CSCI 599: Deep Learning and its Applications

Lecture 5

Fall 2017
Joseph J. Lim

Help from **Shao-Hua Sun**, **Youngwoon Lee**, and **Te-Lin Wu**

Disclaimer

- This course is taught for the 1st time @ USC. This course is 599, and thus an **experimental** course.
- The syllabus, course policy, and grading details **may change** over the semester (**check website!**)
- If you prefer a well-structured course, this is **NOT** a course for you, and I encourage you to take the course next year. We really mean this.
- But, it will be **fun** and **challenging**!

Speakers visiting USC

- Phillip Isola (OpenAI) on October 19th
- Jimmy Ba (Univ. of Toronto) on November 9th
- Carl Vondrick (Google) on November 28th

Important Dates

Subject to change!

- **Midterm: week 7**
- Assignment 1: **week 8** (out: week 5)
- Assignment 2: week 11
- Project
  - ~~Team formation: week 4~~
  - Project proposal: week 8
    - **Project meeting with TA #1: between week 4 - week 8**
    - Project meeting with Instructor #1: week 8 (M-W)
  - Project mid-report: week 12
    - Project meeting with TA #2: between week 8 - week 12
    - Project meeting with Instructor #2: week 11 (M-W)
  - Project report + Final presentation: week 15 (5-9:30pm) **4.5 hours**
    - Project meeting with TA #3: between week 12 - week 15

Assignment 1

- Assignment 1 is released on the website (https://csci599-dl.github.io/)
- DUE October 11th, 2017 (week 8)
- Collaboration is OK! Please list all collaborators' names.

Project

- Report: a blog post (rather than a paper format)
- Lots of "what if we fail" questions

Project

Please take this course to do a fun project, not to get an A

- Report: a blog post (rather than a paper format)
- Lots of "what if we fail" questions

Project meeting with TA #1

- We posted on Piazza your TA assignment
  - Temporary until the proposal due date
- Please sign up for your team on your TA's doodle link.
- Please be available — if not, we may not be able to schedule one with you.

Project meeting with Instructor #1

- We will follow up on this (Monday to Wednesday of week 8).
- Please be available — if not, we may not be able to schedule one with you.
  - I am meeting 58 teams over 3 days 

Note

- **No office hour next Wednesday (9/27).**
- Potentially, also a week after (10/4) — This will be announced on Piazza.

Today's agenda

- Part 1
  - Convolutional Neural Networks
- Part 2
  - Training Neural Networks

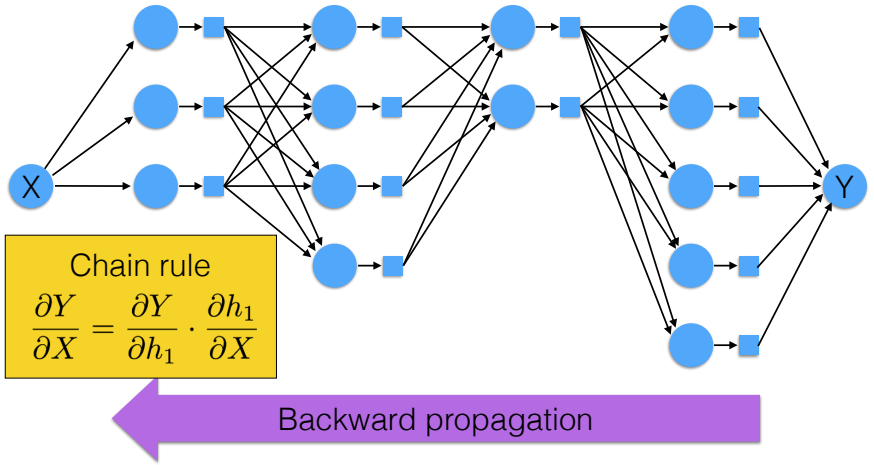
Today's agenda

- Part 1
  - Convolutional Neural Networks
- Part 2
  - Training Neural Networks

Different NN models

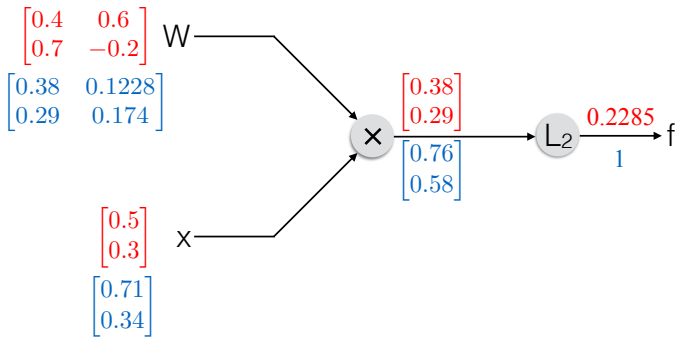
- Feedforward networks (CNNs, …)
- Recurrent networks (RNNs, …)
- Memory networks
- …

Recap: Backward Propagation



Recap: Computational Graph

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \Sigma_{i=1}^{n}(W \cdot x)_i^2$



All useless

Geoff Hinton (the godfather of deep learning) said we need to start over.

On 9/15 (two days after our back-prop lecture), he said he is "**deeply suspicious**" about back-propagation.

Artificial intelligence pioneer says we need to start over

Geoffrey Hinton harbors doubts about AI's current workhorse. (Johnny Guatto / University of Toronto)

In 1986, Geoffrey Hinton co-authored a paper that, three decades later, is central to the explosion of artificial intelligence. But Hinton says his breakthrough method should be dispensed with, and a new path to AI found.

Speaking with Axios on the sidelines of an AI conference in Toronto on Wednesday, Hinton, a professor emeritus at the University of Toronto and a Google researcher, said he is now "deeply suspicious" of back-propagation, the workhorse method that underlies most of the advances we are seeing in the AI field today, including the capacity to sort through photos and talk to Siri. "My view is throw it all away and start again," he said.

The bottom line: Other scientists at the conference said back-propagation still has a core role in AI's future. But Hinton said that, to push materially ahead, entirely new methods will probably have to be invented. "Max Planck said, 'Science progresses one funeral at a time.' The future depends on some graduate student who is deeply suspicious of everything I have said."

How it works: In back propagation, labels or "weights" are used to represent a photo or voice within a brain-like neural layer. The weights are then adjusted and readjusted, layer by layer, until the network can perform an intelligent function with the fewest possible errors.

But Hinton suggested that, to get to where neural networks are able to become intelligent on their own, what is known as "unsupervised learning," "I suspect that means getting rid of back-propagation."

"I don't think it's how the brain works," he said. "We clearly don't need all the labeled data."

https://www.axios.com/ai-pioneer-advocates-starting-over-2485537027.html

All useless

Geoff Hinton (the godfather

More seriously…

watch his capsule talk @ https://www.youtube.com/watch?v=TFIMqt0yT2I

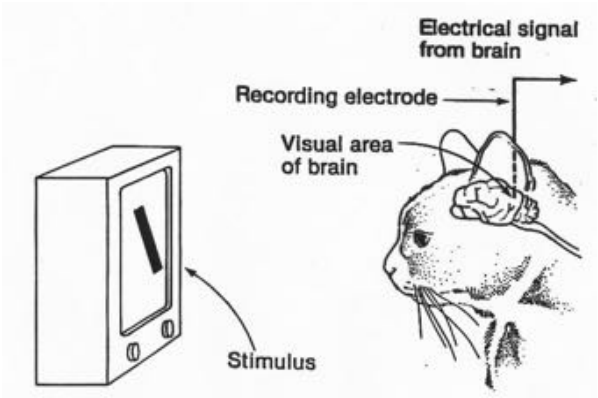
But Hinton suggested that, to get to where neural networks are able to become intelligent on their own, what is known as "unsupervised learning," "I suspect that means getting rid of back-propagation."

"I don't think it's how the brain works," he said. "We clearly don't need all the labeled data."

https://www.axios.com/ai-pioneer-advocates-starting-over-2485537027.html

History

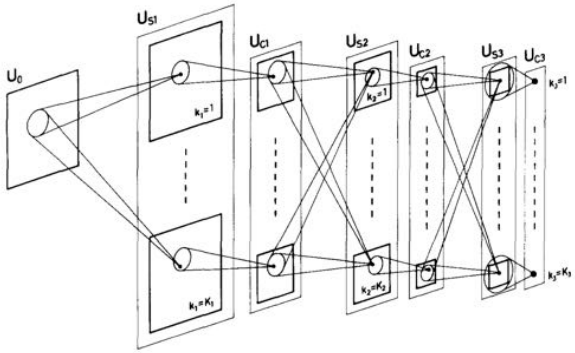
Hubel & Wiesel



Understanding how the system constructs complex representation from simple stimulus features.

History

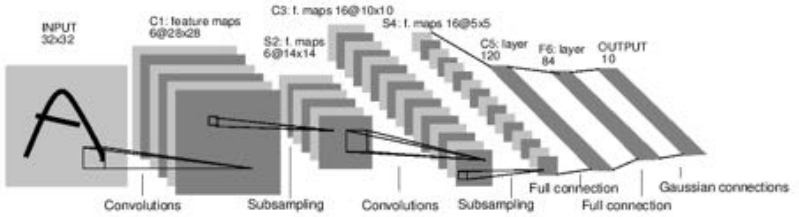
Neocognitron



Fukushima, Kunihiko. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. Competition and cooperation in neural nets 1982.

History

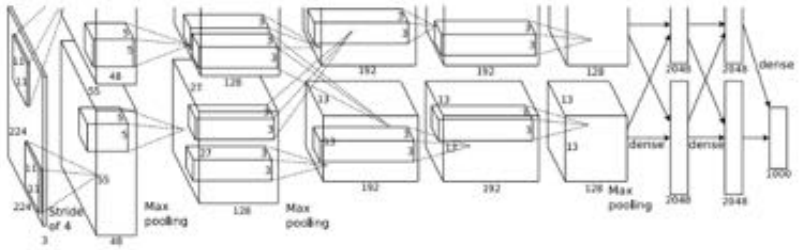
LeNet



Y. LeCun, et. al. Handwritten digit recognition with a back-propagation network. NIPS 1989.

History

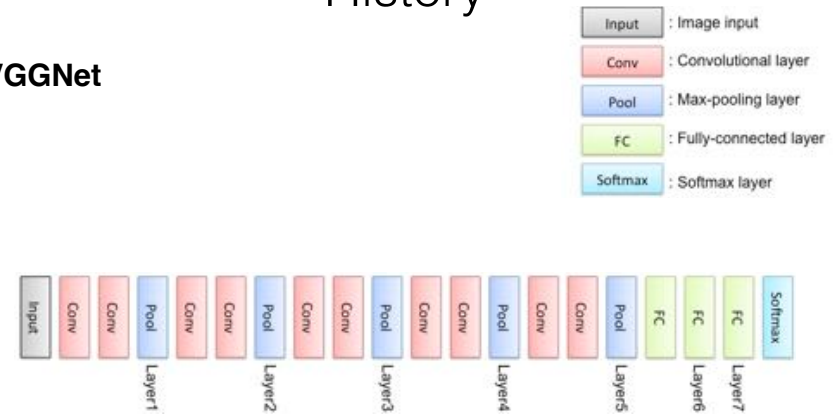
AlexNet



A Krizhevsky, et. al. ImageNet Classification with Deep Convolutional Neural Networks. NIPS 2012.

History

VGGNet



Convolutional NNs

Classification Retrieval

Figures copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Slide credit: CS 231N

Convolutional NNs

Detection Segmentation

Figures copyright Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, 2015. Reproduced with permission.

[Faster R-CNN: Ren, He, Girshick, Sun 2015]

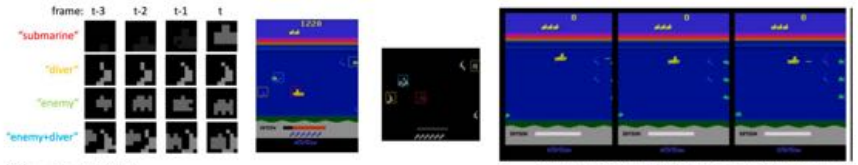
Figures copyright Clement Farabet, 2012. Reproduced with permission.

[Farabet et al., 2012]

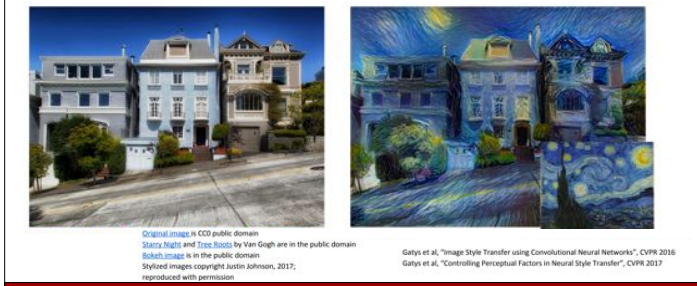
Slide credit: CS 231N

Convolutional NNs

Game AI



[Guo et al. 2014]

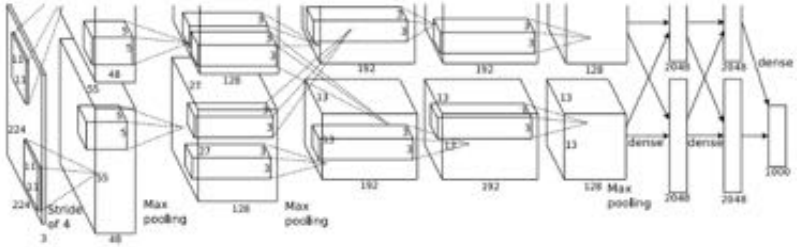


Gatys et al, "Image Style Transfer using Convolutional Neural Networks", CVPR 2016
Gatys et al, "Controlling Perceptual Factors in Neural Style Transfer", CVPR 2017

Arts

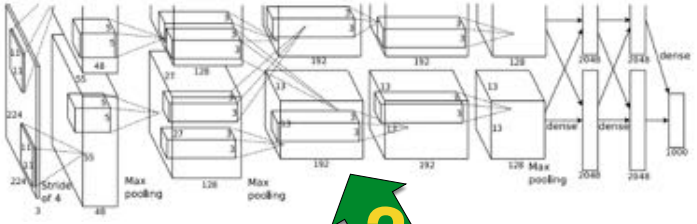
Slide credit: CS 231N

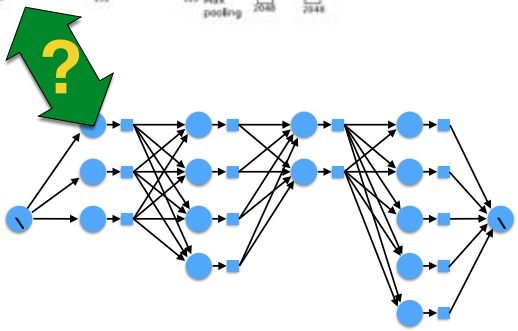
AlexNet in details



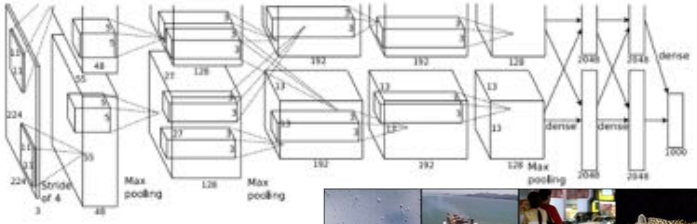
A Krizhevsky, et. al. ImageNet Classification with Deep Convolutional Neural Networks. NIPS 2012.

