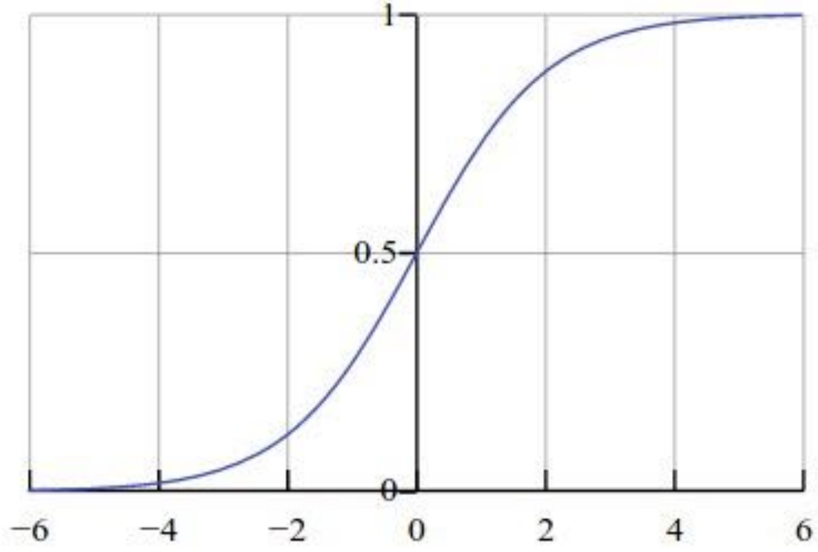
1. Poisoning training sets
    - interfere with the integrity of the training process
    - make modifications to existing training data, or insert additional data in the existing training set
    - increases the prediction error
2. Forcing models to make mistakes instantly with adversarial examples
    - perturb the inputs on which the model makes predictions (after training, during the inference phase)
    - generate “visually random” images that make a lot of sense to a machine learning system, but no sense at all to us

Not just for neural nets

- Linear models
    - Logistic regression
    - Softmax regression
    - SVMs
- Decision trees
- Nearest neighbors

Slide credit: Ian Goodfellow

Lets fool a binary linear classifier: (logistic regression)



$$P(y=1 \mid x; w, b) = \frac{1}{1+e^{-(w^T x+b)}} = \sigma(w^T x + b)$$

Since the probabilities of class 1 and 0 sum to one, the probability for class 0 is $P(y=0 \mid x; w, b) = 1 - P(y=1 \mid x; w, b)$. Hence, an example is classified as a positive example (y = 1) if $\sigma(w^T x + b) > 0.5$, or equivalently if the score $w^T x + b > 0$.

Slide credit: Alex Karpathy

Lets fool a binary linear classifier:

x	2	-1	3	-2	2	2	1	-4	5	1	← input example
w	-1	-1	1	-1	1	-1	1	1	-1	1	← weights

$$P(y=1 \mid x; w, b) = \frac{1}{1+e^{-(w^T x + b)}} = \sigma(w^T x + b)$$

Slide credit: Alex Karpathy

Lets fool a binary linear classifier:

x	2	-1	3	-2	2	2	1	-4	5	1	← input example
w	-1	-1	1	-1	1	-1	1	1	-1	1	← weights

class 1 score = dot product:
= -2 + 1 + 3 + 2 + 2 - 2 + 1 - 4 - 5 + 1 = -3
=> probability of class 1 is 1/(1+e^(-(-3))) = 0.0474
i.e. the classifier is 95% certain that this is class 0 example.

$$P(y=1 \mid x; w, b) = \frac{1}{1+e^{-(w^T x+b)}} = \sigma(w^T x + b)$$

Slide credit: Alex Karpathy

Lets fool a binary linear classifier:

x	2	-1	3	-2	2	2	1	-4	5	1	← input example
w	-1	-1	1	-1	1	-1	1	1	-1	1	← weights
adversarial **x**	?	?	?	?	?	?	?	?	?	?	

class 1 score = dot product:
= -2 + 1 + 3 + 2 + 2 - 2 + 1 - 4 - 5 + 1 = -3
=> probability of class 1 is 1/(1+e^(-(-3))) = 0.0474
i.e. the classifier is 95% certain that this is class 0 example.

$$P(y=1 \mid x; w, b) = \frac{1}{1+e^{-(w^T x+b)}} = \sigma(w^T x + b)$$

Slide credit: Alex Karpathy

Lets fool a binary linear classifier:

x	2	-1	3	-2	2	2	1	-4	5	1	← input example
w	-1	-1	1	-1	1	-1	1	1	-1	1	← weights
adversarial **x**	1.5	-1.5	3.5	-2.5	2.5	1.5	1.5	-3.5	4.5	1.5	

class 1 score before:

-2 + 1 + 3 + 2 + 2 - 2 + 1 - 4 - 5 + 1 = -3

=> probability of class 1 is 1/(1+e^(-(-3))) = 0.0474

-1.5+1.5+3.5+2.5+2.5-1.5+1.5-3.5-4.5+1.5 = 2

=> probability of class 1 is now 1/(1+e^(-(2))) = 0.88

i.e. we improved the class 1 probability from 5% to 88%

$$P(y=1 \mid x; w, b) = \frac{1}{1+e^{-(w^T x + b)}} = \sigma(w^T x + b)$$

Slide credit: Alex Karpathy

Lets fool a binary linear classifier:

x	2	-1	3	-2	2	2	1	-4	5	1	← input example
w	-1	-1	1	-1	1	-1	1	1	-1	1	← weights
adversarial **x**	1.5	-1.5	3.5	-2.5	2.5	1.5	1.5	-3.5	4.5	1.5	

class 1 score before:
-2 + 1 + 3 + 2 + 2 - 2 + 1 - 4 - 5 + 1 = -3
=> probability of class 1 is 1/(1+e^(-(-3))) = 0.0474

-1.5+1.5+3.5+2.5+2.5-1.5+1.5-3.5-4.5+1.5 = 2
=> probability of class 1 is now 1/(1+e^(-(2))) = 0.88
i.e. we improved the class 1 probability from 5% to 88%

This was only with 10 input dimensions. A 224x224 input image has 150,528.

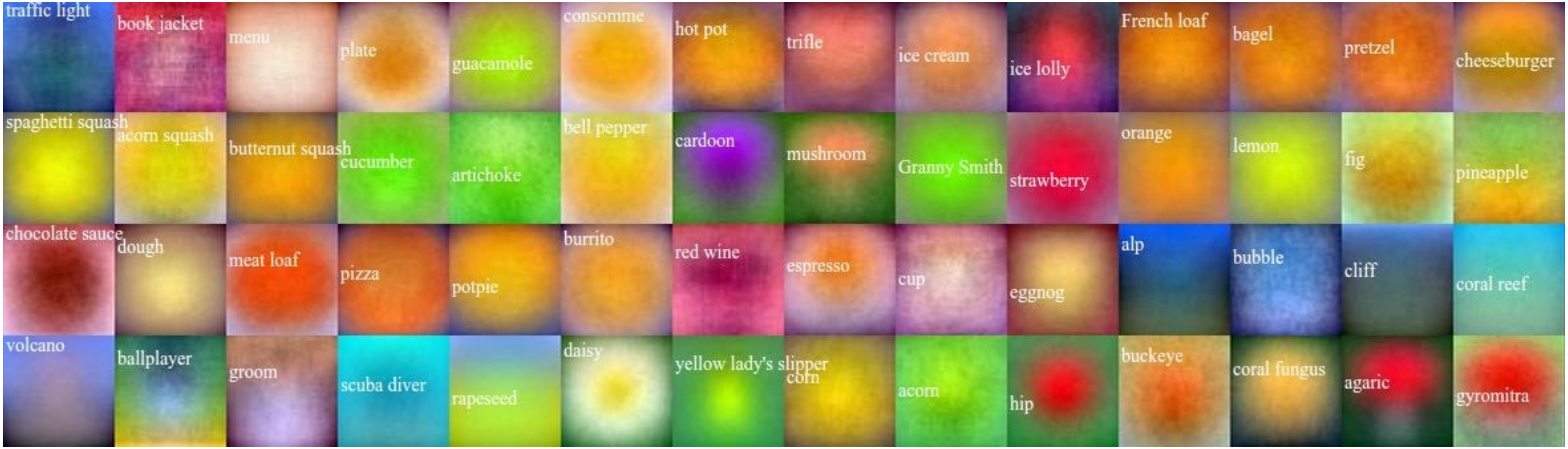
(It's significantly easier with more numbers, need smaller nudge for each)

Slide credit: Alex Karpathy

Blog post: Breaking Linear Classifiers on ImageNet

Recall CIFAR-10 linear classifiers:

ImageNet classifiers:

