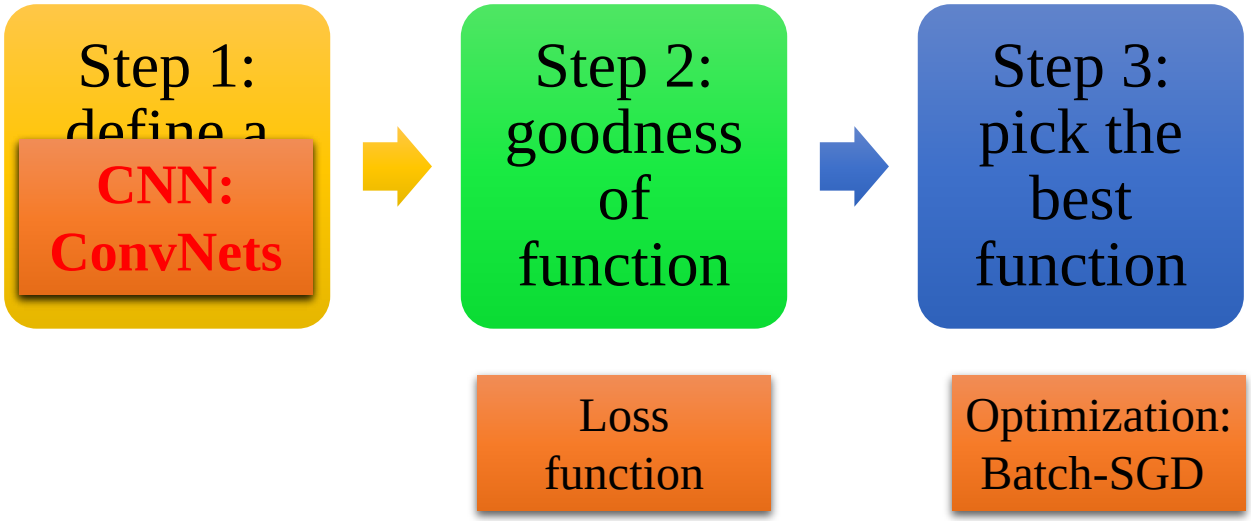
Lecture 3: Image Processing and Convolutional Neural Networks

课程：机器学习与深度学习

Recap: Three Steps for Deep Learning

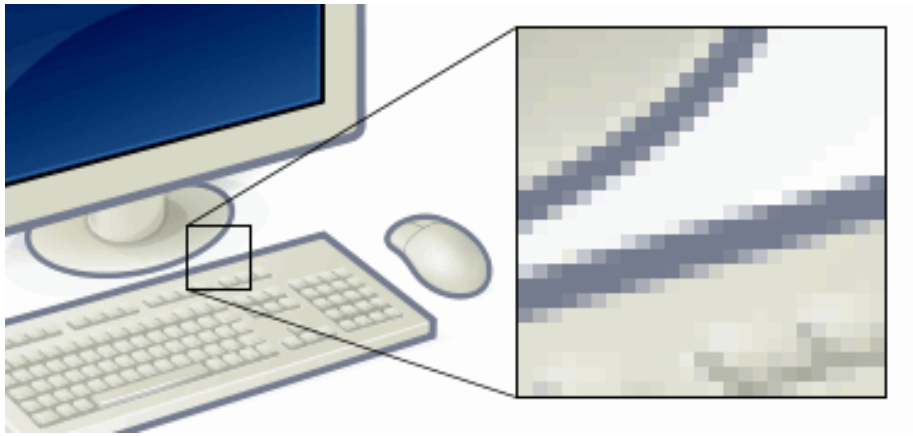


Overview

- Image processing
- Convolutional neural networks
  - Convolution and pooling
  - CNN Architectures
  - CNN outside image classification

Image Processing Basics

- An image is a collection of pixels
- Grayscale (intensity): [0,255]
- Color (three color channels)
  - RGB
  - YCbCr
  - HSV



Wikipedia

$$Y = 0.299R + 0.587G + 0.114B$$
$$Cb = 0.564(B - Y)$$
$$Cr = 0.713(R - Y)$$
$$R = Y + 1.402Cr$$
$$G = Y - 0.344Cb - 0.714Cr$$
$$B = Y + 1.772Cb$$

RGB Color Models

- Red Green Blue are combined to reproduce a broad array of colors
- The choice of primary colors is related to the physiology of the human eye

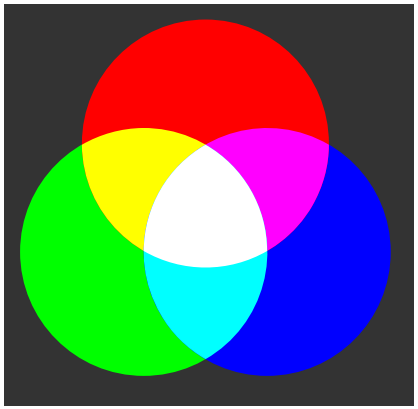


Image as a function: $\mathbb{R}^2 \to \mathbb{R}$

$f(x, y)$ gives the **intensity** at position (x, y)

CFAIR-10 dataset

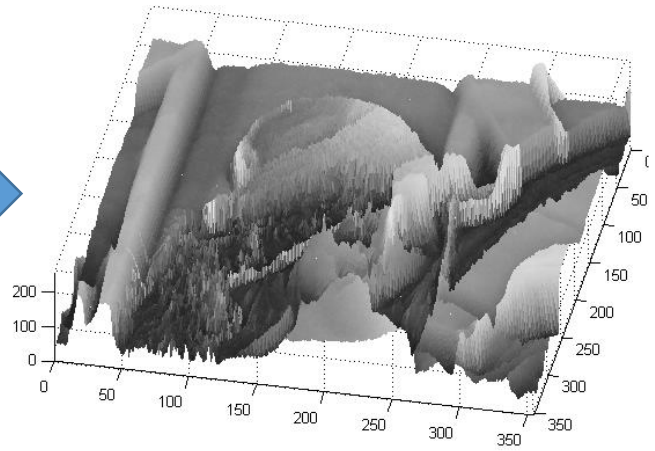
Grayscale image

- $\boldsymbol{f}: [a, b] \times [c, d] \to [0,255]$

Color image

- $\boldsymbol{f} \coloneqq (\boldsymbol{r}, \boldsymbol{g}, \boldsymbol{b})$
  $\boldsymbol{r}, \boldsymbol{g}, \boldsymbol{b}: [a, b] \times [c, d] \to [0,255]$

Color image



Plot of one color channel

Convolution

- Used for extracting interesting features from images, such as *blurring*, *sharpening*, *edge detection*
- Convolution operation is essential for more sophisticated image processing tasks, such as object recognition, segmentation
- The convolution of x and k is defined as

$$s(t) = (x * k)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} x(\tau)k(t-\tau)d\tau$$

  where x is the *input feature* and k is the ***kernel***.
- $x * k = k * x$ (commutative rule)

Convolution

- For image data (discrete and 2-dim), image input $I(i,j)$; $I, K: \mathbb{N} \times \mathbb{N} \to \mathbb{R}$

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

$$= \sum_m \sum_n I(i-m, j-n)K(m,n)$$

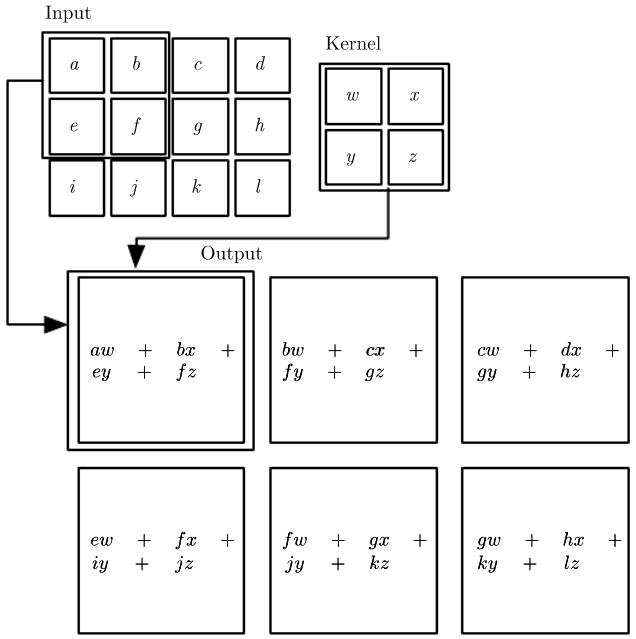
  The later expression is used more often in computer vision

- Cross-correlation:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

The Convolution Operation

- In practice we remove the tails and truncate K to form a kernel matrix



Examples: Gaussian Kernels

- $G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$
- Gaussian kernel reduces the image noise and resolution (image blurring)
- Preprocess the image before feature extraction is done (e.g. edge detection)

$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Gaussian kernel with $\sigma = 1$

Gaussian blur, Wikipedia

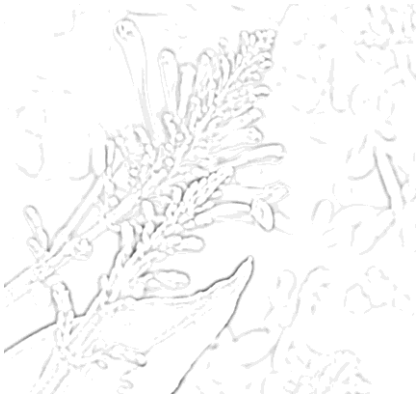
Examples: Feature Detectors

Edge detection

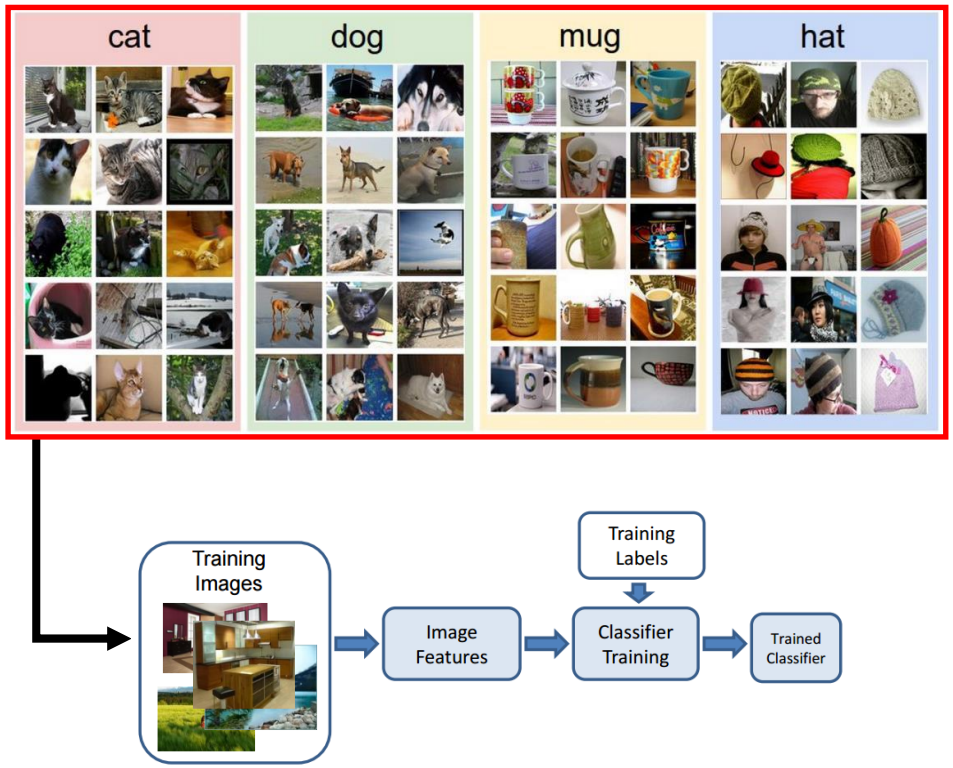
- DoG (Difference of Gaussian)
  - $I * (G(;\sigma_1) - G(;\sigma_2))$
- LoG (Laplacian of Gaussian)
  - $L_{xx} + L_{yy}$ where $L = I * G$

