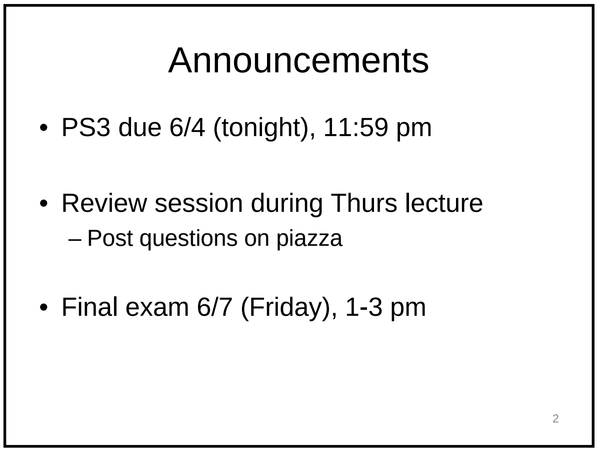
Deep neural networks III

June 4th, 2019

Yong Jae Lee
UC Davis

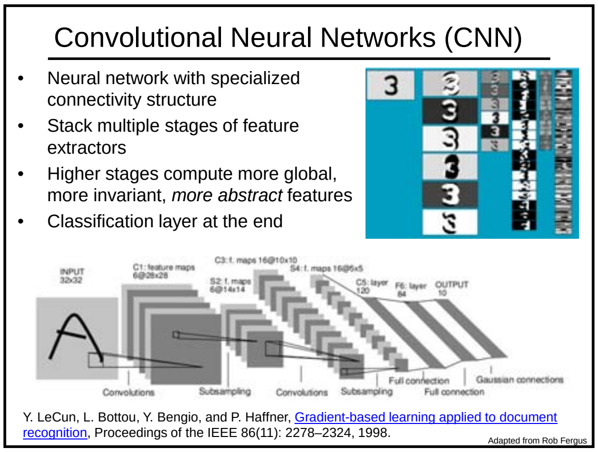
Many slides from Rob Fergus, Svetlana Lazebnik, Jia-Bin Huang, Derek Hoiem, Adriana Kovashka, Andrej Karpathy



Announcements

- PS3 due 6/4 (tonight), 11:59 pm
- Review session during Thurs lecture
  - Post questions on piazza
- Final exam 6/7 (Friday), 1-3 pm

2



Convolutional Neural Networks (CNN)

- Neural network with specialized connectivity structure
- Stack multiple stages of feature extractors
- Higher stages compute more global, more invariant, *more abstract* features
- Classification layer at the end

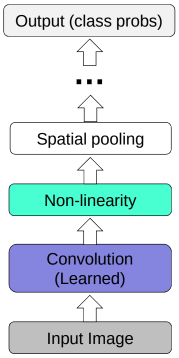
Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86(11): 2278–2324, 1998.

Adapted from Rob Fergus

Convolutional Neural Networks (CNN)

- Feed-forward feature extraction:
  1. Convolve input with learned filters
  2. Apply non-linearity
  3. Spatial pooling (downsample)

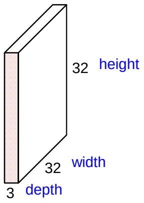
- Supervised training of convolutional filters by back-propagating classification error



Adapted from Lana Lazebnik

Convolutions: More detail

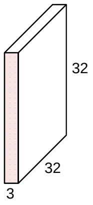
32x32x3 image



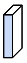
Andrej Karpathy

Convolutions: More detail

32x32x3 image

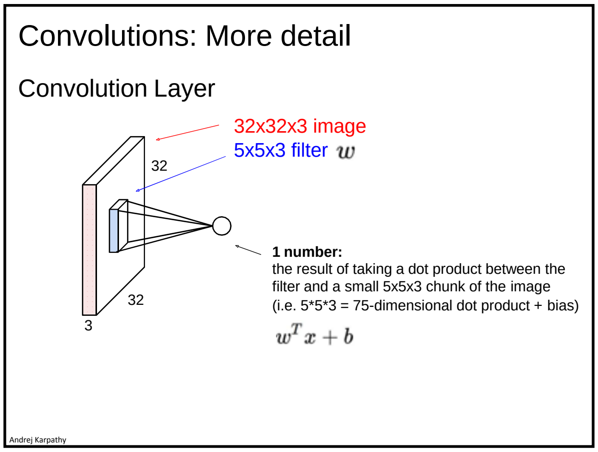


5x5x3 filter



Convolve the filter with the image i.e. "slide over the image spatially, computing dot products"