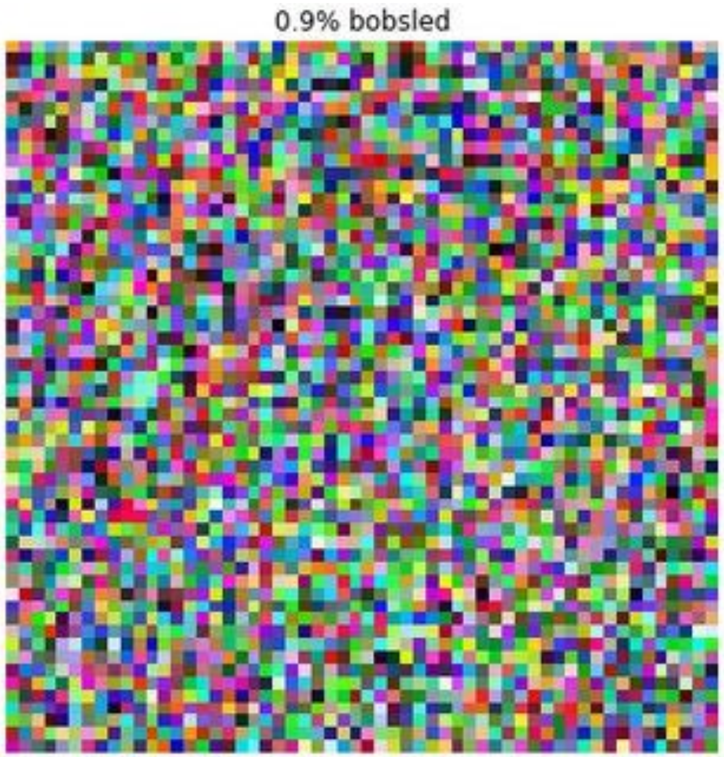


http://karpathy.github.io/2015/03/30/breaking-convnets/

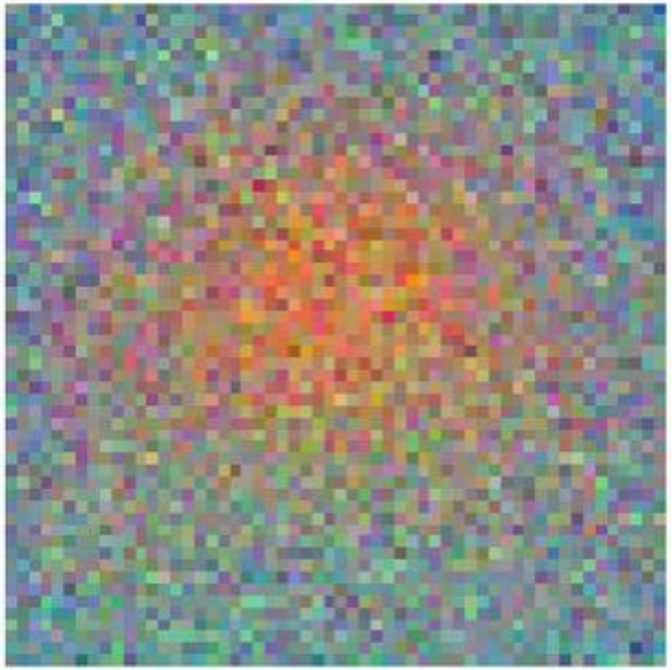
Slide credit: Alex Karpathy

Breaking Linear Classifiers on ImageNet

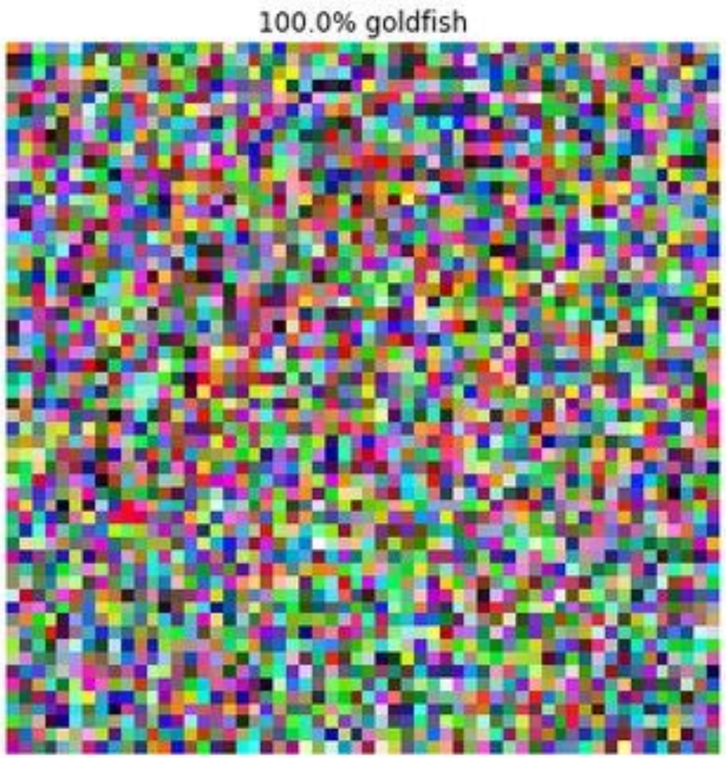
mix in a tiny bit of
Goldfish classifier weights