Before DOG

After DOG



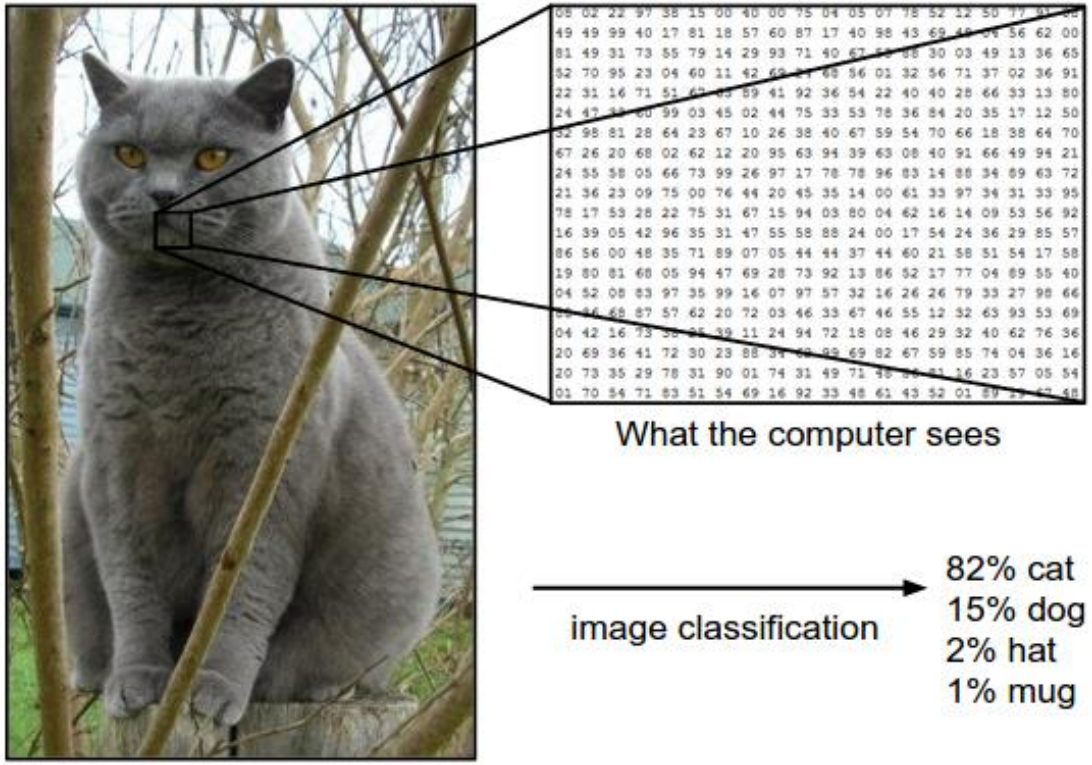
Object Recognition



Before deep learning, OR focuses on feature engineering: hand-picked features + (non)linear classifiers (SVM, Logistic regression)

Image Classification

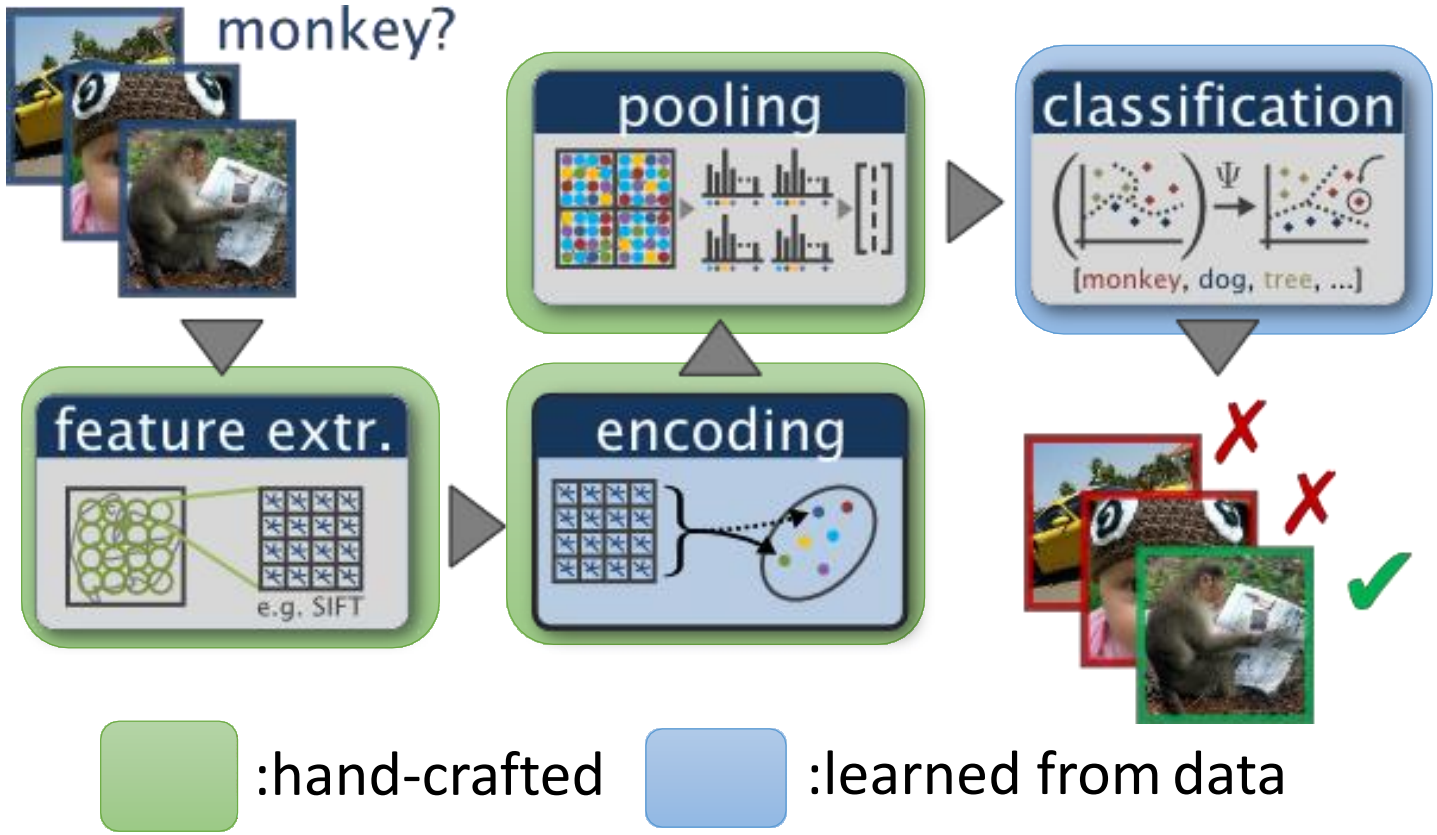
- Recognize a cat



Alex Krizhevsky, Stanford

Image Classification

Shallow Model

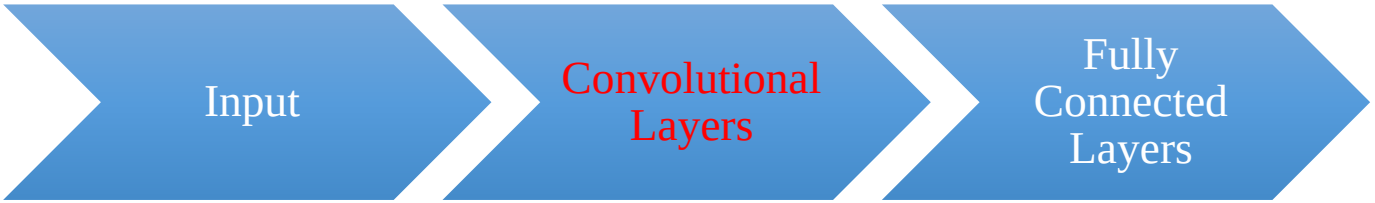


http://www.robots.ox.ac.uk/~vgg/research/encoding_eval/

Comparison of Deep Learning and Others

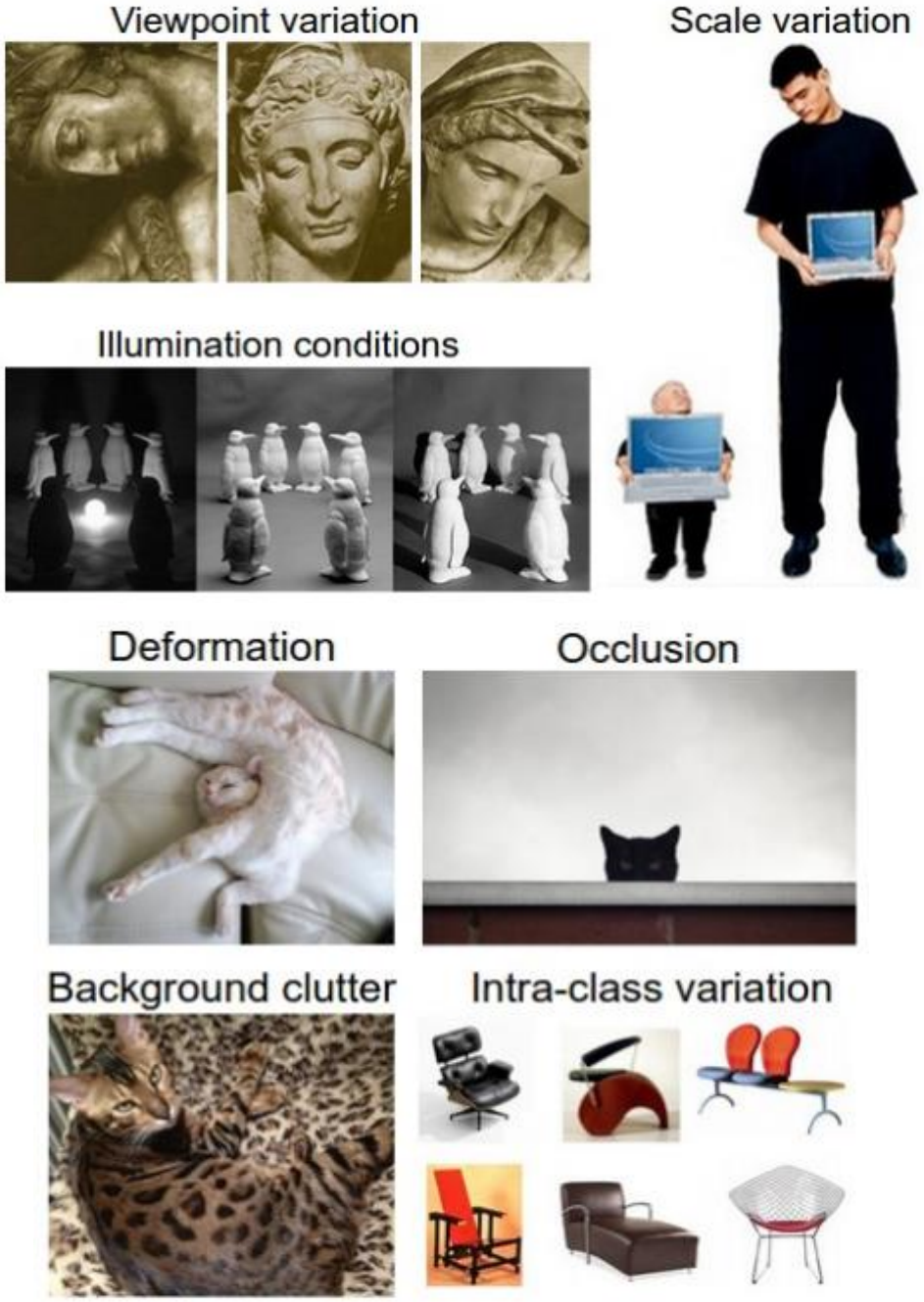
- "Shallow" Models: feature engineering is crucial for model performance

- Deep Learning: automatically learn the features and kernels



Challenges

- Viewpoint variation
- Scale variation
- Deformation
- Occlusion
- Illumination conditions
- Background clutter
- Intra-class variation