Andrej Karpathy



Convolutions: More detail

Convolution Layer

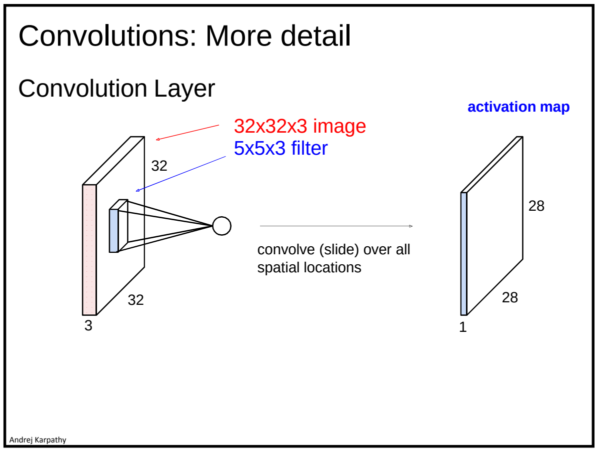
32x32x3 image

5x5x3 filter w

1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

Andrej Karpathy



Convolutions: More detail

Convolution Layer

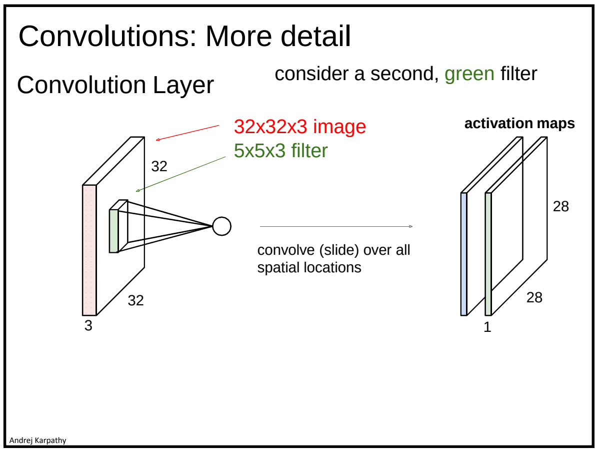
32x32x3 image

5x5x3 filter

activation map

convolve (slide) over all spatial locations

Andrej Karpathy



Convolutions: More detail

Convolution Layer

consider a second, green filter

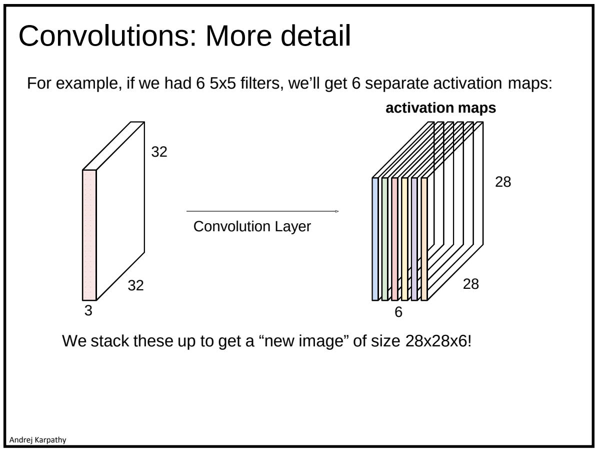
32x32x3 image

5x5x3 filter

activation maps

convolve (slide) over all spatial locations

Andrej Karpathy

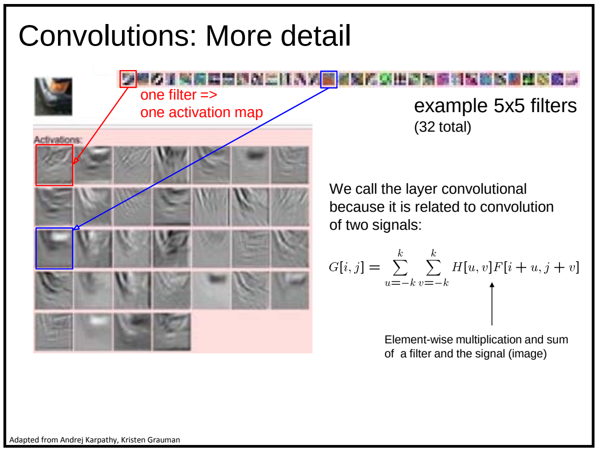


Convolutions: More detail

For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

We stack these up to get a "new image" of size 28x28x6!

Andrej Karpathy



Convolutions: More detail

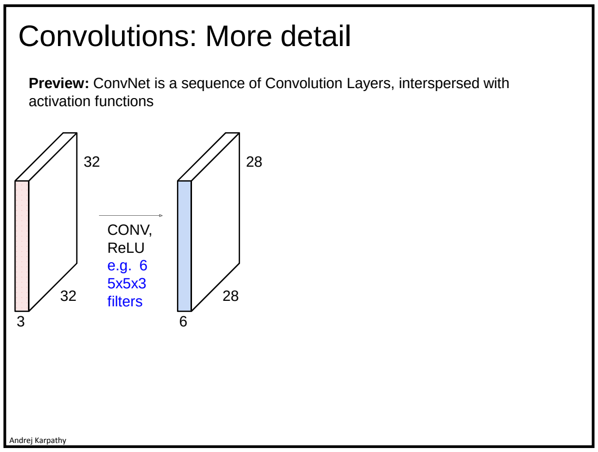
example 5x5 filters
(32 total)

We call the layer convolutional because it is related to convolution of two signals:

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u,j+v]$$

Element-wise multiplication and sum of a filter and the signal (image)

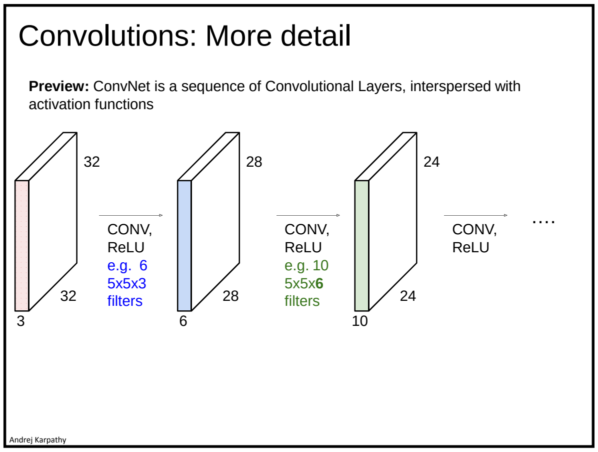
Adapted from Andrej Karpathy, Kristen Grauman