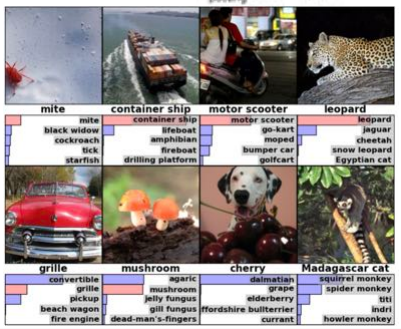
AlexNet in details



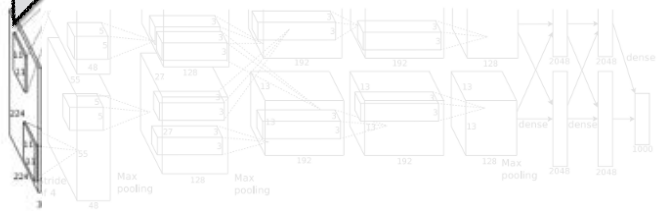


AlexNet in details





AlexNet in details



Image

PC: Daniel Yang

AlexNet in details

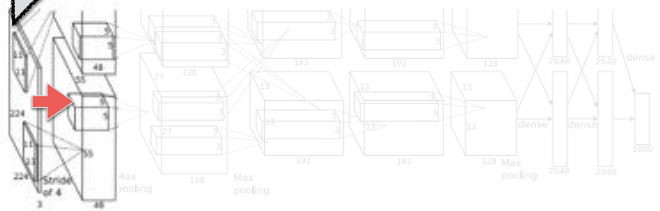


Image Conv

AlexNet in details

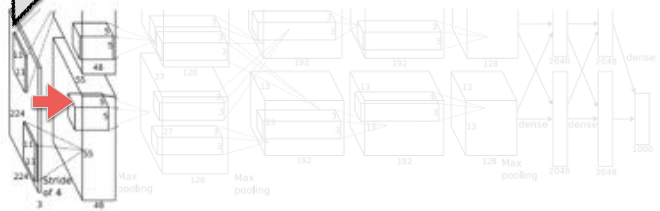


Image Conv Pool

AlexNet in details

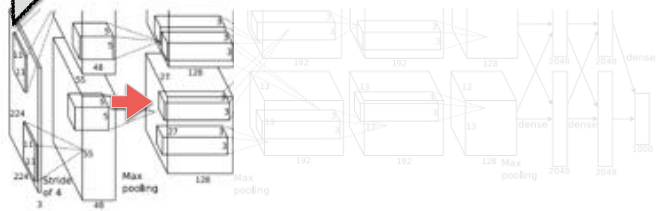


Image Conv Conv Pool

AlexNet in details

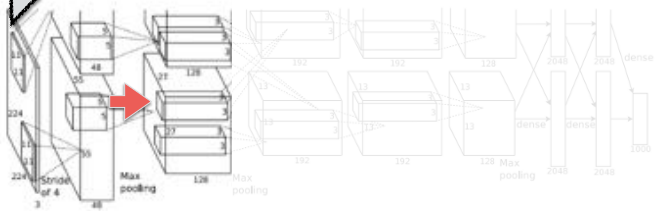


Image | Conv | Pool | Conv | Pool

AlexNet in details

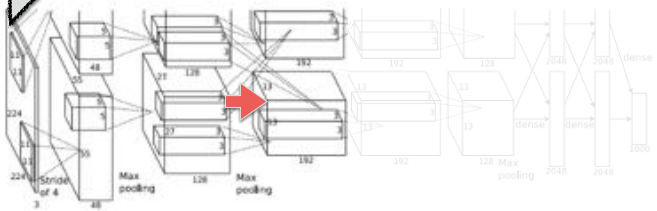


Image | Conv | Pool | Conv | Pool | Conv

AlexNet in details

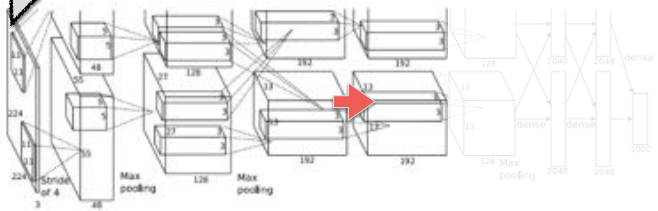


Image | Conv | Pool | Conv | Pool | Conv | Conv

AlexNet in details

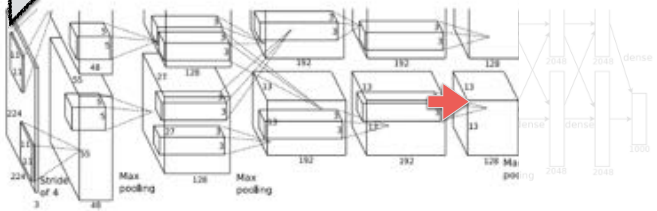


Image | Conv | Pool | Conv | Pool | Conv | Conv | Conv

AlexNet in details

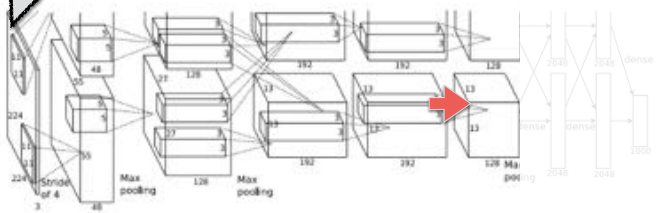


Image | Conv, Pool | Conv, Pool | Conv | Conv | Conv, Pool

AlexNet in details

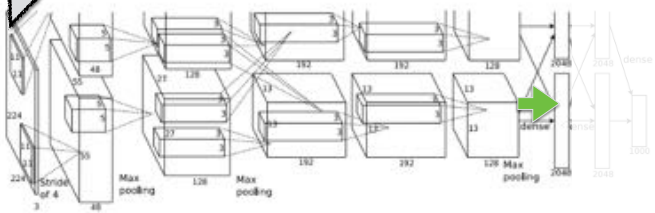


Image | Conv, Pool | Conv, Pool | Conv | Conv | Conv, Pool | FC

AlexNet in details

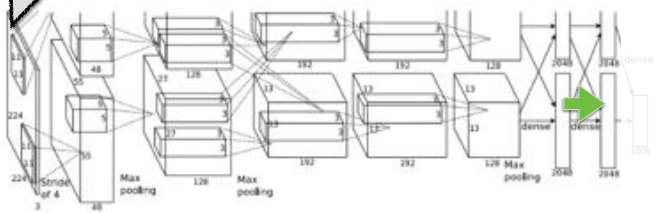
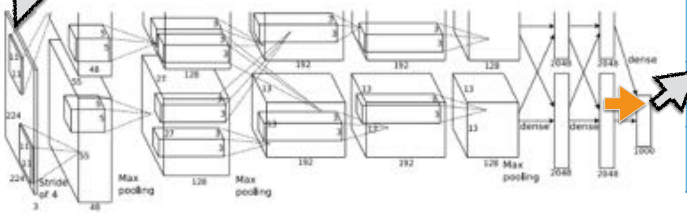


Image | Conv, Pool | Conv, Pool | Conv | Conv | Conv, Pool | FC | FC

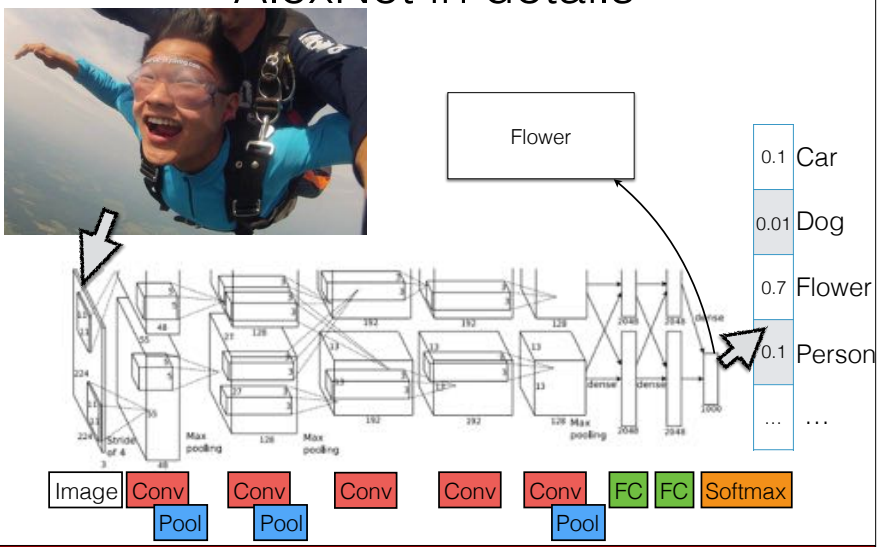
AlexNet in details



0.1 Car
0.01 Dog
0.7 Flower
0.1 Person
… …

Image | Conv, Pool | Conv, Pool | Conv | Conv | Conv, Pool | FC | FC | Softmax

AlexNet in details



What are these layers?

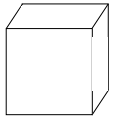
- Convolutional layer Conv
- Pool layer Pool
- Fully connected layer FC
- Softmax layer Softmax

What are these layers?

- Convolutional layer Conv
- Pool layer Pool
- Fully connected layer FC
- Softmax layer Softmax

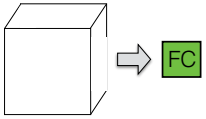
Layer = Function

Fully connected layer (FC)



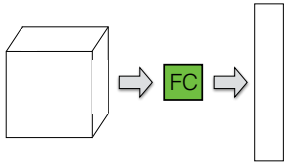
M x N x P

Fully connected layer (FC)



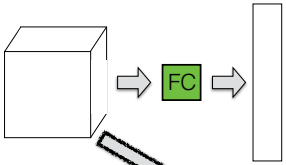
M x N x P

Fully connected layer (FC)



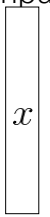
M x N x P Q x 1

Fully connected layer (FC)



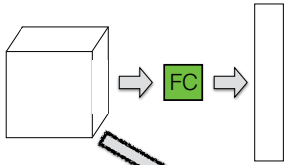
M x N x P Q x 1

FC operation takes an input, stretches it.



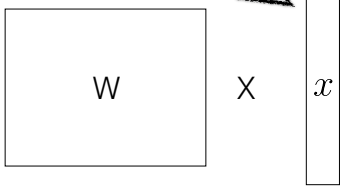
(M*N*P) x 1

Fully connected layer (FC)



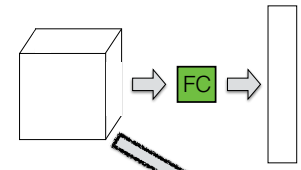
M x N x P Q x 1

FC operation takes an input, stretches it.



Q x (M*N*P) (M*N*P) x 1

Fully connected layer (FC)



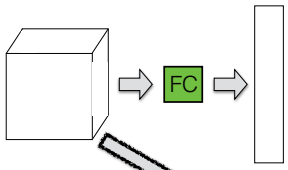
M x N x P Q x 1

FC operation takes an input, stretches it.

W X x = Wx

Q x (M*N*P) (M*N*P) x 1 Q x 1

Fully connected layer (FC)



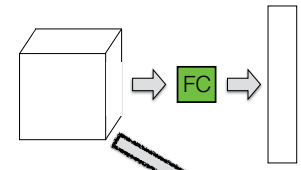
M x N x P Q x 1

FC operation takes an input, stretches it.

W Learned X x = Wx

Q x (M*N*P) (M*N*P) x 1 Q x 1

Fully connected layer (FC)



Why is this called FC?

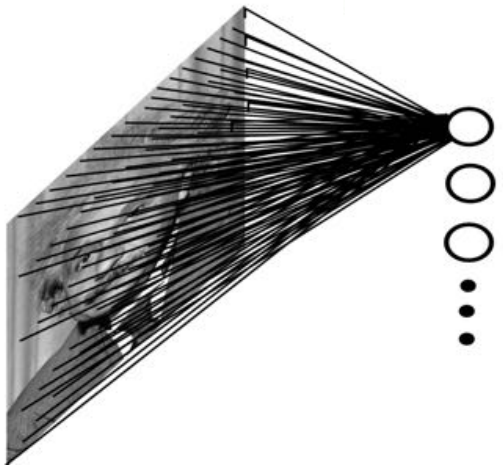
M x N x P Q x 1

FC operation takes an input, stretches it.

W Learned X x = Wx

Q x (M*N*P) (M*N*P) x 1 Q x 1

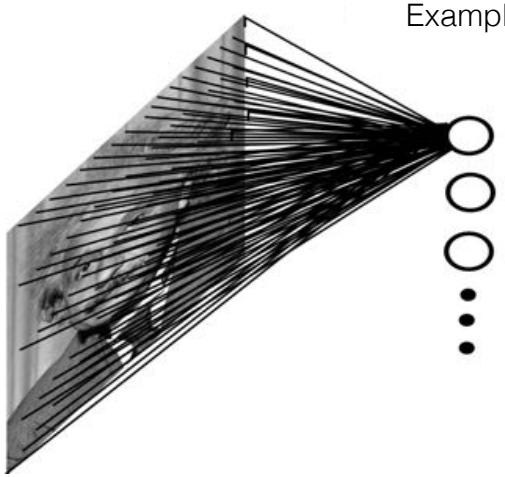
Fully connected layer (FC)



Slide credit: Marc'Aurelio Ranzato

Fully connected layer (FC)

Example: 200x200 image

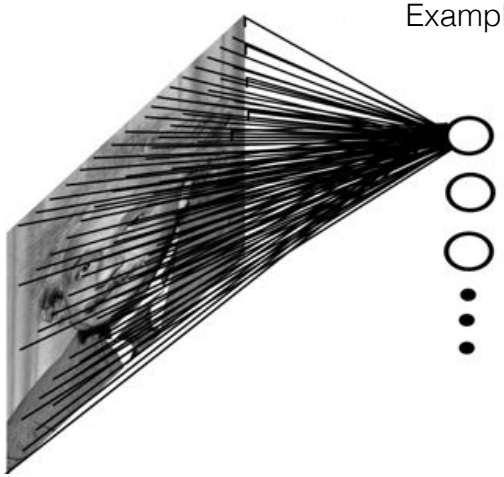


Slide credit: Marc'Aurelio Ranzato

Fully connected layer (FC)

Example: 200x200 image

40k hidden units



Slide credit: Marc'Aurelio Ranzato

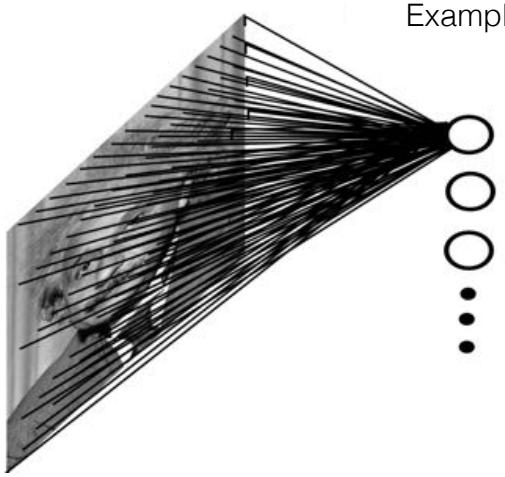
Fully connected layer (FC)

Example: 200x200 image

40k hidden units

~2B parameters!!!

Why?



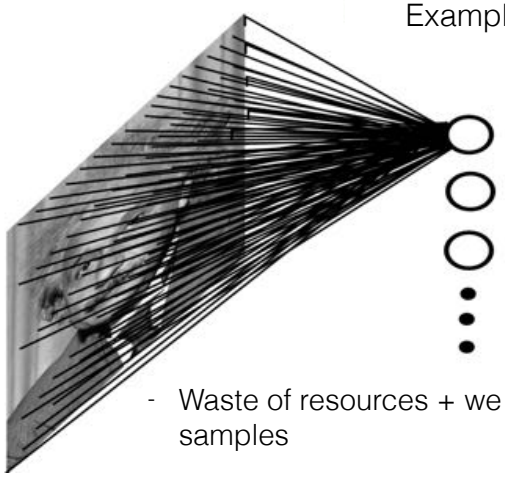
Slide credit: Marc'Aurelio Ranzato

Fully connected layer (FC)

Example: 200x200 image

40k hidden units

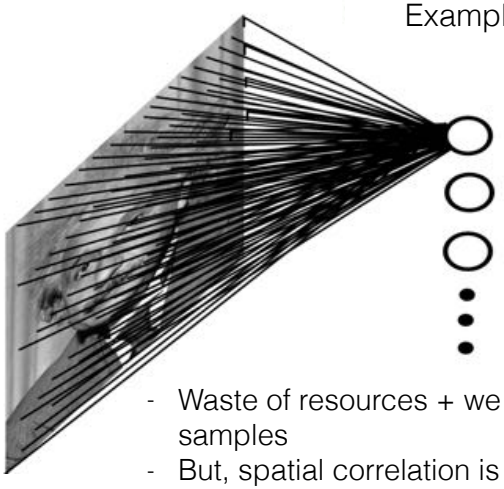
~2B parameters!!!