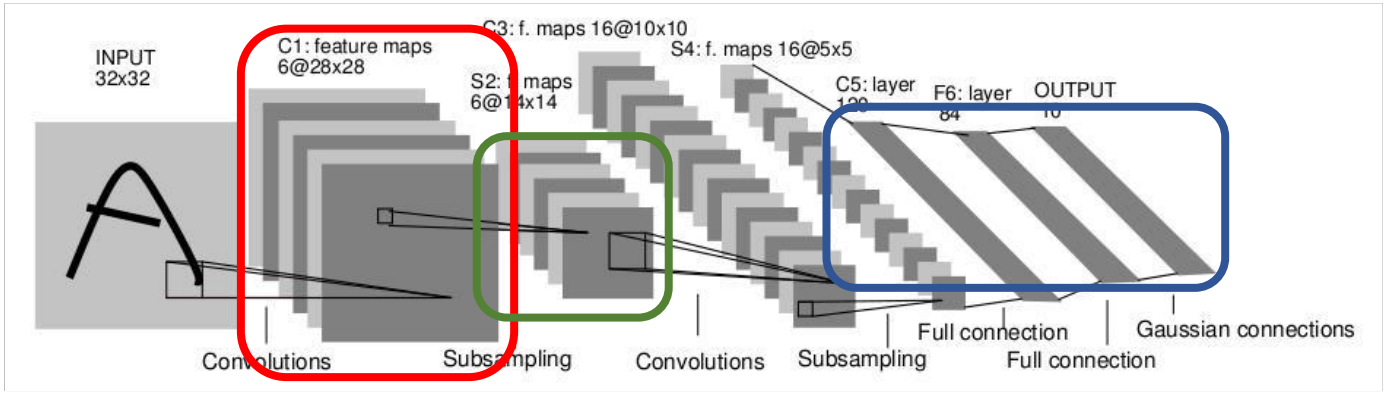
CNN for Image Classification

Overview

- Image processing
- Convolutional neural networks
  - Convolution and Pooling
  - CNN Architectures
  - ConvNets outside image classification

Convolutional Neural Networks (CNN)

- Lecun et al. 1989
- Neural nets with specialized connectivity structure:
  convolution + pooling + fully connected layers
- Widely used in computer vision



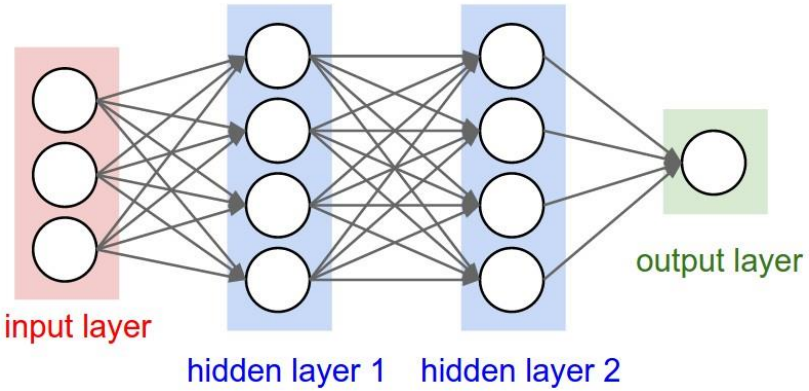
Convolution Pooling Fully connected layers

Y. LeCun , B. Boser , J. S. Denker , D. Henderson , R. E. Howard , W. Hubbard and L. D. Jackel, “Backpropagation applied to handwritten zip code recognition”, Neural Computation, vol. 1, no. 4, pp. 541-551, 1989.

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.

Convolutional Neural Nets

- Recall fully connected feedforward neural networks
  - Each neuron is fully connected to all neurons in the previous layer
  - Neurons in a single layer completely independent

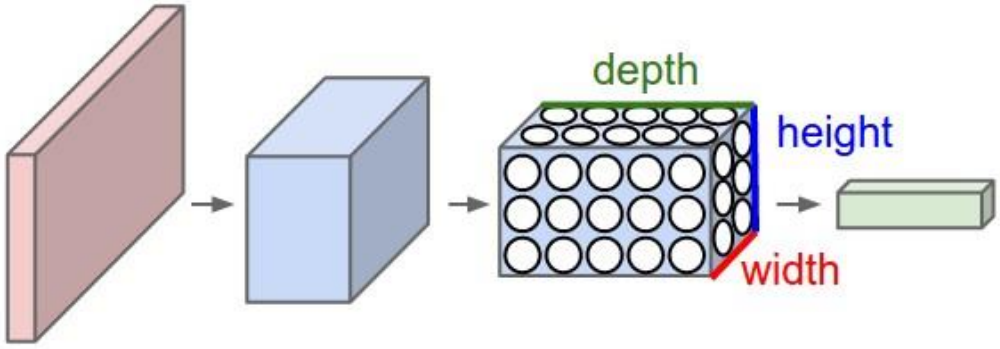


Do not scale well to images, and also not consider their characteristics

→ Convolutional neural nets

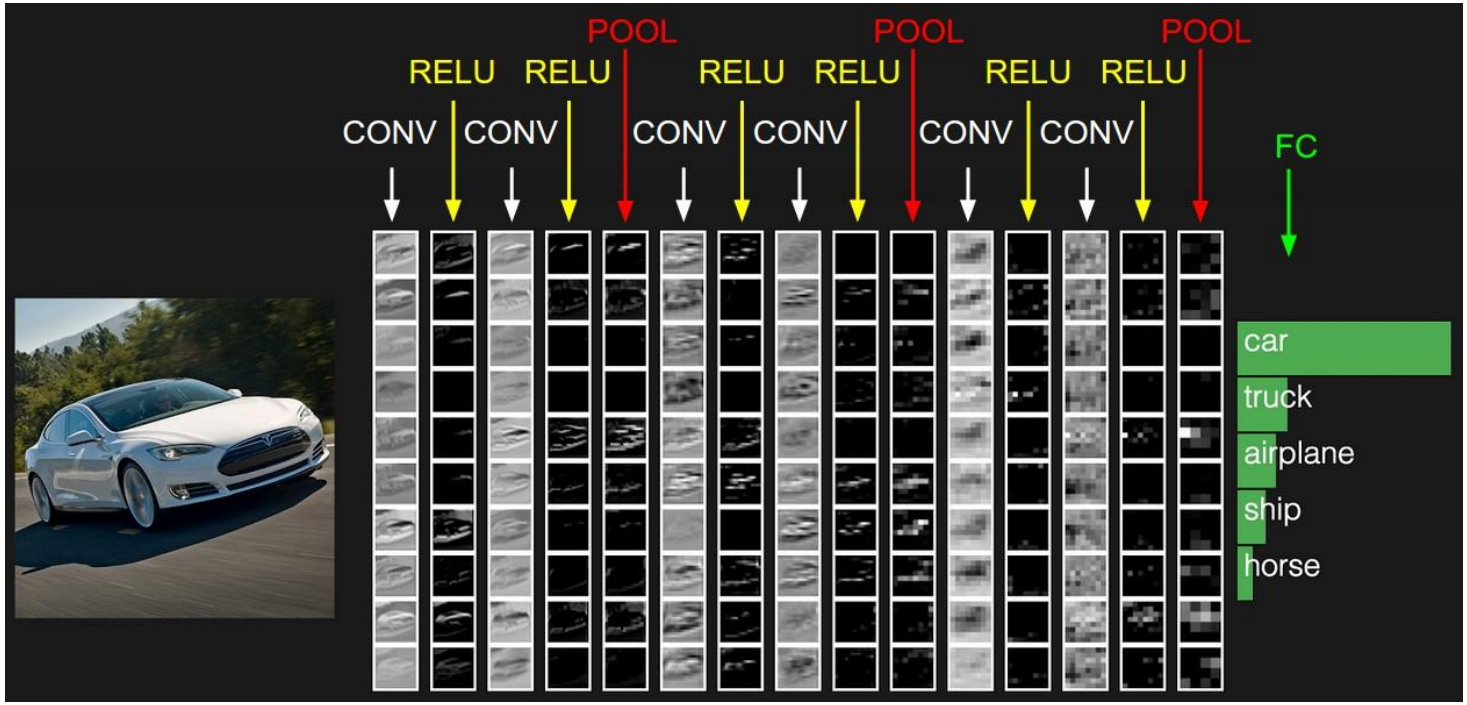
Convolutional Neural Nets

- Structure[1]
  - Convolutional layer (CONV)
    - Local connectivity
    - Parameter sharing
    - Convolution
  - Pooling layer (POOL)
  - Fully-connected layer (FC)



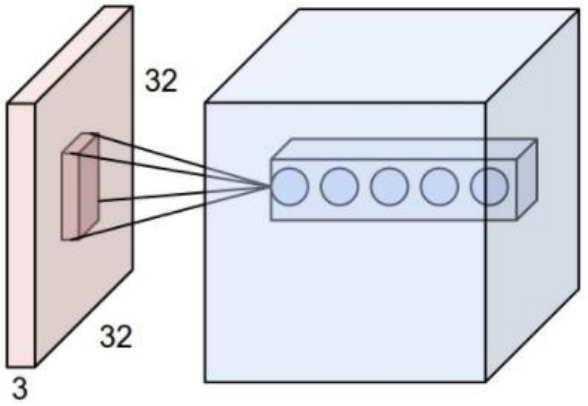
1 CS231n Convolutional Neural Networks for Visual Recognition

Architecture



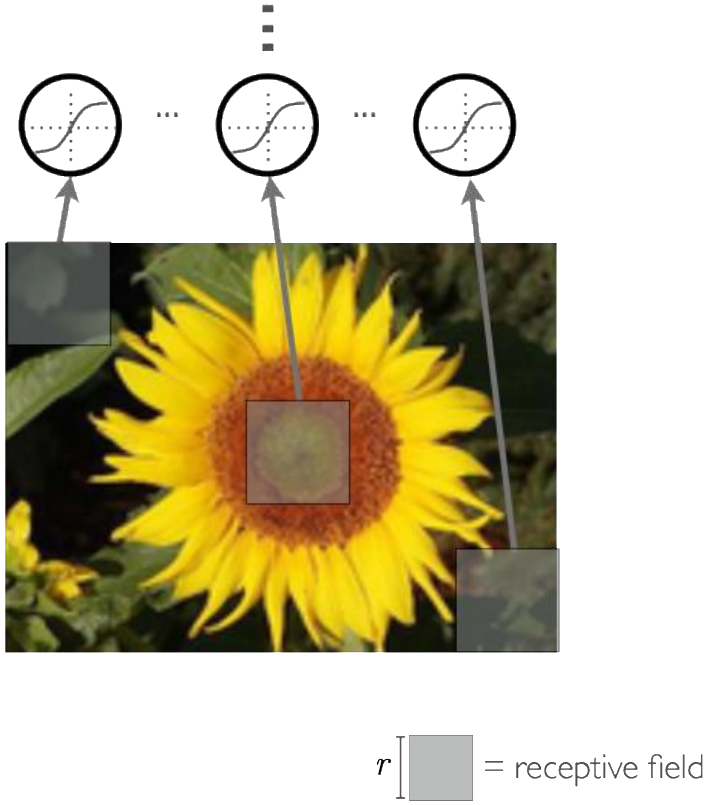
CONV Layer: Overview

- CONV Layer is highly specialized for processing image features
  - Input: preserve spatial structure
  - Process
    - Several filters: learnable parameters
    - Slide filter over the input -> produce 2-d activation map
  - Output: Stack activation maps



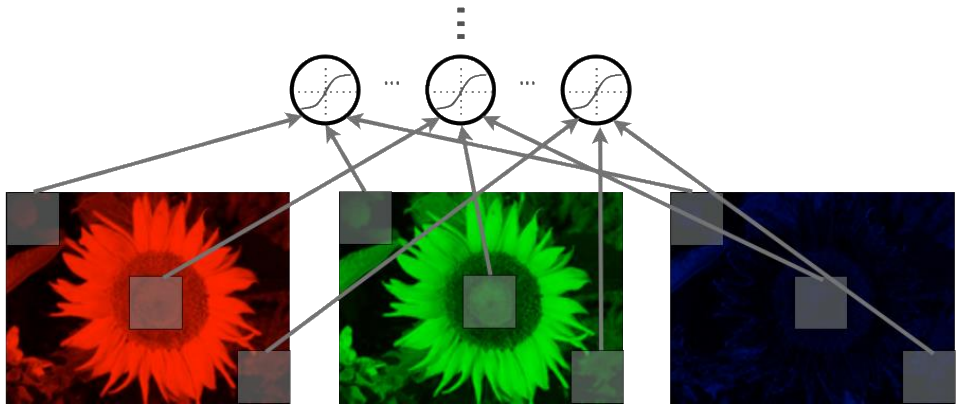
CONV Layer: Local Connectivity

- Neurons in a layer will only be connected to *nearby* neurons in the next layer: **receptive field**
- Why local?
  - Image is high dimensional data, typically > 10000-d. Fully connected layer is expensive