Convolutions: More detail

Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions

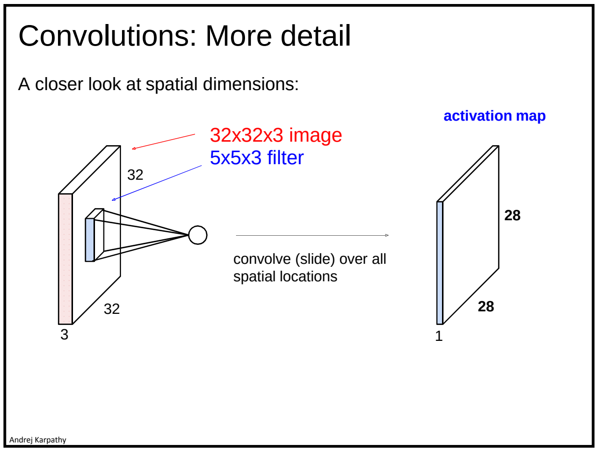
Andrej Karpathy



Convolutions: More detail

Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions

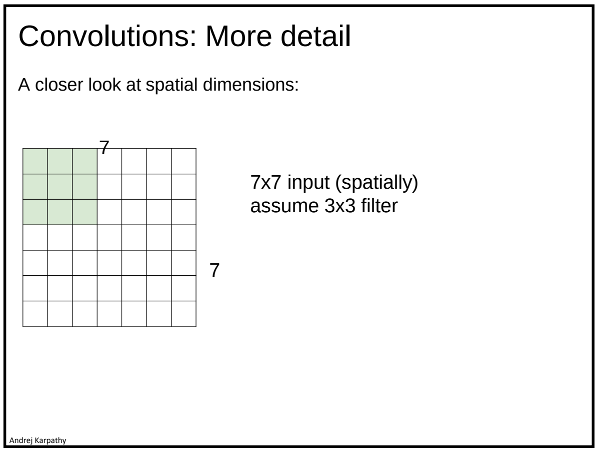
Andrej Karpathy



Convolutions: More detail

A closer look at spatial dimensions:

Andrej Karpathy



Convolutions: More detail

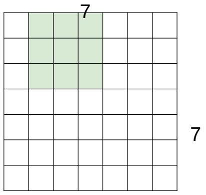
A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter

Andrej Karpathy

Convolutions: More detail

A closer look at spatial dimensions:

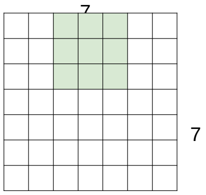


7x7 input (spatially)
assume 3x3 filter

Andrej Karpathy

Convolutions: More detail

A closer look at spatial dimensions:

