University of Rome "La Sapienza"
Dep. of Computer, Control and Management Engineering A. Ruberti

Adversarial Examples

Department of Computer, Control, and Management Engineering Antonio Ruberti

Sapienza Università di Roma

Valsamis Ntouskos, ALCOR Lab
ntouskos@diag.uniroma1.it

Outline

- What is an Adversarial Example?
- Convolutional Neural Networks – review
- Attack methods
- Adversarial Example Properties
- Defense methods
- Other topics

What is an adversarial example?

Sloth or Pain au chocolat?

What is an adversarial example?

Sheepdog or Mop?

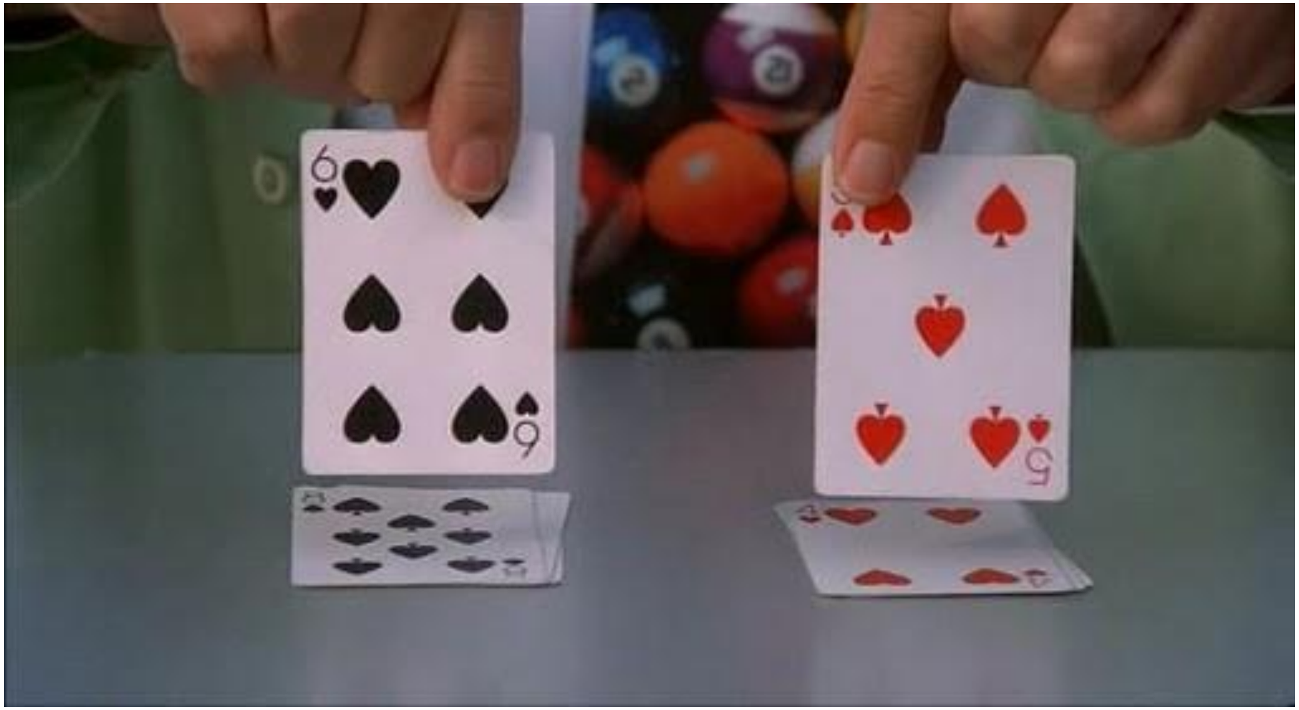
What is an adversarial example?

Chihuahua or Muffin?

What is an adversarial example?

Puppy or Bagel?

What is an adversarial example?

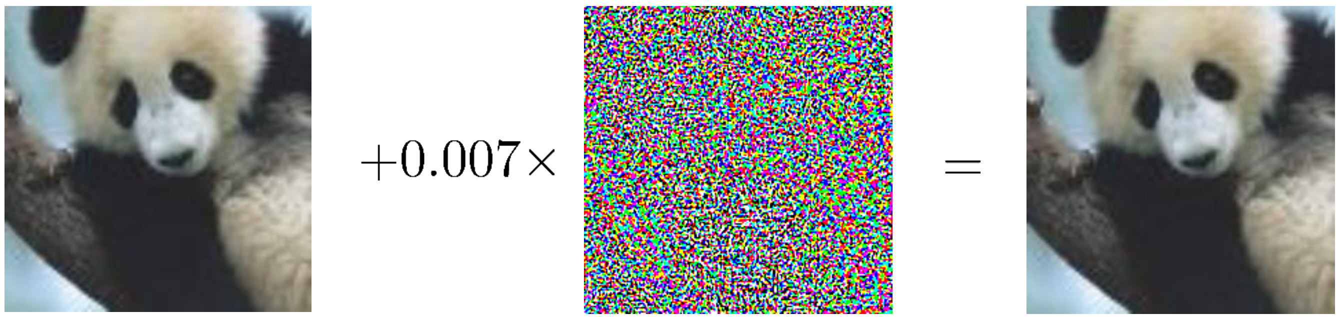


Adversarial examples for CNNs

Garbage Truck
99% confidence

Sports car
85% confidence
Garbage Truck
3% confidence

Adversarial examples for CNNs



Panda
58% confidence

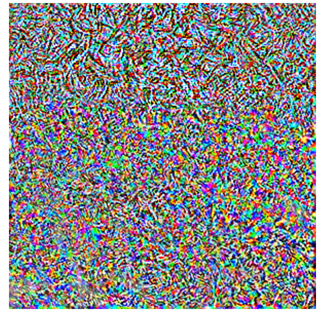
Adversarial
noise

Gibbon
99% confidence

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

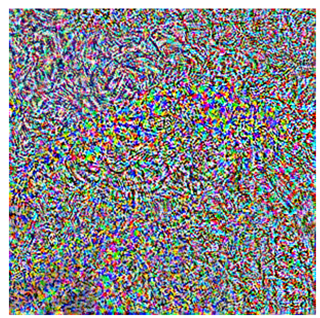
Adversarial examples for CNNs

Alps: 94%



Dog: 100%

Puffer: 98%



Crab: 100%

Dong et al. (2018). Boosting Adversarial Attacks with Momentum. *CVPR*

Image classification

xkcd: Tasks

"The Virtually Impossible"

Slides from Caffe framework tutorial @ CVPR2015

Deep Learning with CNNs

Compositional Models

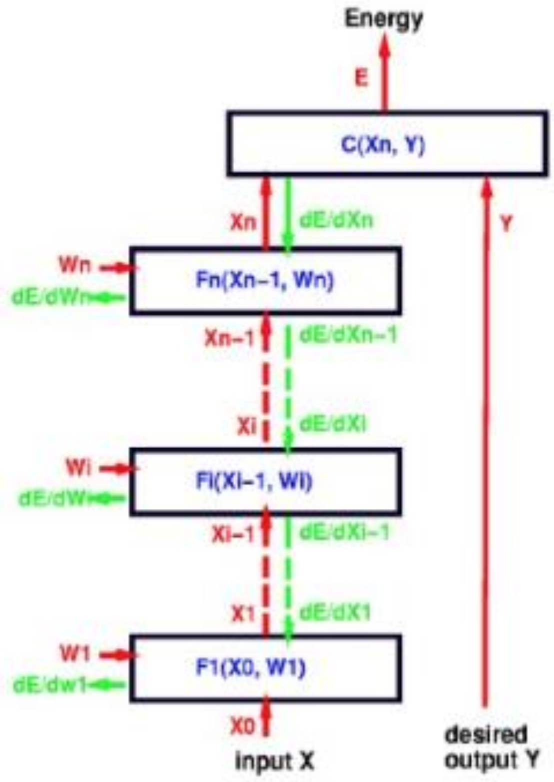
Learned End-to-End

Hierarchy of Representations

- vision: pixel, motif, part, object
- text: character, word, clause, sentence
- speech: audio, band, phone, word

concrete → abstract

learning

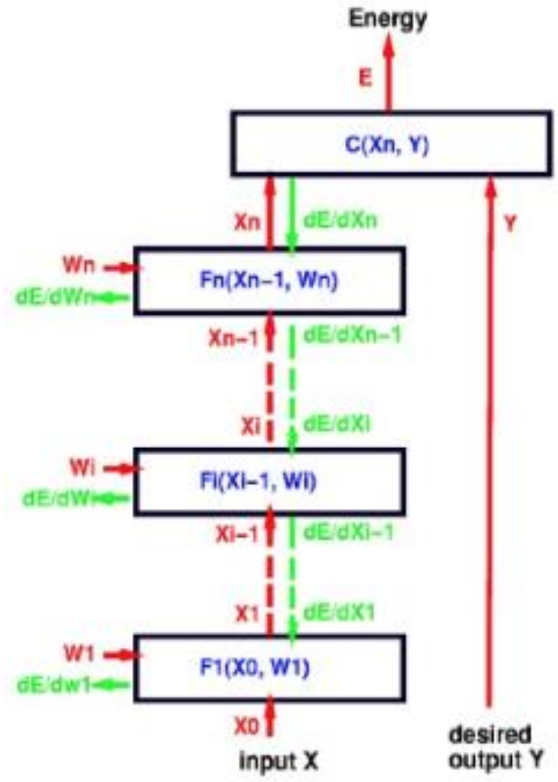


Slides from Caffe framework tutorial @ CVPR2015

Deep Learning with CNNs

Compositional Models
Learned End-to-End

Back-propagation jointly learns all of the model parameters to optimize the output for the task.



Slides from Caffe framework tutorial @ CVPR2015

Motivation - Why Convolutional?

Inputs usually treated as general feature vectors

In some cases inputs have special structure:

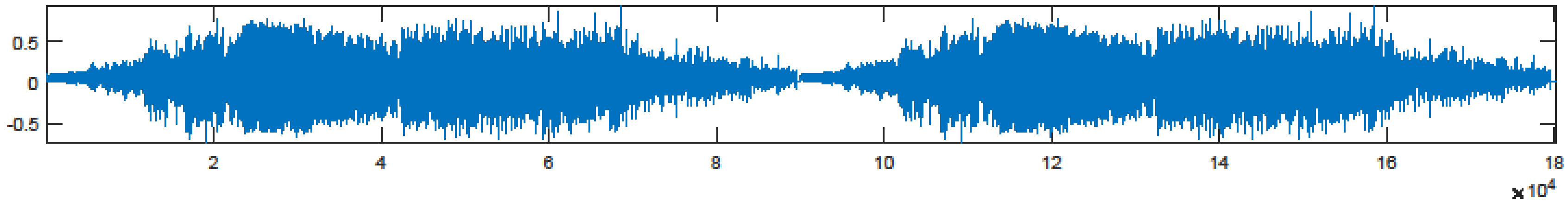
- Audio
- Images
- Videos

Signals: Numerical representations of physical quantities

Deep learning can be directly applied on signals by using suitable operators

Motivation - Why Convolutional?

Audio

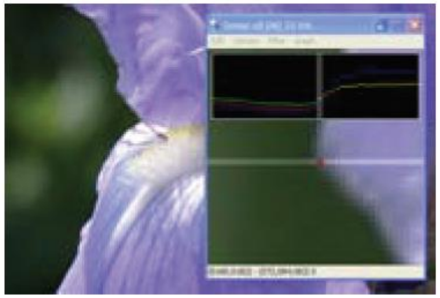


...	0.0468	0.0468	0.0468	0.0390	0.0390	0.0390	0.0546	0.0625	0.0625	0.0390	0.0312	0.0468	0.0625	...

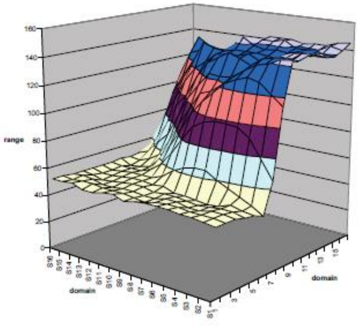
1D data - (variable length) vectors

Motivation - Why Convolutional?