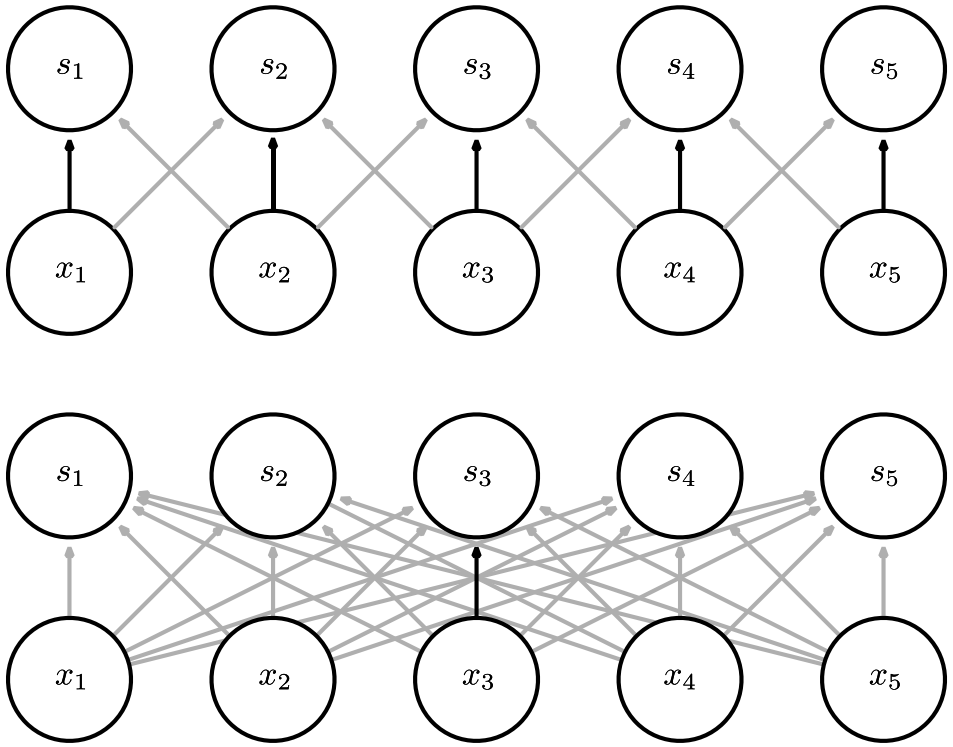


CON Layer: Local Connectivity

- They are connected to all channels: 1 for grayscale; 3 for color image (R/G/B)
- Thus, connectivity is *local* in *space* but *full* in *depth*

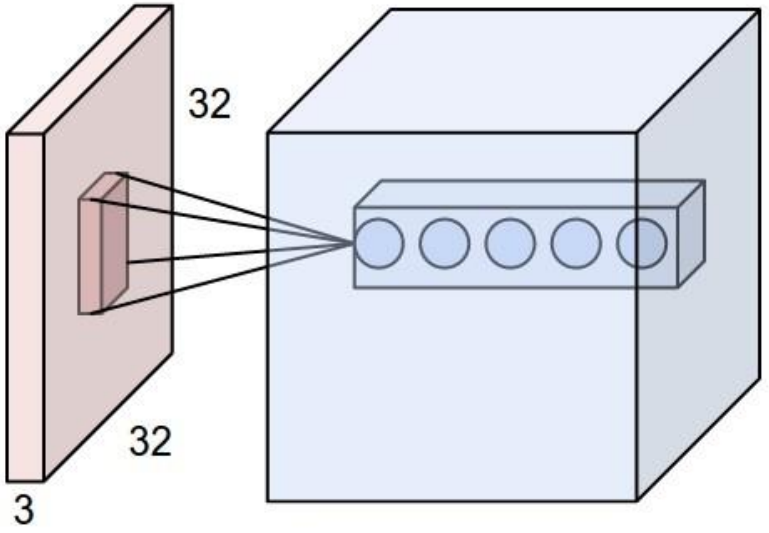


CONV Layer: Local Connectivity



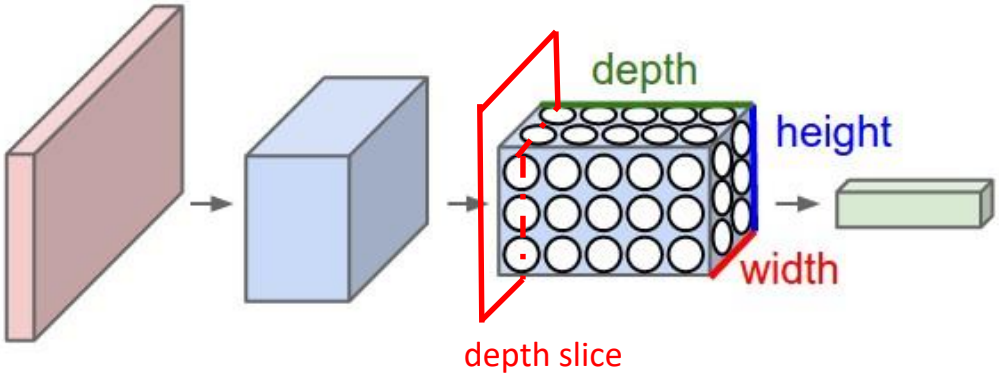
CONV Layer: Example

- Suppose that the input volume has size [32x32x3], (e.g. an RGB CIFAR-10 image).
- If the receptive field (or the filter size) is 5x5, then each neuron in the Conv Layer will have weights to a 5x5x3 box in the input volume, for a total of 5*5*3 = 75 weights (and +1 bias parameter).
- Connectivity is local in ***space*** but full in ***depth***, w.r.t. all the conv layers



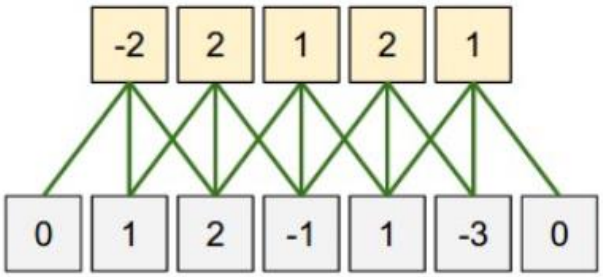
CONV Layer: Spatial Arrangement

- Each layer is arranged in three dimension: **width, height, depth**
  - Depth of the output: number of filters
  - Stride: when slide the filter
  - **Zero-padding**
- A single 2-d slice of depth is called a depth slice

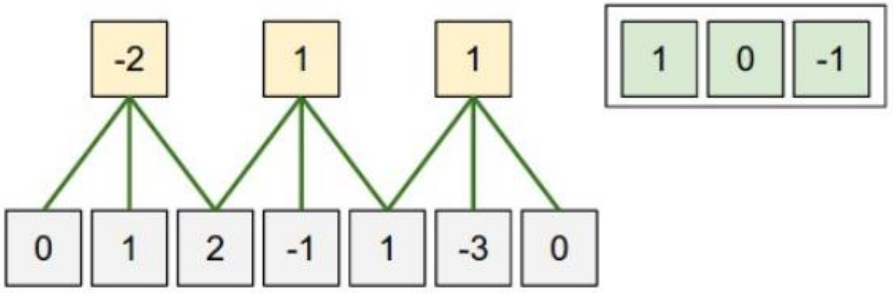


CONV Layer: Spatial Arrangement

Stride=1



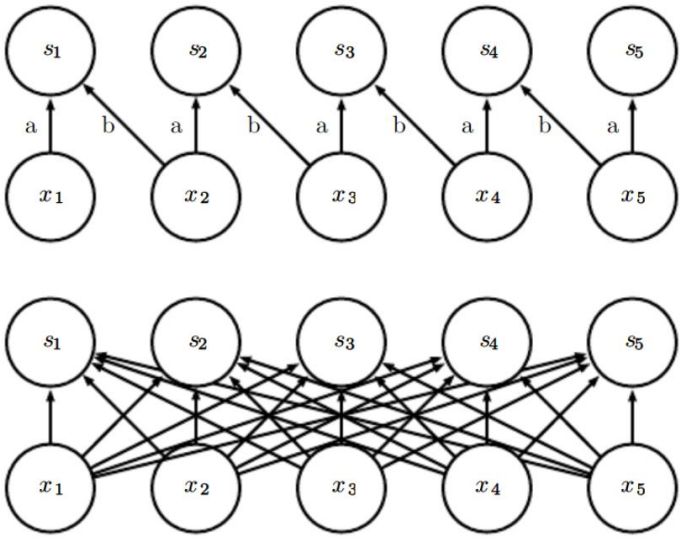
Stride=2



Input: one spatial dimension
Input size=5
Zero-padding=1
Size of the receptive field=3

Convolutional Layer: Parameters Sharing

- If one feature is useful to compute at some spatial position (x, y), then it should also be useful to compute at a different position (x_2, y_2).



Parameters Sharing

- Each depth slice uses the same parameter, the forward propagation of CONV is computed as the *convolution* of the neuron's weights
- Sometimes, when images have centered structures
  - No parameter sharing, but *locally-connected layer*

If not CONV, fully connected layer

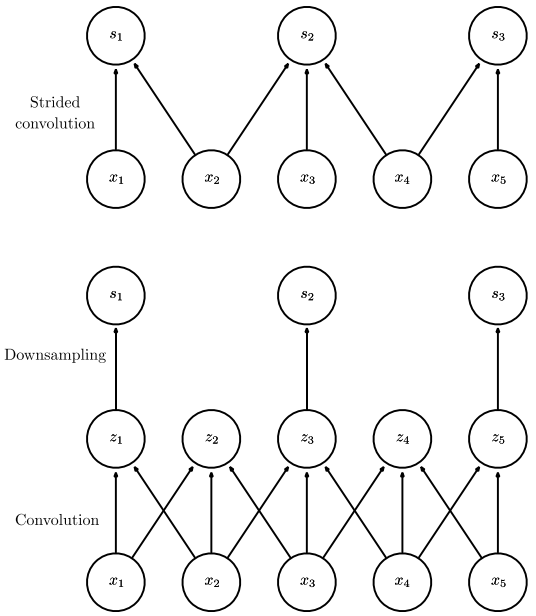
- No hidden layer, input +output layer
  - Input: $32 \times 32 \times 3$ image -> stretch to 3072×1
  - Output: 10 neurons
  - The number of weights: 3072*10

Hyper-parameters of Conv Layers

- Depth K: number of filters for each receptive field
- Spatial Extent F: receptive field
- Zero-Padding P: zeros around the input
- Stride S: number of pixels the filter move at a time

Variants of Basic Convolution Function

- Convolution with stride S to reduce complexity
- Conv Layer ($S > 1$) is equivalent to:
  Conv Layer ($S = 1$) + down-sampling



Variants of Basic Convolution Function

Use zero paddings to control spatial extents

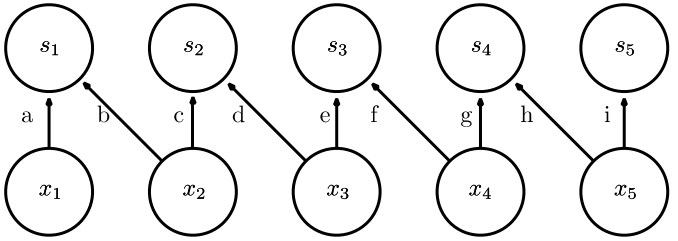
- **Valid convolution**: no padding, shrinking spatial domain
  - If input is $m \times m$ and kernel is $k \times k$, then output is $(m - k + 1) \times (m - k + 1)$
- **Same convolution**: add enough padding, no shrinkage, but pixels near the border has small influence
  - If input is $m \times m$ and kernel is $k \times k$, then output is $m \times m$
- **Full convolution**: add enough padding, each pixels visited k times
  - If input is $m \times m$ and kernel is $k \times k$, then output is$(m + k -$

Variants of Basic Convolution Function

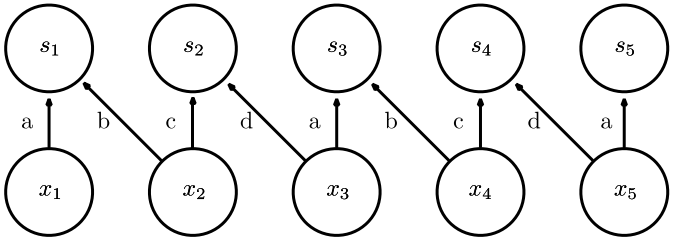
- **Unshared convolution**: local connectivity + no parameter sharing
- **Local convolution in depth**: each output channel only connects to parts of the input channels.
- **Tiled convolution**: learn a set of kernels that we rotate through as we move through space