- Waste of resources + we have not enough training samples

Slide credit: Marc'Aurelio Ranzato

Fully connected layer (FC)



Example: 200x200 image

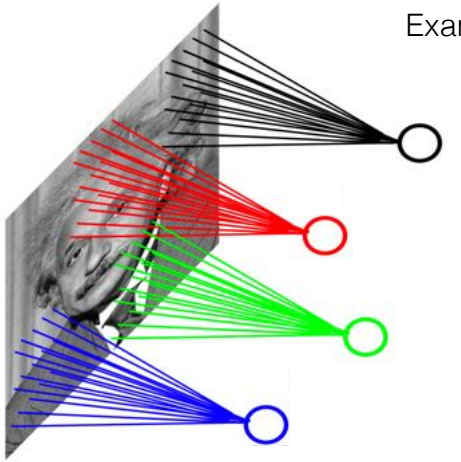
40k hidden units

~2B parameters!!!

- Waste of resources + we have not enough training samples
- But, spatial correlation is local!

Slide credit: Marc'Aurelio Ranzato

Locally connected layer



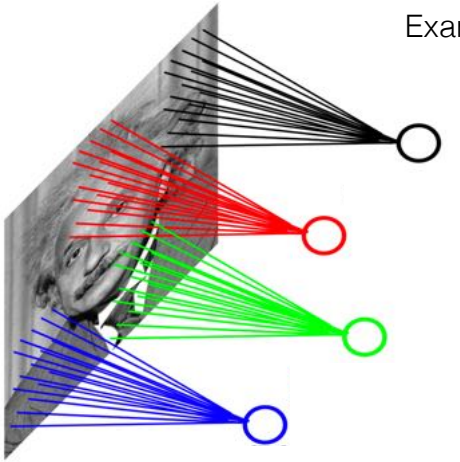
Example: 200x200 image

40k hidden units

Filter size: 10x10

Slide credit: Marc'Aurelio Ranzato

Locally connected layer



Example: 200x200 image

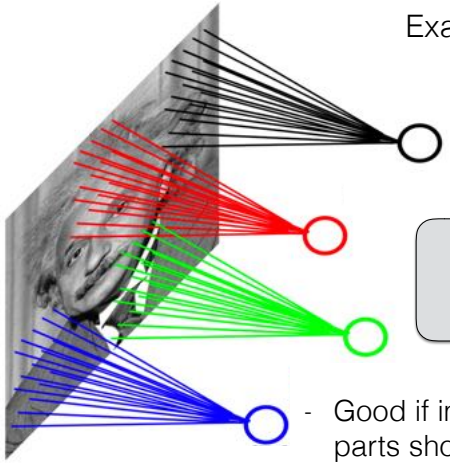
40k hidden units

Filter size: 10x10

4M parameters

Slide credit: Marc'Aurelio Ranzato

Locally connected layer



Example: 200x200 image

40k hidden units

Filter size: 10x10

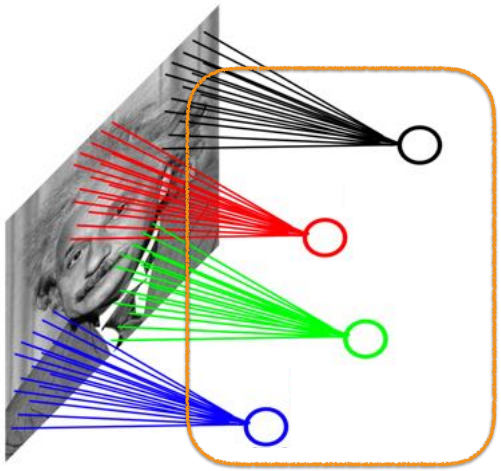
4M parameters

What can we do?

- Good if images are registered (i.e. parts show up at the same locations)

Slide credit: Marc'Aurelio Ranzato

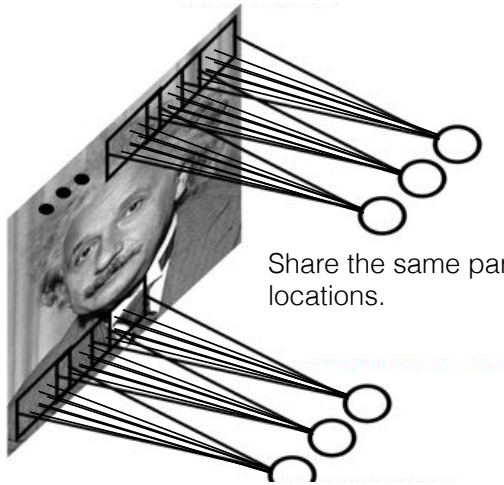
Locally connected layer



Share all local weights

Slide credit: Marc'Aurelio Ranzato

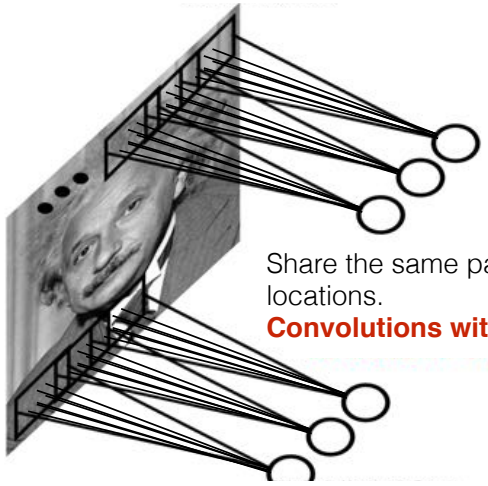
Convolutional layer



Share the same parameters across different locations.

Slide credit: Marc'Aurelio Ranzato

Convolutional layer

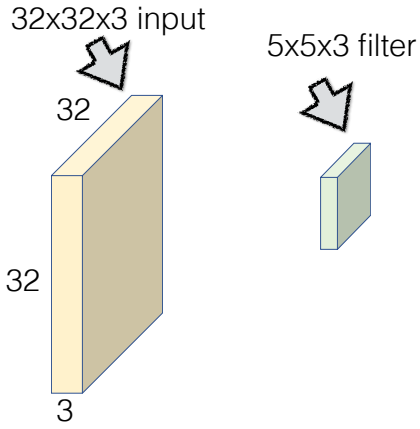


Share the same parameters across different locations.

Convolutions with learned kernels (weights)

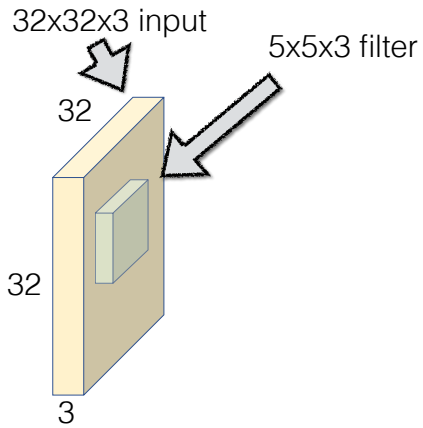
Slide credit: Marc'Aurelio Ranzato

Convolutional layer



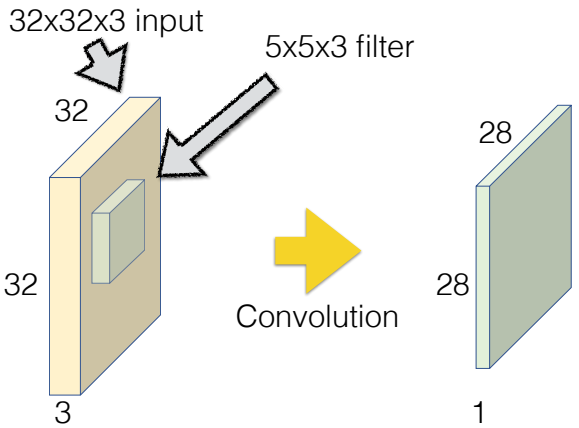
Slide credit: CS 231N

Convolutional layer



Slide credit: CS 231N

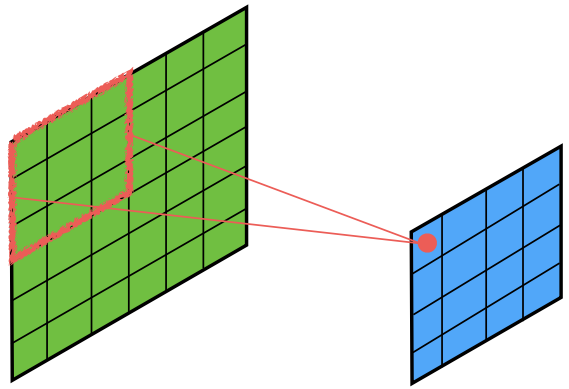
Convolutional layer



Slide credit: CS 231N

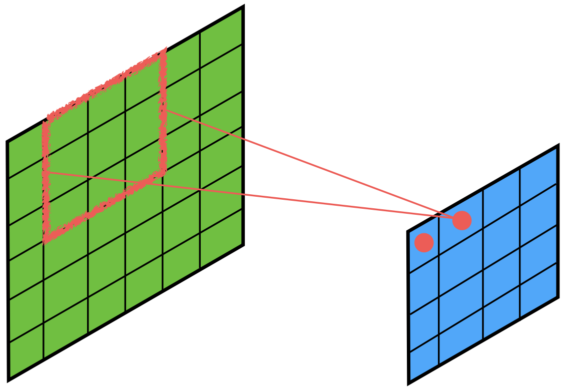
Convolution?

Convolutional layer



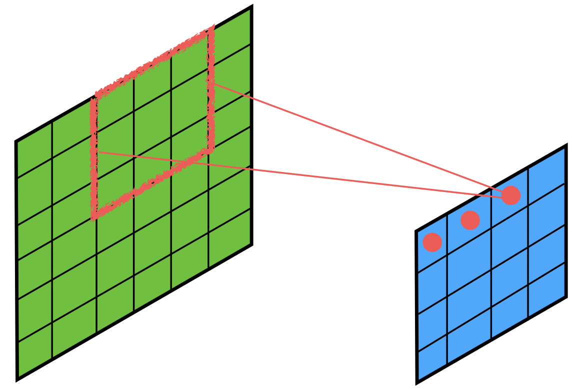
Inspired by Marc'Aurelio Ranzato's slides

Convolutional layer



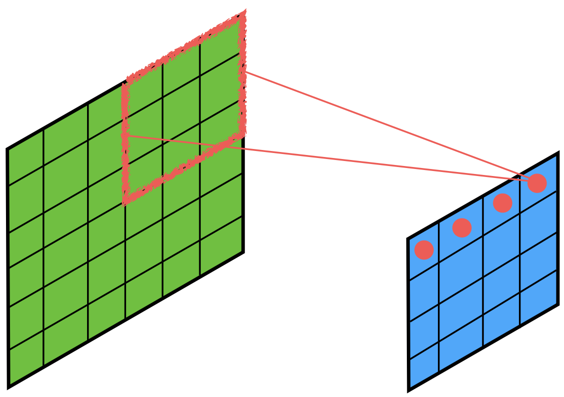
Inspired by Marc'Aurelio Ranzato's slides

Convolutional layer



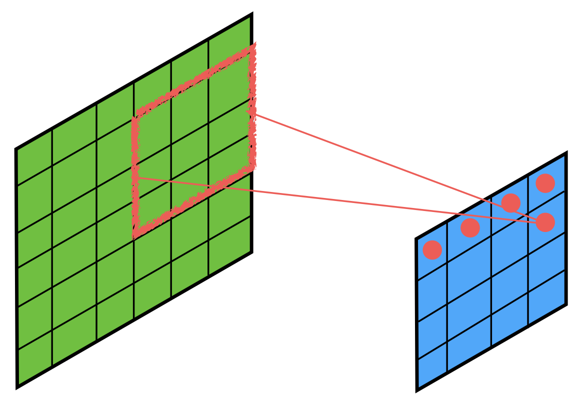
Inspired by Marc'Aurelio Ranzato's slides

Convolutional layer



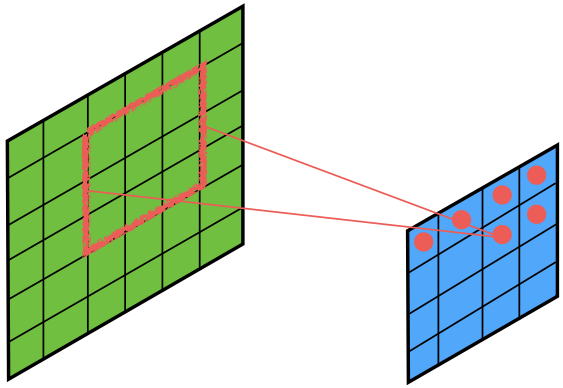
Inspired by Marc'Aurelio Ranzato's slides

Convolutional layer



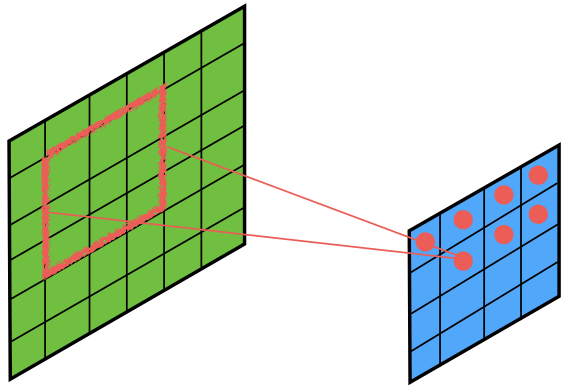
Inspired by Marc'Aurelio Ranzato's slides