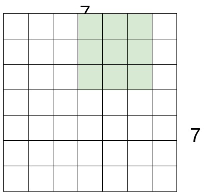


7x7 input (spatially)
assume 3x3 filter

Andrej Karpathy

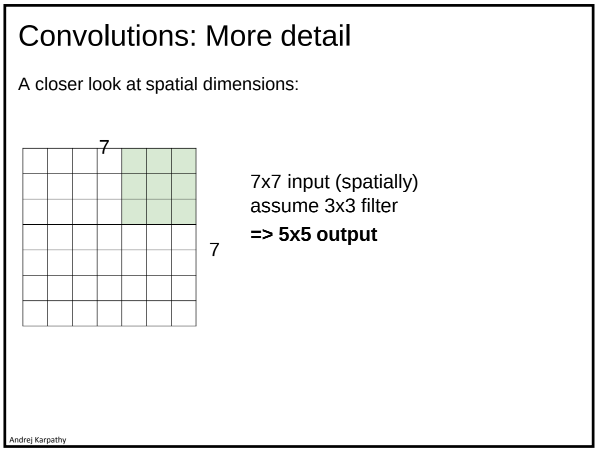
Convolutions: More detail

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

Andrej Karpathy

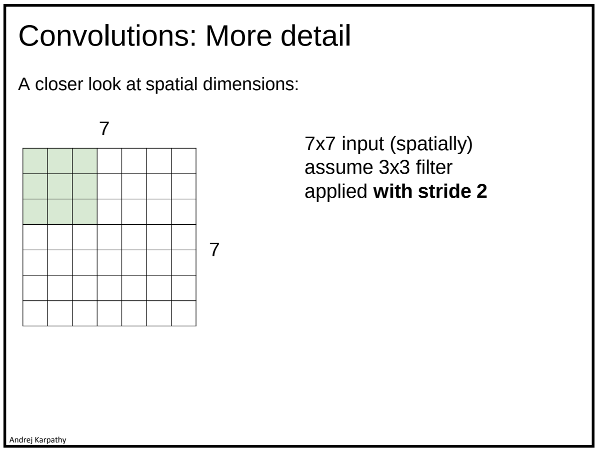


Convolutions: More detail

A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter
=> 5x5 output

Andrej Karpathy

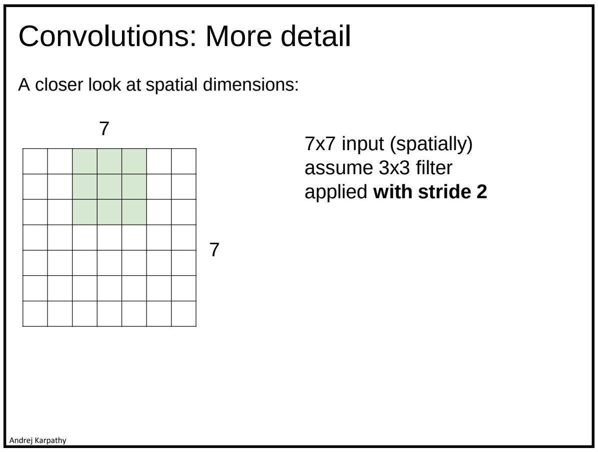


Convolutions: More detail

A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Andrej Karpathy

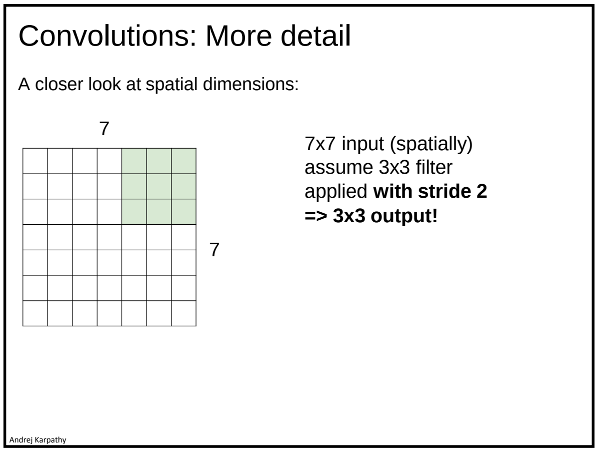