Variants of Basic Convolution Function

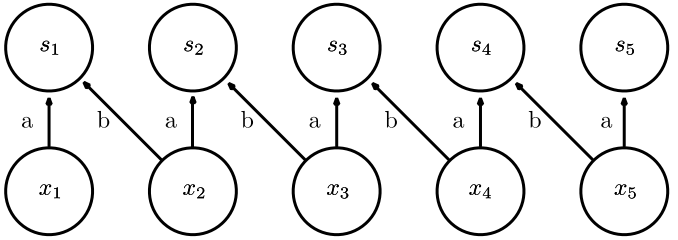
Local connection
(no sharing)



Tiled convolution
(cycle between groups of
shared parameters)



Convolution
(one group shared every where)

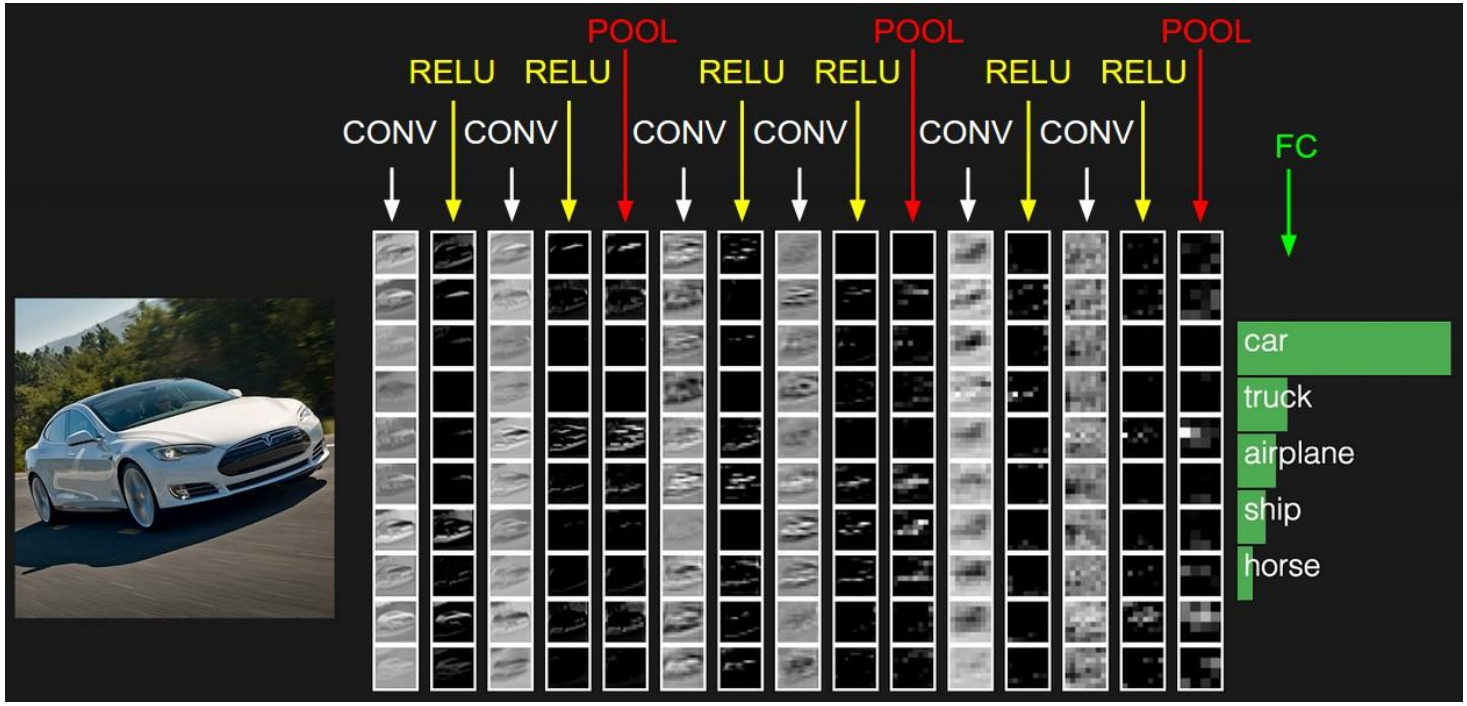


Summary of Conv Layer Parameters

- Input image size: $W_1 \times H_1 \times D_1$
- Require four hyper-parameters
  - Number of filters K, filter spatial extent F, the stride S, zero Padding P
- Output image size: $W_2 \times H_2 \times D_2$ where
  - $W_2 = (W_1 - F + 2P)/S+1$
  - $H_2 = (H_1 - F + 2P)/S+1$
  - $D_2 = K$
- With parameter sharing, $F \times F \times D_1$ weights per filter, total: $(F \times F \times D_1) \times K$

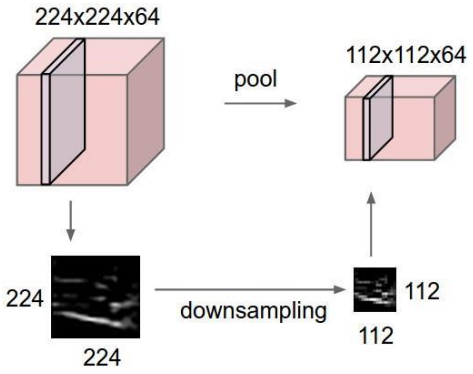
Tips: input layers divisible by 2 many times; small F (3 or 5); most common stride of 1

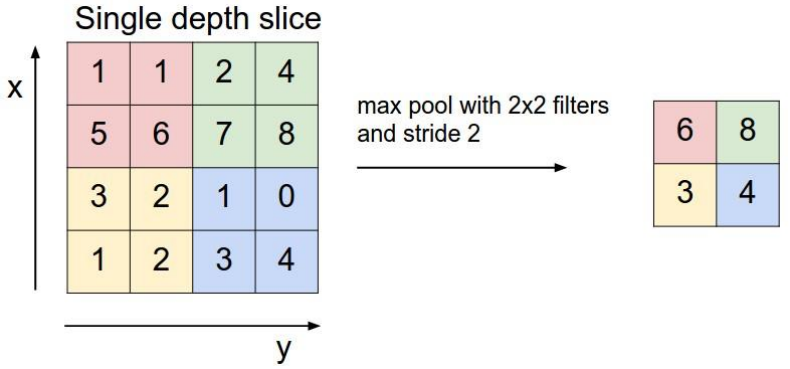
Architectures



Pooling Layer

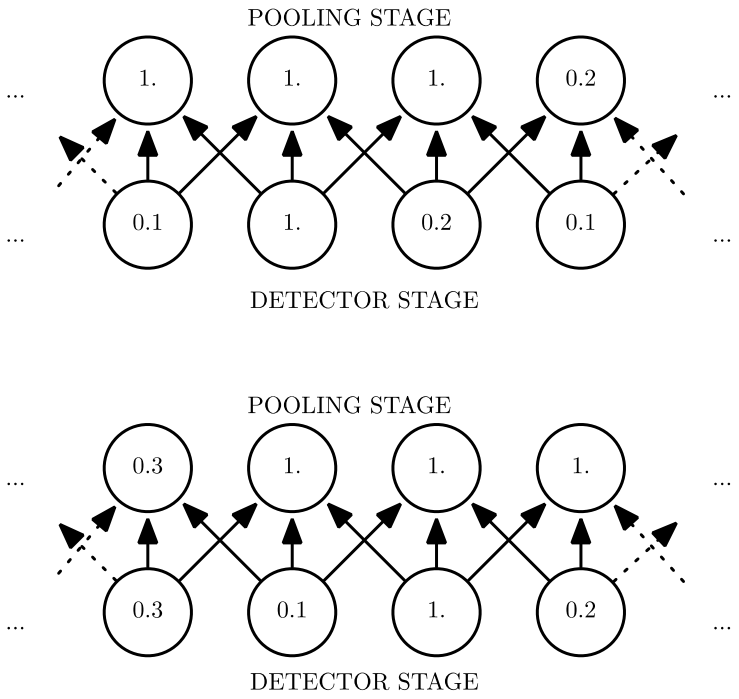
- A pooling layer
  - replace the output with a *summary statistic* of the nearby points
  - help to make the representation *invariant* to small translation





Max Pooling

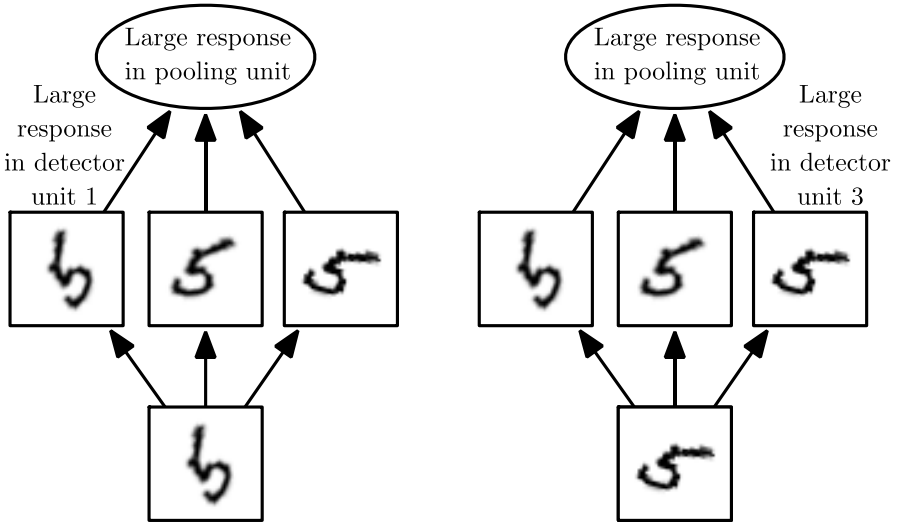
- Learn invariance to shifting
- Top: a view of the middle of the output of a Conv Layer
- Bottom: same network after the input shifted to the right by one pixel



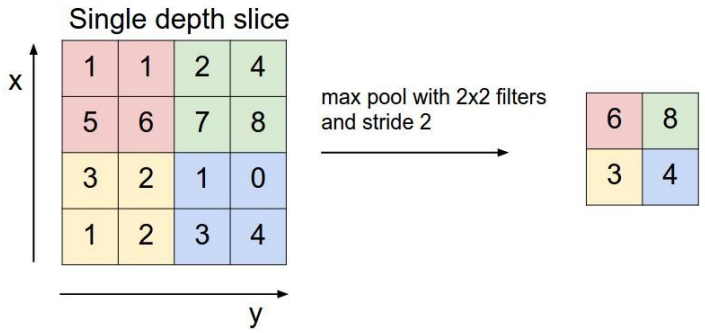
Max Pooling

- Learn invariance to rotation
- All three filters are intended to detect a hand written 5
- Max pooling unit has a large activation regardless of which detector unit was activated.