+



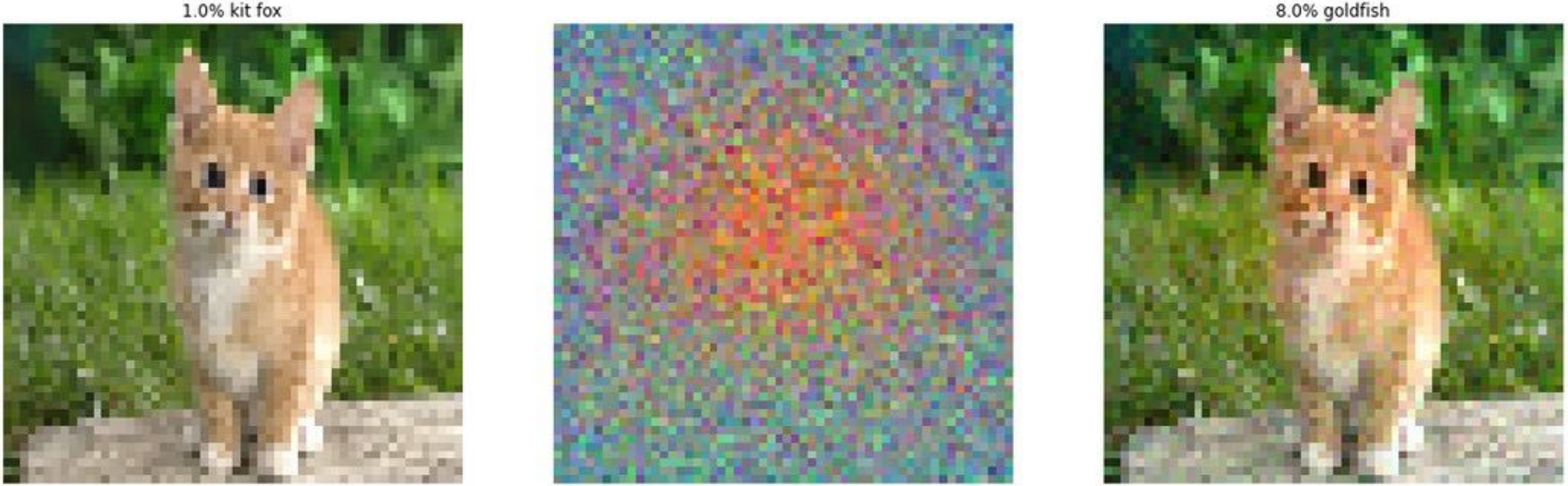
=



100% Goldfish

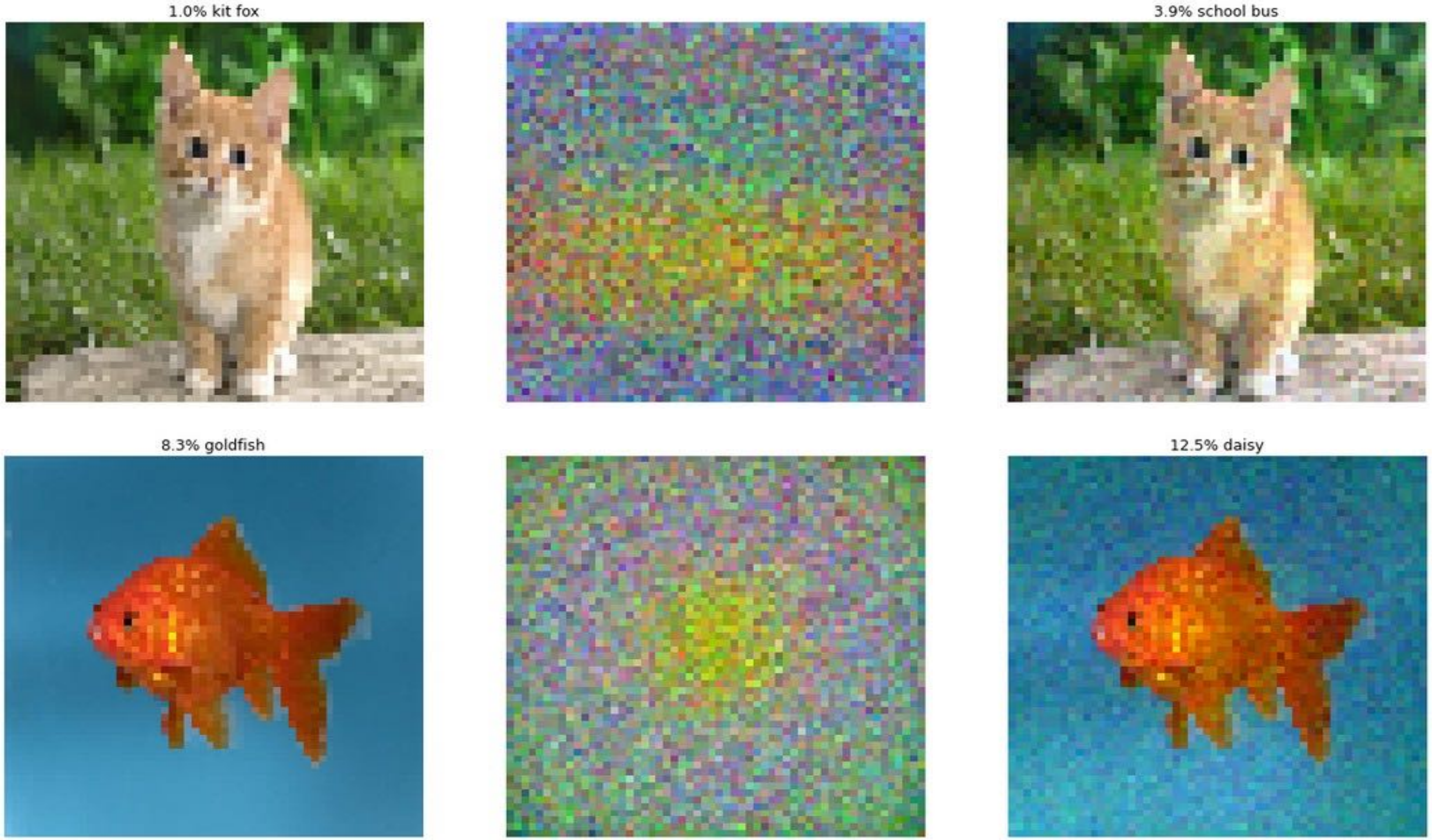
Slide credit: Alex Karpathy

Breaking Linear Classifiers on ImageNet



Slide credit: Alex Karpathy

Breaking Linear Classifiers on ImageNet



Slide credit: Alex Karpathy

Intriguing Properties of Neural Networks

(Szegedy et al., 2013)

correct +distort ostrich correct +distort ostrich

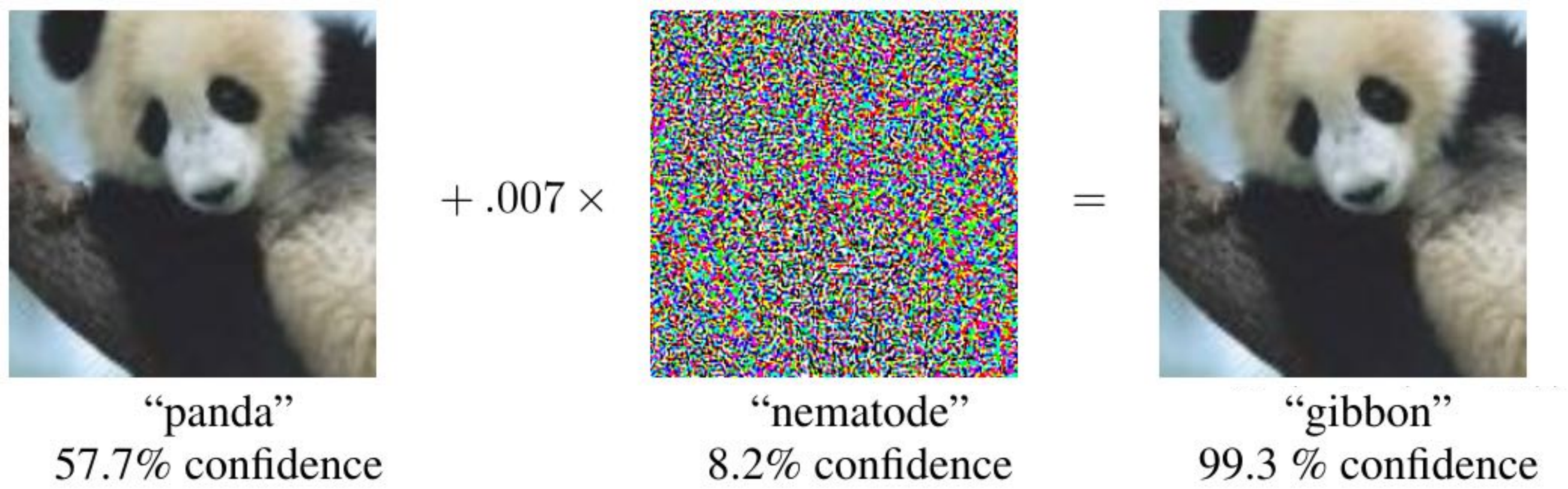
Minimize $\|r\|_2$ subject to:

1. $f(x+r) = l$
2. $x + r \in [0,1]^m$

f: classifier function
x: input image
r: distortion
l: target label

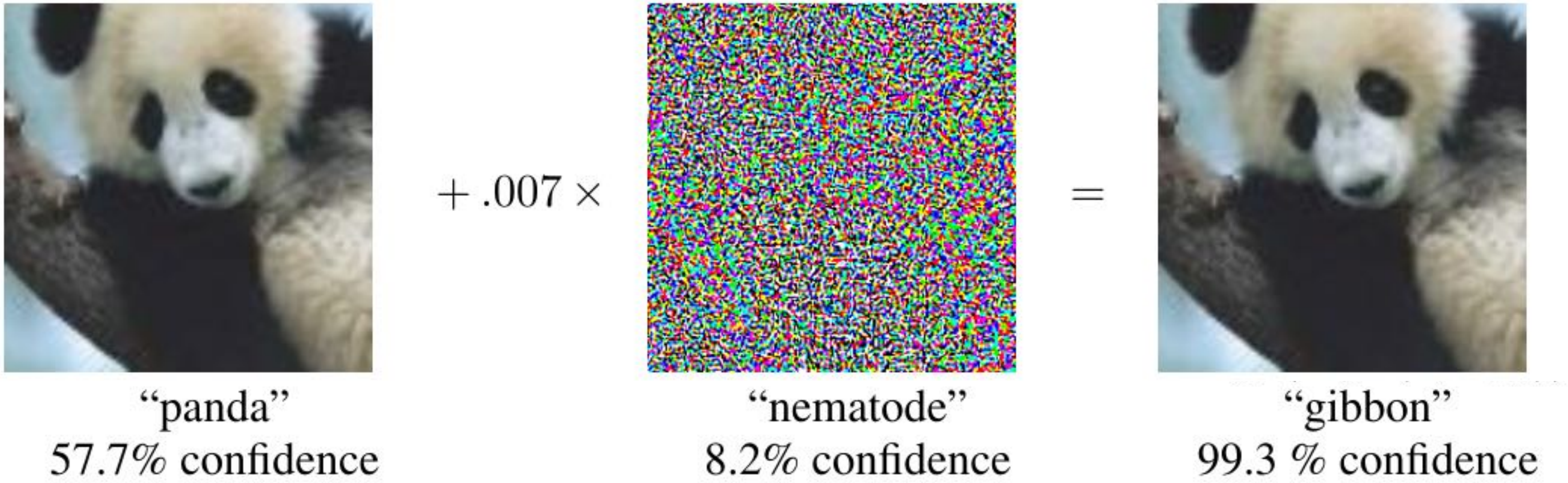
Minimize $c|r| + \text{loss}_f(x+r, l)$
subject to $x + r \in [0,1]^m$

Explaining and Harnessing Adversarial Examples
(Goodfellow et al., 2014)



Slide credit: Ian Goodfellow

Explaining and Harnessing Adversarial Examples
(Goodfellow et al., 2014)



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

Score of label y_{true}, given input image X
e.g. cross entropy loss

Explaining and Harnessing Adversarial Examples
(Goodfellow et al., 2014)

- Perturbation is computed to minimize a specific norm in the input domain while increasing the model's prediction error

$$J(\tilde{\boldsymbol{x}}, \boldsymbol{\theta}) \approx J(\boldsymbol{x}, \boldsymbol{\theta}) + (\tilde{\boldsymbol{x}} - \boldsymbol{x})^\top \nabla_{\boldsymbol{x}} J(\boldsymbol{x}).$$

Maximize

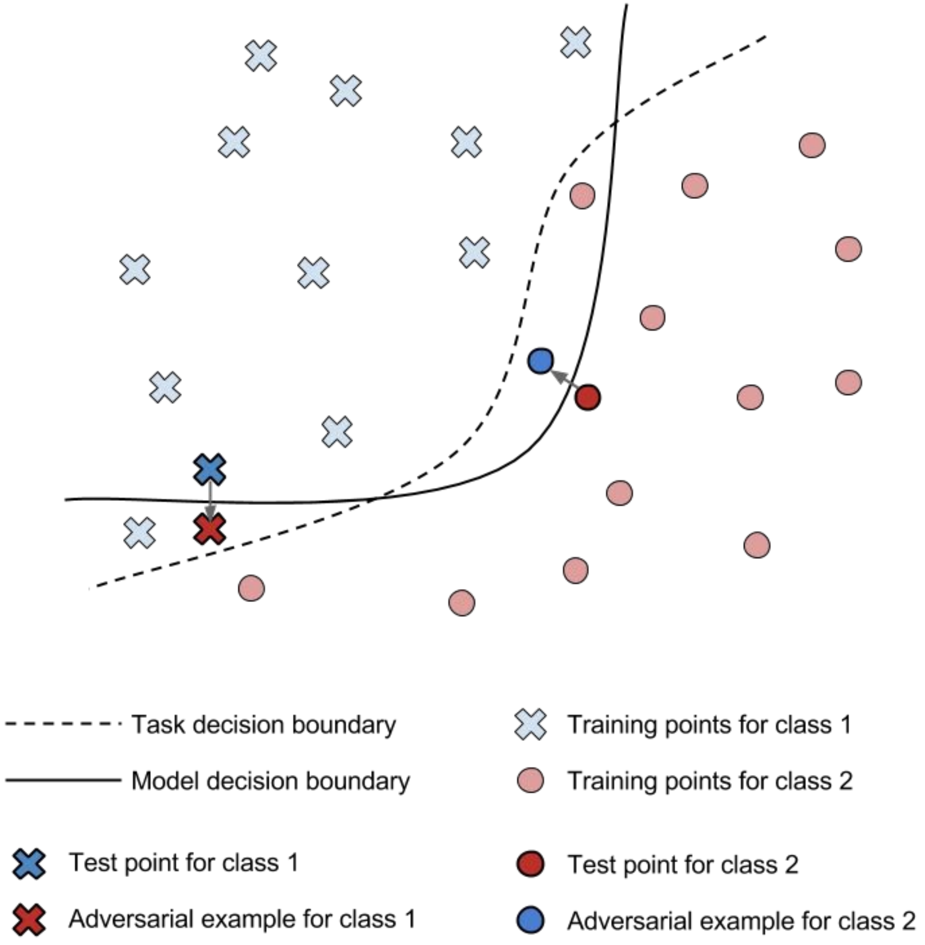
$$J(\boldsymbol{x}, \boldsymbol{\theta}) + (\tilde{\boldsymbol{x}} - \boldsymbol{x})^\top \nabla_{\boldsymbol{x}} J(\boldsymbol{x})$$