Images



45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120



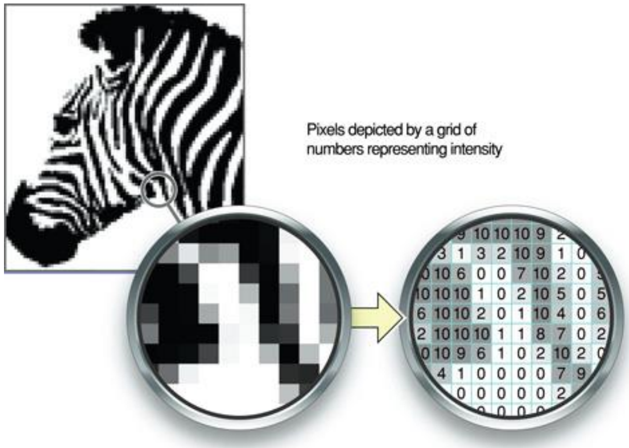
2D data - matrices

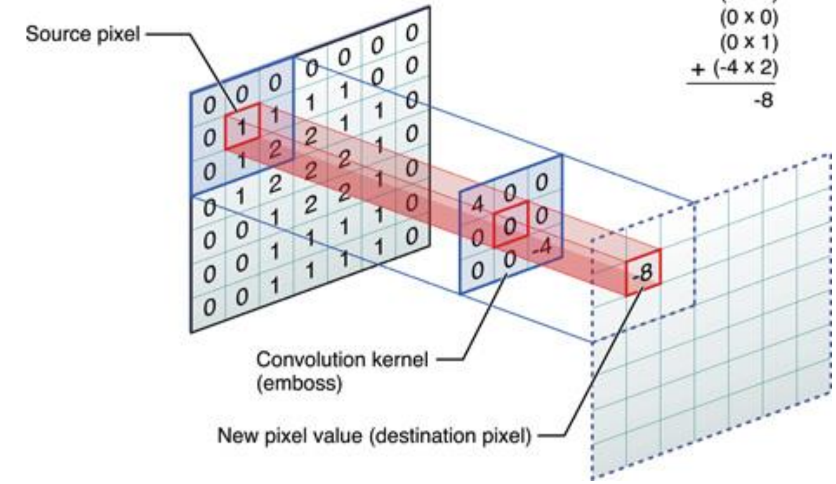
Video

A sequence of images sampled through time - 3D data

Some theory

Convolution





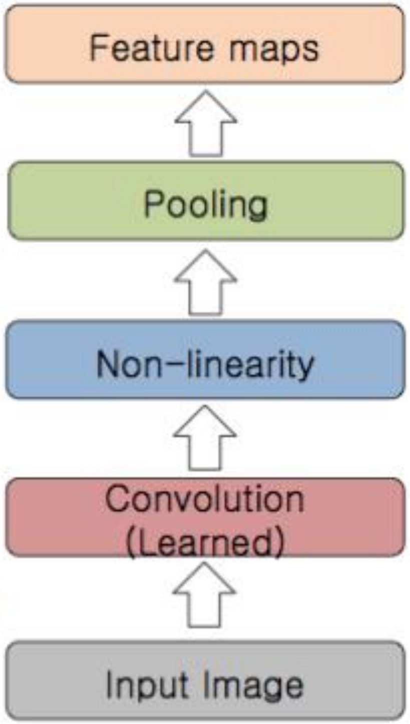
- Image filtering is based on convolution with special kernels

Original

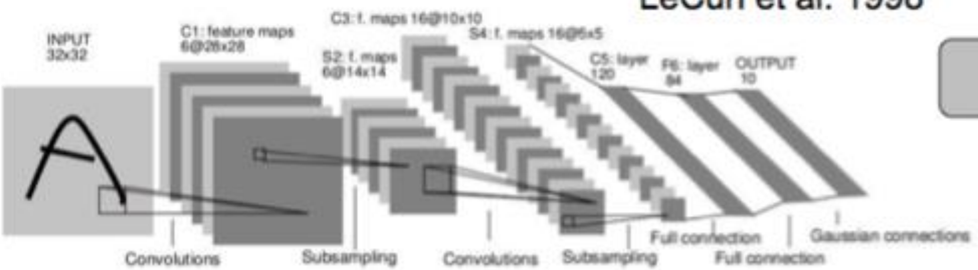
Emboss

Some theory

- Feed-forward:
  - Convolve input
  - Non-linearity (rectified linear)
  - Pooling (local max)
- Supervised
- Train convolutional filters by back-propagating classification error



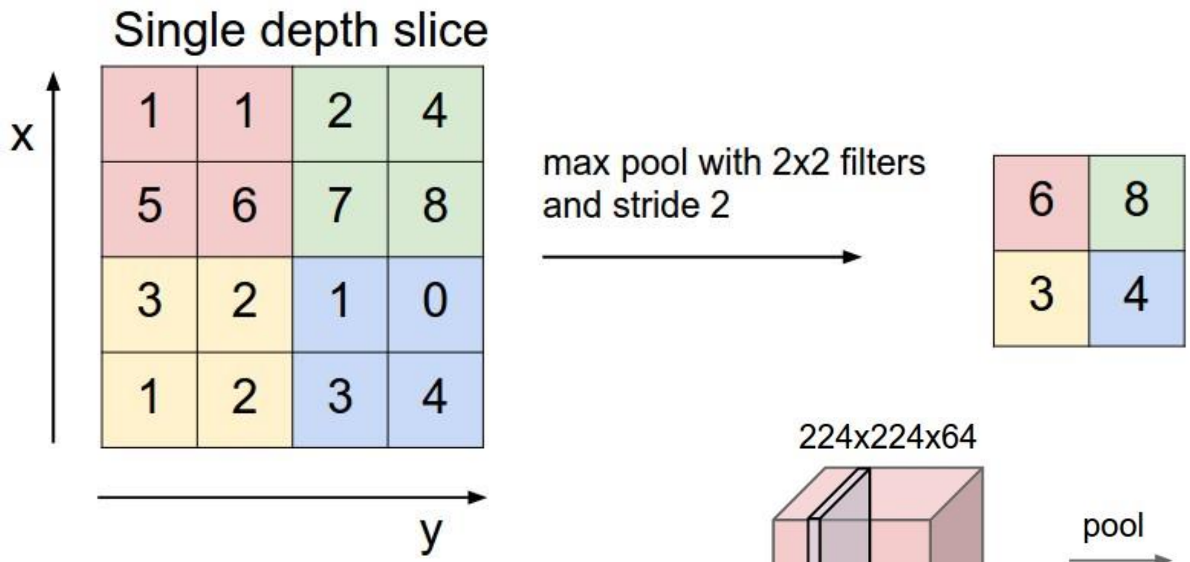
LeCun et al. 1998



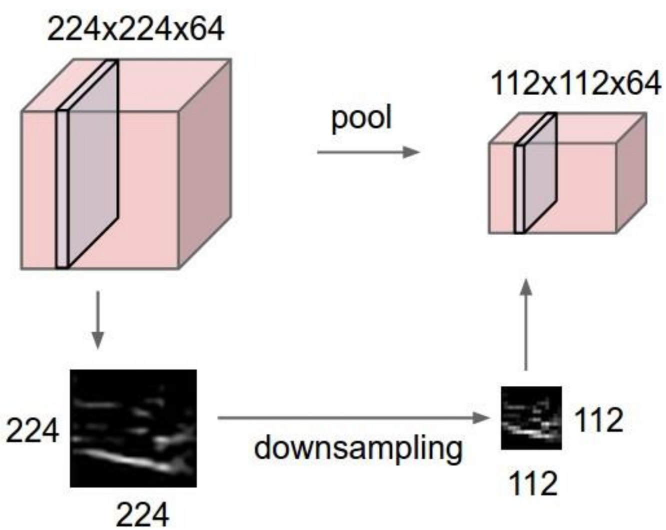
Slide: R. Fergus

Some theory

Pooling

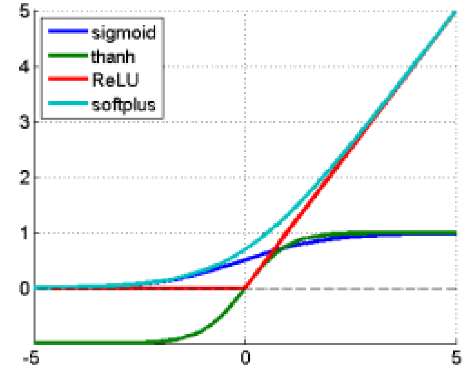


- Introduces subsampling



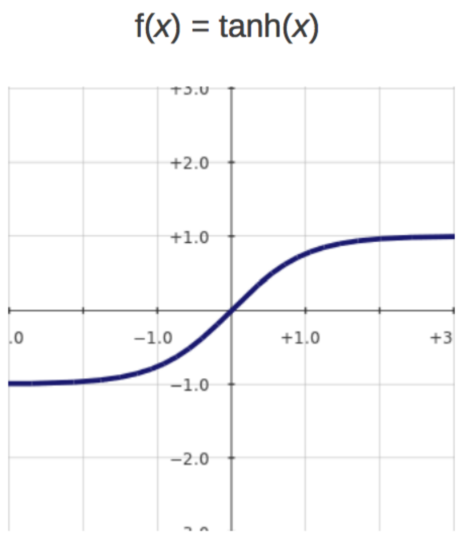
Some theory

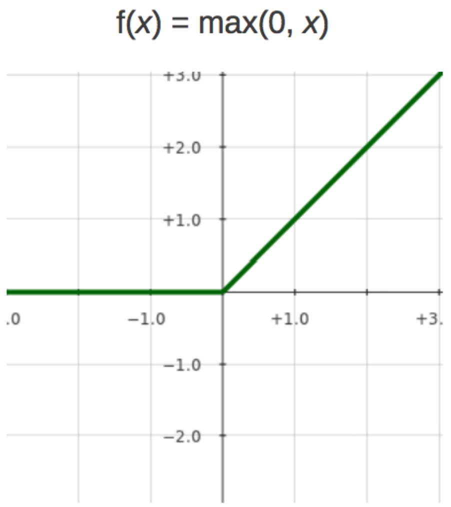
Activation



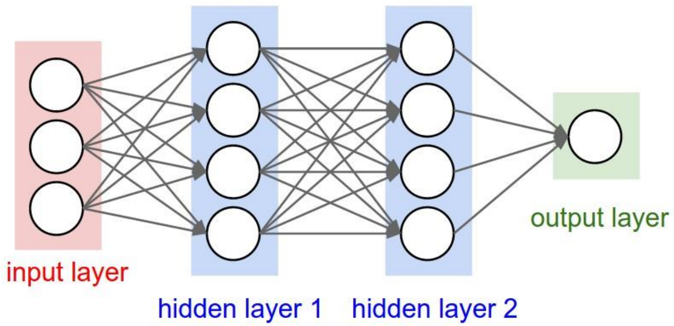
Standard way to model a neuron
f(x) = tanh(x) or *f(x) = (1 + e^{-x})^{-1}* → $f(x) = \tanh(x)$ or $f(x) = (1 + e^{-x})^{-1}$
Very slow to train (saturation)

Non-saturating nonlinearity (RELU)
$f(x) = \max(0, x)$
Quick to train

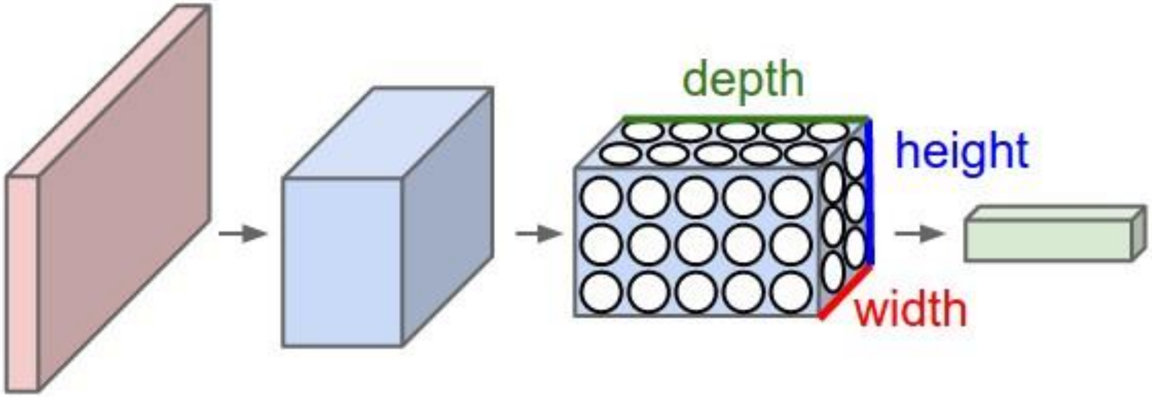




Some theory



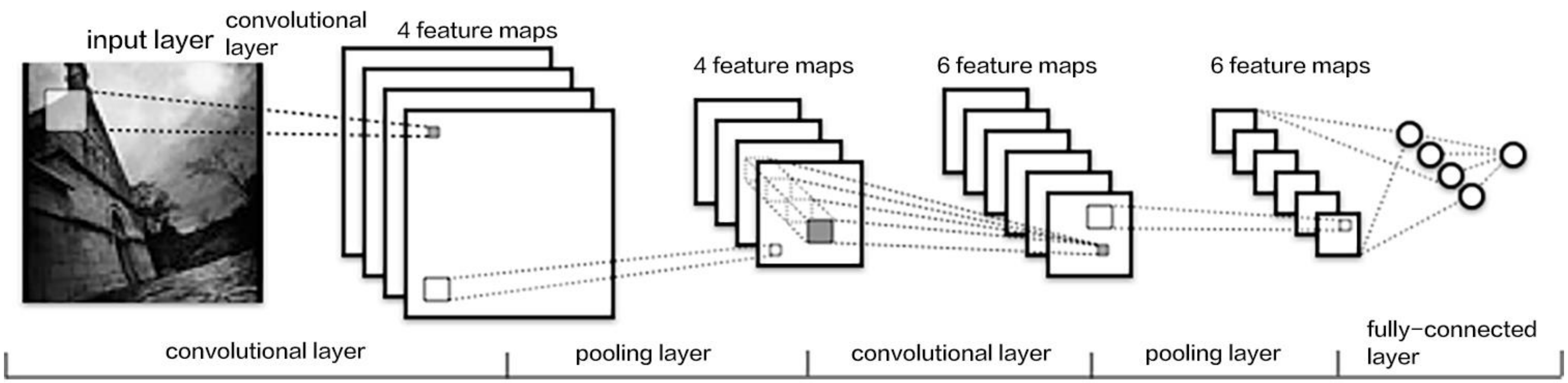
A regular 3-layer Neural Network

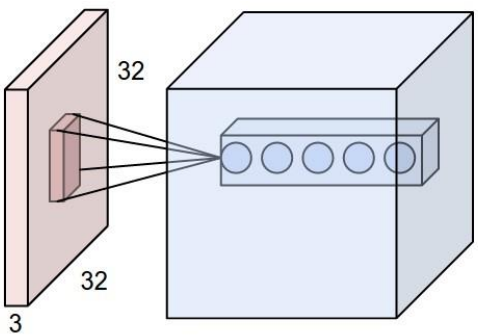


Every convolutional layer of a CNN transforms the 3D input volume to a 3D output volume of neuron activations.