Convolutions: More detail

A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Andrej Karpathy

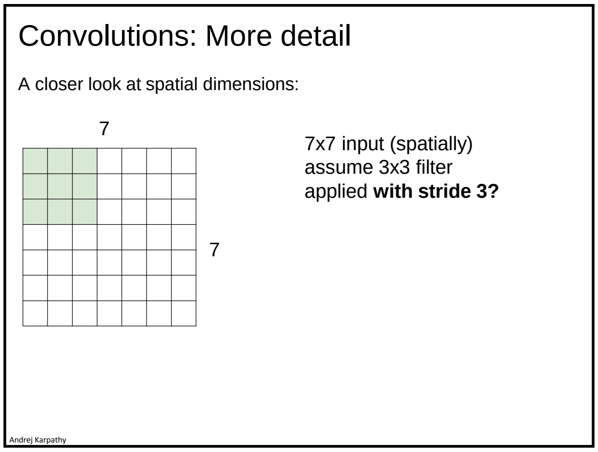


Convolutions: More detail

A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

Andrej Karpathy

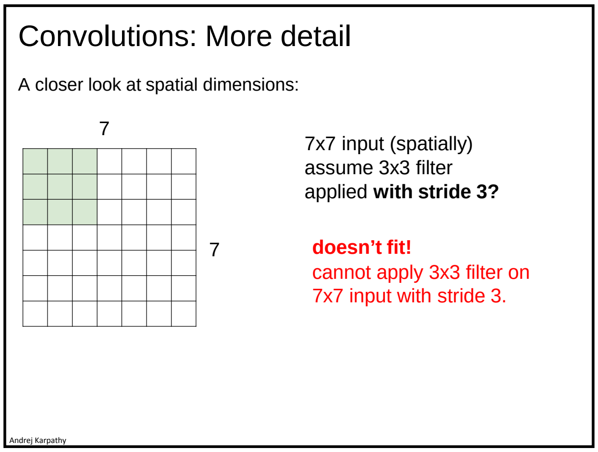


Convolutions: More detail

A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

Andrej Karpathy



Convolutions: More detail

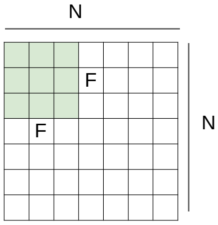
A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

doesn’t fit!
cannot apply 3x3 filter on
7x7 input with stride 3.