subject to

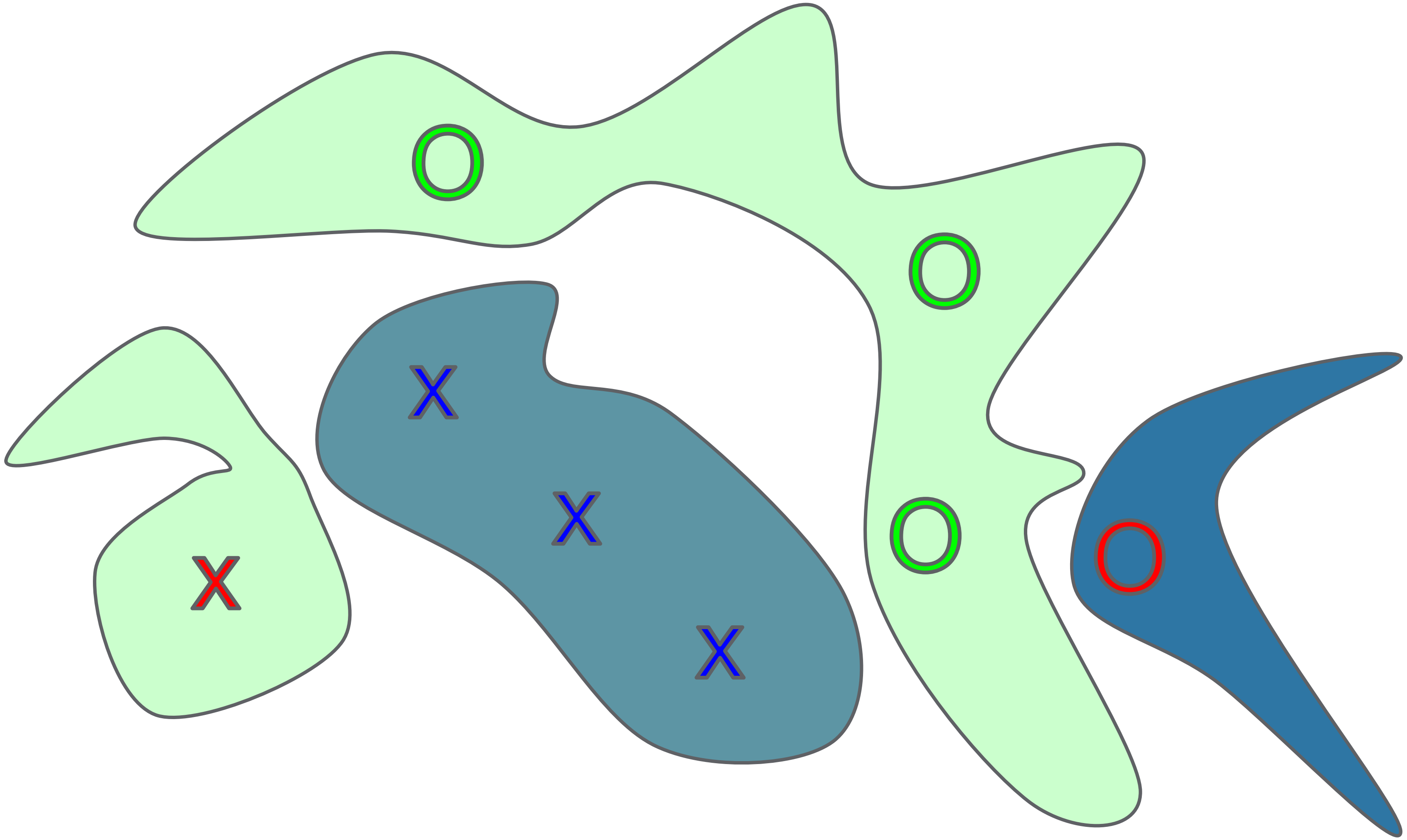
$$||\tilde{\boldsymbol{x}} - \boldsymbol{x}||_\infty \leq \epsilon$$

$$\Rightarrow \tilde{\boldsymbol{x}} = \boldsymbol{x} + \epsilon \mathrm{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{x})\right).$$

The Fast Gradient Sign Method

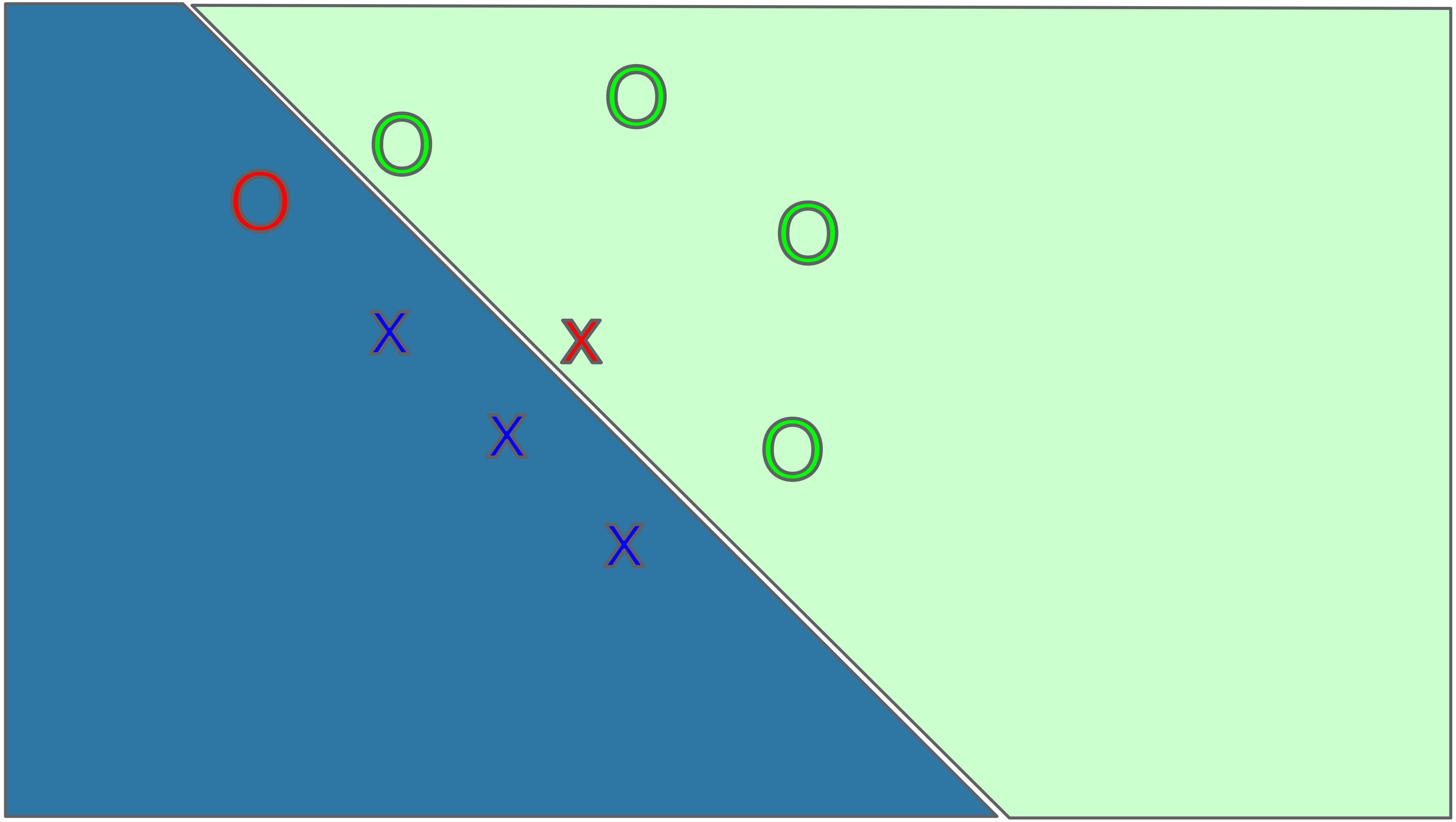


Adversarial Examples from Overfitting



Slide credit: Ian Goodfellow

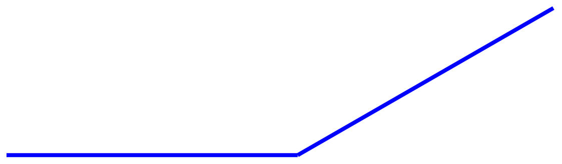
Adversarial Examples from Excessive Linearity