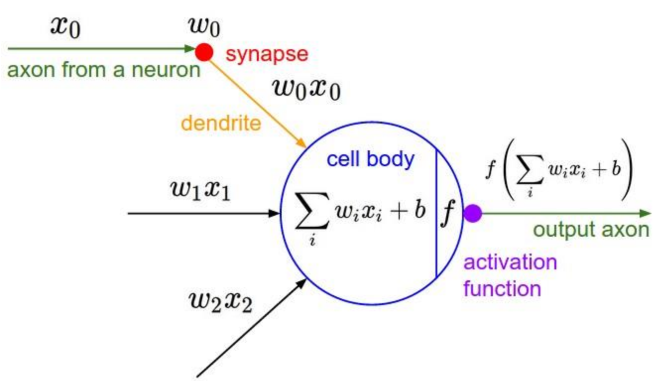
Material from Fei-Fei's group

Some theory





Each neuron is connected to a local region in the input volume spatially, but to all channels



The neurons still compute a dot product of their weights with the input followed by a non-linearity

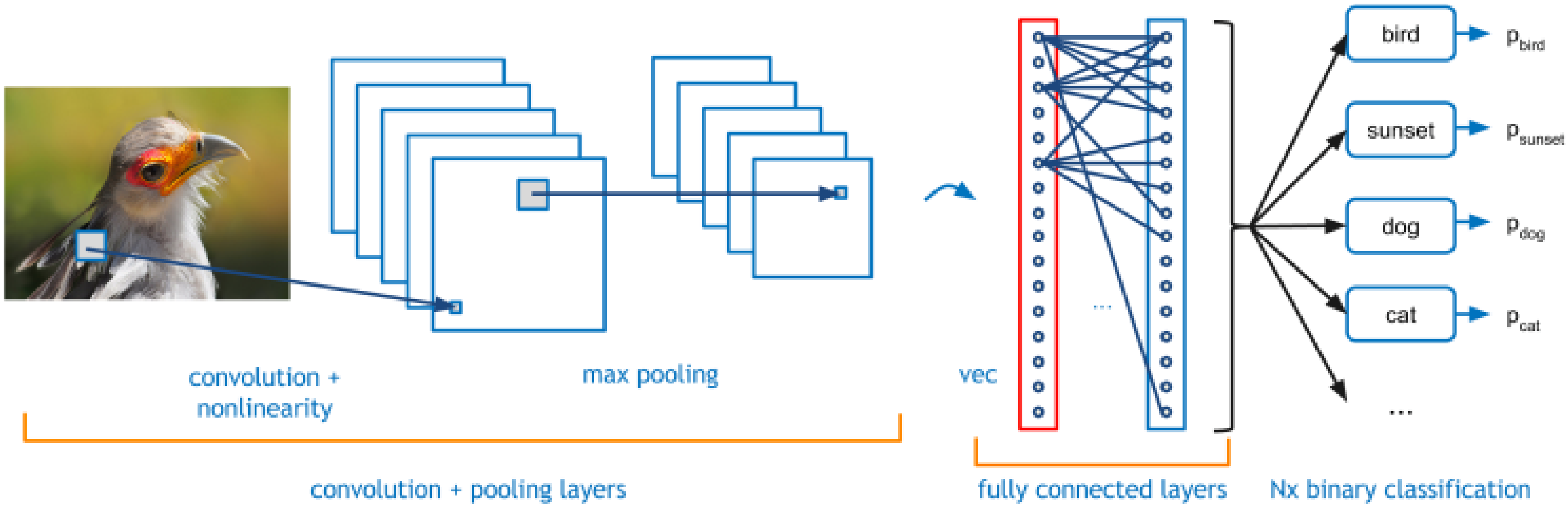
Material from Fei-Fei's group

Algorithms

- Each* neuron/layer is differentiable!
- Backpropagation algorithm (chain-rule)
- Use standard gradient-based optimization algorithms (SGD, AdaGrad, …)
- *The devil lies in the details* though …
  - Choosing hyperparameters / loss-function
  - Exploding/Vanishing gradients – batch normalization
  - Overfitting – Regularization
  - Cost of performing experiments
  - Convergence
  - …

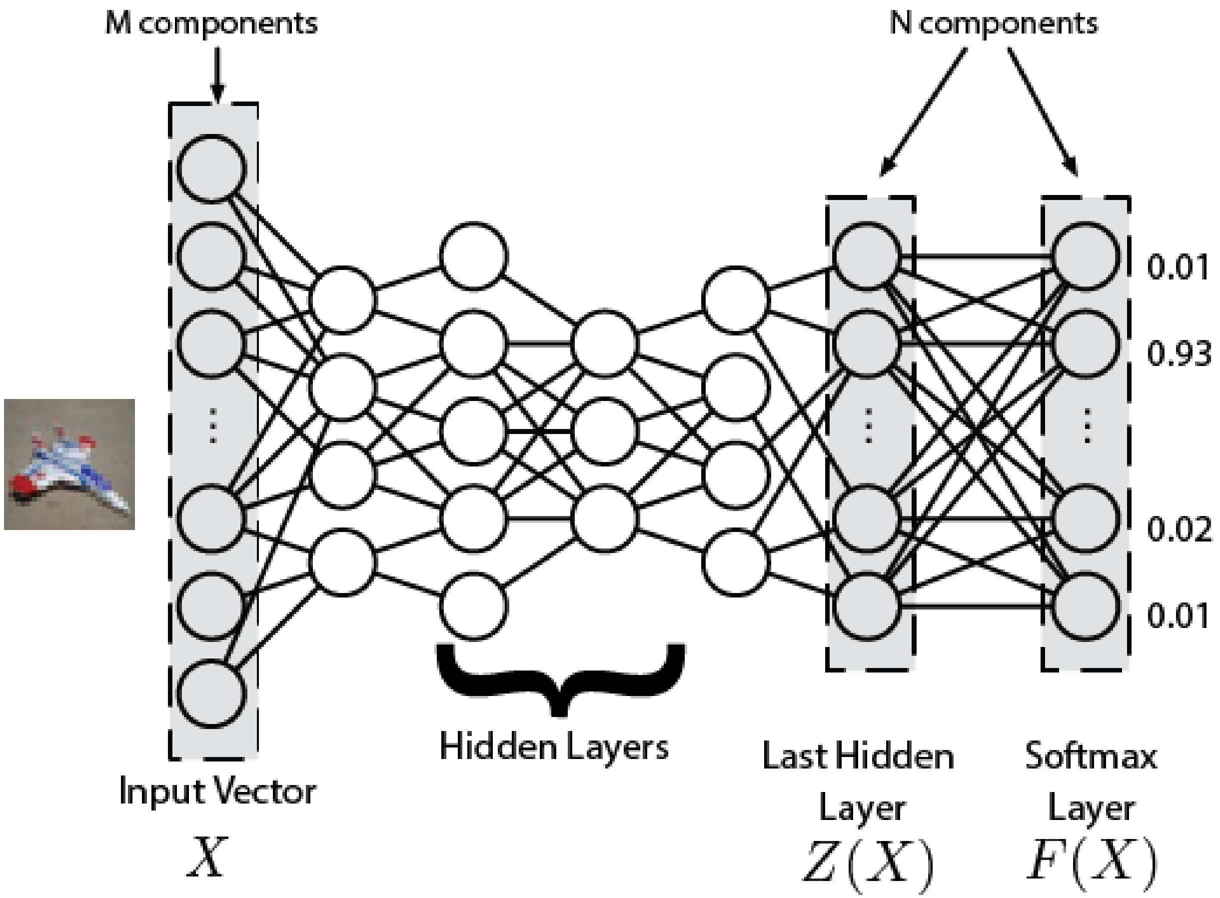
*what about max-pooling?

Image classification with CNNs



Slides from Caffe framework tutorial @ CVPR2015

Image classification with CNNs



Slides from Caffe framework tutorial @ CVPR2015

Cost function

Multiclass classication

Softmax activation function

$$y = \mathrm{softmax}(Z)_i = \frac{\exp(Z_i)}{\sum_j \exp(Z_j)}$$

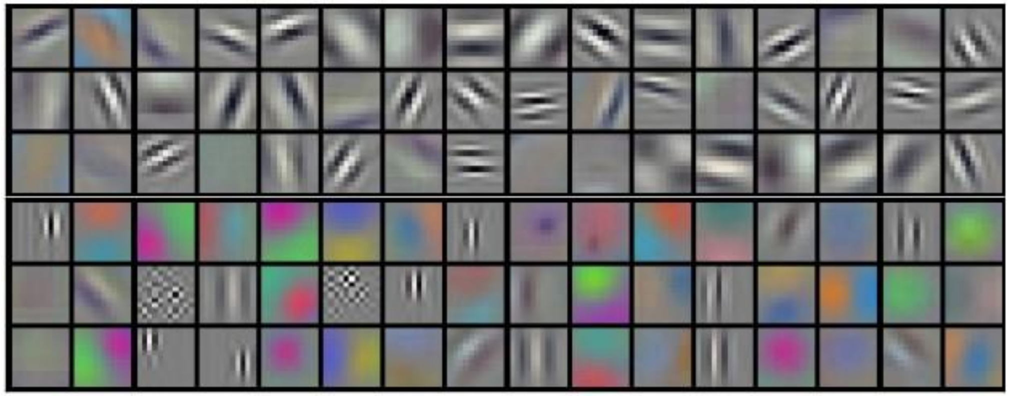
Likelihood corresponds to a Multinomial distribution

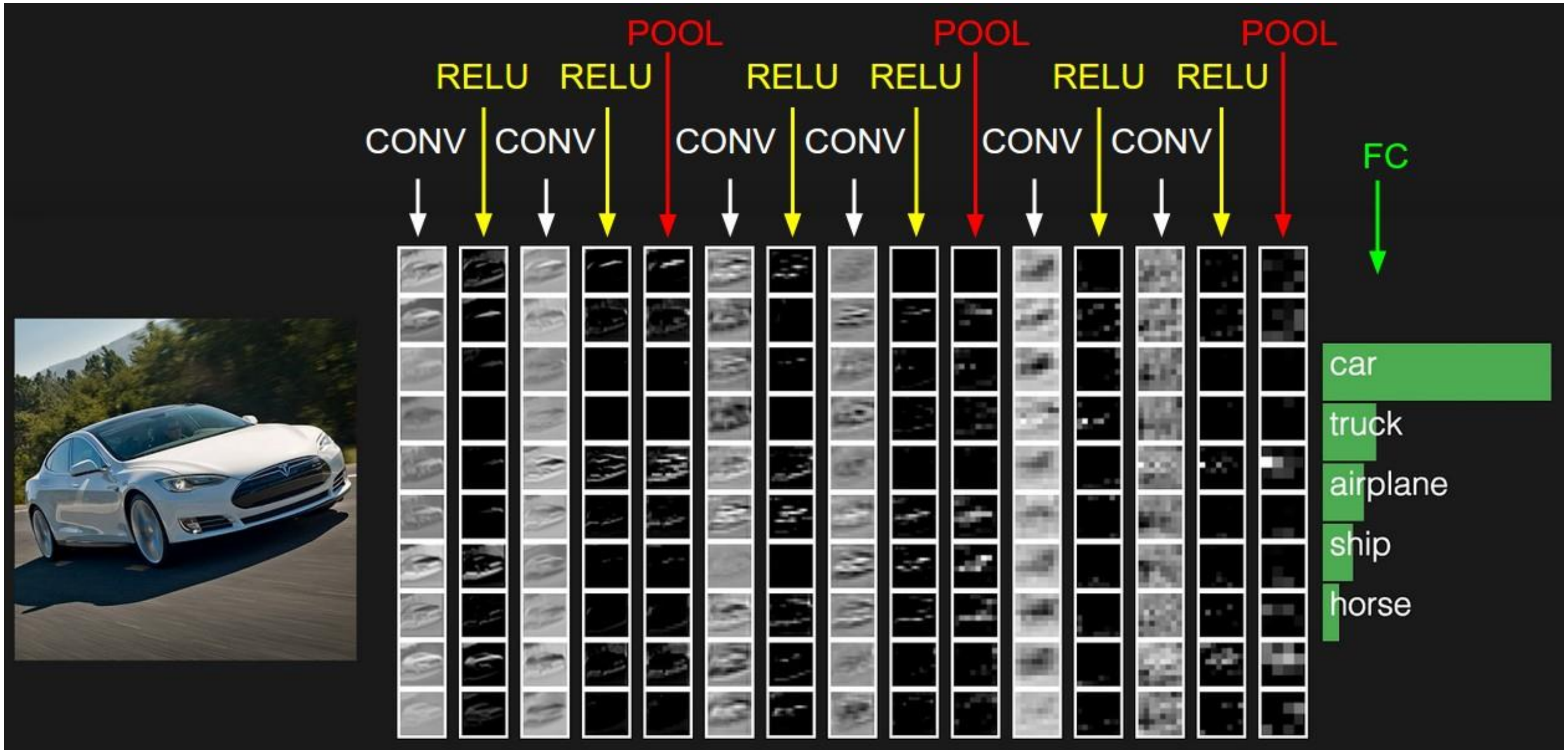
$$J_i = -\ln \mathrm{softmax}(Z)_i = \ln \sum_j \exp(Z_j) - Z_i$$

Train network by minimizing the cross-entropy loss

$$\mathcal{L} = \sum_{i=1}^{N} y_i J_i = -\sum_{i=1}^{N} y_i \ln \mathrm{softmax}(Z)_i$$

Kernels and Feature maps





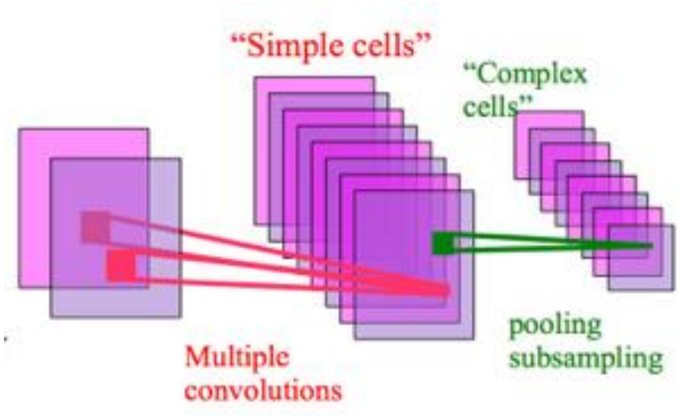
Material from Fei-Fei's group

Brief history of CNNs

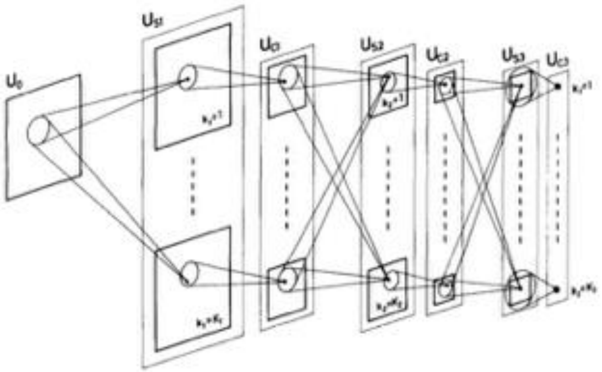
Foundational work done in the middle of the 1900s