Andrej Karpathy

Convolutions: More detail



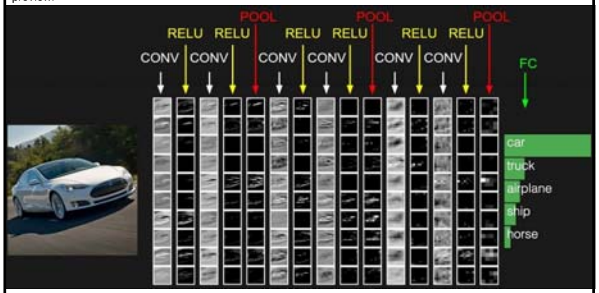
Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

Andrej Karpathy

Convolutions: More detail

preview:



Andrej Karpathy

A Common Architecture: AlexNet

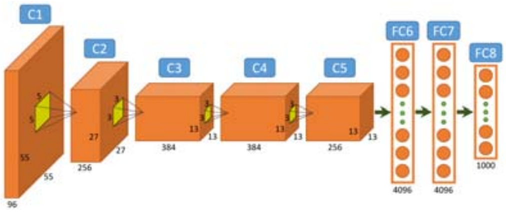


Figure from http://www.mdpi.com/2072-4292/7/11/14680/htm

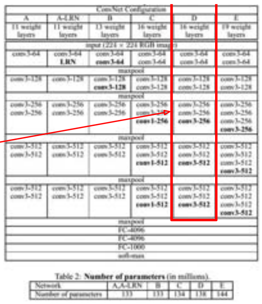
Case Study: VGGNet

[Simonyan and Zisserman, 2014]

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

best model

11.2% top 5 error in ILSVRC 2013
->
7.3% top 5 error



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

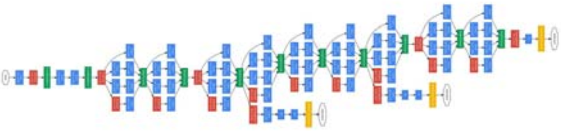
Table 2: **Number of parameters** (in millions).

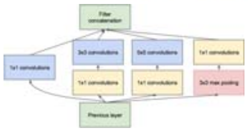
Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Andrej Karpathy

Case Study: GoogLeNet

[Szegedy et al., 2014]





Inception module

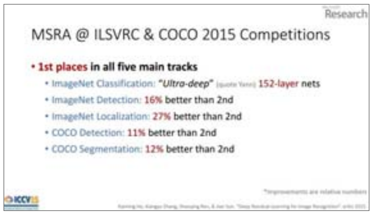
ILSVRC 2014 winner (6.7% top 5 error)

Andrej Karpathy

Case Study: ResNet

[He et al., 2015]

ILSVRC 2015 winner (3.6% top 5 error)



Research

MSRA @ ILSVRC & COCO 2015 Competitions

- 1st places in all five main tracks
  - ImageNet Classification: *"Ultra-deep"* (quote Yann) 152-layer nets
  - ImageNet Detection: 16% better than 2nd
  - ImageNet Localization: 27% better than 2nd
  - COCO Detection: 11% better than 2nd
  - COCO Segmentation: 12% better than 2nd

*improvements are relative numbers

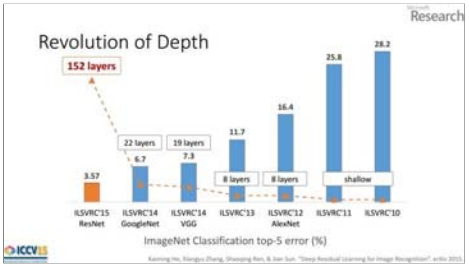
ICCV15

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

Slide from Kaiming He's recent presentation https://www.youtube.com/watch?v=1PGLj-uKT1w

Andrej Karpathy

Case Study: ResNet



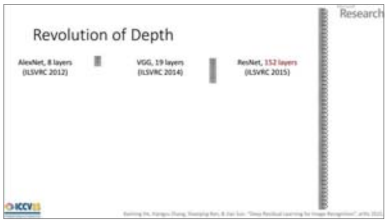
(slide from Kaiming He's recent presentation)

Andrej Karpathy

Case Study: ResNet

[He et al., 2015]

ILSVRC 2015 winner (3.6% top 5 error)



2-3 weeks of training on 8 GPU machine

(slide from Kaiming He's recent presentation)