Convolutional layer



Inspired by Marc'Aurelio Ranzato's slides

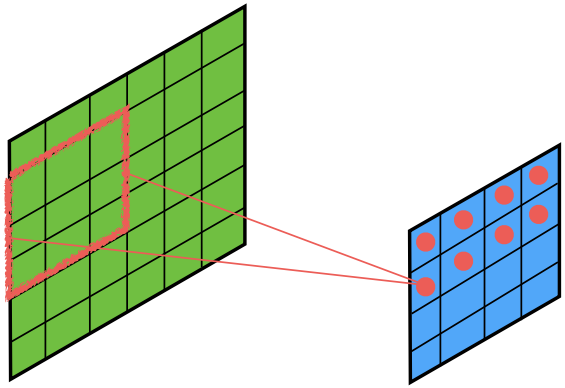
Convolutional layer



Inspired by Marc'Aurelio Ranzato's slides

Joseph J. Lim CSCI 599 @ USC Lecture 5

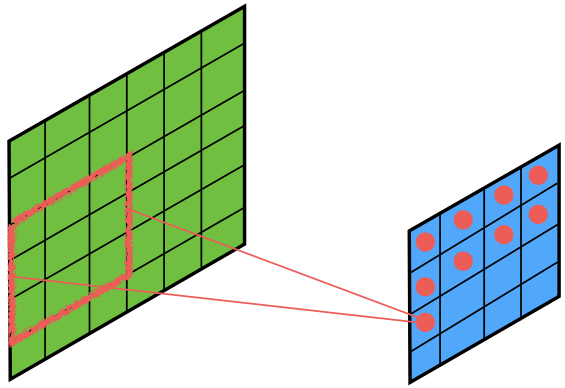
Convolutional layer



Inspired by Marc'Aurelio Ranzato's slides

Joseph J. Lim CSCI 599 @ USC Lecture 5

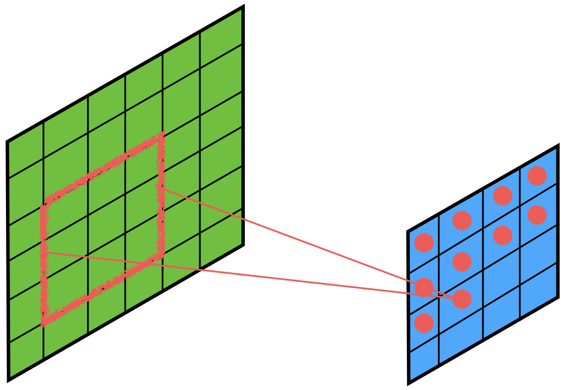
Convolutional layer



Inspired by Marc'Aurelio Ranzato's slides

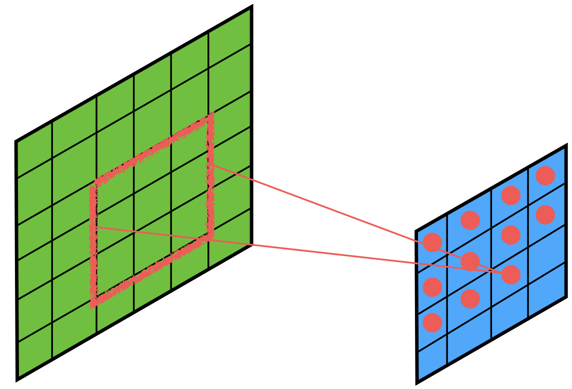
Joseph J. Lim CSCI 599 @ USC Lecture 5

Convolutional layer



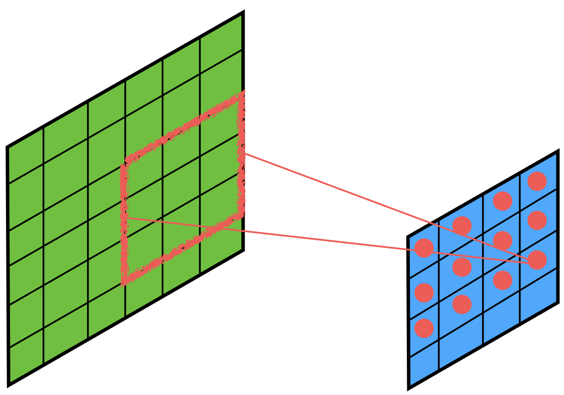
Inspired by Marc'Aurelio Ranzato's slides

Convolutional layer



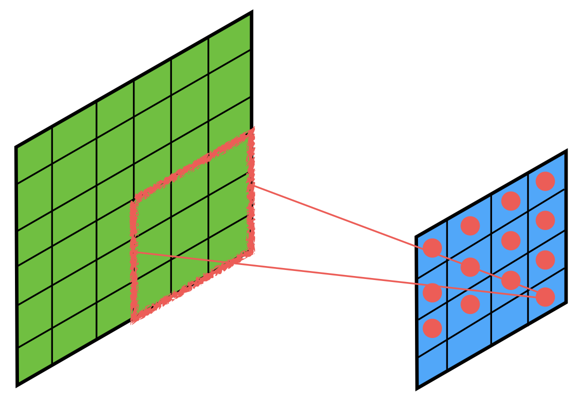
Inspired by Marc'Aurelio Ranzato's slides

Convolutional layer



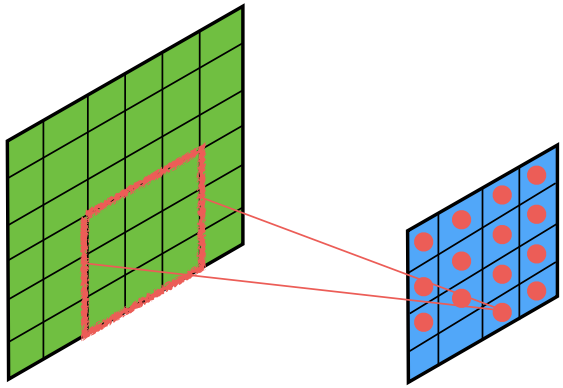
Inspired by Marc'Aurelio Ranzato's slides

Convolutional layer



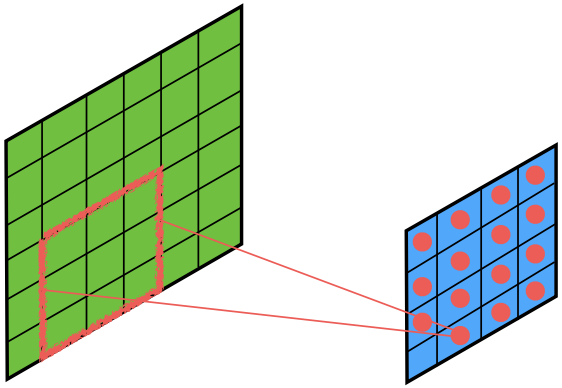
Inspired by Marc'Aurelio Ranzato's slides

Convolutional layer



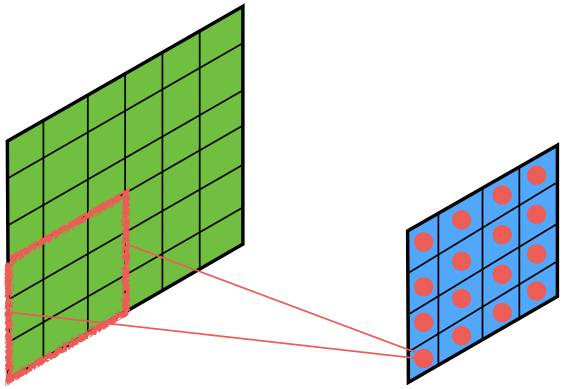
Inspired by Marc'Aurelio Ranzato's slides

Convolutional layer



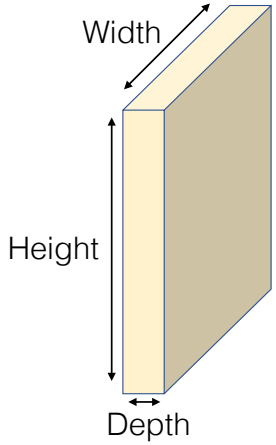
Inspired by Marc'Aurelio Ranzato's slides

Convolutional layer

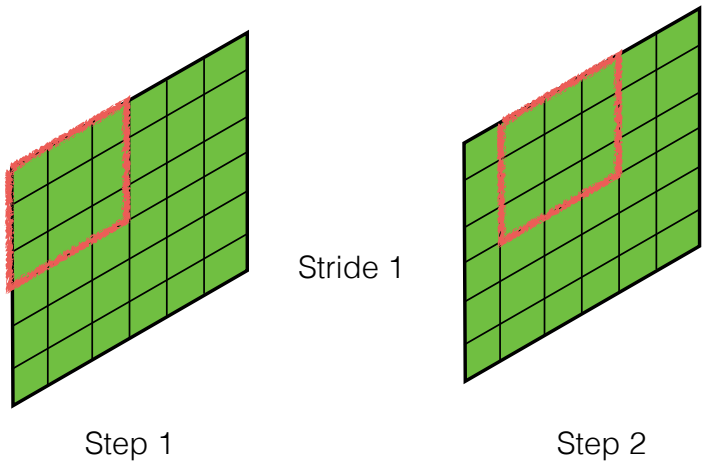


Inspired by Marc'Aurelio Ranzato's slides

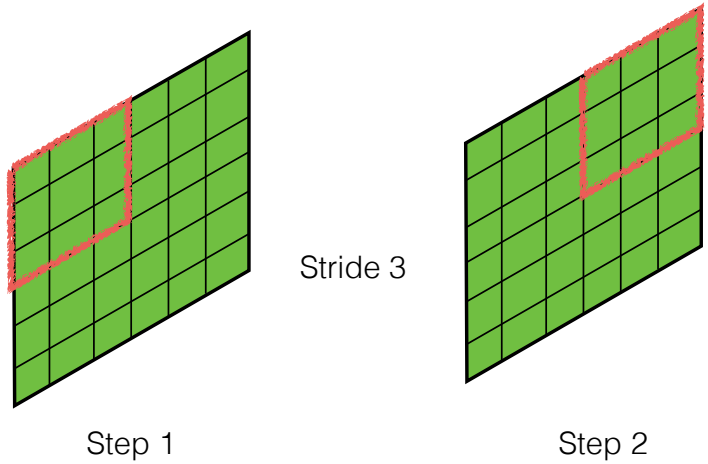
Terminologies



Terminologies



Terminologies



Terminologies

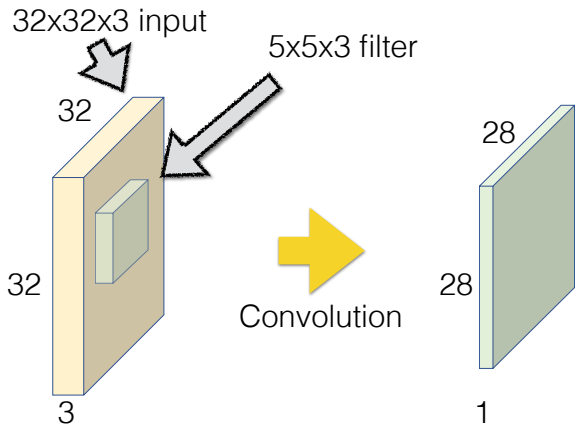
a	b	c	d	e
f	g	h	i	j
k	l	m	n	o
p	q	r	s	t
u	v	w	x	y

Terminologies

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	a	b	c	d	e	0	0
0	0	f	g	h	i	j	0	0
0	0	k	l	m	n	o	0	0
0	0	p	q	r	s	t	0	0
0	0	u	v	w	x	y	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

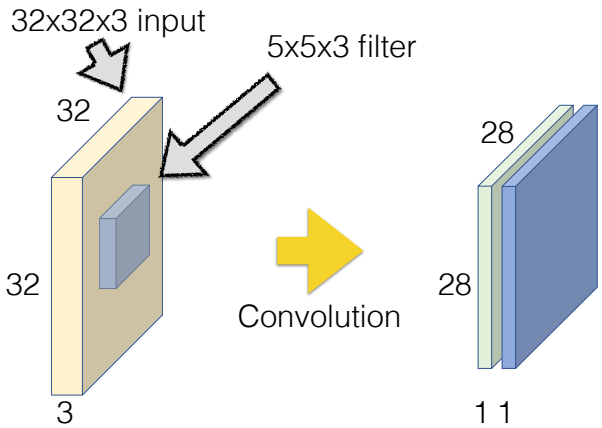
Zero-padding: 2

Convolutional layer



Slide credit: CS 231N

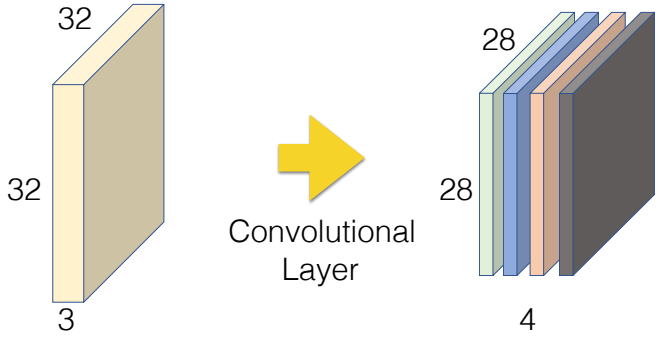
Convolutional layer



Slide credit: CS 231N

Convolutional layer

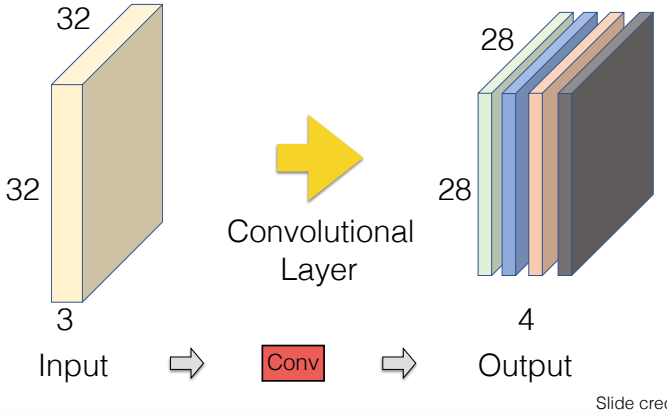
With 4 5x5x3 filters, we will get 4 activation outputs.
Hence, the final output is 28x28x4.



Slide credit: CS 231N

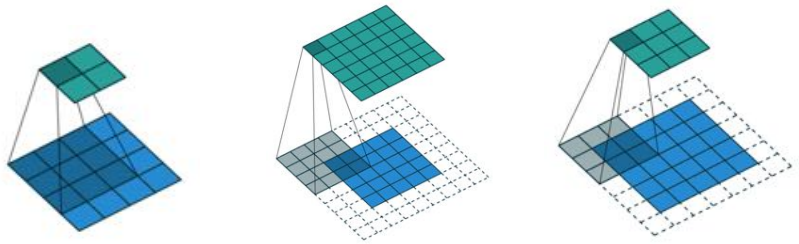
Convolutional layer

With 4 5x5x3 filters, we will get 4 activation outputs.
Hence, the final output is 28x28x4.



Slide credit: CS 231N

Convolutional layer



padding 0, stride 1 padding 2, stride 1 padding 1, stride 2

Note: it can have **depth** (3rd dimension).

Image credit: vdumoulin (github): https://github.com/vdumoulin/conv_arithmetic

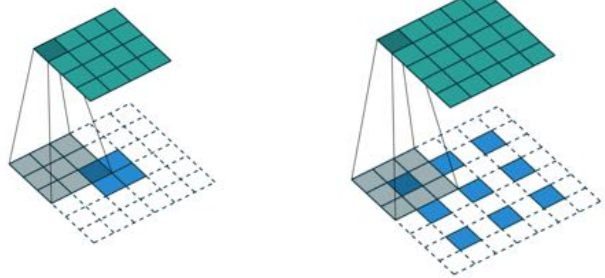
Deconvolutional layer

Deconvolutional layer

Warning: this is not about **de-**convolution!

Deconvolutional layer

Warning: this is not about **de-**convolution!



padding 0, stride 1 padding 1, stride 1

Why do we need this?

Image credit: vdumoulin (github): https://github.com/vdumoulin/conv_arithmetic

Fully connected vs Convolutional

- Fully connected layer
- Locally connected layer
- Convolutional layer

- Which layer is the best? Why?

What are these layers?