- 1940s-1960s: Cybernetics [McCulloch and Pitts 1943, Hebb 1949, Rosenblatt 1958]
- 1980s-mid 1990s: Connectionism [Rumelhart 1986, Hinton 1989]
- 1990s: modern convolutional networks [LeCun et al. 1998], LSTM [Hochreiter & Schmidhuber 1997, MNIST and other large datasets]

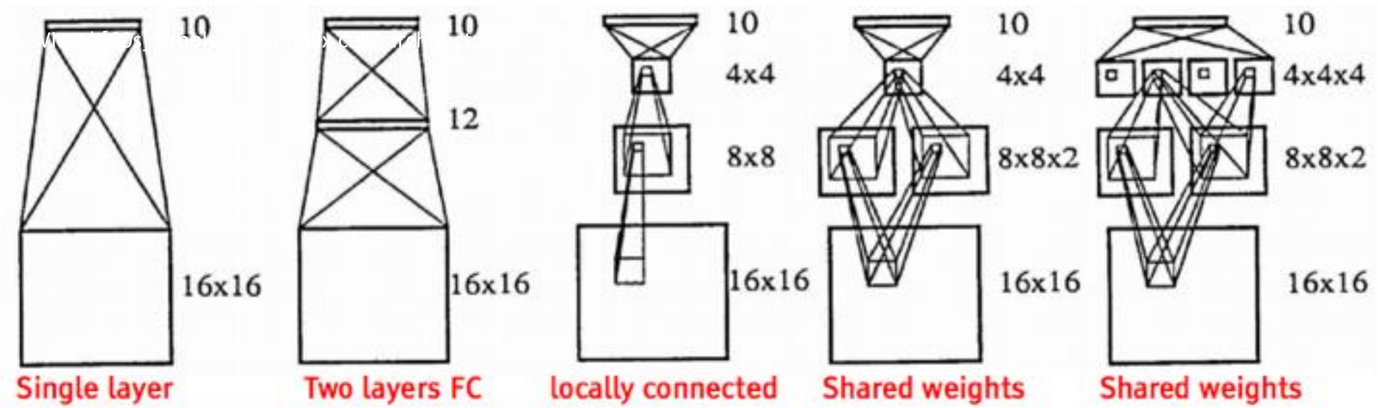
Brief history of CNNs



Hubel & Wiesel [60s] Simple & Complex cells architecture

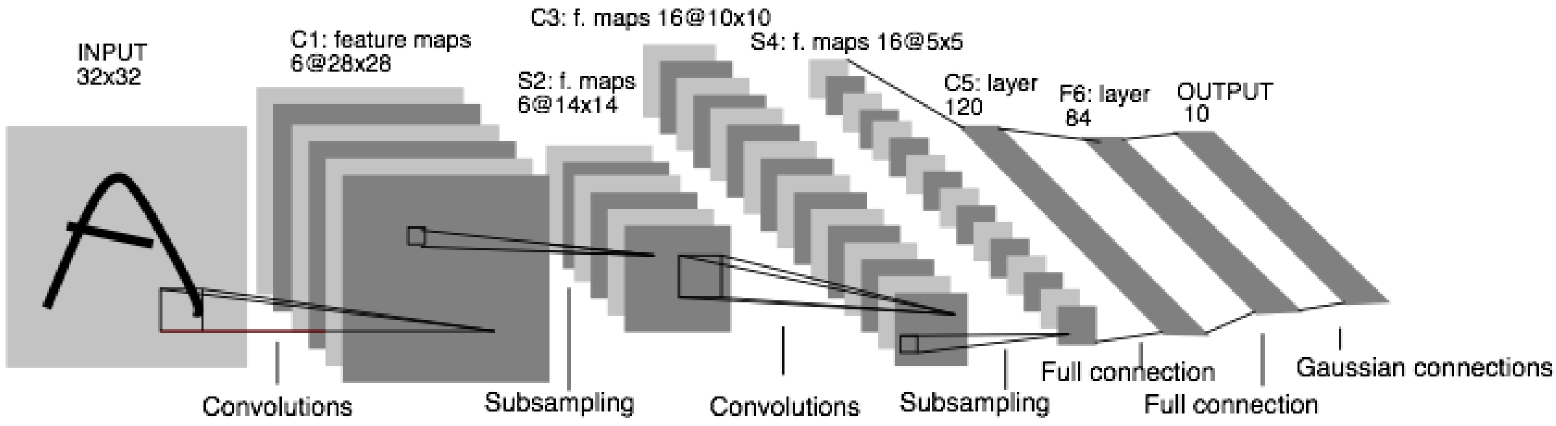


Fukushima's Neocognitron [70s]



Yann LeCun's Early CNNs [80s]:

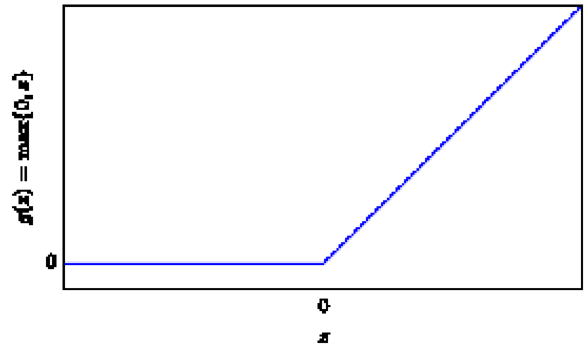
Convolutional Networks: 1989

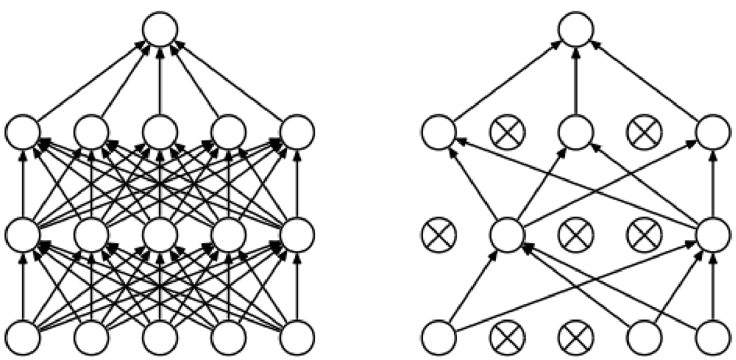


LeNet: a layered model composed of convolution and subsampling operations followed by a holistic representation and ultimately a classifier for handwritten digits. [LeNet]

Recent success

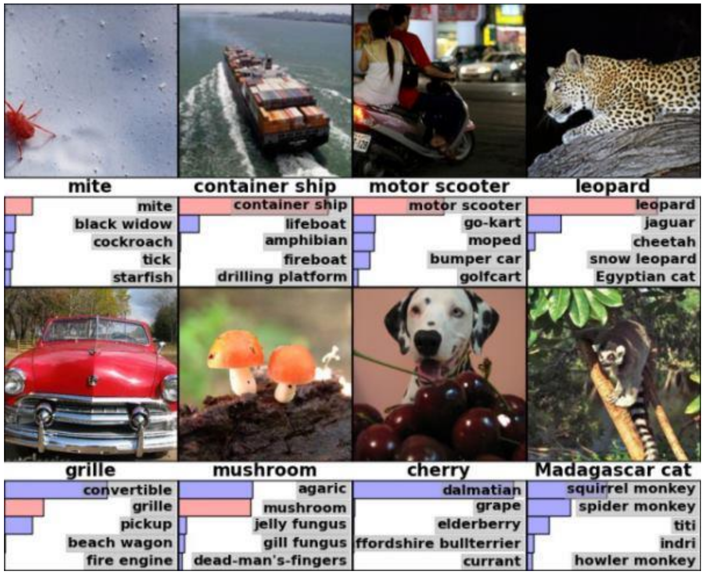
- Parallel Computation (GPU)
- Larger training sets
- International Competitions
- Theoretical advancements
  - Dropout
  - ReLUs
  - Batch Normalization





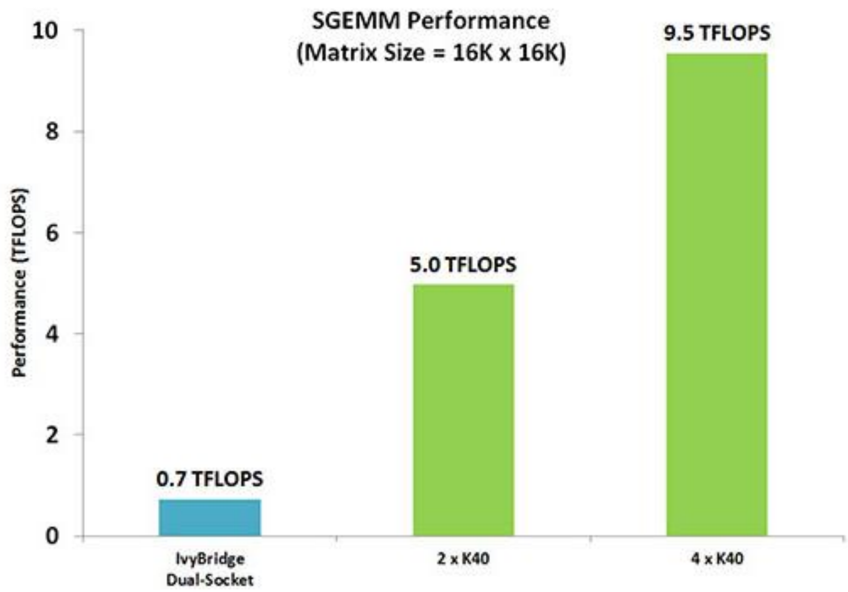
Standard Neural Net

After applying dropout.



Recent success

Better Hardware – GPUs



CUDA Jetson TX1, TK1

OpenCL branch

Android lib, demo

Recent success

Larger training sets

ImageNet

- Over 15M labeled high resolution images
- Roughly 22K categories
- Collected from web and labeled by Amazon Mechanical Turk

Recent success

Competitions

ILSVRC

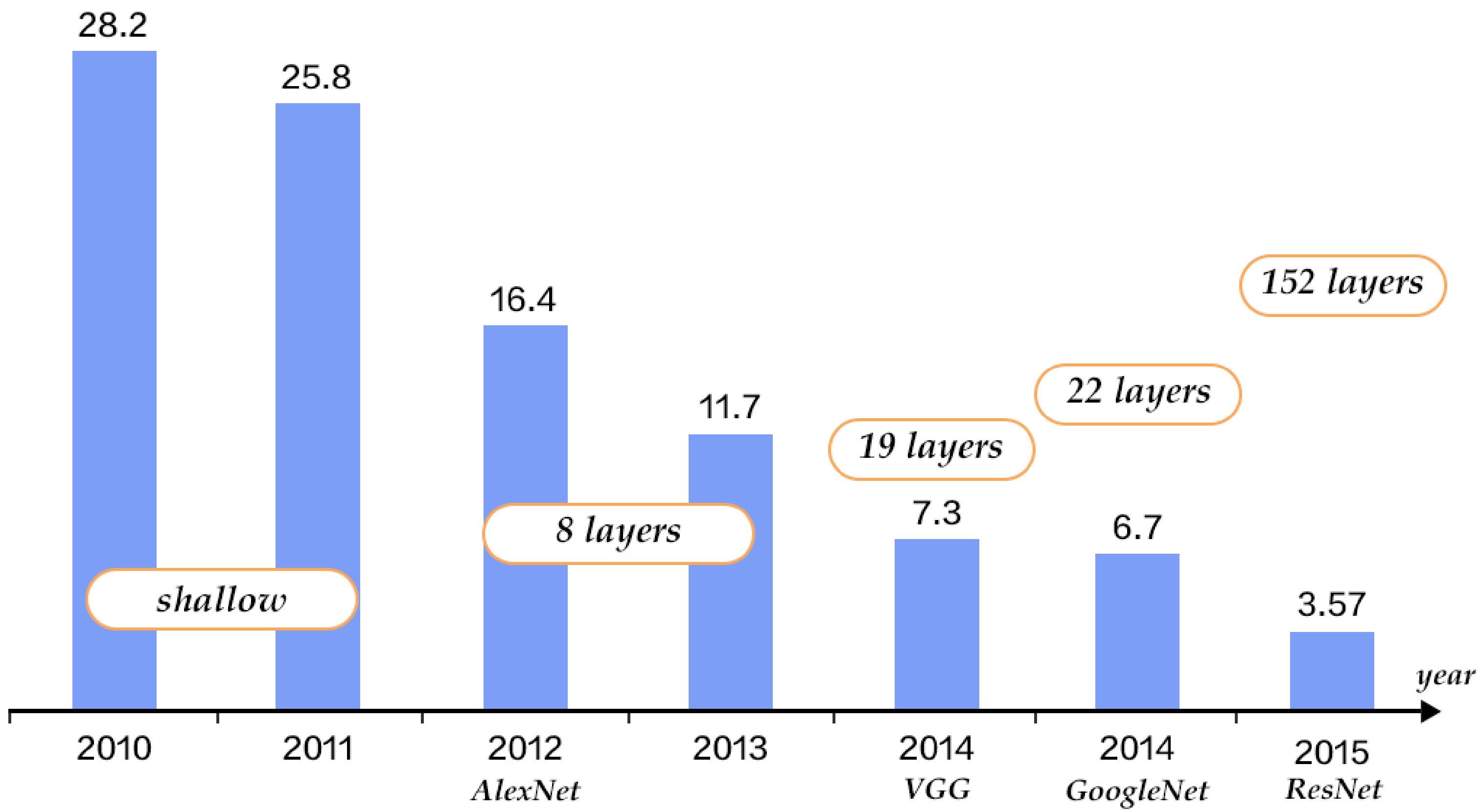
- Annual competition of image classification at large scale
- 1.2M images in 1K categories
- Classification: make 5 guesses about the image label