Andrej Karpathy

Practical matters

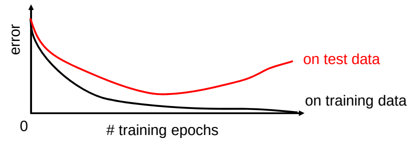
Comments on training algorithm

- Not guaranteed to converge to zero training error, may converge to local optima or oscillate indefinitely.
- However, in practice, does converge to low error for many large networks on real data.
- Thousands of epochs (epoch = network sees all training data once) may be required, hours or days to train.
- To avoid local-minima problems, run several trials starting with different random weights (*random restarts*), and take results of trial with lowest training set error.
- May be hard to set learning rate and to select number of hidden units and layers.
- Neural networks had fallen out of fashion in 90s, early 2000s; back with a new name and significantly improved performance (deep networks trained with dropout and lots of data).

Ray Mooney, Carlos Guestrin, Dhruv Batra

Over-training prevention

- Running too many epochs can result in over-fitting.

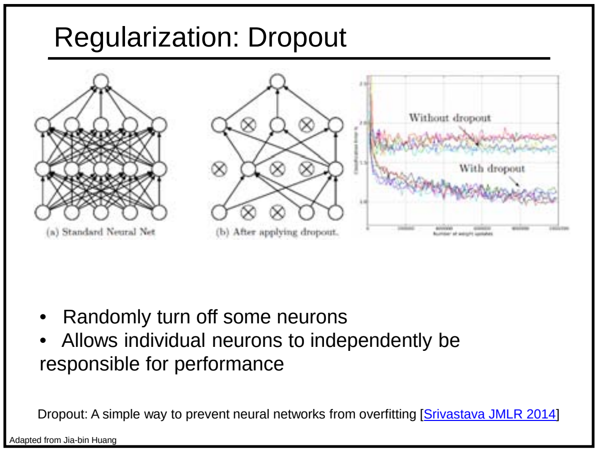


- Keep a hold-out validation set and test accuracy on it after every epoch. Stop training when additional epochs actually increase validation error.

Adapted from Ray Mooney

Training: Best practices

- Use mini-batch
- Use regularization
- Use cross-validation for your parameters
- Use RELU or leaky RELU, don’t use sigmoid
- Center (subtract mean from) your data
- Learning rate: too high? too low?
- Use Batch Normalization

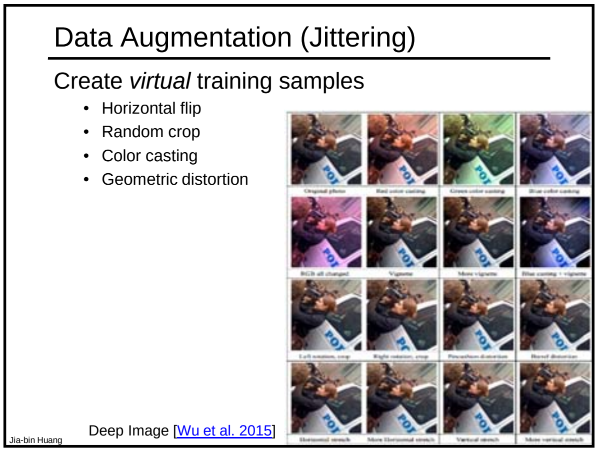


Regularization: Dropout

- Randomly turn off some neurons
- Allows individual neurons to independently be responsible for performance

Dropout: A simple way to prevent neural networks from overfitting [Srivastava JMLR 2014]

Adapted from Jia-bin Huang



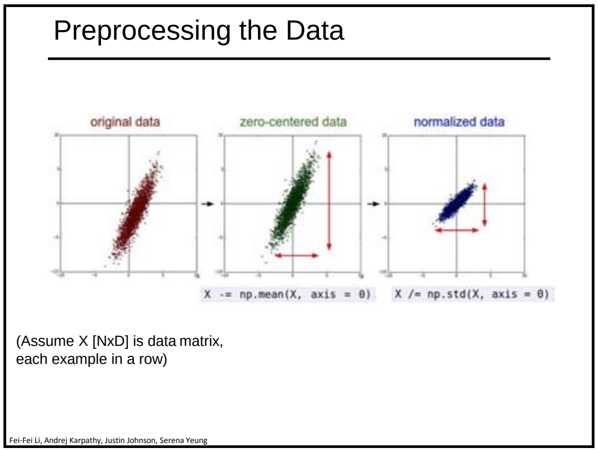
Data Augmentation (Jittering)