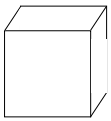
- Convolutional layer 

- Pool layer 

- Fully connected layer 

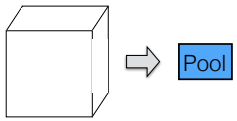
- Softmax layer 

Max pooling layer



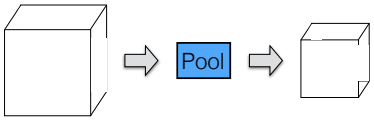
M x N x P

Max pooling layer



M x N x P

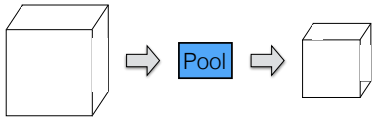
Max pooling layer



M x N x P

m x n x P

Max pooling layer



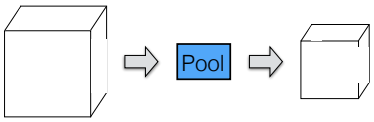
M x N x P

m x n x P

- Shrinks the feature smaller (compact + manageable)

Slide credit: CS 231N

Max pooling layer



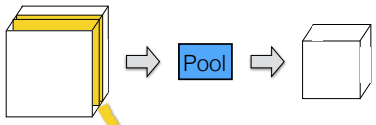
M x N x P

m x n x P

- Shrinks the feature smaller (compact + manageable)
- Each activation map (3rd dim) is handled separately

Slide credit: CS 231N

Max pooling layer

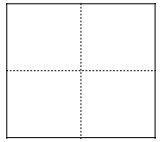


M x N x P

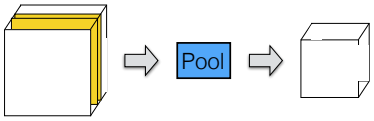
m x n x P

1	3	2	7	2
5	1	3	0	5
4	8	5	0	6
3	2	6	0	8
0	7	4	6	9

max pool with
3x3 filters & stride 2



Max pooling layer



M x N x P

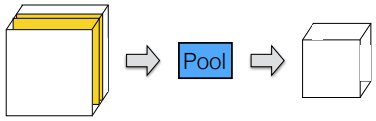
m x n x P

1	3	2	7	2
5	1	3	0	5
4	8	5	0	6
3	2	6	0	8
0	7	4	6	9

max pool with
3x3 filters & stride 2

8	

Max pooling layer



M x N x P

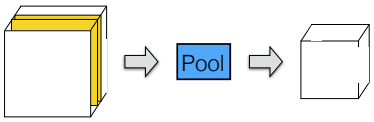
m x n x P

1	3	2	7	2
5	1	3	0	5
4	8	5	0	6
3	2	6	0	8
0	7	4	6	9

max pool with
3x3 filters & stride 2

8	7

Max pooling layer



M x N x P

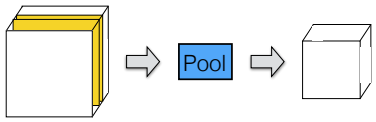
m x n x P

1	3	2	7	2
5	1	3	0	5
4	8	5	0	6
3	2	6	0	8
0	7	4	6	9

max pool with
3x3 filters & stride 2

8	7
8	

Max pooling layer



M x N x P

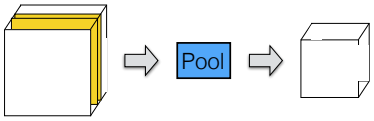
m x n x P

1	3	2	7	2
5	1	3	0	5
4	8	5	0	6
3	2	6	0	8
0	7	4	6	9

max pool with
3x3 filters & stride 2

8	7
8	9

Max pooling layer



M x N x P

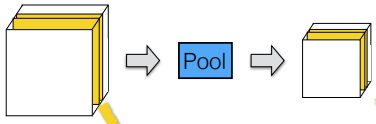
m x n x P

1	3	2	7	2
5	1	3	0	5
4	8	5	0	6
3	2	6	0	8
0	7	4	6	9

max pool with
3x3 filters & stride 2

8	7
8	9

Max pooling layer



M x N x P

m x n x P

1	3	2	7	2
5	1	3	0	5
4	8	5	0	6
3	2	6	0	8
0	7	4	6	9

max pool with
3x3 filters & stride 2

8	7
8	9

Pooling layers

- **Max pooling** (most common)
- Average pooling
- L2-norm pooling
- …

Why pooling layers?

Why pooling layers?

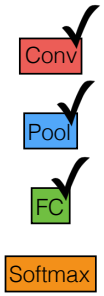
- Reduces # of parameters in the following layers

Why pooling layers?

- Reduces # of parameters in the following layers
  - Hence, less overfitting!

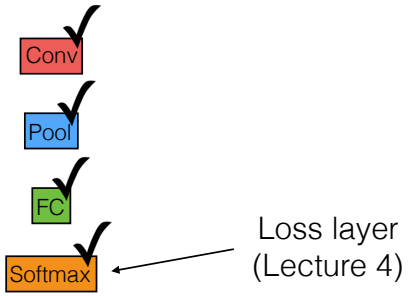
What are these layers?

- Convolutional layer
- Pool layer
- Fully connected layer
- Softmax layer



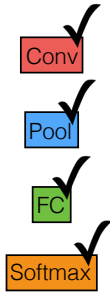
What are these layers?

- Convolutional layer
- Pool layer
- Fully connected layer
- Softmax layer



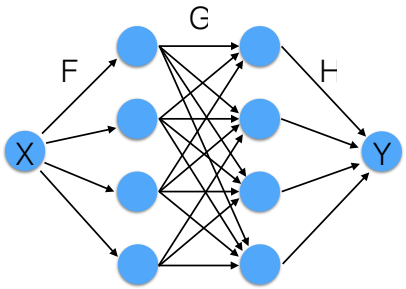
All these are linear!!!

- Convolutional layer
- Pool layer
- Fully connected layer
- Softmax layer

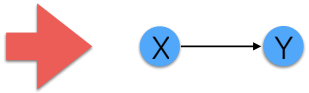


Loss layer
(Lecture 4)

Activation functions



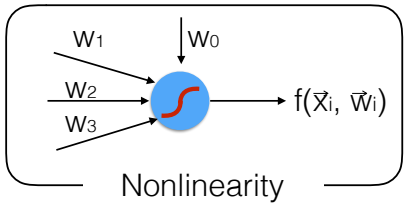
Y=H(G(F(X)))



Y=(HGF)(X)

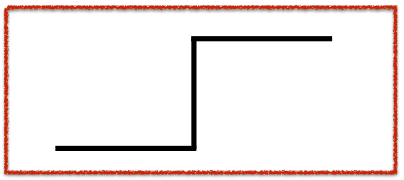
Without nonlinear activation,
multiple layers are reduced to one matrix multiplication.

Activation functions



Nonlinearity makes a neural network deeper

Step function

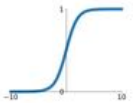


- Firing after a threshold
- Is not differentiable
- Cannot apply back-propagation

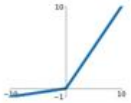
Anything else we can use?

Activation functions

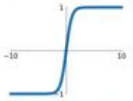
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



Leaky ReLU
$\max(0.1x, x)$

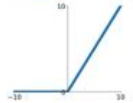


tanh
$\tanh(x)$



Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ReLU
$\max(0, x)$



ELU
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

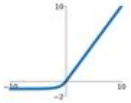
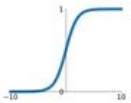


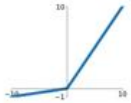
Figure from Stanford cs231n lecture slides

Activation functions

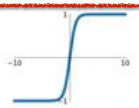
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



Leaky ReLU
$\max(0.1x, x)$

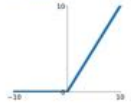


tanh
$\tanh(x)$



Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ReLU
$\max(0, x)$



ELU
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

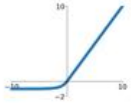
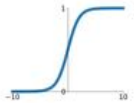


Figure from Stanford cs231n lecture slides

Sigmoid

Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$

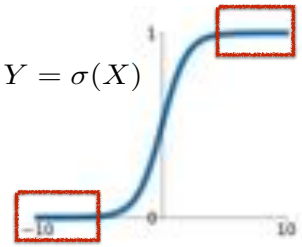


- Similar to step functions, but differentiable
- (-) easily saturated (no gradients)

Figure from Stanford cs231n lecture slides

Gradient vanishing problem

$Y = \sigma(X)$



Chain rule

$$\frac{\partial Y}{\partial X} = \frac{\partial Y}{\partial \sigma} \cdot \frac{\partial \sigma}{\partial X}$$

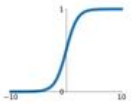
is almost 0 for most of X

Thus, $\frac{\partial Y}{\partial X}$ also goes to 0.

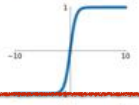
Figure from Stanford cs231n lecture slides

Activation functions

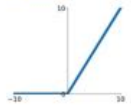
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



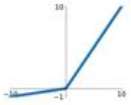
tanh
$\tanh(x)$



ReLU
$\max(0, x)$



Leaky ReLU
$\max(0.1x, x)$



Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU
$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$

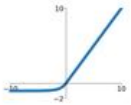
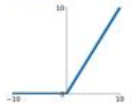


Figure from Stanford cs231n lecture slides

ReLU

ReLU
$\max(0, x)$

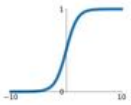


- Prevent gradient vanishing when x > 0
- Computationally efficient
- Biologically plausible
- (-) Still loose gradient when x < 0

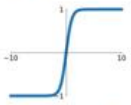
Figure from Stanford cs231n lecture slides

Activation functions

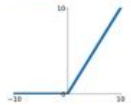
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



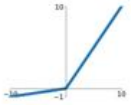
tanh
$\tanh(x)$



ReLU
$\max(0, x)$



Leaky ReLU
$\max(0.1x, x)$



Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU
$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$

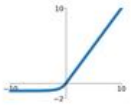
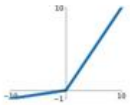


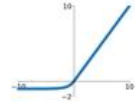
Figure from Stanford cs231n lecture slides

Leaky ReLU/ELU

Leaky ReLU
$\max(\alpha x, x)$



ELU
$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$



- No gradient vanishing problem
- Can be used instead of ReLU (e.g. Leaky ReLU for deconvolution)

Figure from Stanford cs231n lecture slides

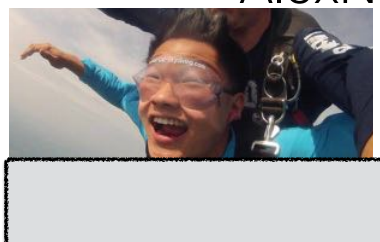
AlexNet in details

Flower

0.1 Car
0.01 Dog
0.7 Flower
0.1 Person
… …

Image Conv Pool ReLU Conv Pool ReLU Conv ReLU Conv ReLU Conv Pool ReLU FC ReLU FC ReLU Softmax

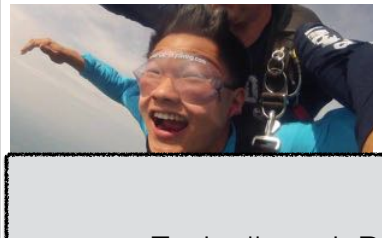
AlexNet in details



Flower

Typically omit ReLU for the visualization

AlexNet in details



Flower

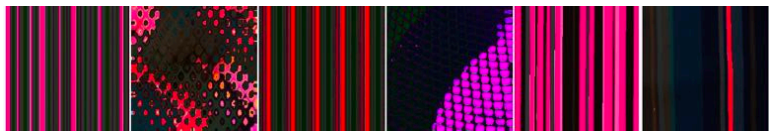
Typically omit ReLU for the visualization

Image Conv Pool Conv Pool Conv Conv Conv Pool FC FC Softmax

What did AlexNet learn?

Imag Conv Conv Conv Conv Conv FC FC Softmax

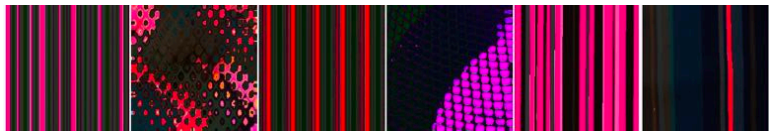
red



Bau, Zhou, and et. al. Network Dissection: Quantifying Interpretability of Deep Visual Representations. CVPR 2017.

What did AlexNet learn?

red

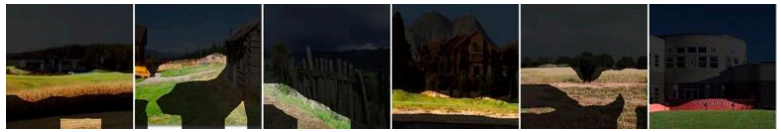


blue

Bau, Zhou, and et. al. Network Dissection: Quantifying Interpretability of Deep Visual Representations. CVPR 2017.

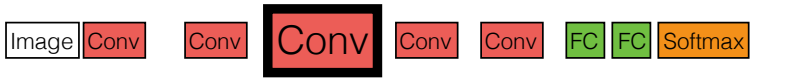
What did AlexNet learn?

grass

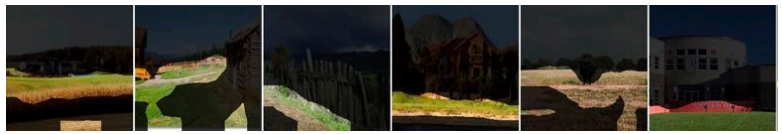


Bau, Zhou, and et. al. Network Dissection: Quantifying Interpretability of Deep Visual Representations. CVPR 2017.

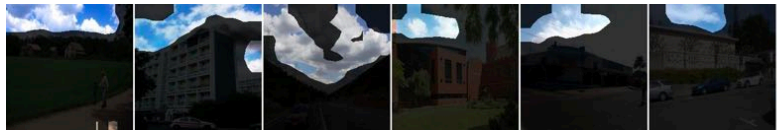
What did AlexNet learn?



grass

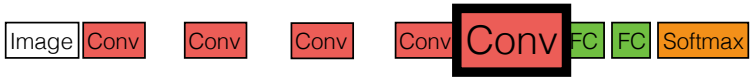


sky



Bau, Zhou, and et. al. Network Dissection: Quantifying Interpretability of Deep Visual Representations. CVPR 2017.

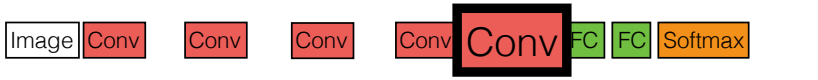
What did AlexNet learn?



dog

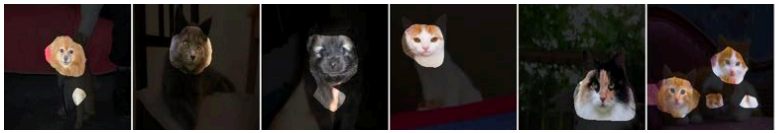
Bau, Zhou, and et. al. Network Dissection: Quantifying Interpretability of Deep Visual Representations. CVPR 2017.

What did AlexNet learn?



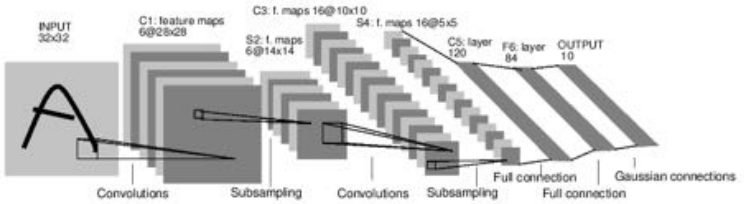
dog

cat



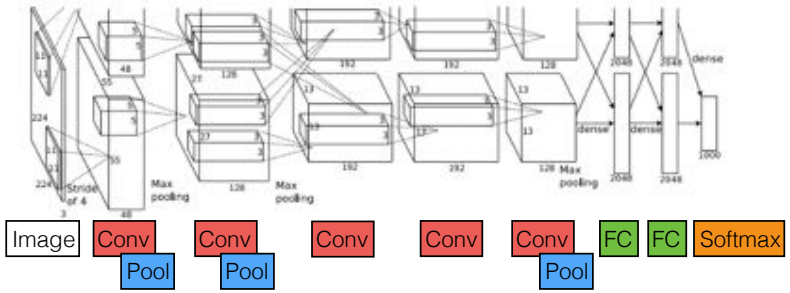
Bau, Zhou, and et. al. Network Dissection: Quantifying Interpretability of Deep Visual Representations. CVPR 2017.

CNNs: LeNet



LeCun et. al. Handwritten digit recognition: Applications of neural net chips and automatic learning. IEEE Communication 1989.

CNNs: AlexNet



A Krizhevsky, et. al. ImageNet Classification with Deep Convolutional Neural Networks. NIPS 2012.