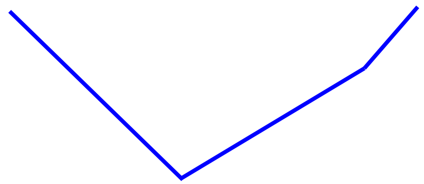
Slide credit: Ian Goodfellow

Modern deep nets are very piecewise linear

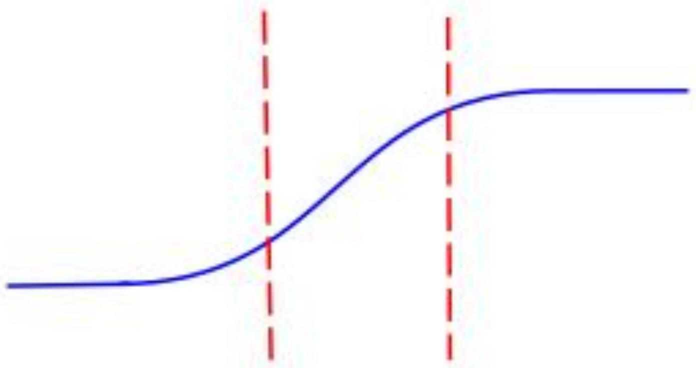
Rectified linear unit



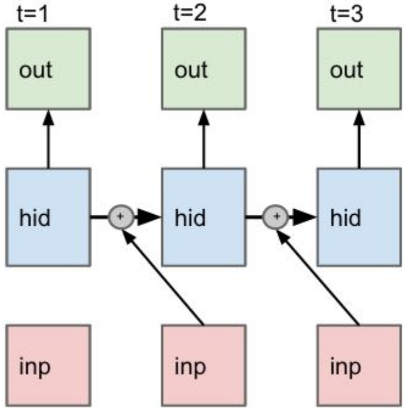
Maxout



Carefully tuned sigmoid



LSTM



Slide credit: Ian Goodfellow

Gradient-based Adversarial Examples

- Fast Gradient Sign (Goodfellow et al., 2014)

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

- Basic Iterative Method (Kurakin et al., 2017)

$$\boldsymbol{X}_0^{adv} = \boldsymbol{X}, \quad \boldsymbol{X}_{N+1}^{adv} = Clip_{X,\epsilon}\Big\{\boldsymbol{X}_N^{adv} + \alpha\,\mathrm{sign}\big(\nabla_X J(\boldsymbol{X}_N^{adv}, y_{true})\big)\Big\}$$

$$Clip_{X,\epsilon}\left\{\boldsymbol{X}'\right\}(x,y,z) = \min\Big\{255, \boldsymbol{X}(x,y,z)+\epsilon, \max\big\{0, \boldsymbol{X}(x,y,z)-\epsilon, \boldsymbol{X}'(x,y,z)\big\}\Big\}$$

- Iterative Least-Likely Class Method (Kurakin et al., 2017)

$$y_{LL} = \arg\min_y \big\{p(y|\boldsymbol{X})\big\}$$

$$\boldsymbol{X}_0^{adv} = \boldsymbol{X}, \quad \boldsymbol{X}_{N+1}^{adv} = Clip_{X,\epsilon}\left\{\boldsymbol{X}_N^{adv} - \alpha\,\mathrm{sign}\left(\nabla_X J(\boldsymbol{X}_N^{adv}, y_{LL})\right)\right\}$$

Gradient-based Adversarial Examples

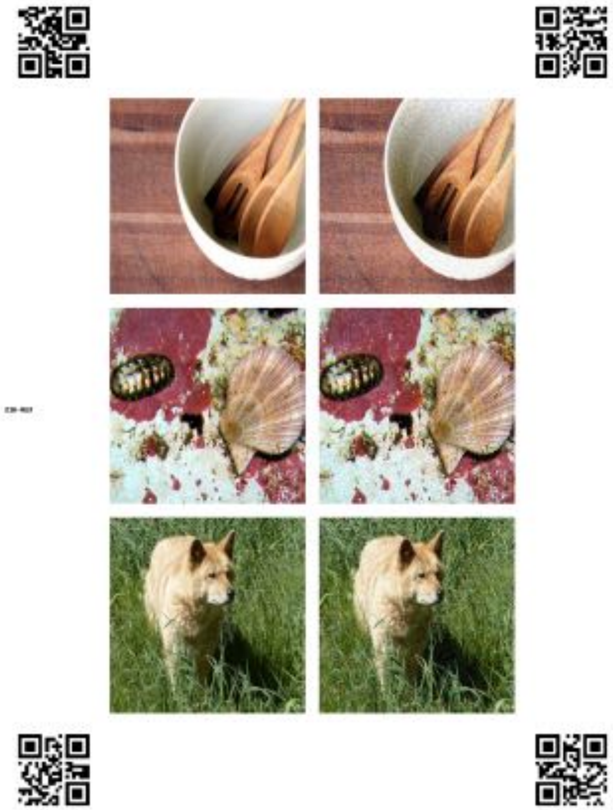
Original sample

Fast Gradient

Basic Iterative Method

Iterative Least-Likely Class Method

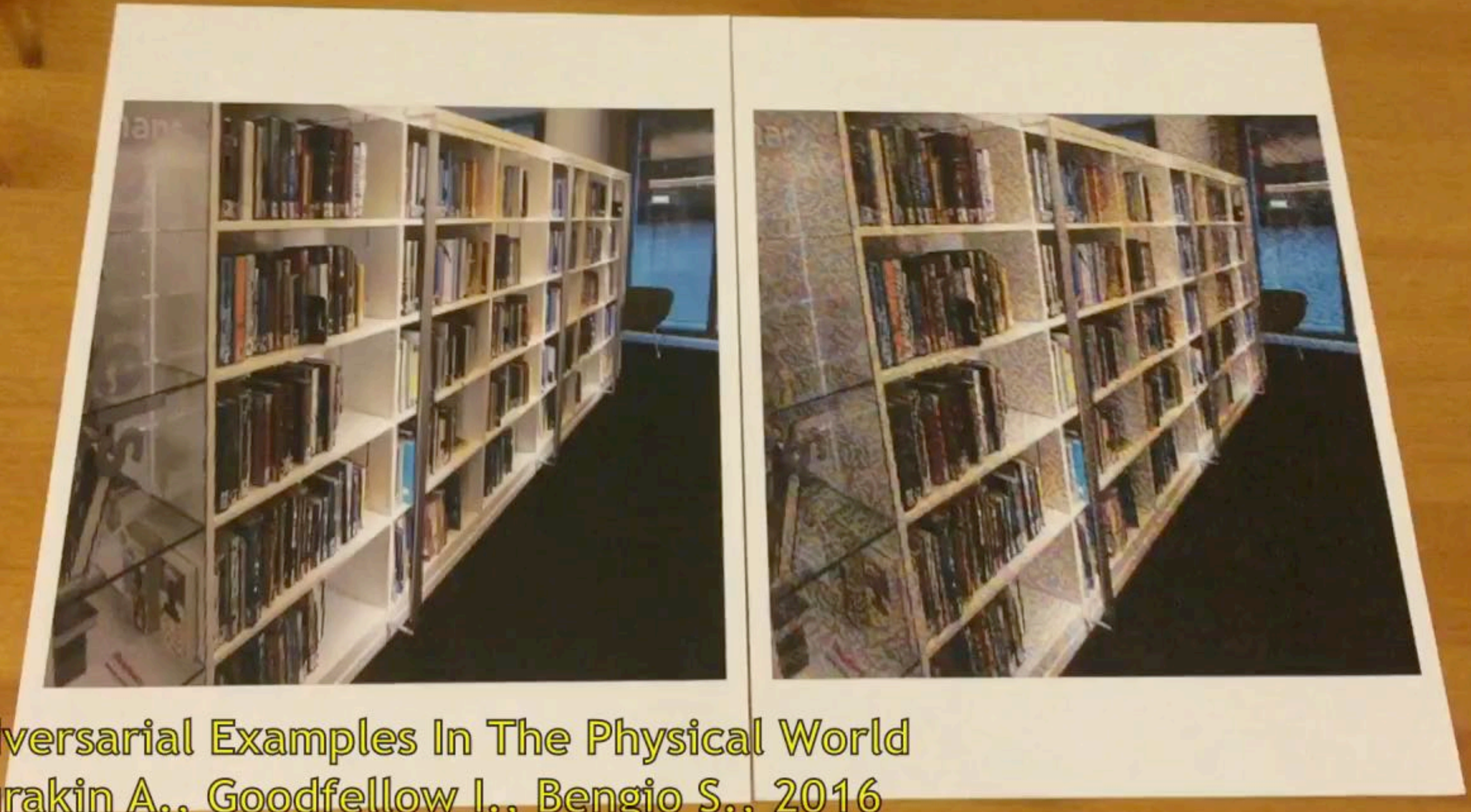
Adversarial Examples in the Physical World
(Kurakin, Goodfellow, Bengio, 2017)