EntleBucher

Appenzeller

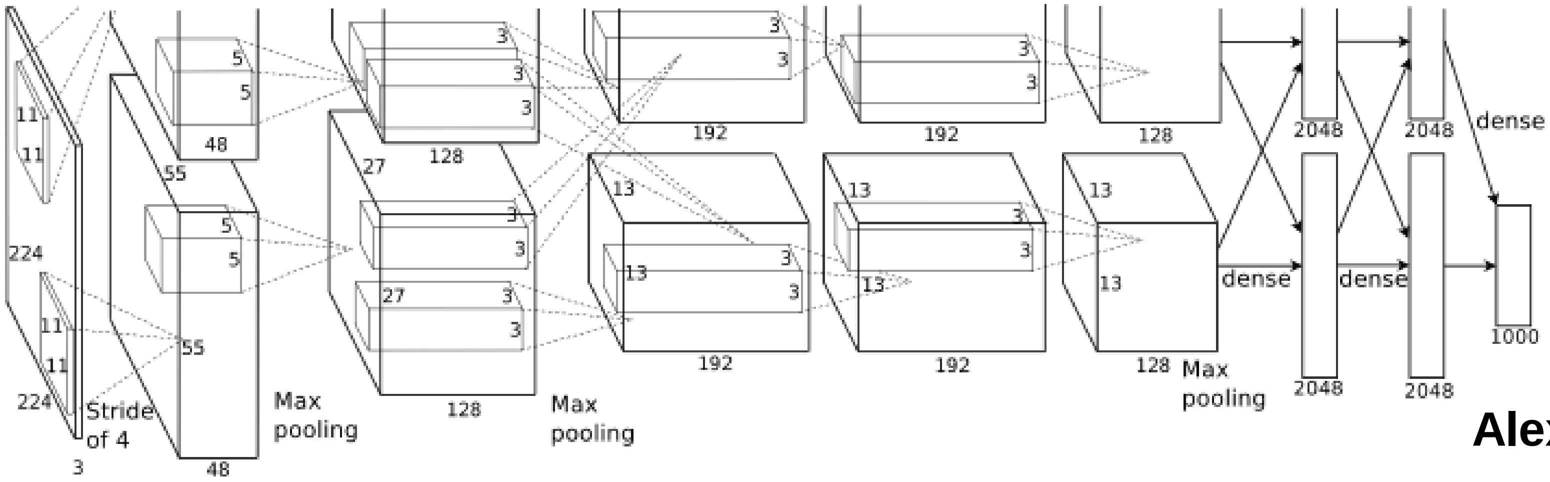
CNNs in Computer Vision

- Image classification



Evolution of CNNs for image classification

Convolutional Nets: 2012



AlexNet

Model	Top-1	Top-5
Sparse coding [2]	*47.1%*	*28.2%*
SIFT + FVs [24]	*45.7%*	*25.7%*
CNN	**37.5%**	**17.0%**

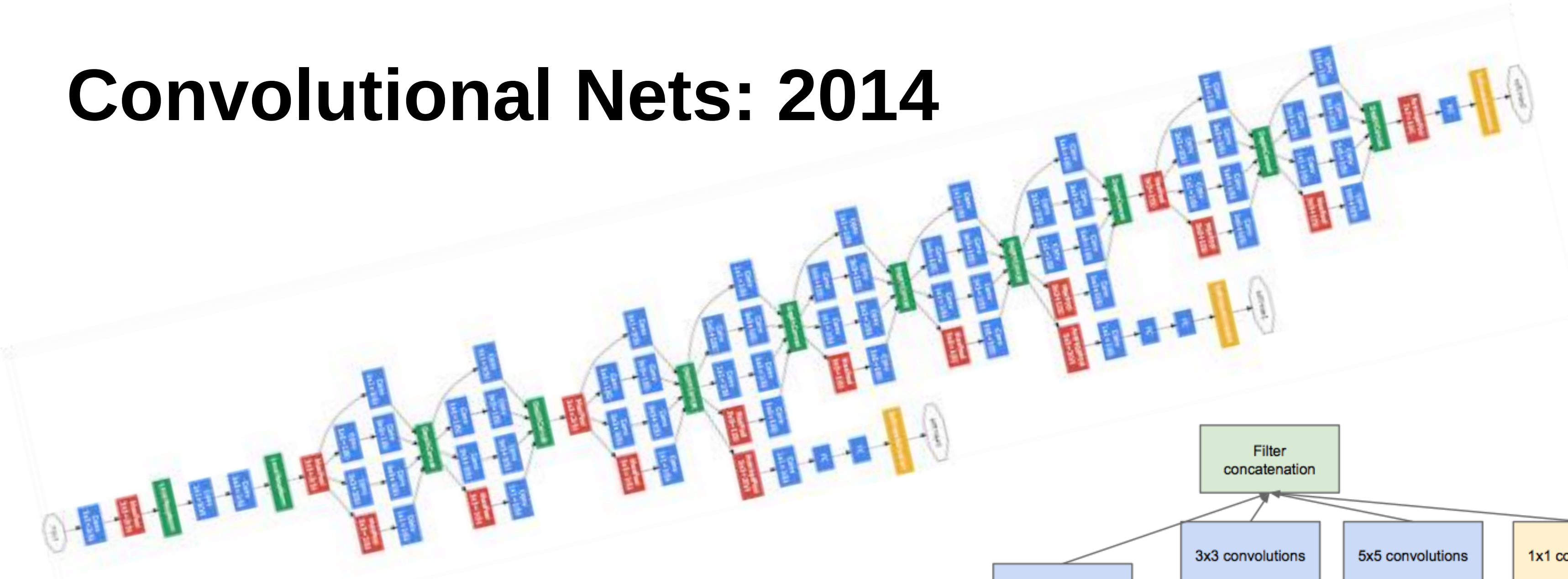
Table 1: Comparison of results on ILSVRC-2010 test set. In *italics* are best results achieved by others.

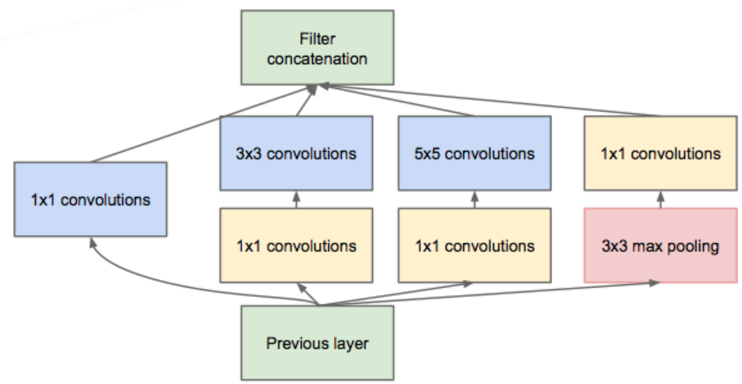
Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SIFT + FVs [7]	—	—	*26.2%*
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	**16.4%**
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	**15.3%**

Table 2: Comparison of error rates on ILSVRC-2012 validation and test sets. In *italics* are best results achieved by others. Models with an asterisk* were "pre-trained" to classify the entire ImageNet 2011 Fall release. See Section 6 for details.

Evolution of CNNs for image classification

Convolutional Nets: 2014





ILSVRC14 Winners: ~6.6% Top-5 error

- **GoogLeNet**: composition of multi-scale dimension-reduced modules

\+ depth
\+ data
\+ dimensionality reduction

Evolution of CNNs for image classification

Convolutional Nets: 2014

ILSVRC14 Winners: ~6.6% Top-5 error

- **VGG**: 16 layers of 3x3 convolution interleaved with max pooling + 3 fully-connected layers

+ depth
+ data
+ dimensionality reduction

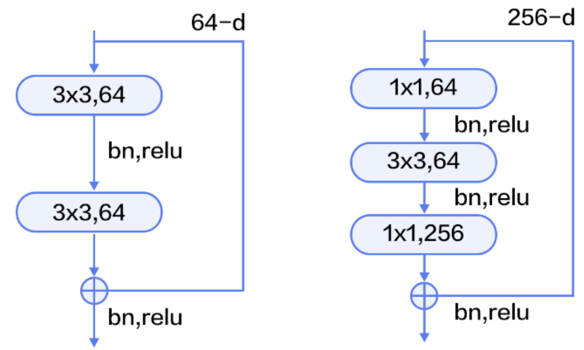
Evolution of CNNs for image classification

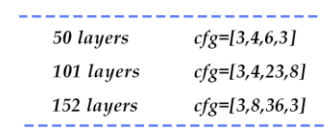
Convolutional Nets: 2015

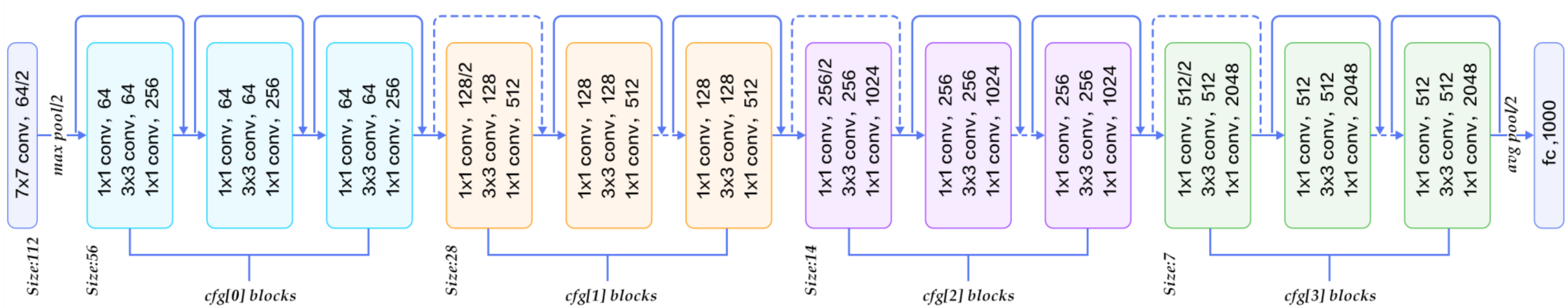
ResNet

ILSVRC15 Winner: ~3.6% Top-5 error

Intuition: Easier to learn zero than identity function







Adversarial attack methods

- White-box attacks
  - The network is “transparent” to the attacker – both the architecture and the weights are known

- Black-box attacks
  - The attacker has only access to the input and output of the network

- Gray-box attacks
  - The attacker knows the network architectures but not the weights

White-box attack methods

Fast Gradient Sign Method (FGSM)

- Classifier (e.g. ResNet-50)

$$\tilde{y} = f(\theta, \mathbf{x})$$

- Find adversarial image $\mathbf{x}'$ that maximizes the loss:

$$\mathcal{L}(\mathbf{x}', y) = \mathcal{L}(f(\theta, \mathbf{x}'), y)$$

- Bounded perturbation:

$\|\mathbf{x}' - \mathbf{x}\|_\infty \leq \epsilon$, ϵ the attack strength

Optimal adversarial image:

$$\mathbf{x}' = \mathbf{x} + \epsilon \cdot \mathrm{sign}\left(\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, y)\right)$$

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

White-box attack methods

Iterative Fast Gradient Sign Method (IFGSM)

- Similar to FGSM
- Generates enhanced attacks

$$\mathbf{x}^{(m)} = \mathbf{x}^{(m-1)} + \epsilon \cdot \mathrm{sign}\left(\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^{(m-1)}, y)\right)$$

with $\mathbf{x}^{(0)} = \mathbf{x}$ and $\mathbf{x}' = \mathbf{x}^{(M)}$, where M is the number of iterations

Both FGSM and IFGSM are fix-perturbation attacks

Kurakin et al. (2016). Adversarial examples in the physical world. *arXiv*

White-box attack methods

Step Least Likely (l.l.) attack

- Similar to FGSM

$$\mathbf{x}' = \mathbf{x} - \epsilon \cdot \mathrm{sign}\left(\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, y_{l.l.})\right)$$

where $y_{l.l.}$ the least likely class predicted by the network on clean image $\mathbf{x}$

- Strong attack as it emphasizes least likely class

Kurakin et al. (2016). Adversarial examples in the physical world. *arXiv*

White-box attack methods

CW-L2 attack (Carlini and Wagner)

- zero-confidence attack
- for all $t \neq y$ find the adversarial image that will be classified as t by solving the problem:

$$\min_{\delta} \|\delta\|_2^2$$

subject to

$$f(\mathbf{x} + \delta) = t, \;\; \mathbf{x} + \delta \in [0, 1]^n$$

- Finding the exact solution is difficult

Carlini & Wagner (2016). Towards evaluating the robustness of neural networks. *ESSP*

White-box attack methods

CW-L2 attack (Carlini and Wagner) (cont.)

- Relaxed version:

$$\min_{\delta} \|\delta\|_2^2 + c \cdot g(\mathbf{x} + \delta)$$

subject to

$$\mathbf{x} + \delta \in [0, 1]^n, \;\; c \geq 0$$

Letting $Z(\mathbf{x})$ be the neural net activations before the output layer (logits)

$$g(\mathbf{x}) = \max\left(\max_{i \neq t}(Z(x)_i) - Z(x)_t, 0\right)$$

White-box attack methods

CW-L2 attack (Carlini and Wagner) (cont.)