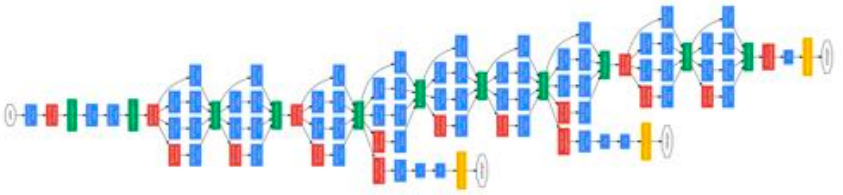
What made AlexNet AlexNet?

- GPU?
- Data?
- ReLU?
- Convolutional?
- Hyperparameter?

CNNs: VGGNet

Simonyan and Zisserman. Very Deep Convolutional Networks for Large-scale Image Recognition. ICLR 2015.

CNNs: GoogLeNet



Szegedy, et. al. Going Deeper with Convolution. CVPR 2015.

CNNs: ResNet

He, et. al. Deep **Residual** Learning for Image Recognition. CVPR 2016.

Encoder

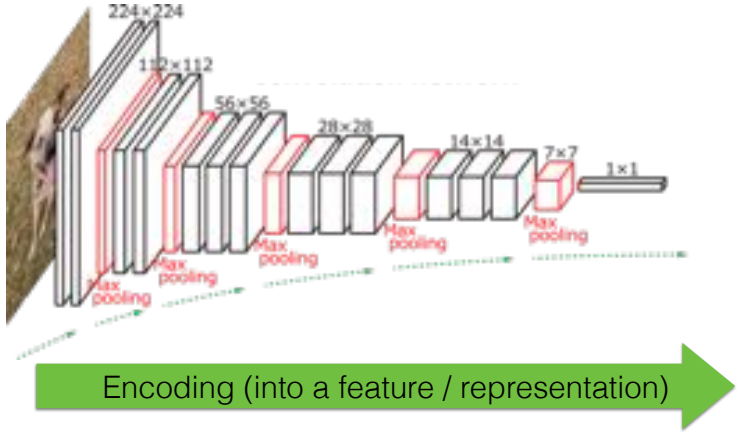


Image credit: Noh, et. al. Learning Deconvolution Network for Semantic Segmentation. ICCV 2015.

Decoder

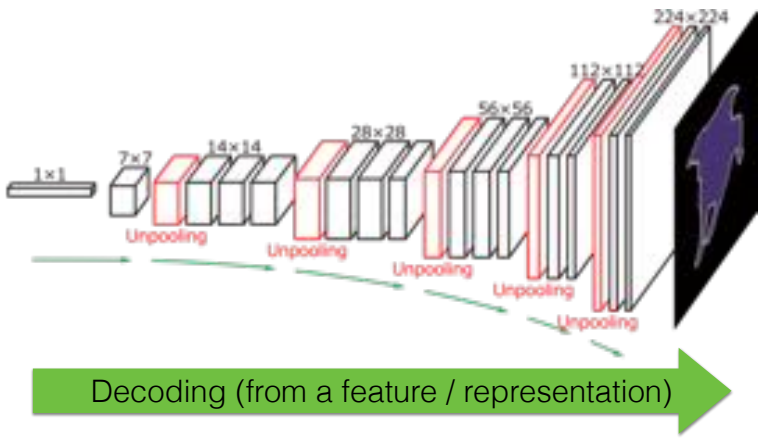


Image credit: Noh, et. al. Learning Deconvolution Network for Semantic Segmentation. ICCV 2015.

Congratulations!

All basic vocabs are covered.

Let's revisit CV tasks!

CNN applications

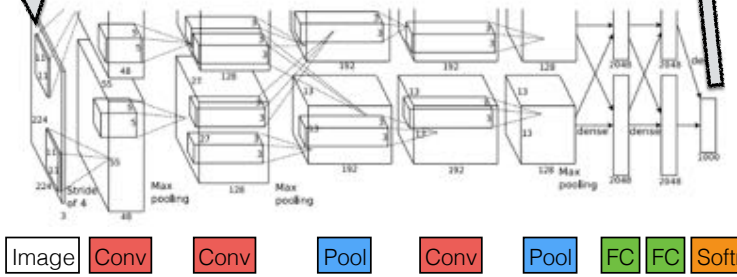
- Image classification
- Object detection
- Object segmentation
- Image resolution
- Human pose detection
- Action classification
- Image captioning

Image Classification

Car

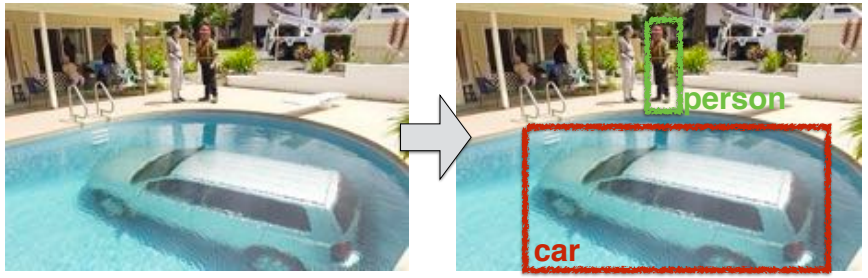
Image Classification

Car



Object Detection

Object Detection



Object Detection

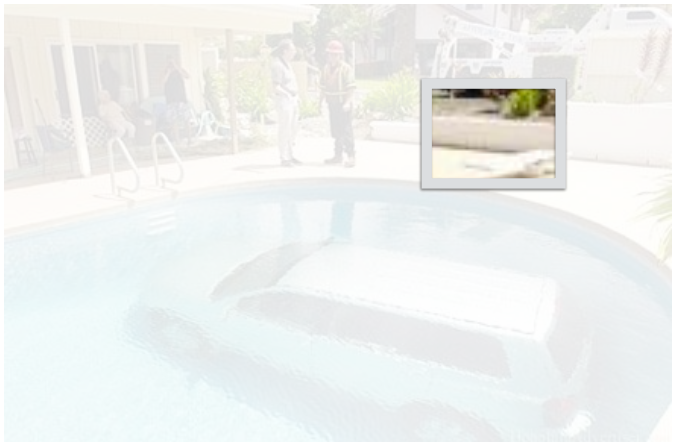
Object Detection

1. Object proposal

Object Detection

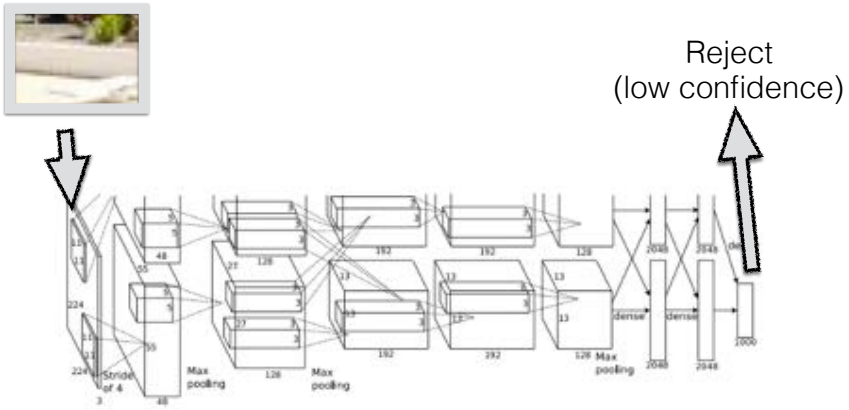
2. Evaluate each proposal

Object Detection



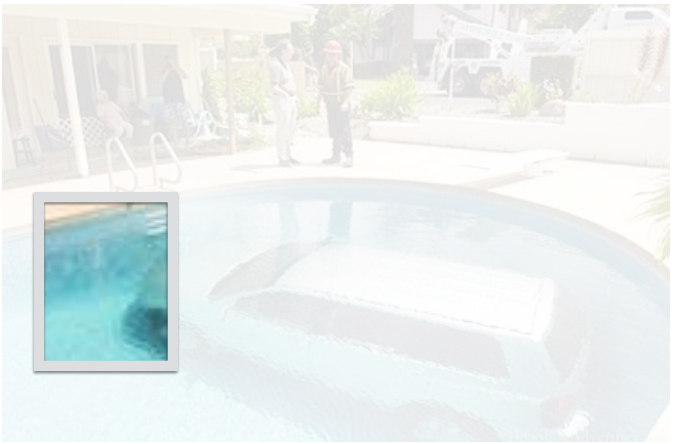
2. Evaluate each proposal

Object Detection



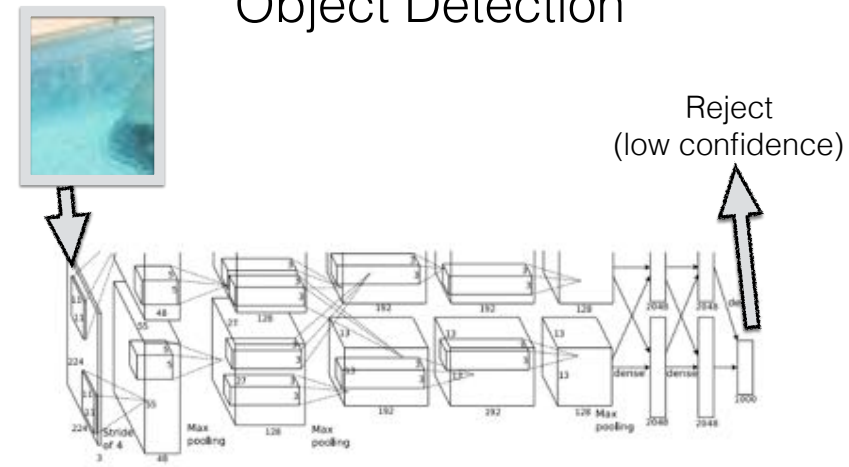
2. Evaluate each proposal

Object Detection



2. Evaluate each proposal

Object Detection

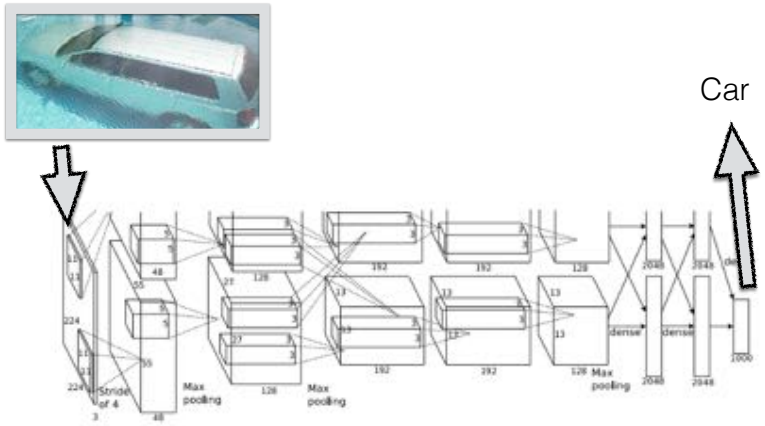


2. Evaluate each proposal

Object Detection

2. Evaluate each proposal

Object Detection



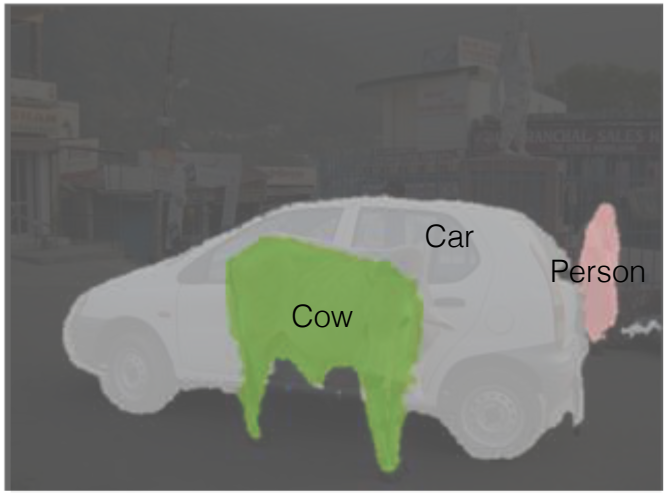
2. Evaluate each proposal

Object Detection

For more:
R-CNN (https://github.com/rbgirshick/rcnn) and Fast R-CNN (https://github.com/rbgirshick/fast-rcnn)