Create *virtual* training samples

- Horizontal flip
- Random crop
- Color casting
- Geometric distortion

Jia-bin Huang

Deep Image [Wu et al. 2015]



Preprocessing the Data

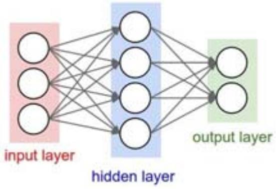
```
X -= np.mean(X, axis = 0)
```

```
X /= np.std(X, axis = 0)
```

(Assume X [NxD] is data matrix,
each example in a row)

Fei-Fei Li, Andrej Karpathy, Justin Johnson, Serena Yeung

Weight Initialization



Q: what happens when W=constant init is used?

Fei-Fei Li, Andrej Karpathy, Justin Johnson, Serena Yeung

Weight Initialization

- Another idea: **Small random numbers**
(gaussian with zero mean and 1e-2 standard deviation)

```
W = 0.01* np.random.randn(D,H)
```

Works ~okay for small networks, but problems with deeper networks.

Make variance of input and output in each layer similar
- Xavier initialization [Glorot et al. 2010]
- He initialization [He et al. 2015]

Fei-Fei Li, Andrej Karpathy, Justin Johnson, Serena Yeung

Batch Normalization

[Ioffe and Szegedy, 2015]

“you want zero-mean unit-variance activations? just make them so.”

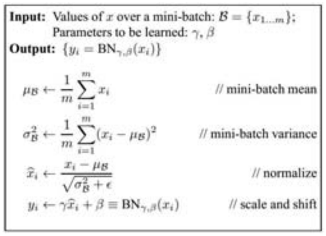
consider a batch of activations at some layer. To make each dimension zero-mean unit-variance, apply:

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

Fei-Fei Li, Andrej Karpathy, Justin Johnson, Serena Yeung

Batch Normalization

[Ioffe and Szegedy, 2015]



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

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

- Improves gradient flow through the network
- Allows higher learning rates
- Reduces the strong dependence on initialization
- Acts as a form of regularization

Fei-Fei Li, Andrej Karpathy, Justin Johnson, Serena Yeung

Transfer Learning

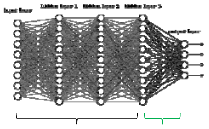
"You need a lot of a data if you want to train/use CNNs"

BUSTED

Andrej Karpathy

Transfer Learning with CNNs

- The more weights you need to learn, the more data you need
- That's why with a deeper network, you need more data for training than for a shallower network
- One possible solution:



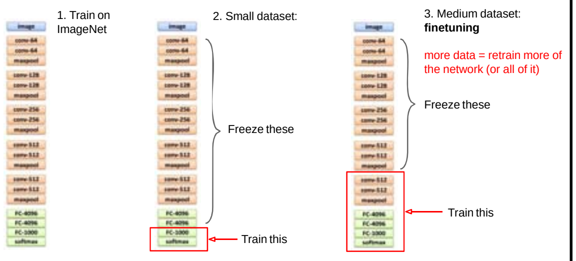
Set these to the already learned weights from another network

Learn these on your own task

Transfer Learning with CNNs

Source: classification on ImageNet

Target: some other task/data



1. Train on ImageNet

2. Small dataset:

Freeze these

Train this

3. Medium dataset: **finetuning**

more data = retrain more of the network (or all of it)

Freeze these

Train this

Adapted from Andrej Karpathy

Summary

- We use deep neural networks because of their strong performance in practice
- Convolutional neural networks (CNN)
  - Convolution, nonlinearity, max pooling
- Training deep neural nets
  - We need an objective function that measures and guides us towards good performance
  - We need a way to minimize the loss function: stochastic gradient descent
  - We need backpropagation to propagate error through all layers and change their weights
- Practices for preventing overfitting
  - Dropout; data augmentation; transfer learning