(a) Printout

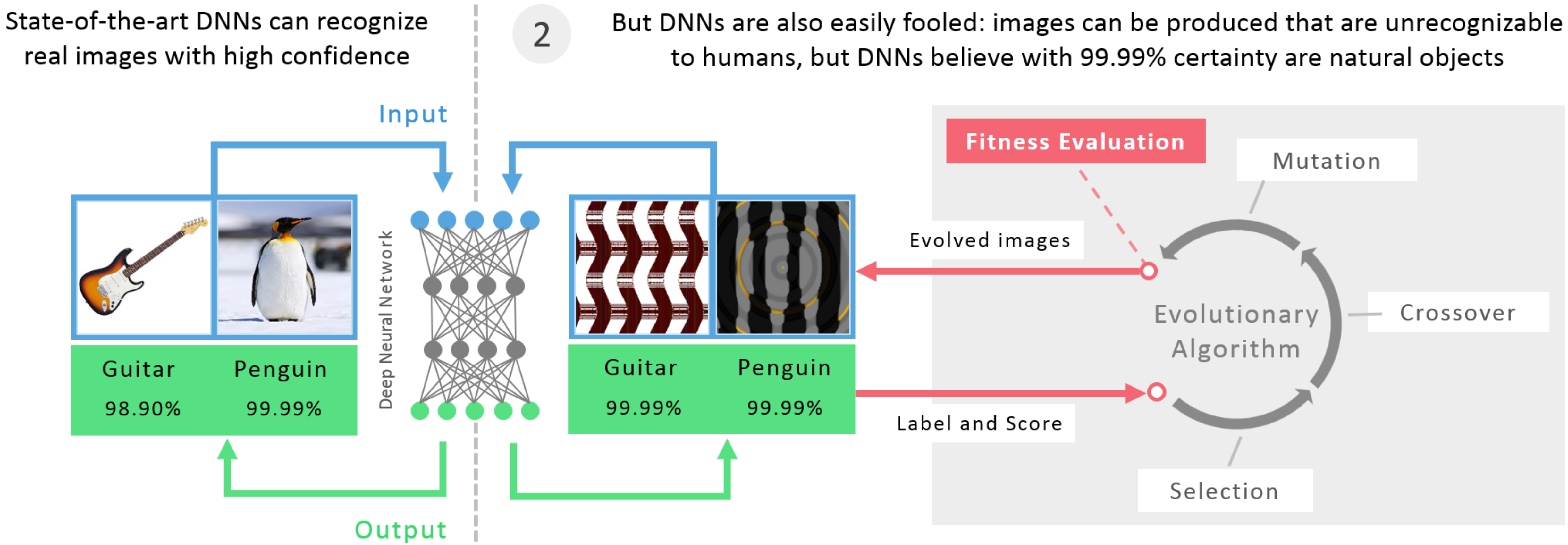
(b) Photo of printout

(c) Cropped image



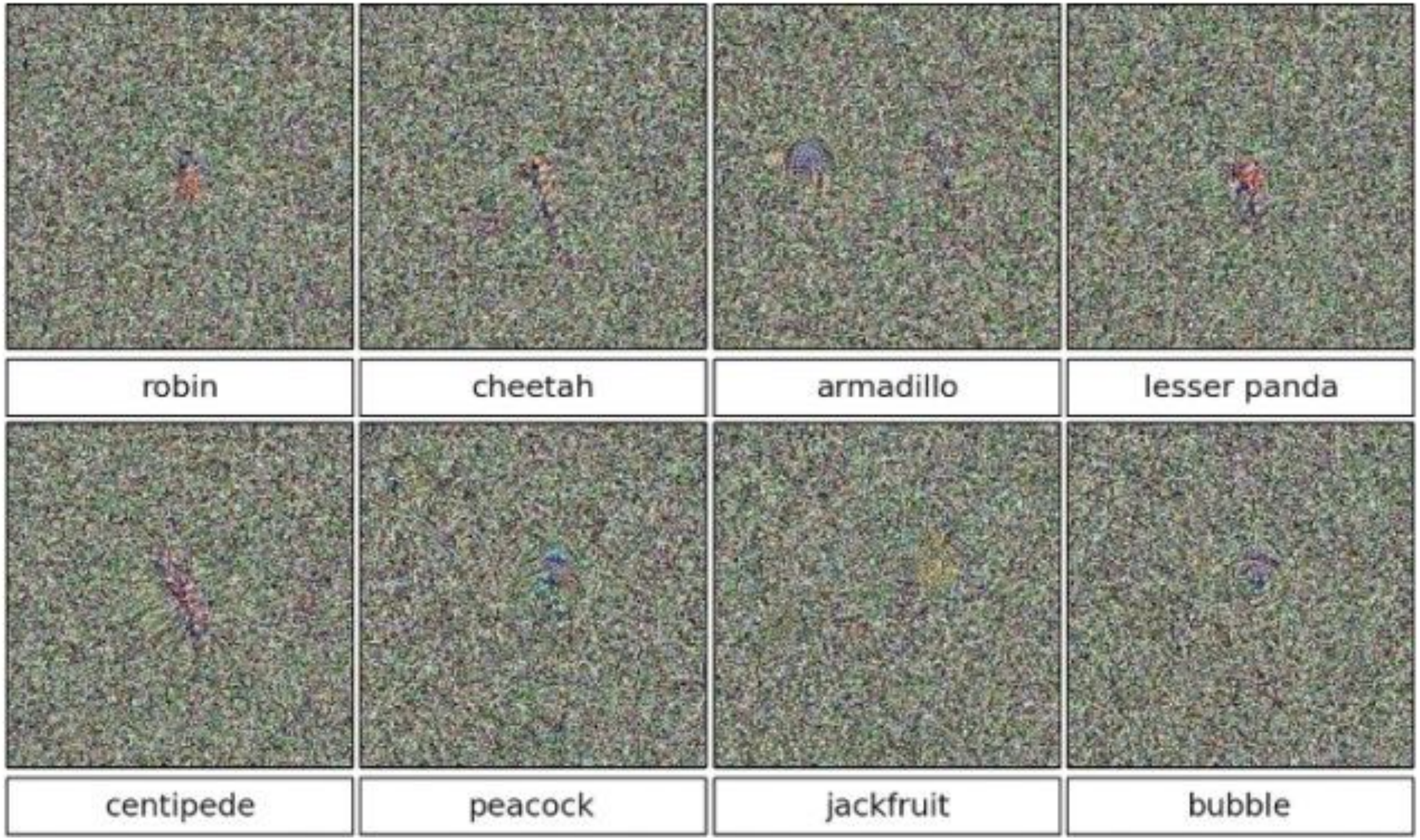
Adversarial Examples In The Physical World
Kurakin A., Goodfellow I., Bengio S., 2016

Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images
(Nguyen, Yosinski, Clune, 2014)



Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images
(Nguyen, Yosinski, Clune, 2014)

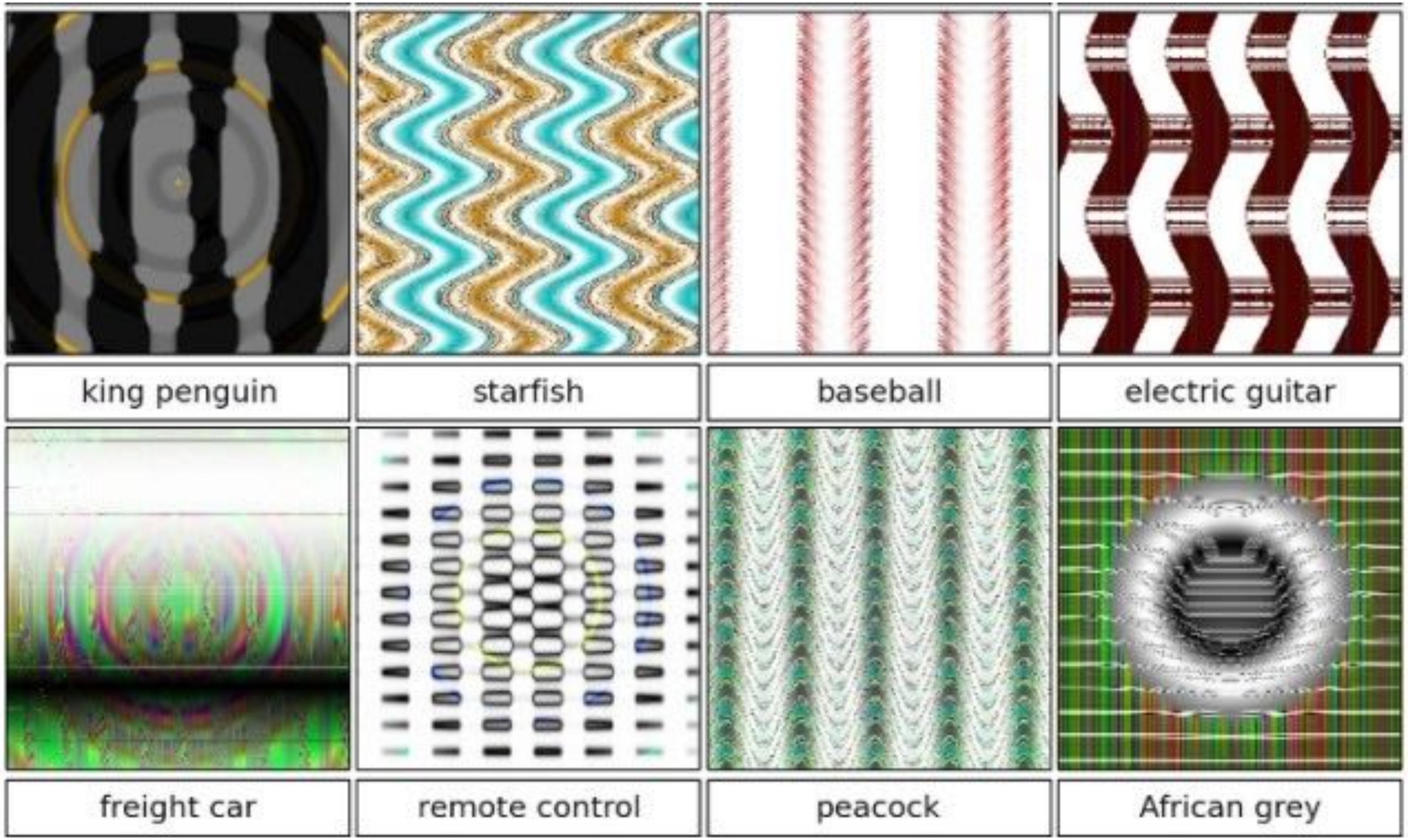
>99.6% confidences



Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images

(Nguyen, Yosinski, Clune, 2014)