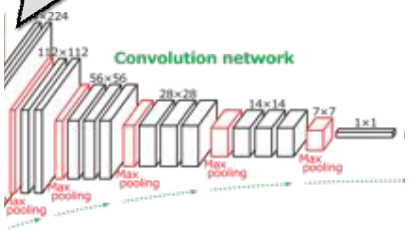
Semantic Segmentation

Semantic Segmentation

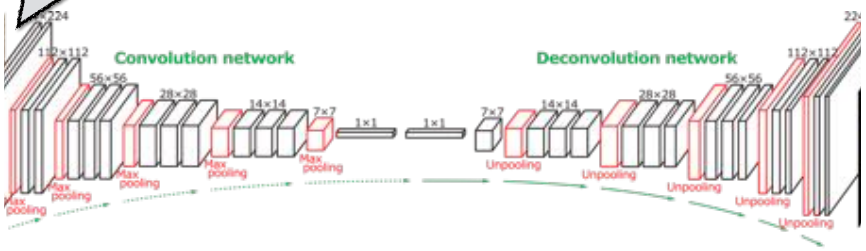


Semantic Segmentation

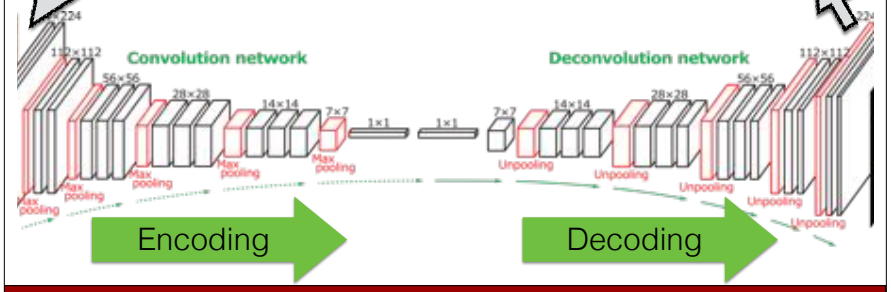
Semantic Segmentation



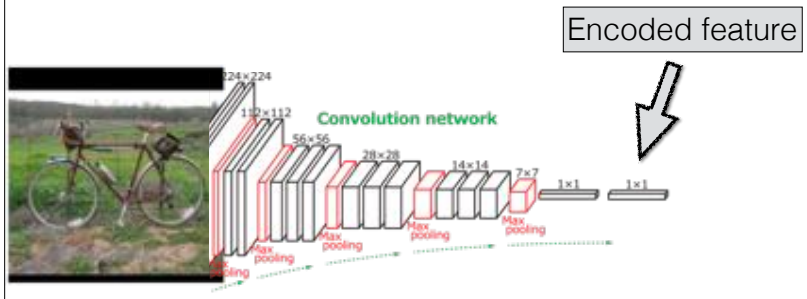
Semantic Segmentation



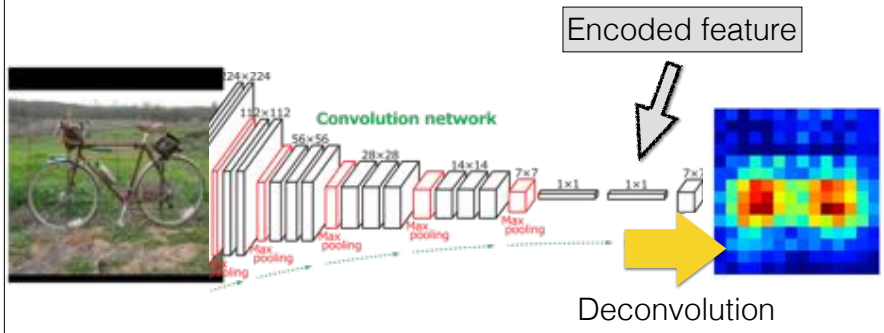
Semantic Segmentation



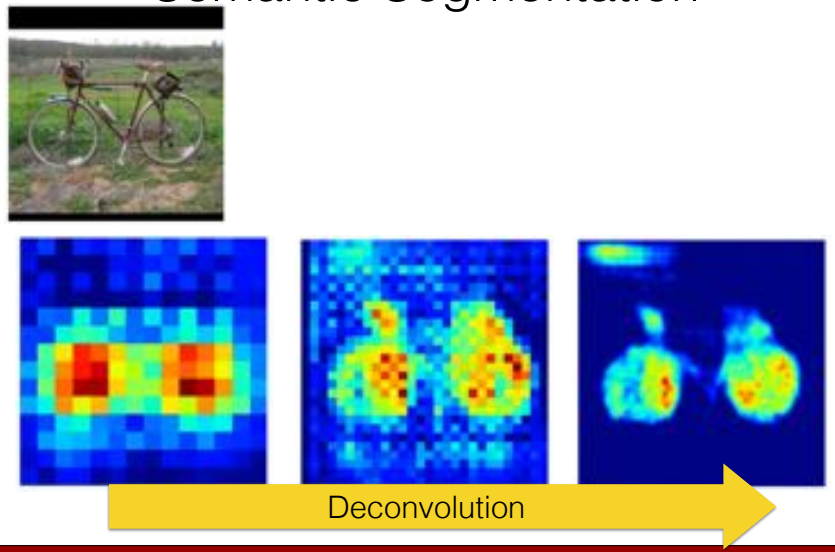
Semantic Segmentation



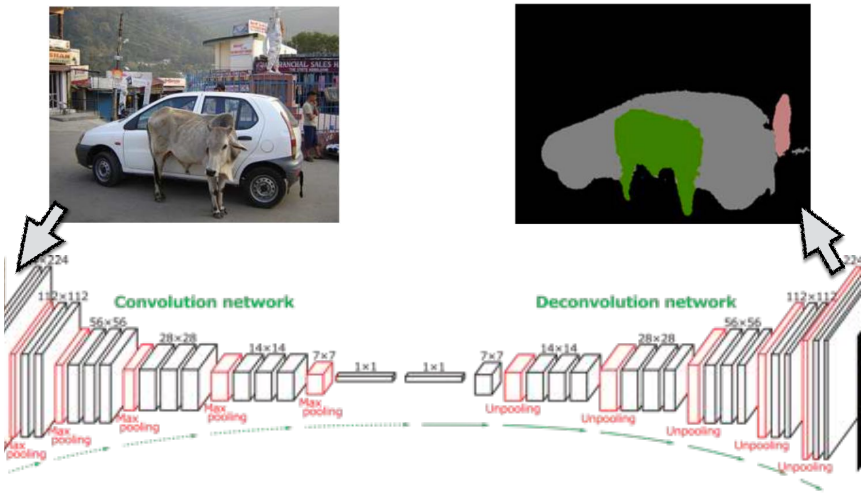
Semantic Segmentation



Semantic Segmentation

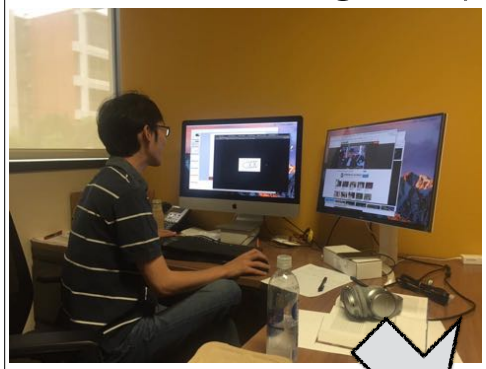


Semantic Segmentation



For more: Noh, et.al. Learning Deconvolution Network for Semantic Segmentation. ICCV 2015.

Image captioning



Hao Li (USC prof.) said

Josepo ah! Let me tell you!
Hard working Deep Nearning
Propessor at USC university!

Image captioning

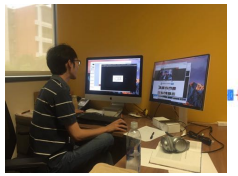
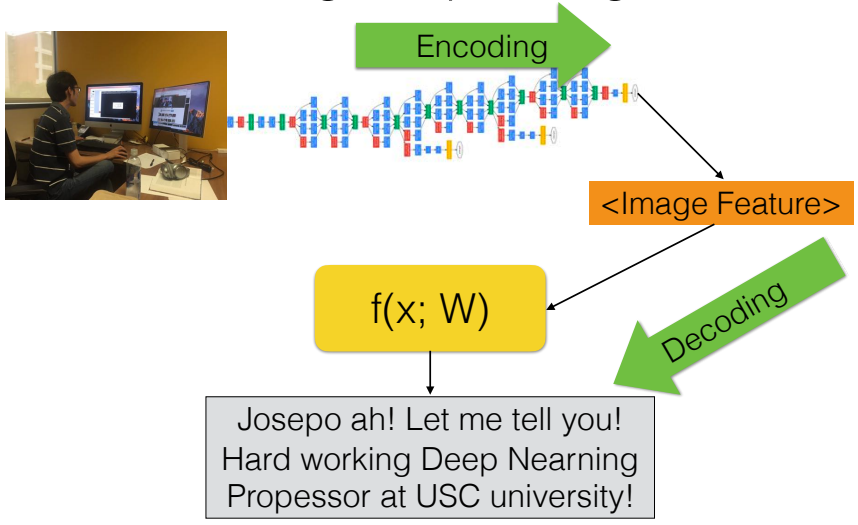


Image captioning



Demo

- 3D Face Reconstruction from a Single Image

  http://www.cs.nott.ac.uk/~psxasj/3dme/

Demo

- 3D Face Reconstruction from a Single Image

  http://www.cs.nott.ac.uk/~psxasj/3dme/

Demo

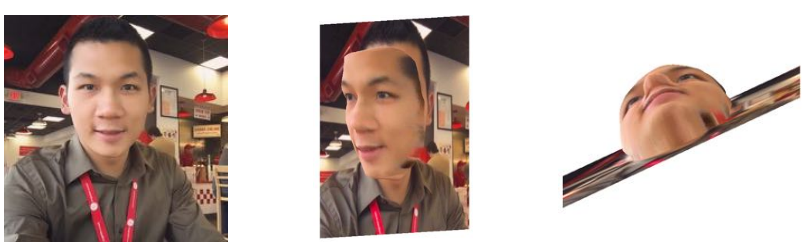
- 3D Face Reconstruction from a Single Image

  http://www.cs.nott.ac.uk/~psxasj/3dme/

Demo

- 3D Face Reconstruction from a Single Image

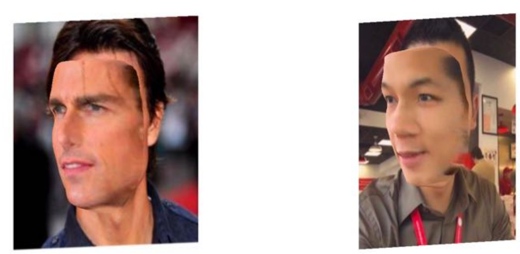
  http://www.cs.nott.ac.uk/~psxasj/3dme/



Demo

- 3D Face Reconstruction from a Single Image

  http://www.cs.nott.ac.uk/~psxasj/3dme/



Summary

- Fully connected layer
- Convolutional layer
- Deconvolutional layer
- Pooling layers
- Activation layers (e.g. ReLU)

Break Time

See you in 20 mins!

Today's agenda

- Part 1
  - Convolutional Neural Networks
- Part 2
  - Training Neural Networks

All inputs are just numbers

Image

a	b	c	d	e
f	g	h	i	j
k	l	m	n	o
p	q	r	s	t
u	v	w	x	y

Sound

a	b	c	d	e

Data preprocessing

- Zero-center data
- Normalize data
- PCA Whitening

Data preprocessing

Why do we need preprocessing?

- Zero-center data
- Normalize data
- PCA Whitening

Data preprocessing

Why do we need preprocessing?

- Zero-center data
- Normalize data
- PCA Whitening

Activation functions (ReLU) works around **zero**.

If input data is biased toward positive or negative, randomly initialized layers produce biased output.

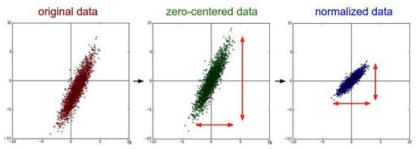


Figure from Stanford cs231n lecture slides

Data preprocessing

Why do we need preprocessing?

- Zero-center data
- Normalize data
- PCA Whitening

A feature with small scale has a negligible effect on backprop.

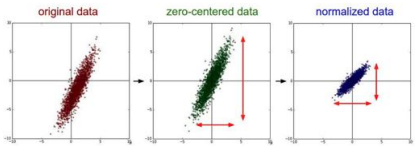


Figure from Stanford cs231n lecture slides

Data preprocessing

Why do we need preprocessing?

- Zero-center data
- Normalize data
- PCA Whitening

Remove correlation between input features

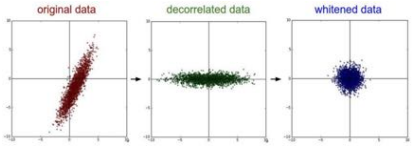
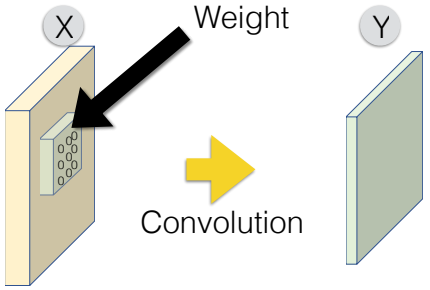


Figure from Stanford cs231n lecture slides

Weight initialization

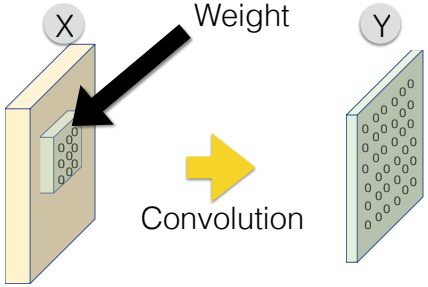
If all weights are 0?



Slide credit: CS 231N

Weight initialization

If all weights are 0?



Gradient vanishing problem

Output **Y** will be always 0 and then gradients will go to 0.

Slide credit: CS 231N

Weight initialization

- Small random numbers
- Xavier
- Xavier / 2 (He et. al.)

Slide credit: CS 231N

Small Random Numbers

- Sample from a gaussian distribution with 0 mean and given standard deviation (such as 0.01)

$$W \sim \mathcal{N}(\mu, \sigma^2)$$

- Doesn't work for deeper networks

Small Random Numbers

If the weights are too small,

the activation signals shrink quickly

If the weights are too large,

the activation signals explode

Xavier Initialization

- Set the variance according to # of input neurons

$$W \sim \mathcal{N}(\mu, \sigma^2) \longleftarrow Var(W) = \frac{1}{n_{in}}$$

- Consider # output neurons for back-prop

$$Var(W) = \frac{2}{n_{in} + n_{out}}$$

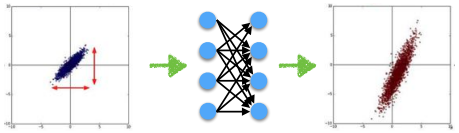
He et. al. Initialization

$$\frac{Random\ Numbers\ of\ dimension\ (D_{in}, D_{out})}{\sqrt{D_{in}\ /\ 2}}$$

- Default choice of training a deep network

Normalization

- After each layer, the distribution of activation signals changes (Internal Covariate Shift)

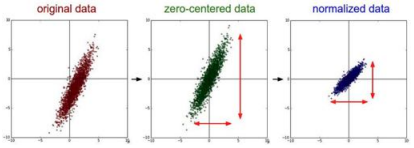


- As a network becomes deeper, distribution shifts more.
- Recall why we need data preprocessing

Figure from Stanford cs231n lecture slides

Normalization

- Normalize the distribution of activations to have zero mean and unit variance



Same as data preprocessing

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

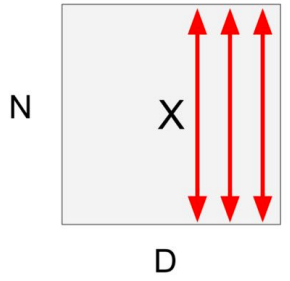
Figure from Stanford cs231n lecture slides

Batch normalization

- Why do we need normalization?
  - Internal Covariate Shift
  - To train deeper networks on larger data
- Make each dimension Unit Gaussian

Batch normalization

- Compute the empirical mean and variance independently for each dimension and normalize.



Batch normalization

- Vanilla Normalization:

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

Batch normalization

- Squashing the range:

$$y^{(k)} = \gamma^{(k)}\widehat{x}^{(k)} + \beta^{(k)}$$

- Network can learn:

$$\gamma^{(k)} = \sqrt{\mathrm{Var}[x^{(k)}]}$$
$$\beta^{(k)} = \mathrm{E}[x^{(k)}]$$

Batch normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$
$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$
$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

[Ioffe and Szegedy, 2015]

Batch normalization

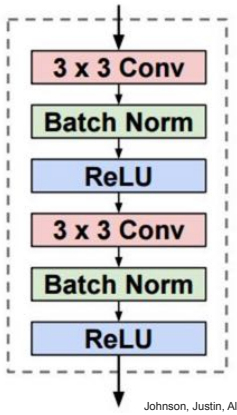
- Why is it good?
  - Improves gradient flow
  - Allows higher learning rates
  - Reduces strong dependence on initialization
  - Regularization

Batch normalization

- At test time
  - The mean and variance are not computed based on the batch.
  - Empirical mean of activations during training is used.

Batch normalization

- Usually added after convolutional layers or fully connected layers
- before non-linearities
- Question: Which is better?
  - Before of after non-linearities



Johnson, Justin, Alexandre Alahi, and Li Fei-Fei. "Perceptual losses for real-time style transfer and super-resolution." *European Conference on Computer Vision*. Springer International Publishing, 2016.