- Average pooling is used historically, but demonstrated to perform worse in practice



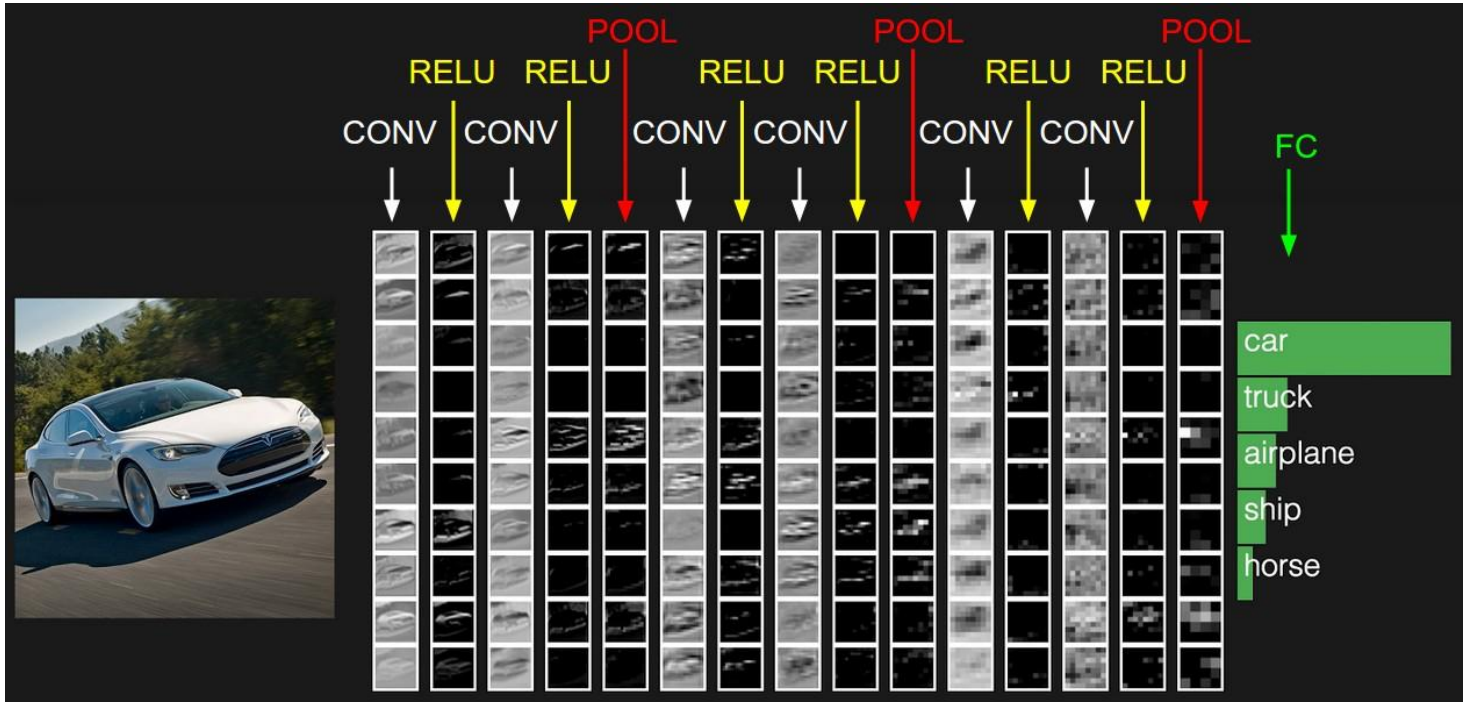
Pooling Layer



- Input size: $W_1 \times H_1 \times D_1$
- Hyper-parameters:
  - Spatial Extent F: receptive field, Stride S: number of pixels the filter move at a time
- Output size: $W_2 \times H_2 \times D_2$ where:
  - $W2 = (W1 - F)/S + 1$
  - $H_2 = \frac{H_1 - F}{S} + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layer

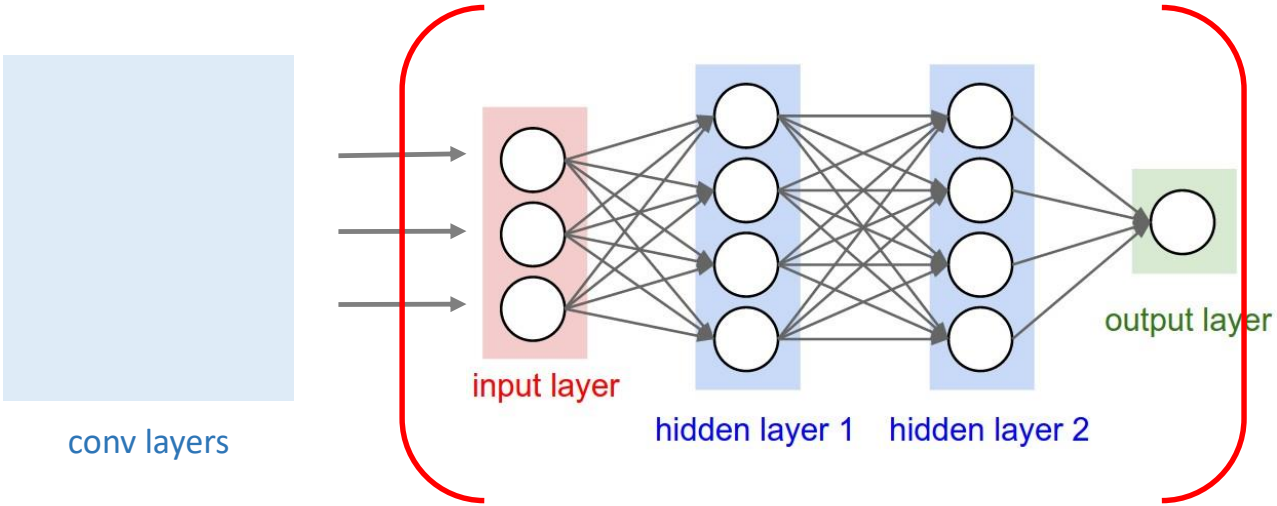
Tips: most common to see F and S of 2

Architectures

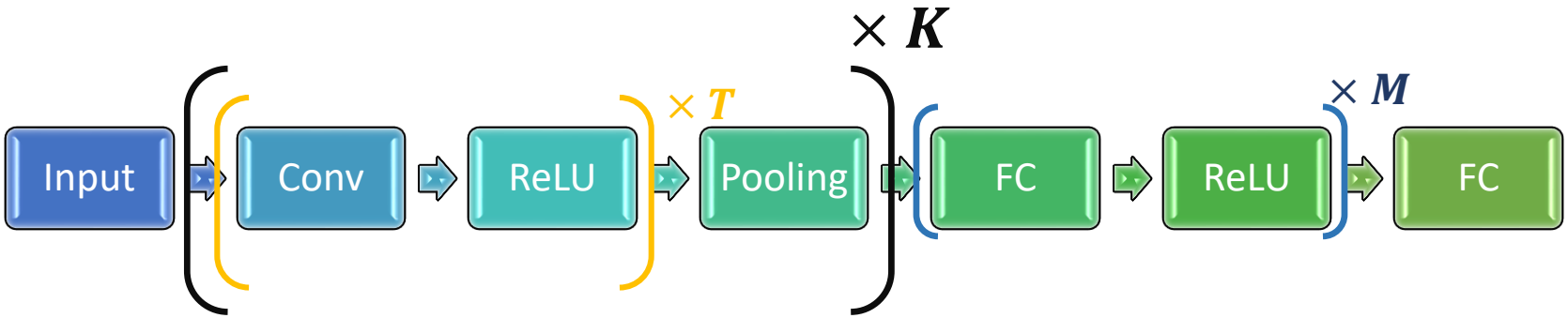


Fully Connected Layers

- Has full connection to all the neurons in the previous layers
- Essentially the same as what we discussed before.



Typical Architectures



Generally, $T \leq 3, K \geq 0, M < 3$

Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3431-3440).

Training CNN

- Similar with fully connected feedforward NN
  - We apply (mini-batch) stochastic gradient descent
  - Max pooling: loss J; i, j, k for depth, width, height

$$y_{ijk} = h_{ijk}(x) \stackrel{\text{def}}{=} \max_{pq} x_{ij+pk+q}$$

then

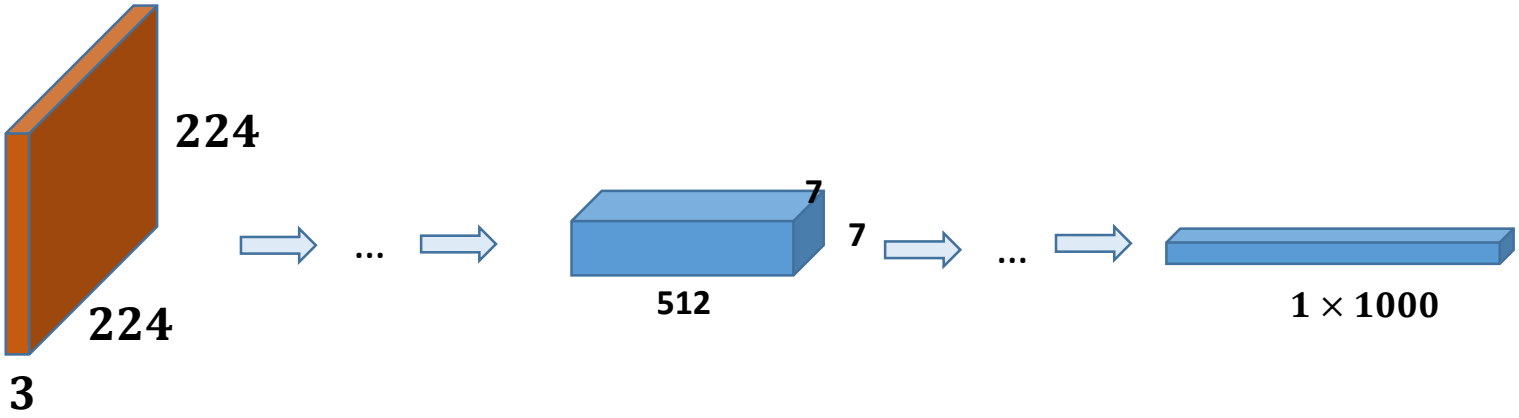
$$\frac{\partial J}{\partial x_{ij'k'}} = \sum_{jk: y_{ijk} = x_{ij'k'}} \frac{\partial J}{\partial y_{ijk}}$$

Conversion between CONV and FC Layers

- CONV => FC layers
  - Conv layers are FC layers with some weights constrained to be zero.
- FC => CONV Layers "Fully convolutional network"
  - E.g.: an FC layer with $K = 4096$ is equivalent to a conv layer with $P = 0, S = 1, K = 4096$

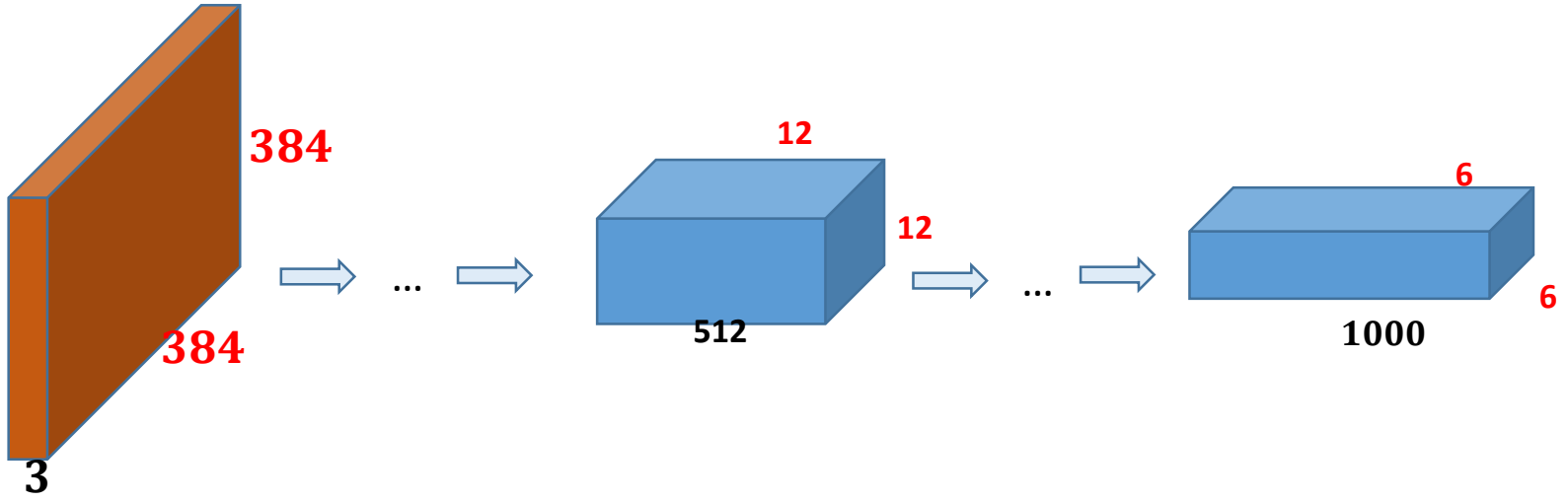
Conversion between Conv and FC Layers

- Cast a classification net to a fully convolutional network
- original data :$224 \times 224 \times 3$



Conversion between Conv and FC Layers

- Cast a classification net to a fully convolutional network
- Forward on expanded data $384 \times 384 \times 3$



- Efficiency: amortizing the computation of overlapping receptive fields.