>99.6% confidences



Adversarial Learning – Failed Defenses

Generative pretraining

Removing perturbation with an autoencoder

Adding noise at test time

Ensembles

Confidence-reducing perturbation at test time

Error correcting codes

Multiple glimpses

Weight decay

Double backprop

Adding noise at train time

Various non-linear units

Dropout

Slide credit: Ian Goodfellow

Adversarial Learning – Defense Techniques

- Two defense techniques

1. Adversarial training (Szegedy et al., 2013)
   - a brute force solution where adversarial examples are generated and the model is explicitly trained not to be fooled by each of them.
   - improves the generalization of a machine learning model
2. Defensive distillation (Hinton et al., 2015; Papernot and McDaniel, 2016)
   - a strategy where the model is trained to output probabilities of different classes, rather than hard decisions about which class to output
   - smooths the model's decision surface in adversarial directions exploited by the adversary

Adversarial Training

Labeled as bird

Decrease probability of bird class

Still has same label (bird)

Slide credit: Ian Goodfellow

Adversarial Training

- Generate adversarial examples and use them while training
- Introduce an adversarial regularization term to the general loss function

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

training target

adversarial regularization

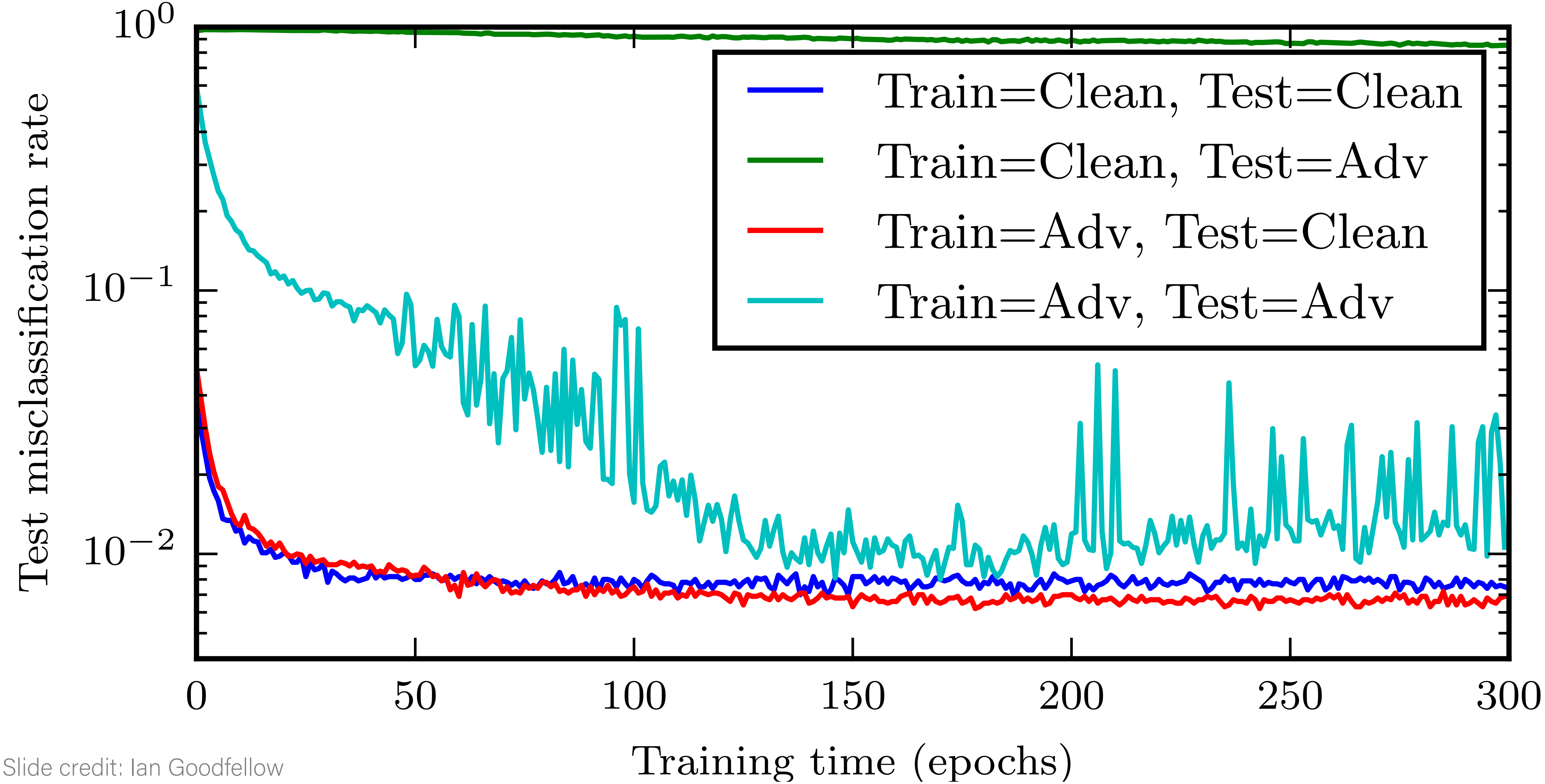
Virtual Adversarial Training

Unlabeled; model guesses it's probably a bird, maybe a plane

New guess should match old guess (probably bird, maybe plane)

Adversarial perturbation intended to change the guess

Training on Adversarial Examples



Slide credit: Ian Goodfellow

Adversarial Training of Other Models

- Linear models: SVM / linear regression cannot learn a step function, so adversarial training is less useful, very similar to weight decay

- k-NN: adversarial training is prone to overfitting.

- Takeway: neural nets can actually become more secure than other models.
- Adversarially trained neural nets have the best empirical success rate on adversarial examples of any machine learning model.

Defensive Distillation

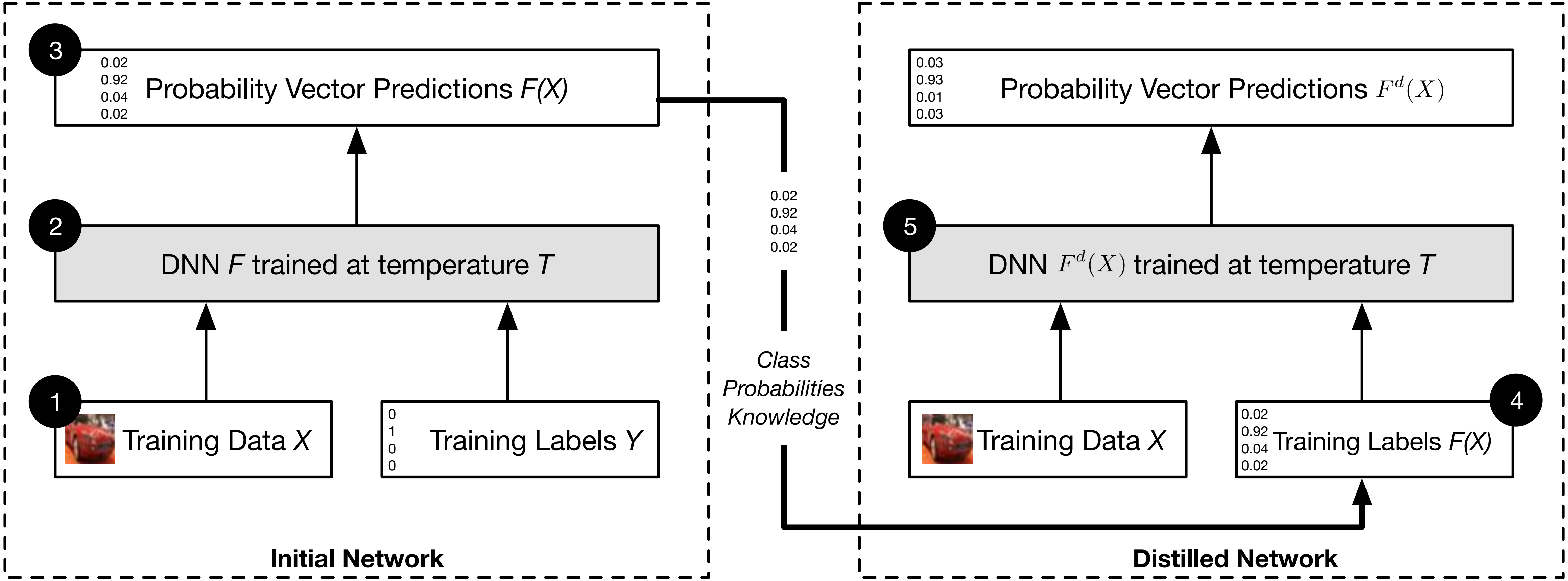
- Neural networks typically produce class probabilities by using a “softmax” output layer:

$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

T: a temperature that is normally set to 1
q_i: class probability

- Defensive distillation changes the training procedure essentially by re-configuring this “softmax” layer.
- It smooths the model’s decision surface, eliminates overfitting, and thus increase robustness of the deep neural network model.
- <u>Simplest form:</u> Use the original model's predictions as the groundtruth labels to train the distilled model.