- Let

$$\delta = \frac{1}{2}(\tanh(\mathbf{w}) + 1) - \mathbf{x}$$

We get the following unconstrained optimization problem:

$$\min_{\mathbf{w}} \| \frac{1}{2}(\tanh(\mathbf{w}) + 1) - \mathbf{x} \|_2^2 + c \cdot$$
$$\max\left\{0, \max_{i \neq t}(Z(\frac{1}{2}(\tanh(\mathbf{w}) + 1))_i) - Z(\frac{1}{2}(\tanh(\mathbf{w}) + 1))_t\right\}$$

- powerful attack method
- resists many defense methods

White-box attack methods

Other norms

- For a bound based on L_2 norm:

$$\|\mathbf{x}' - \mathbf{x}\| \leq \epsilon$$

FGSM solution becomes:

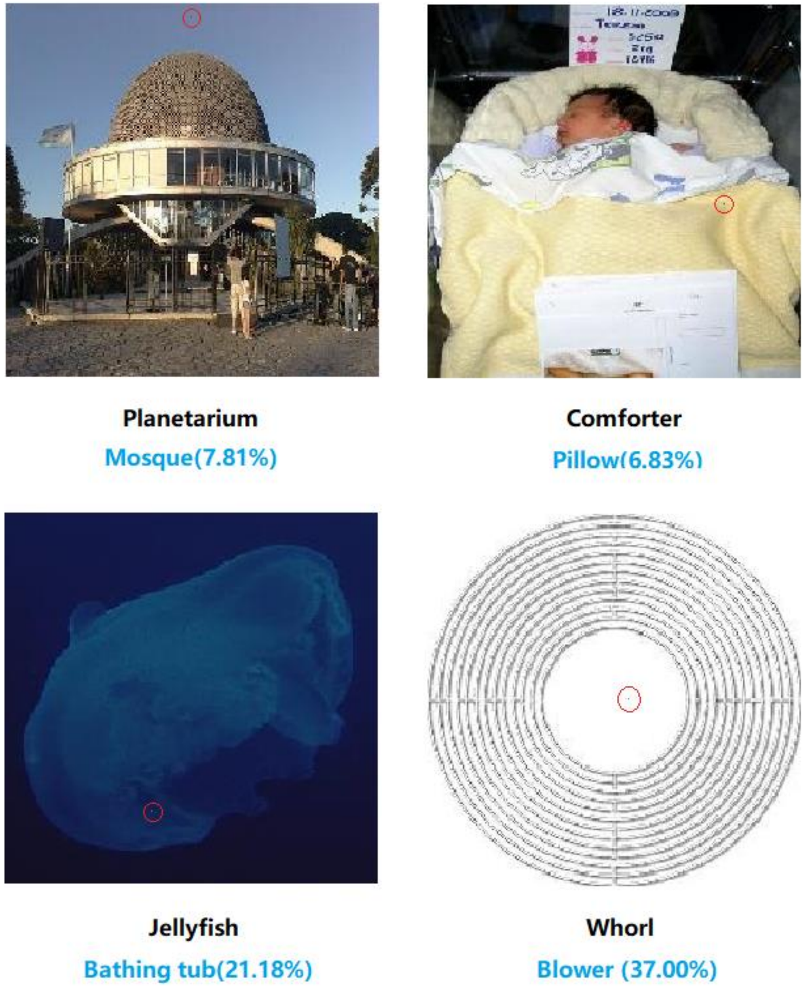
$$\mathbf{x}' = \mathbf{x} + \epsilon \frac{\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, y)}{\|\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, y)\|}$$

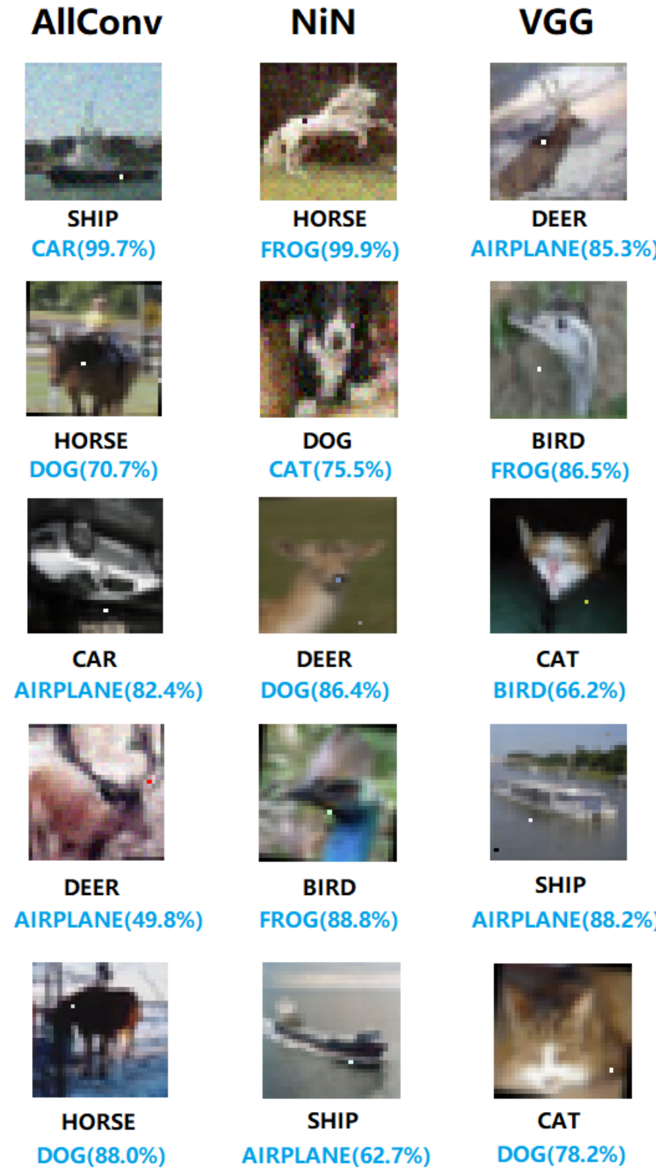
- For bounds based on L_1 and L_0 norms:
  - sparse perturbation patterns
  - e.g. single-pixel attack

Adversarial Examples for different norms

Shafahi et al. (2019). Are adversarial examples inevitable? *ICLR (to appear)*

Single Pixel attack





Su et al. (2017). One Pixel Attack for Fooling Deep Neural Networks. *IEEE Trans. Ev. Comp.*

Black-box attack methods

Transferability

- adversarial examples are highly transferable
- it is very likely that an adversarial example of one network can fool another network
- transferability depends on the type of attack
  - e.g. examples built with FGSM are highly transferable

Black-box attack methods

Main Idea

- train a substitute network based on the input/output pairs of the target network
- build adversarial examples for the substitute network
- attack the target network with the examples built for the substitute network
- due to transferability the attack is very likely to succeed

Black-box attack methods

Observations

- need “suitable” architecture for substitute network
  - High-level knowledge about the problem is required (e.g. for images convolutional layers are needed)
- collection of a sufficient number of input/output pairs from the target may be costly/impractical
  - collect a limited number of samples for each class
  - augment the dataset (e.g. using the network Jacobian)

Papernot et al. (2016). Practical Black-Box Attacks against Machine Learning. *CCS*

Adversarial Example Properties

Adversarial examples success for small ϵ depends:

- Dimensionality of input space
  - The larger the dimensionality the easier to find AE
  - Theoretical results based on isoperimetric inequality
- Image complexity
  - Datasets with more “complex” classes are more susceptible

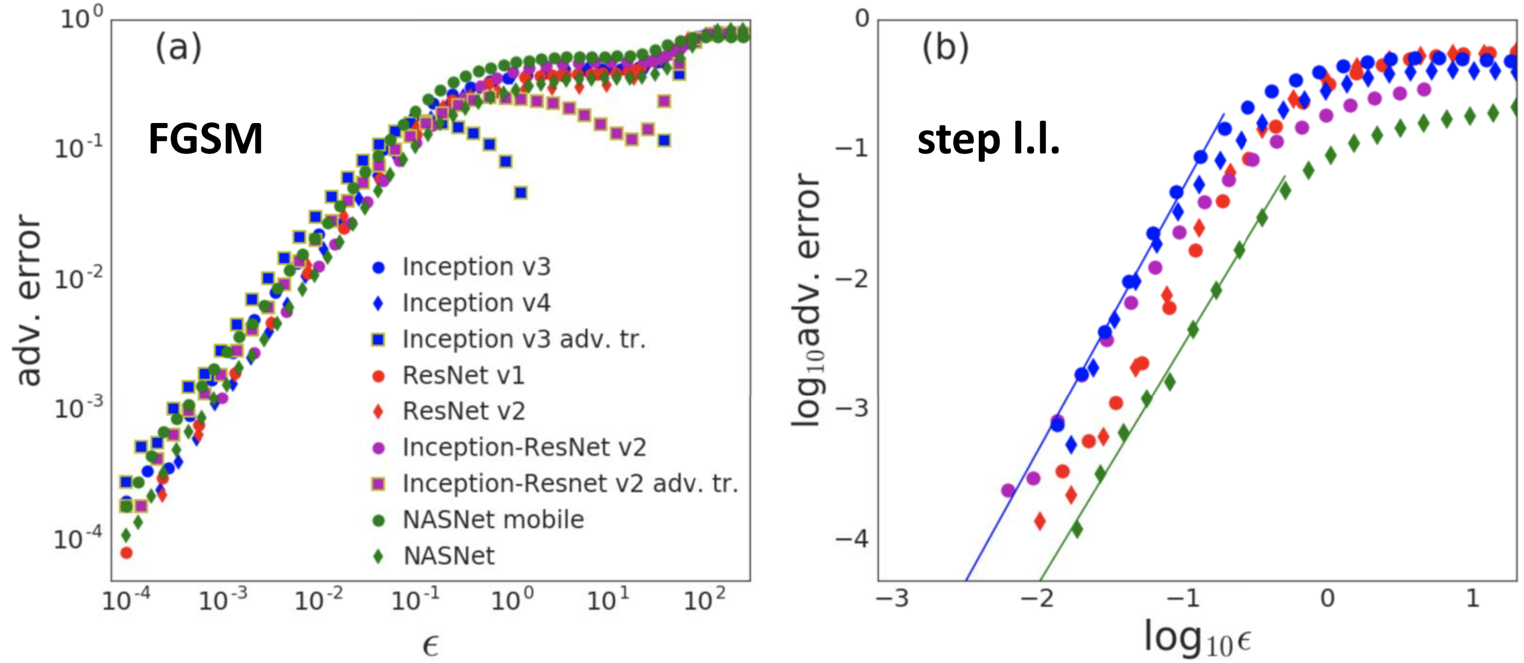
Does not depend on:

- Dataset size
- Network structure / classifier

Shafahi et al. (2019). Are adversarial examples inevitable? *ICLR (to appear)*

Adversarial Example Properties

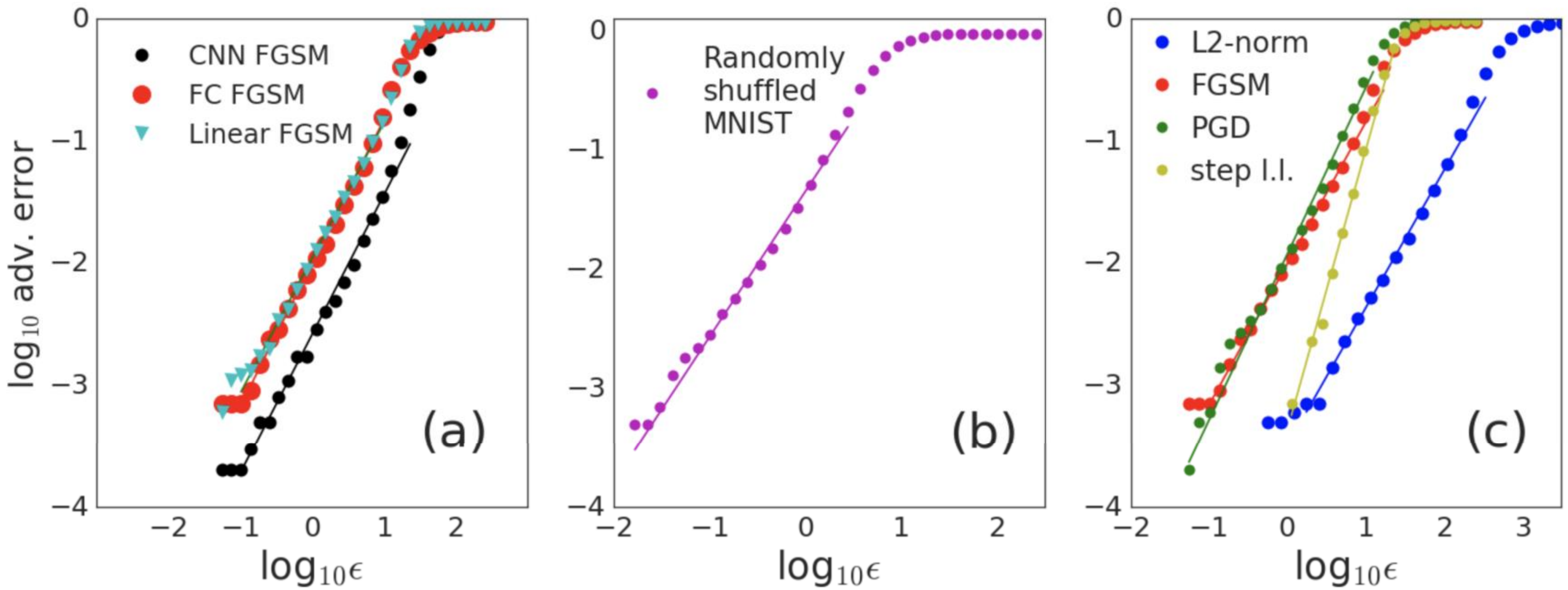
Adversarial Examples seem to follow a power law for small ϵ