Visualization

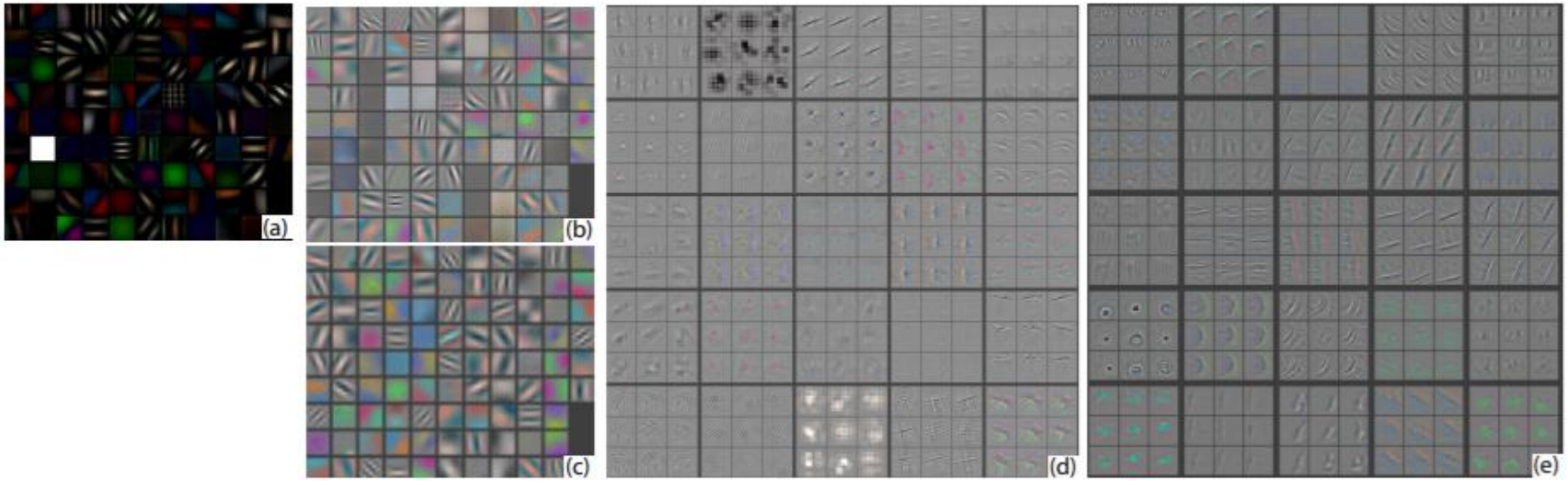


Figure 6. (a): 1st layer features without feature scale clipping. Note that one feature dominates. (b): 1st layer features from (Krizhevsky et al., 2012). (c): Our 1st layer features. The smaller stride (2 vs 4) and filter size (7x7 vs 11x11) results in more distinctive features and fewer "dead" features. (d): Visualizations of 2nd layer features from (Krizhevsky et al., 2012). (e): Visualizations of our 2nd layer features. These are cleaner, with no aliasing artifacts that are visible in (d).

Zeiler M D, Fergus R. Visualizing and understanding convolutional networks[C]//European conference on computer vision. Springer, Cham, 2014: 818-833.

Deconvnets: Zeiler, M., Taylor, G., Fergus, R.: Adaptive deconvolutional networks for mid and high level feature learning. In: ICCV (2011)

Visualization

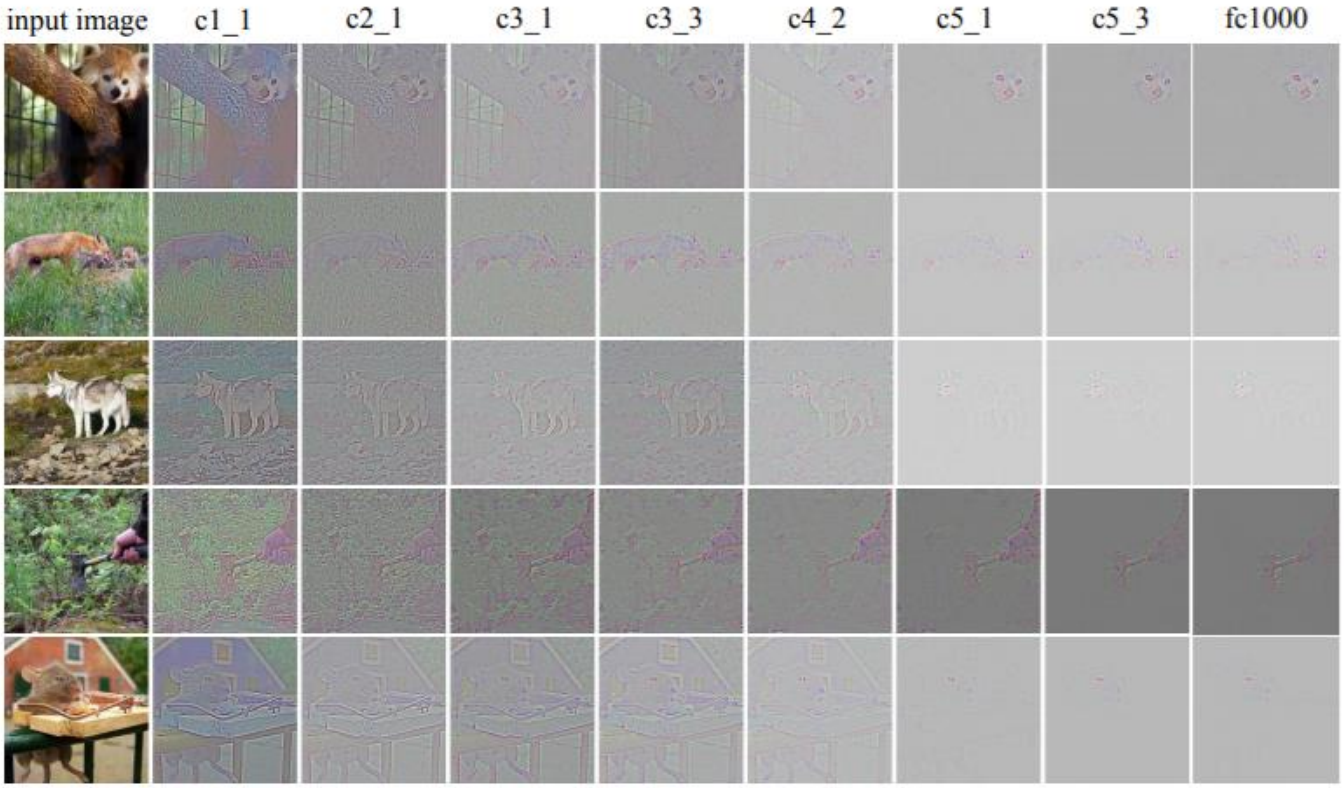


Figure 4. Visualization of visual information processing through different layers. The first column shows five input images, the following 7 columns show the reconstruction results of every two convolutional layers of VGG-16, and the last column shows the reconstruction result of last fully-connect layer. From top to bottom, the images are from *lesser panda, kit fox, Siberian husky, hatchet* and *mousetrap* respectively. (Best viewed electronically)

Yu W, Yang K, Bai Y, et al. Visualizing and comparing AlexNet and VGG using deconvolutional layers[C]//Proceedings of the 33 rd International Conference on Machine Learning. 2016.

A Quick Summary

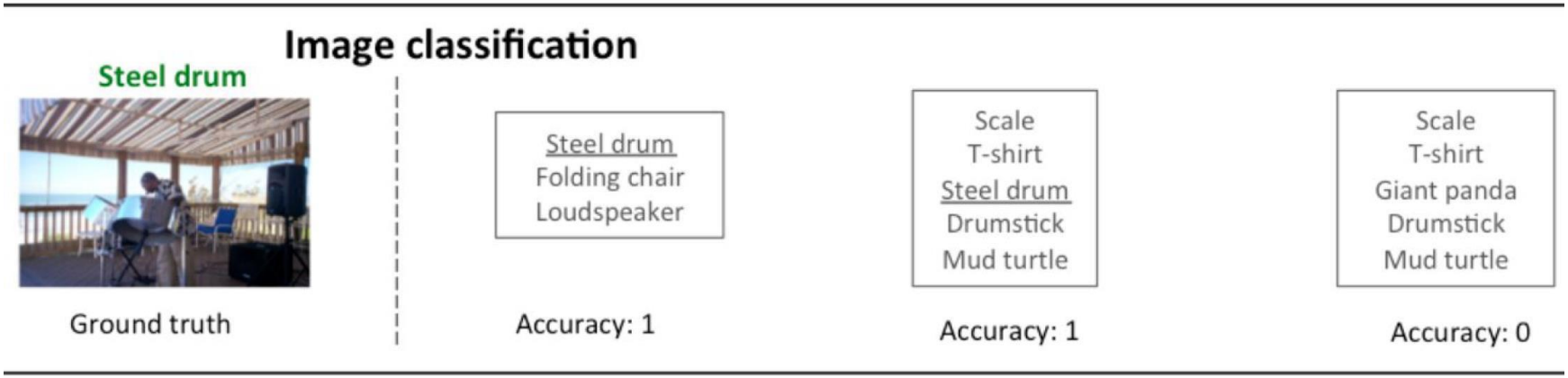
- CNN stack CONV, POOL, FC layers
- Trend towards smaller filters and deeper architectures
- Trend towards getting rid of POOL/FC layers

Overview

- Image processing
- Convolutional neural networks
  - Convolution and pooling
  - CNN Architectures
  - ConvNets outside image classification

ILSVRC Challenge Evaluation for Classification

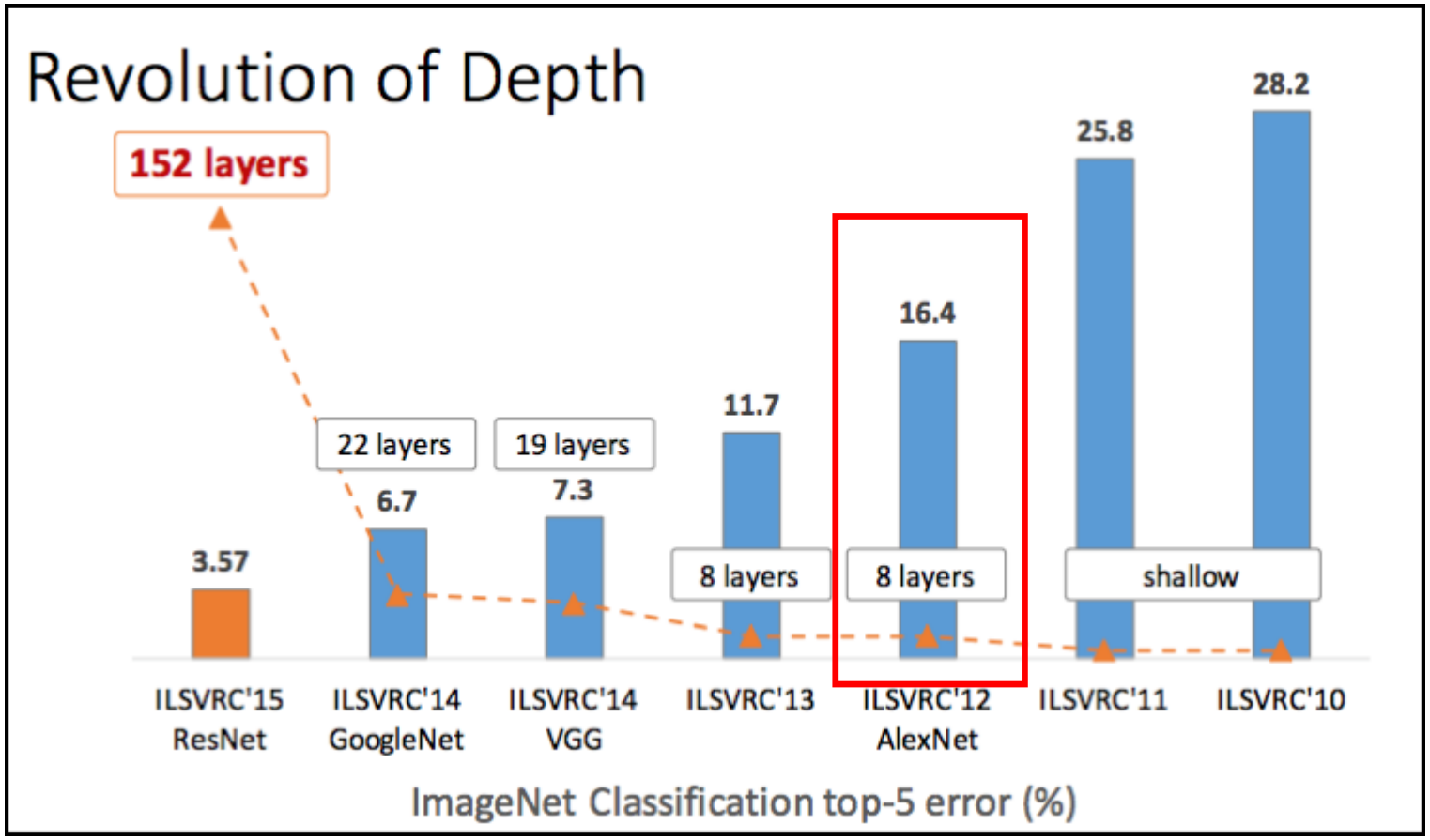
- Top 5 error rate:
- You get 5 guesses to get the correct label