Training details

- Learning rate
- Batch size
- Hyperparameter
- Loss curve

Learning Rate

How **fast** or **slow** we update the model parameters

$$\theta_{t+1} = \theta_t - \eta_t \nabla f(\theta_t)$$

Learning Rate

How **fast** or **slow** we update the model parameters

$$\theta_{t+1} = \theta_t - \eta_t \nabla f(\theta_t)$$

Gradients

Learning Rate

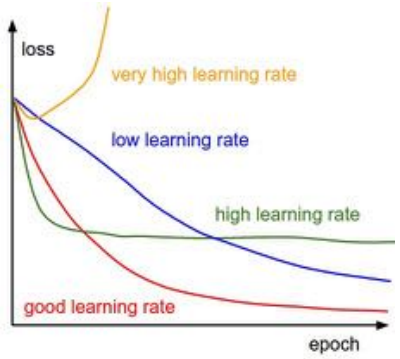
Too large

- Explode

Too small

- Get stuck in local minimum

Learning Rate



ref: http://cs231n.github.io/neural-networks-3/#anneal

Learning Rate

Too large

- Explode

Too small

- Get stuck in local minimum

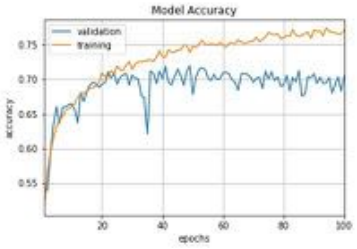
Anneal the learning rate

Learning Rate

- Constant
- Step decay
- Exponential decay
- Time-based decay
- Etc

Learning Rate

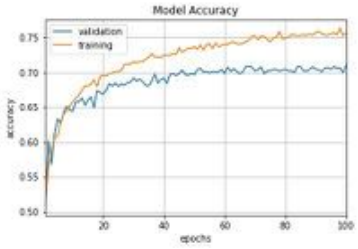
Constant $lr = lr_0$



Dataset: CIFAR-10 ref: https://goo.gl/xnrb3N

Learning Rate

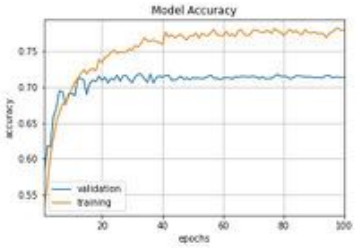
Time-based decay $lr = \frac{lr_0}{1 + k * iter}$

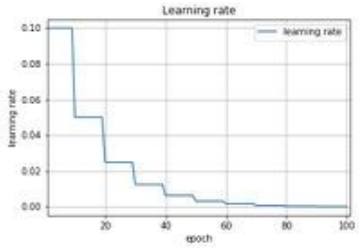


Dataset: CIFAR-10 ref: https://goo.gl/xnrb3N

Learning Rate

Step decay $lr = lr_0 * floor(\frac{epoch}{epoch\_drops})$

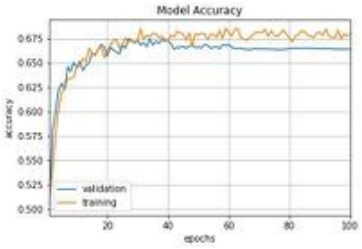


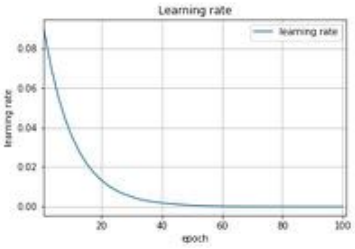


Dataset: CIFAR-10 ref: https://goo.gl/xnrb3N

Learning Rate

Exponential decay $lr = \frac{lr_0}{\exp^{k*iter}}$

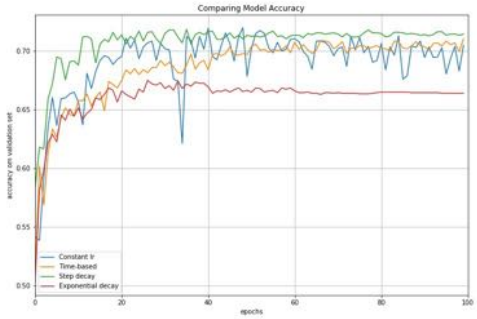




Dataset: CIFAR-10 ref: https://goo.gl/xnrb3N

Learning Rate

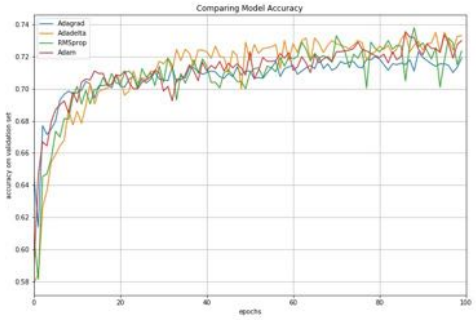
Comparisons



Dataset: CIFAR-10 ref: https://goo.gl/xnrb3N

Learning Rate

Adaptive learning rate methods



Dataset: CIFAR-10 ref: https://goo.gl/xnrb3N

Batch Size

Number of samples in one forward/backward pass

Batch Size

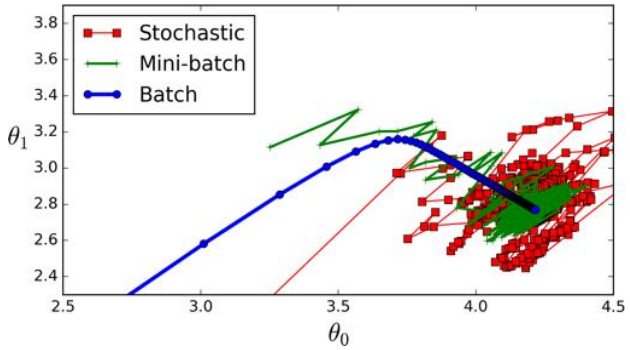
Too large

- Memory inefficient
- Computation inefficient
- Converge to sharp minimizers of the training function

Too small

- Less accurate gradients at each iteration
- Converge to flat minimizers of the training function

Batch Size



Stochastic: batch size is 1

ref: https://stats.stackexchange.com/questions/153531/what-is-batch-size-in-neural-network

Optimizers and Algorithms

Gradient Descent

- Gradient descent
- Stochastic gradient descent
- Mini-batch gradient descent
- Momentum
- Nesterov accelerated gradient
- Etc

Gradient Descent

Computes the gradient of the cost function w.r.t. to the parameters θ for **the entire training dataset at each iteration**

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta_t} f(\theta_t)$$

Stochastic Gradient Descent

Computes the gradient of the cost function w.r.t. to the parameters θ for **each data point at each iteration**

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta_t} f(\theta_t; x^{(i)})$$

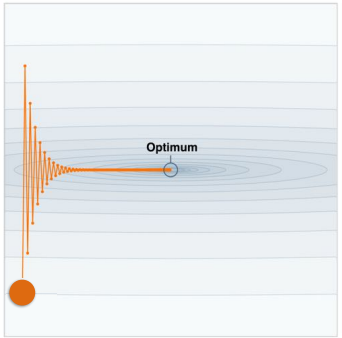
Mini-batch Gradient Descent

Computes the gradient of the cost function w.r.t. to the parameters θ for performs an update for **every mini-batch of n training examples at each iteration**

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta_t} f(\theta_t; x^{(i:i+n)})$$

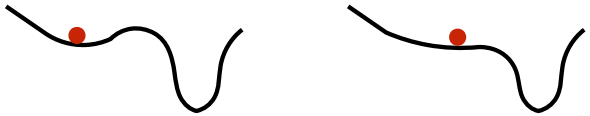
Momentum

GD family updates parameters inefficiency



Momentum

GD family gets stuck at local minimums and saddle points easily



local minimum saddle point

Momentum

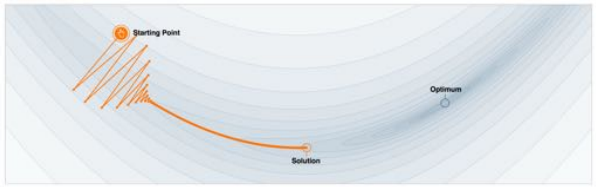
Consider the history:
Adding a fraction γ of **the update vector of the past time step** to the current update vector

$$\nu_t = \gamma \nu_{t-1} + \eta \nabla_{\theta_t} f(\theta_t)$$

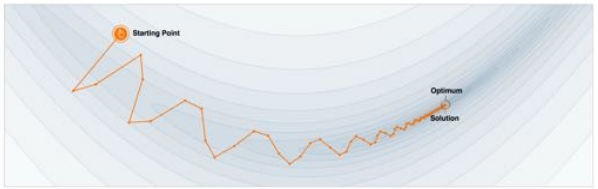
$$\theta_{t+1} = \theta_t - \nu_t$$

Momentum

Without momentum

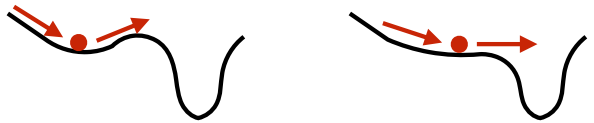


With momentum



Momentum

GD family gets stuck at local minimums and saddle points easily



local minimum saddle point

Nesterov accelerated gradient

Calculate the gradient not w.r.t. to our current parameters but w.r.t. **the approximate future position of our parameters**:

$$\nu_t = \gamma \nu_{t-1} + \eta \nabla_{\theta_t} f(\theta_t - \gamma \nu_{t-1})$$

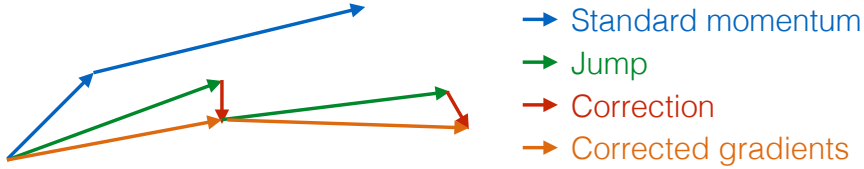
$$\theta_{t+1} = \theta_t - \nu_t$$

Nesterov accelerated gradient

Calculate the gradient not w.r.t. to our current parameters but w.r.t. **the approximate future position of our parameters**:

$$\nu_t = \gamma \nu_{t-1} + \eta \nabla_{\theta_t} f(\theta_t - \gamma \nu_{t-1})$$

$$\theta_{t+1} = \theta_t - \nu_t$$



Regularization

- Extra terms
- Dropout
- Data augmentation

Regularization

$$L = L_{original} + \lambda R(\theta)$$

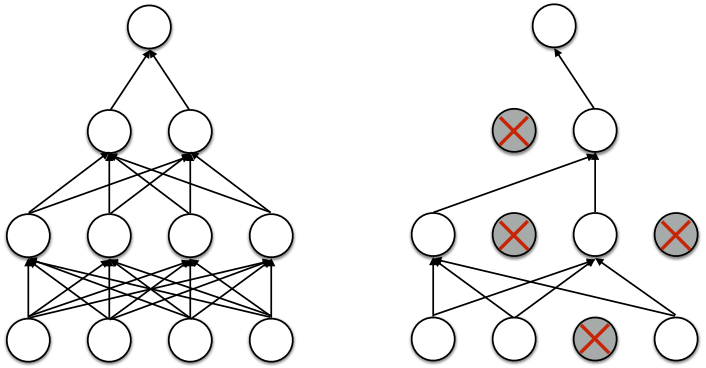
L2 norm regularizer $R(\theta) = \Sigma_i \Sigma_j \theta_{i,j}^2$

L1 norm regularizer $R(\theta) = \Sigma_i \Sigma_j |\theta_{i,j}|$

L1+L2 norm regularizer $R(\theta) = \Sigma_i \Sigma_j |\theta_{i,j}| + \alpha \theta_{i,j}^2$

Dropout

At each iteration, randomly shutdown some neurons with a certain probability (usually 0.5)



Dropout

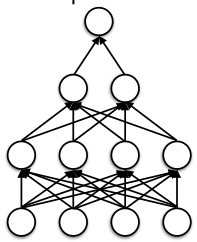
Intuition

- Forces the network to have a redundant representation
- Training a bunches of small models sharing parameters
  - Each binary mask is one model

Dropout

Testing phase

- All neurons are active always
- Scale the activations so that for each neuron
  - Testing output = expected output at training time



Data Augmentation

Deep networks need to be trained on a huge number of training images

What if… we don't have enough amount of data?

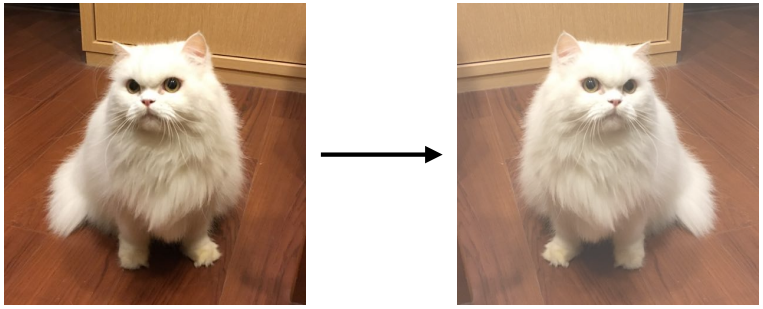
Data Augmentation

Some tricks for image data augmentation

- Flip
- Scale
- Rotate
- Crop
- Color jitter
- etc

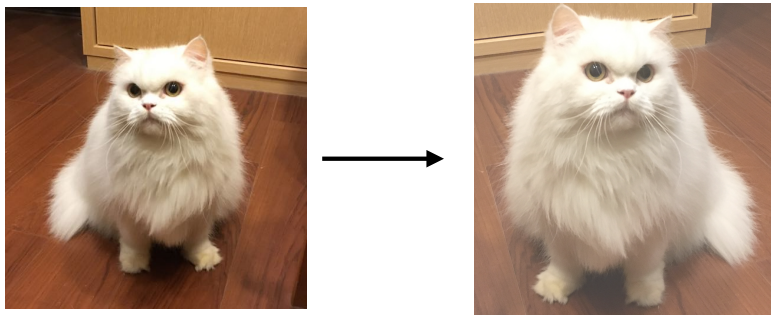
Data Augmentation

Flip



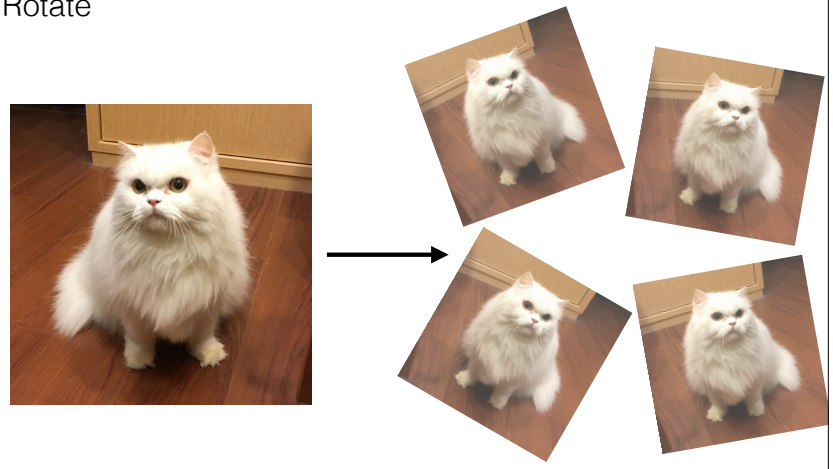
Data Augmentation

Scale



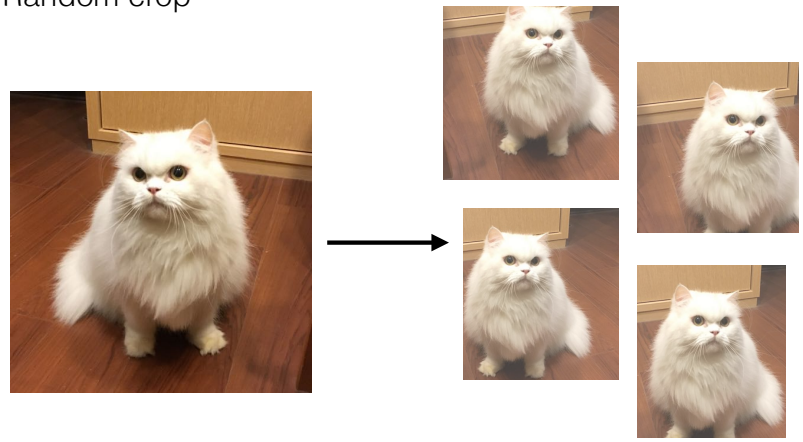
Data Augmentation

Rotate

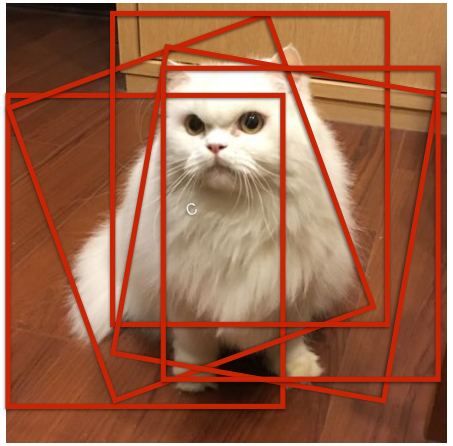


Data Augmentation

Random crop



Data Augmentation



Summary

Deep learning is all about getting everything right…

- Data
- Model
  - Architecture
- Optimizer
- Hyperparameters
- Training tricks
  - Initialization, normalization, regularization, etc

Next week

- Deep learning software
- Cloud services (Google cloud, Amazon AWS)

Todo

- Midterm on week 7
- Assignment #1 & Project proposal due week 8
- Project Meeting with TA #1 by week 8

Questions?