Cubuk et al. (2018). Intriguing Properties of Adversarial Examples. *ICLR*

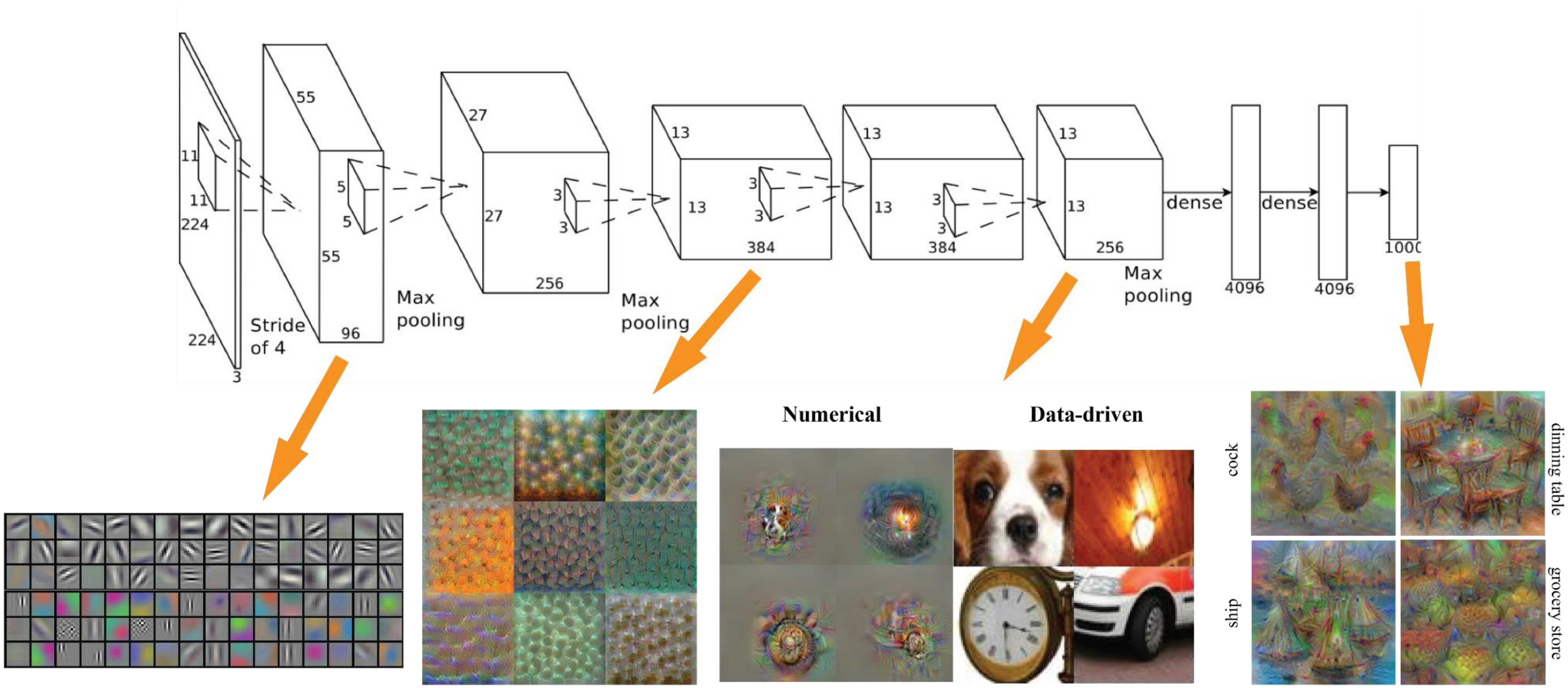
Adversarial Example Properties

Adversarial Examples seem to follow a power law for small ϵ



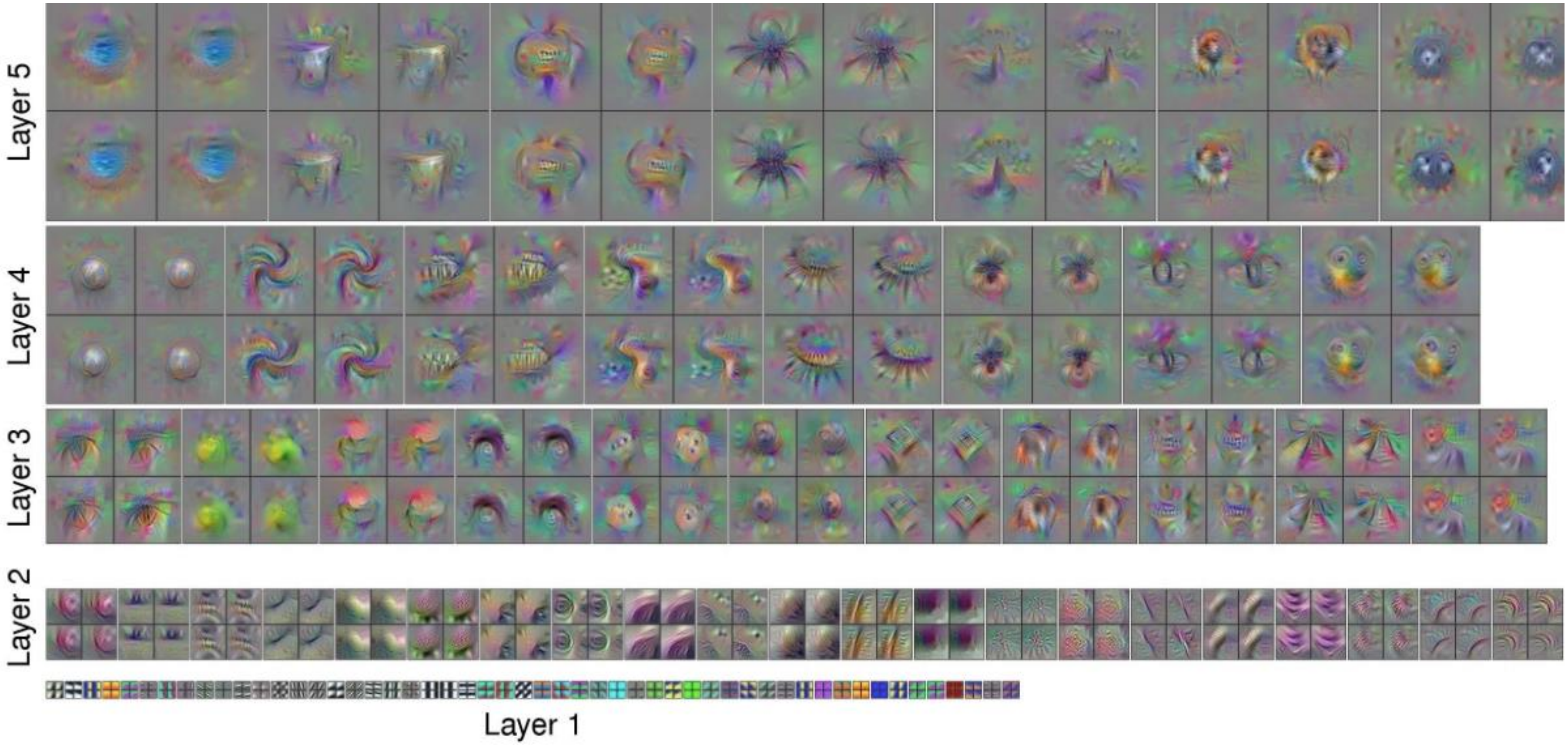
Cubuk et al. (2018). Intriguing Properties of Adversarial Examples. *ICLR*

What does the network see?



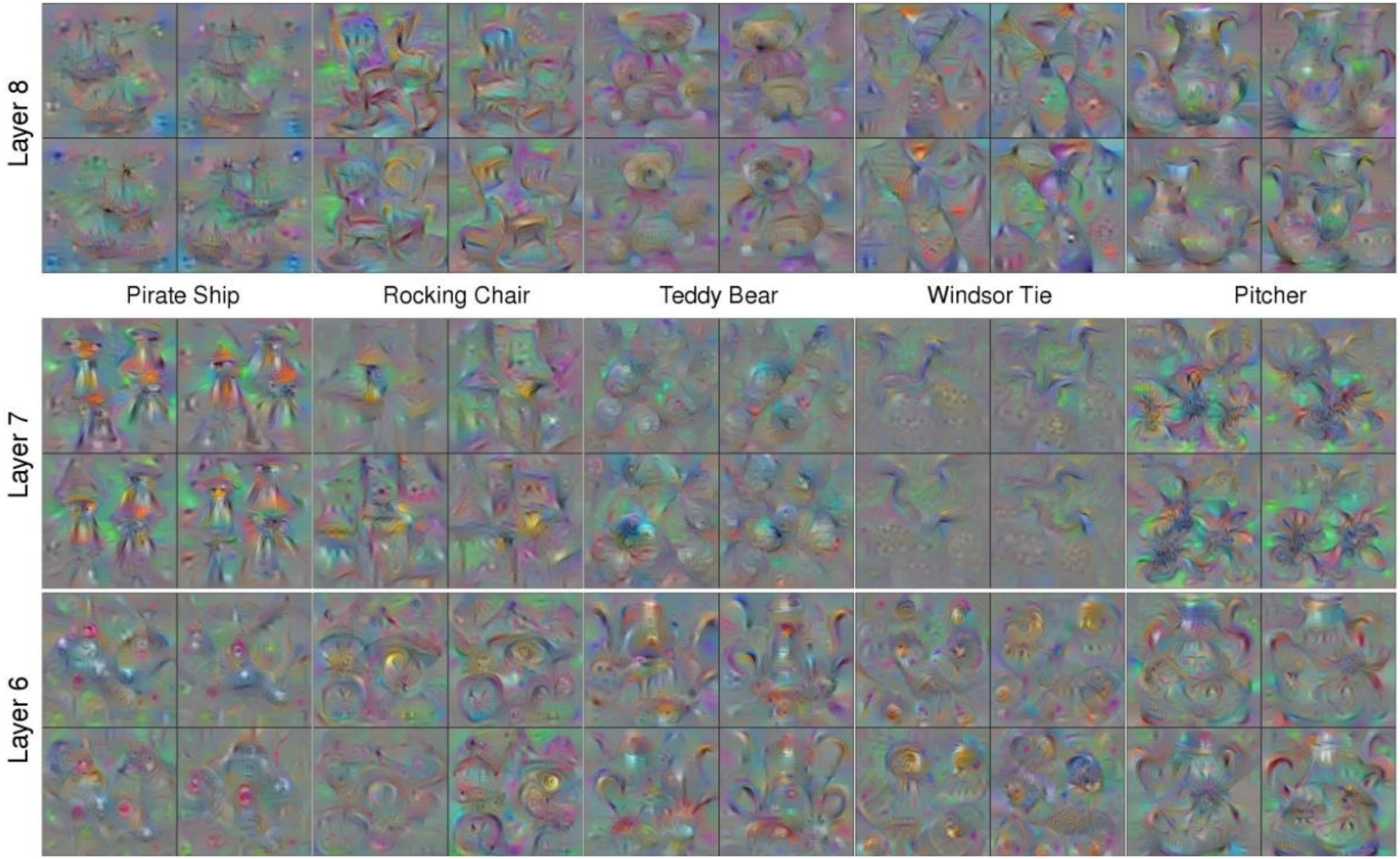
Conv 1: Edge+Blob
Conv 3: Texture
Conv 5: Object Parts
Fc8: Object Classes

What does the network see?



Understanding Neural Networks Through Deep Visualization
Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, Hod Lipson

What does the network see?



Understanding Neural Networks Through Deep Visualization
Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, Hod Lipson

What does the network see?

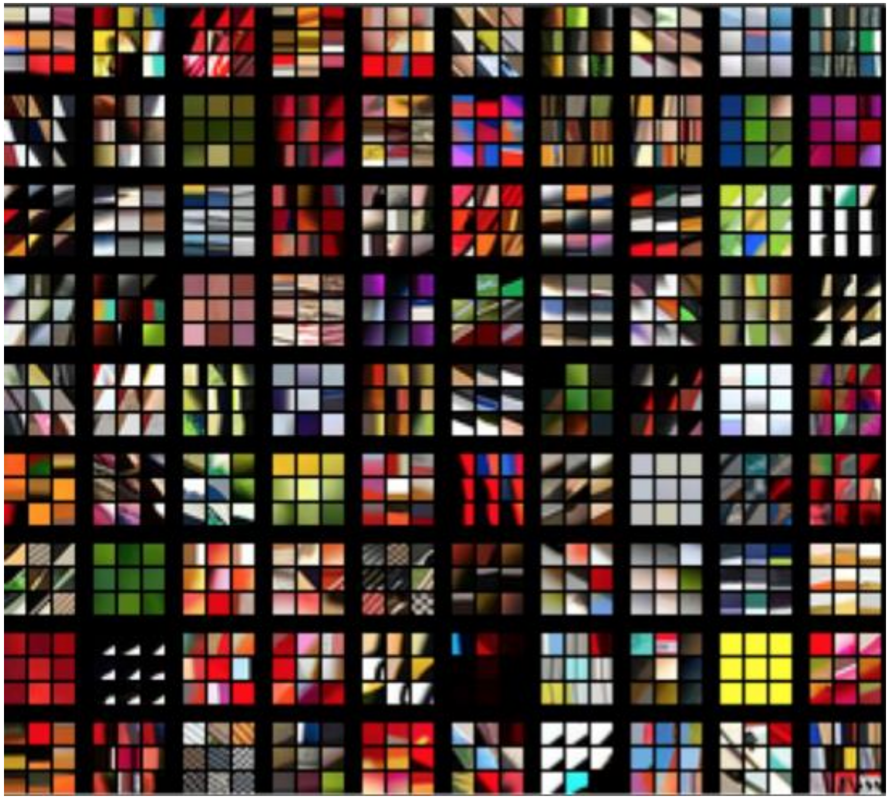
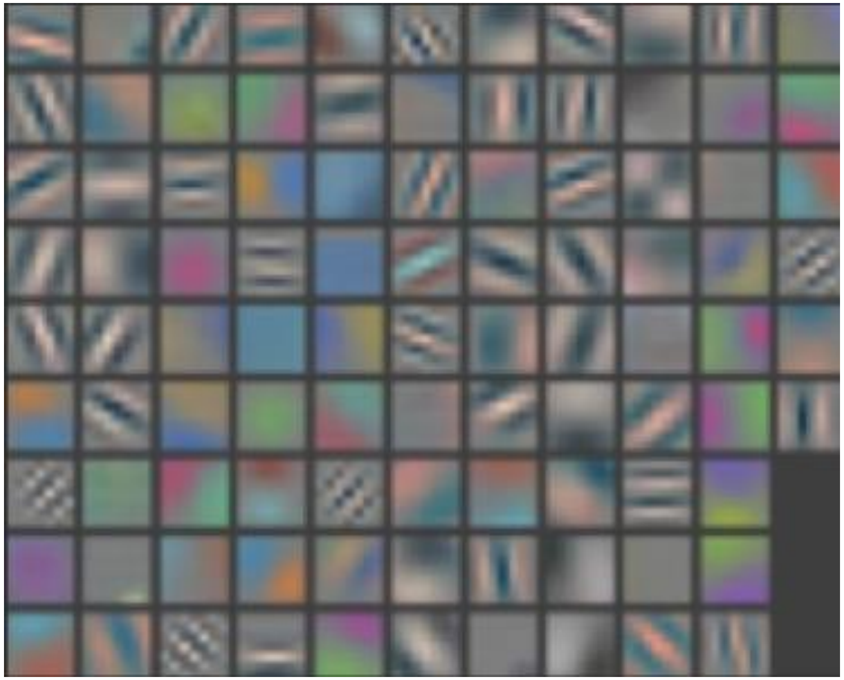


image patches that strongly activate 1st layer filters



1st layer filters

[Zeiler-Fergus]