Defensive Distillation



$$\frac{\partial C}{\partial z_i} = \frac{1}{T}(q_i - p_i) = \frac{1}{T}\left(\frac{e^{z_i/T}}{\sum_j e^{z_j/T}} - \frac{e^{v_i/T}}{\sum_j e^{v_j/T}}\right)$$

Papernot et al., 2016

Defensive Distillation

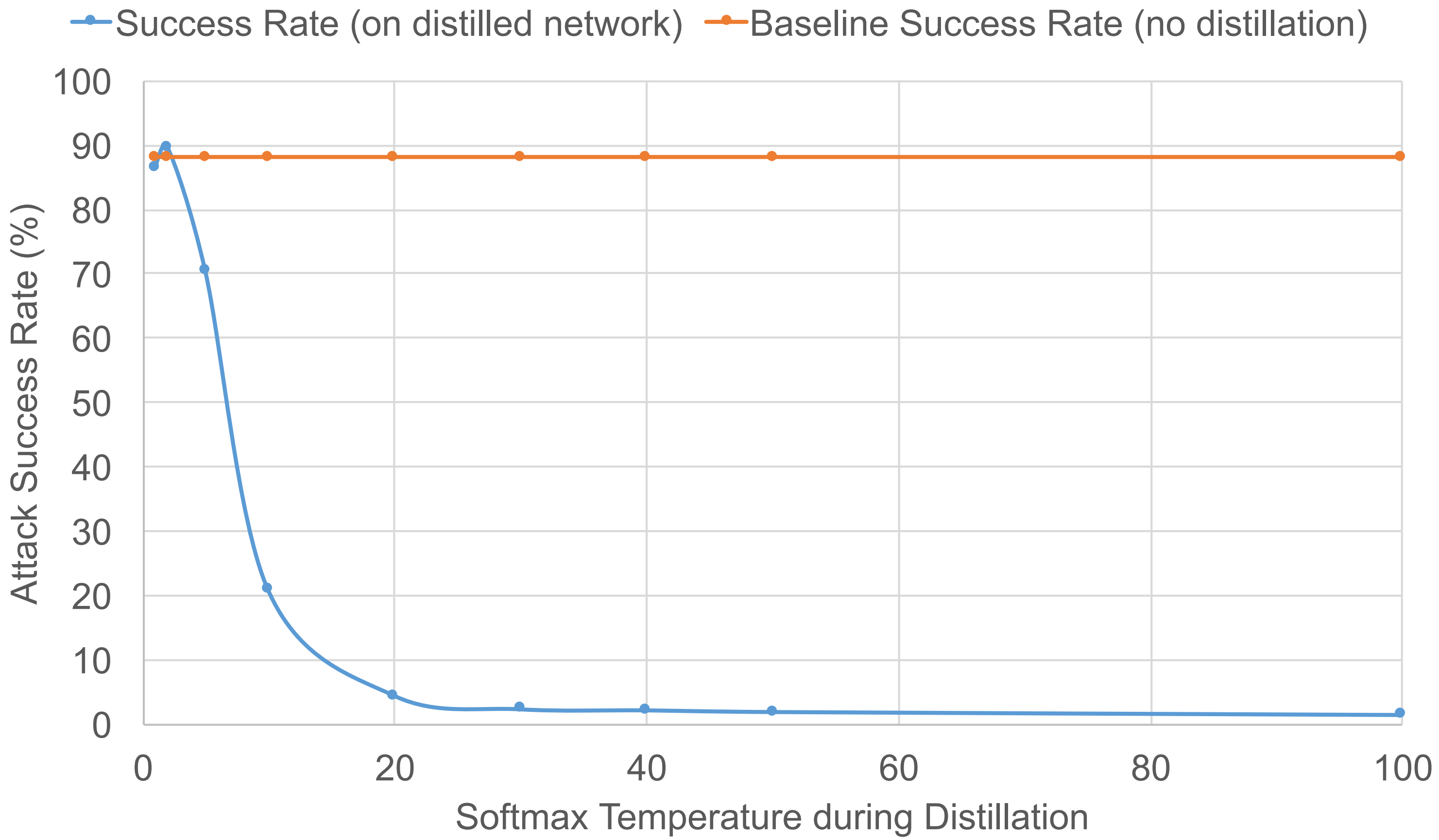
- This strategy include the following steps (Papernot and McDaniel, 2016):

1. Train a first instance of the neural network by using the training data (X, Y) where the labels Y indicate the correct class of samples X.
2. Infer predictions of the training data and provide a new training dataset (X,f(X)) where the new class labels are the probability vectors quantifying the likeliness of X being in each class.
3. Train a *distilled* instance of the neural network f using this newly labeled dataset (X,f(X)).

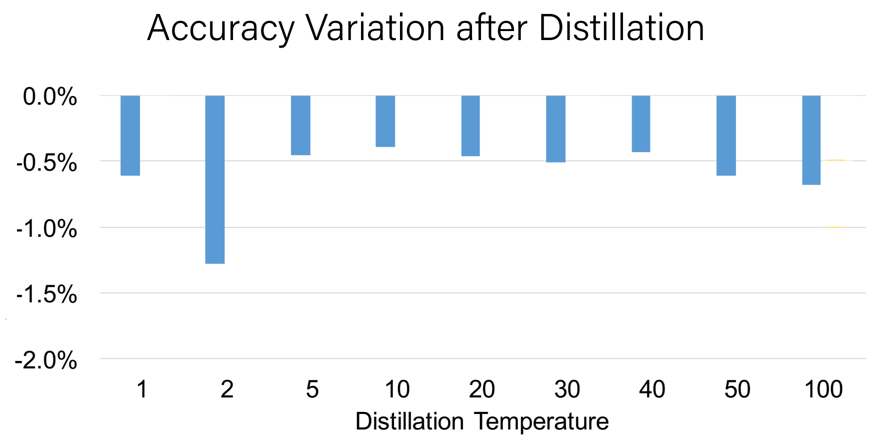
While training the first and the distilled network, use the same high T values. At test time, deploy the distilled network by setting T back to 1.

Defensive Distillation

- Distillation at high temperatures improves the smoothness of the network, and reduces its sensitivity to small input variations.

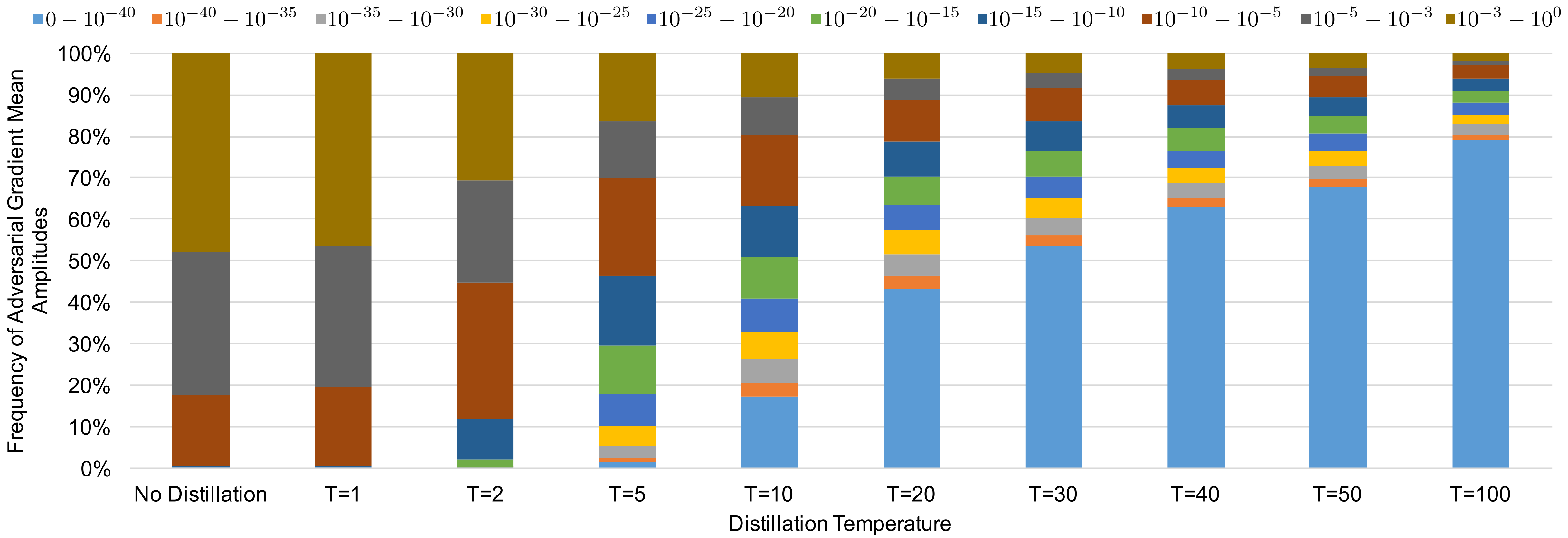


On the MNIST model - a 9 layer deep neural network with a 99.5% test accuracy (Papernot and McDaniel, 2016)



Distillation and Sensitivity

- Distillation reduces gradients exploited by adversaries to craft perturbations.



10,000 samples from the CIFAR10 test set into bins according to the mean value of their adversarial gradient amplitude. (Papernot et al., 2016)

Summary

- Big gains in performance on perceptual tasks by using deep neural network models.
- Machine learning has not yet reached true human-level performance.
- Adversarial examples show that many modern machine learning algorithms can be easily fooled.
- Many different ways of attacking deep neural network models.
- Very few ways of defending deep neural network models.

- Recent work (Papernot et al., 2017) considers more realistic threat models
  - The adversary have no knowledge of the machine learning architecture and model parameters.