Defense Mechanisms – Adversarial Training

Main idea

Augment the training dataset with adversarial examples

Pros:

- simple to implement
- works well for the considered attack types

Cons:

- depends on specific attack type / strength
- less effective against black-box attacks
- leads to accuracy drop of unperturbed images

Bruna et al. (2014). Intriguing Properties of Neural Networks. *ICLR*

Defense Mechanisms – Gradient Masking

Main idea

Build a model that does not have useful gradients

- e.g. replacing the last layers with nearest neighbor classifier

Pros:

- simple to implement
- effective against white-box attacks

Cons:

- Not effective against black-box attacks
- leads to accuracy drop of unperturbed images

Papernot et al. (2016). Practical Black-Box Attacks against Machine Learning. *CCS*

Defense Mechanisms – PGD Adversarial Training

Main idea

Instead of simply training the network with adversarial examples solve the saddle point problem:

$$\min_{\theta} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} \left[\max_{\delta \in S} \mathcal{L}(f_\theta(\mathbf{x}+\delta), y) \right]$$

Pros:

- State-of-the-art performance

Cons:

- depends on specific attack type

Madry et al. (2018). Towards deep learning models resistant to adversarial attacks. *ICLR*

Defense Mechanisms – DefenseGANs

Main idea

Train a Generative Adversarial Network (GAN) that generates unperturbed images

Instead of classifying a given input image, use the closest image generated by the GAN

Pros:

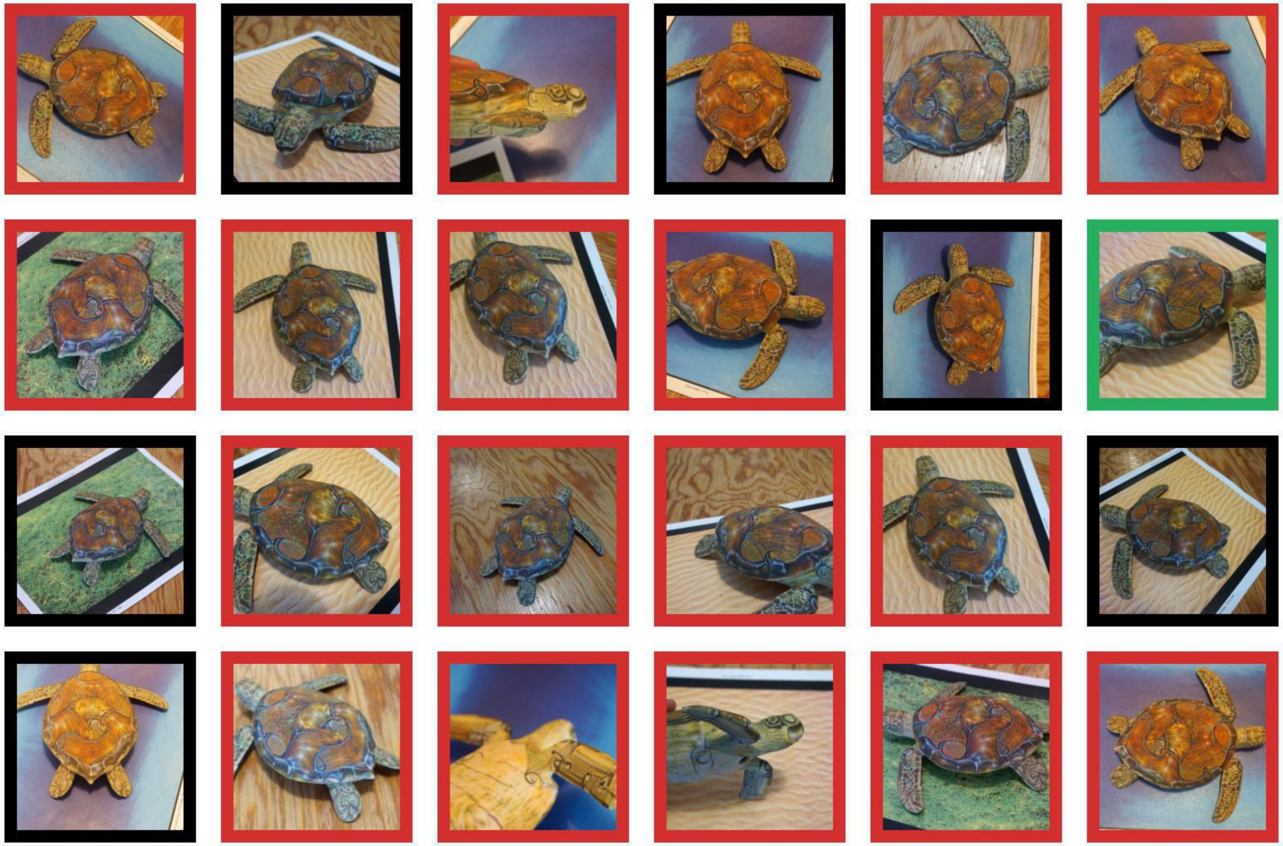
- effective against white-box and black-box attacks
- no accuracy drop (theoretically)

Cons:

- complex method
- difficult to train GAN

Samangouei et al. (2018). Defense-gan: Protecting classifiers against adversarial attacks using generative models. *ICLR*

3D Adversarial Objects



Classified as turtle Classified as other Classified as rifle

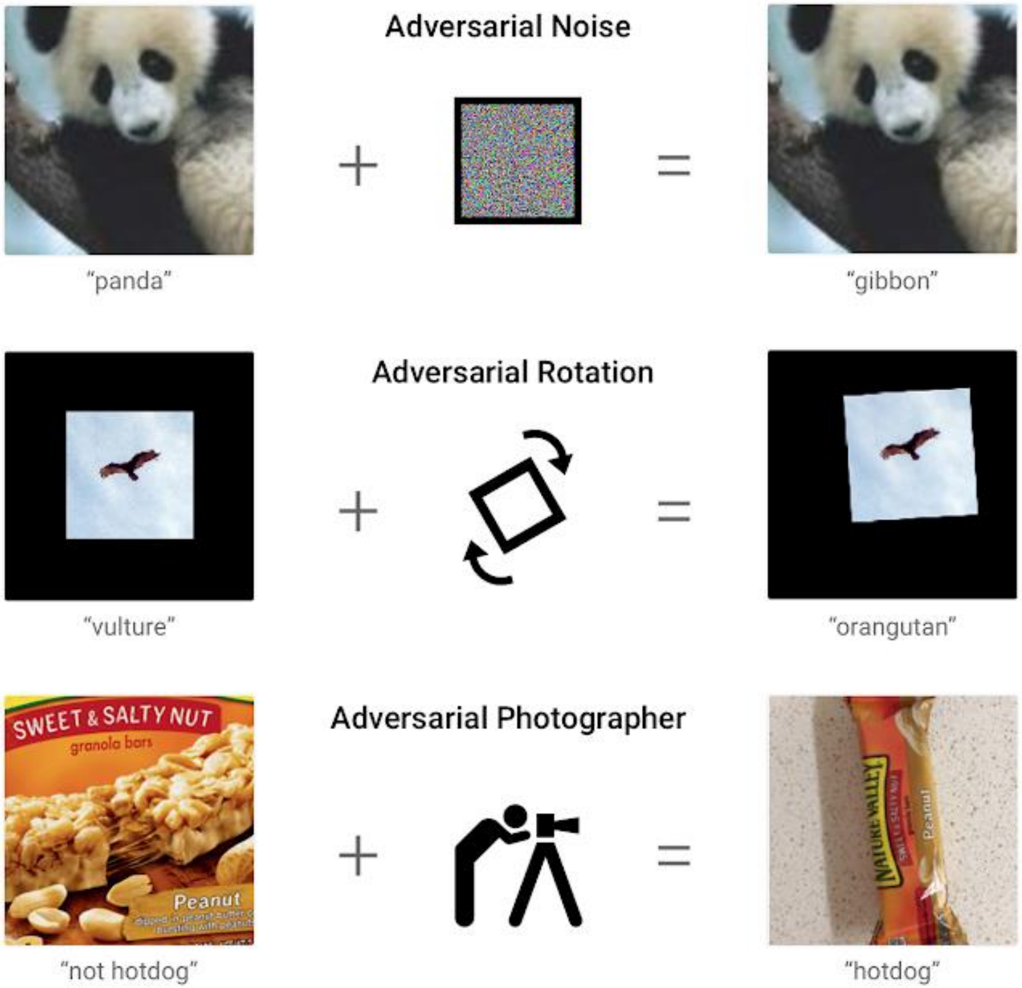
Athalye et al. (2018). Synthesizing Robust Adversarial Examples. *PMLR*

3D Adversarial Objects

Classified as baseball   Classified as other   Classified as espresso

Athalye et al. (2018). Synthesizing Robust Adversarial Examples. *PMLR*

Other types of attack



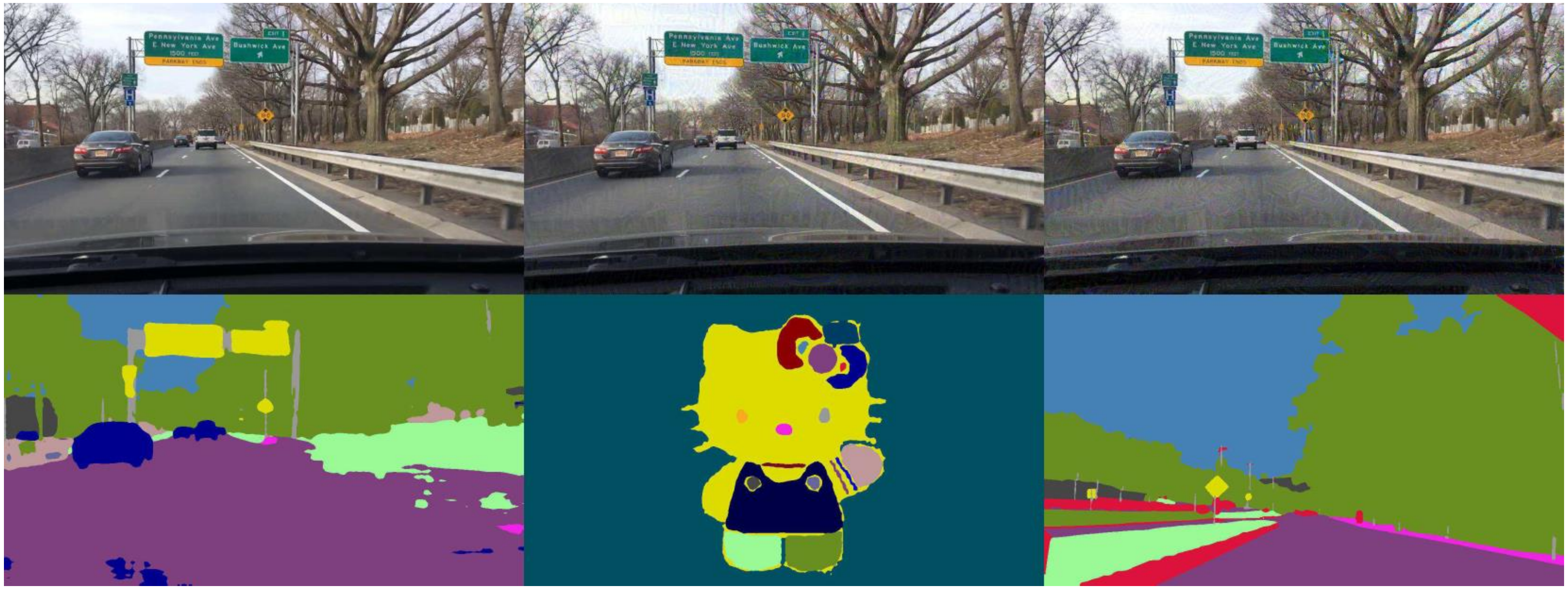
Brown et al. (2018). Unrestricted Adversarial Examples. *arXiv*

Adversarial Examples in Semantic Segmentation

Each color represents a different class:
(road, traffic sign, car, sky, building, etc.)

Xiao et al. (2018). Characterizing Adversarial Examples Based on Spatial Consistency Information for Semantic Segmentation. *ECCV*

Adversarial Examples in Semantic Segmentation



Each color represents a different class:
(road, traffic sign, car, sky, building, etc.)

Xiao et al. (2018). Characterizing Adversarial Examples Based on Spatial Consistency Information for Semantic Segmentation. *ECCV*

Thank you!

Resources

Frameworks:

- Caffe/Caffe 2 (UC Berkeley) | C/C++, Python, Matlab
- TensorFlow (Google) | C/C++, Python, Java, Go
- Theano (U Montreal) | Python
- CNTK (Microsoft) | Python, C++ , C#/.Net, Java
- Torch/PyTorch (Facebook) | Lua/Python
- MxNet (DMLC) | Python, C++, R, Perl, …
- Darknet (Redmon J.) | C
- …

Resources

High-level libraries:

- Keras | Backends: TensorFlow (TF), Theano

Models:

- Depends on the framework, e.g.
  - https://github.com/BVLC/caffe/wiki/Model-Zoo (Caffe)
  - https://github.com/tensorflow/models/tree/master/research (TF)

Interactive Interfaces:

- DIGITS (NVIDIA) | Caffe, TF, Torch
- TensorBoard (TF)

Tools:

- http://ethereon.github.io/netscope (for networks defined in protobuf)