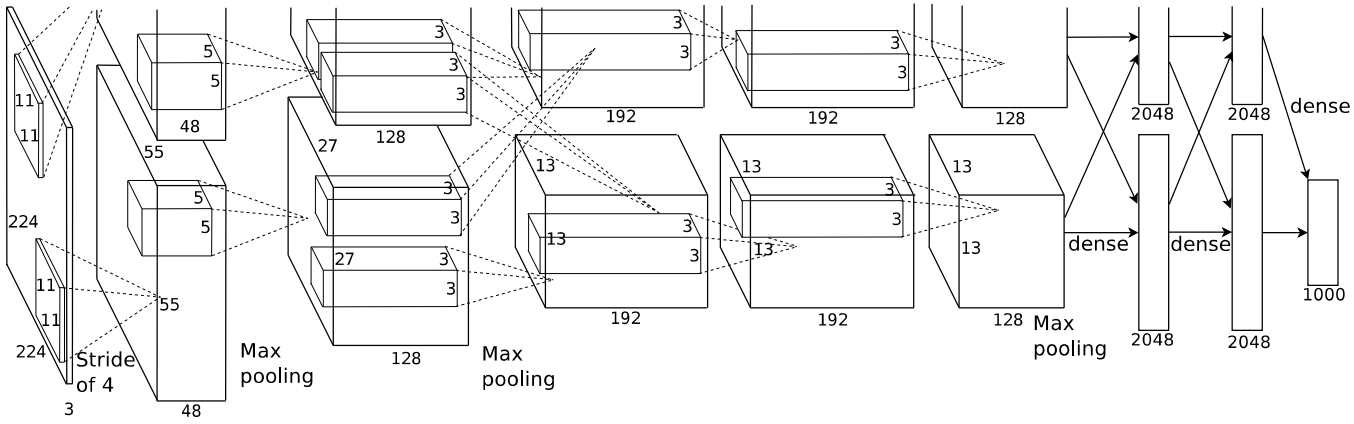
CNN Architectures

- **AlexNet (2012) 15.4%** (8 layers 61 million parameters)
- **ZFNet (2013) 15.4% to 11.2% (**8 layers. More filters. Denser stride.)
- **VGGNet (2014): 11.2% to 7.3%**
  - Beautifully uniform:
    3x3 conv, stride 1, pad 1, 2x2 max pool
  - 16 layers
  - 138 million parameters
- **GoogLeNet (2014): 11.2% to 6.7%**
  - Inception modules
  - 22 layers
  - 5 million parameters
  - (throw away fully connected layers)
- **ResNet (2015): 6.7% to 3.57%**
  - More layers = better performance
  - 152 layers

CNN Architectures



AlexNet



Input: 224*224*3

First layer: (CONV1): 96 11*11 filters with Stride 4, output=55*55*96 Parameters: 96*11*11*3

(POOL1): 3*3 filters with Stride 2, output=27*27*96 Parameters: 0

(Norm1): Normalization

Second layer: (CONV2): 256 5*5 filters with Stride 1, pad 2, output=27*27*256 Parameters: 256*5*5*96

(POOL2): 3*3 filters with Stride 2, output=13*13*256

(Norm2): Normalization

Third layer: (CONV3): 384 3*3 filters with Stride 1, pad 1, output=13*13*384 Parameters: 384*3*3*256

Fourth layer: (CONV4): 384 3*3 filters with Stride 1, pad 1, output=13*13*384 Parameters: 384*3*3*384

Fifth layer: (CONV5): 256 3*3 filters with Stride 1, pad 1, output=13*13*256 Parameters: 256*3*3*384

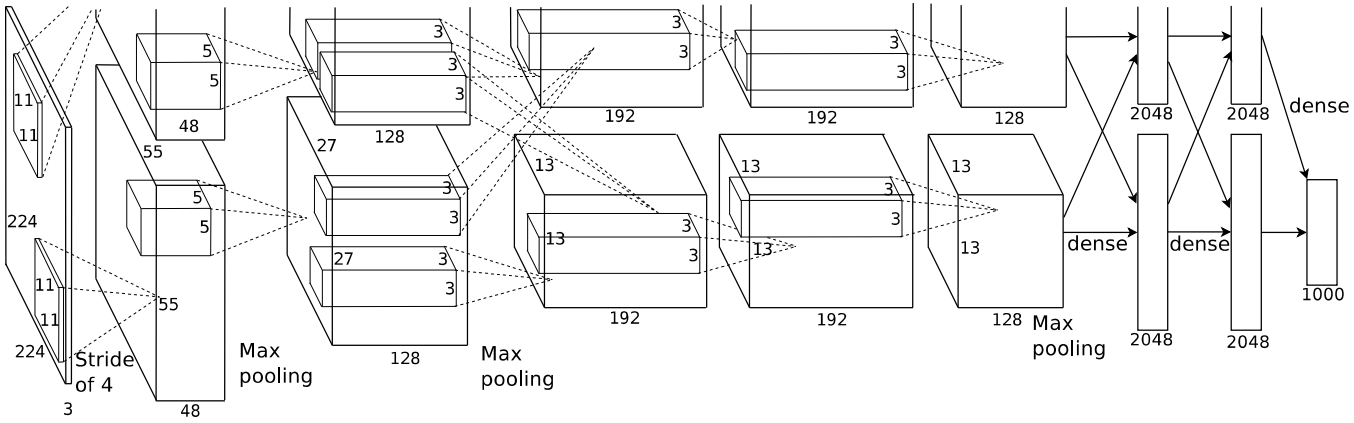
(POOL3): 3*3 filters with Stride 2, output=6*6*256

Sixth layer: (FC6): 4096 neurons Parameters: 6*6*256*4096

Seventh layer: (FC7): 4096 neurons Parameters: 4096*4096

Eighth layer: (FC8): 1000 neurons Parameters: 4096*1000

AlexNet

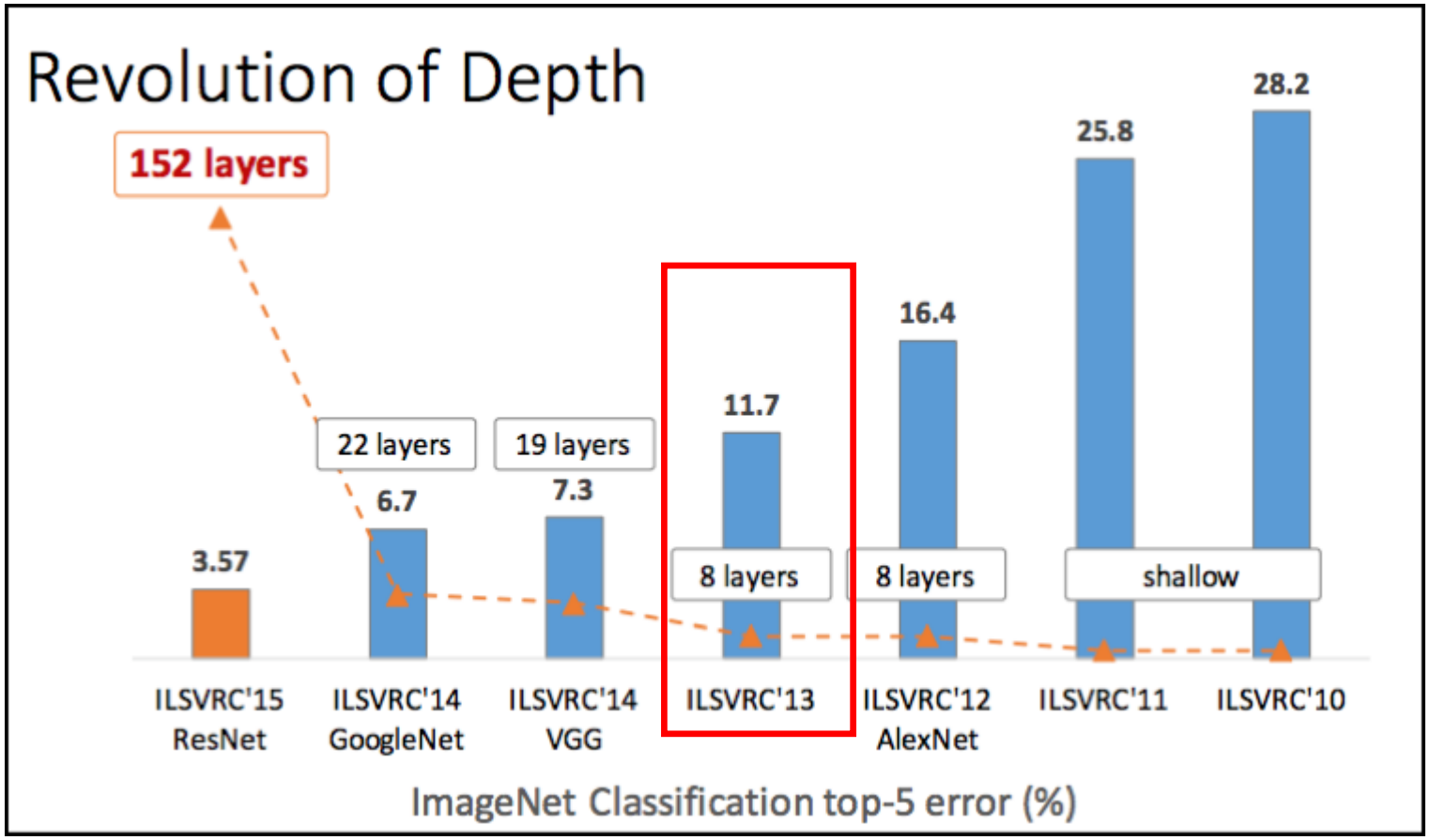


- 8 layers
- 61 million parameters
- ReLU instead of sigmoid
- Local response normalization
  - improved performance by 1.2%
- Data Augmentation
  - Improved performance by 1%+
- Dropout

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

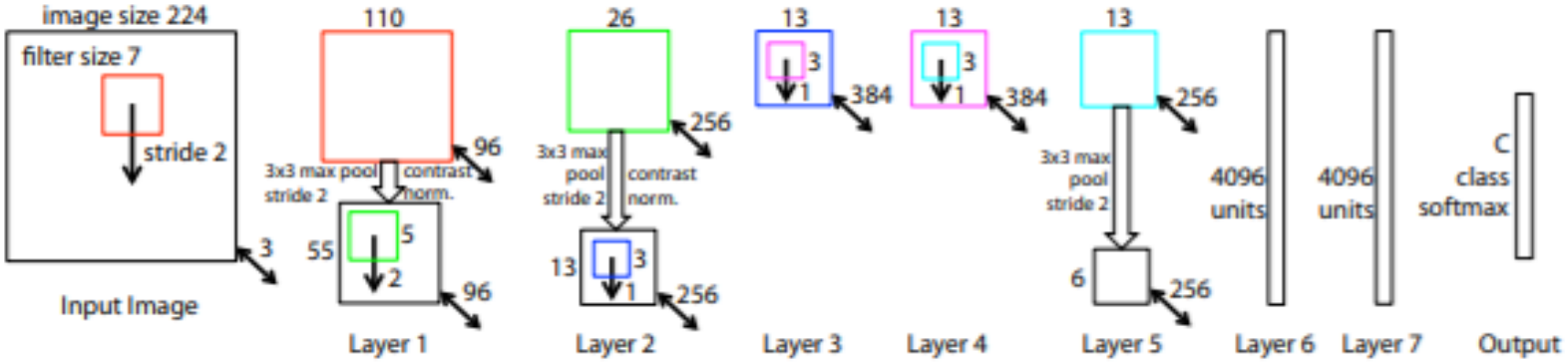
Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105

CNN Architectures



Zeiler, M. D., & Fergus, R. (2014, September). Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818-833). Springer, Cham.

ZFNet

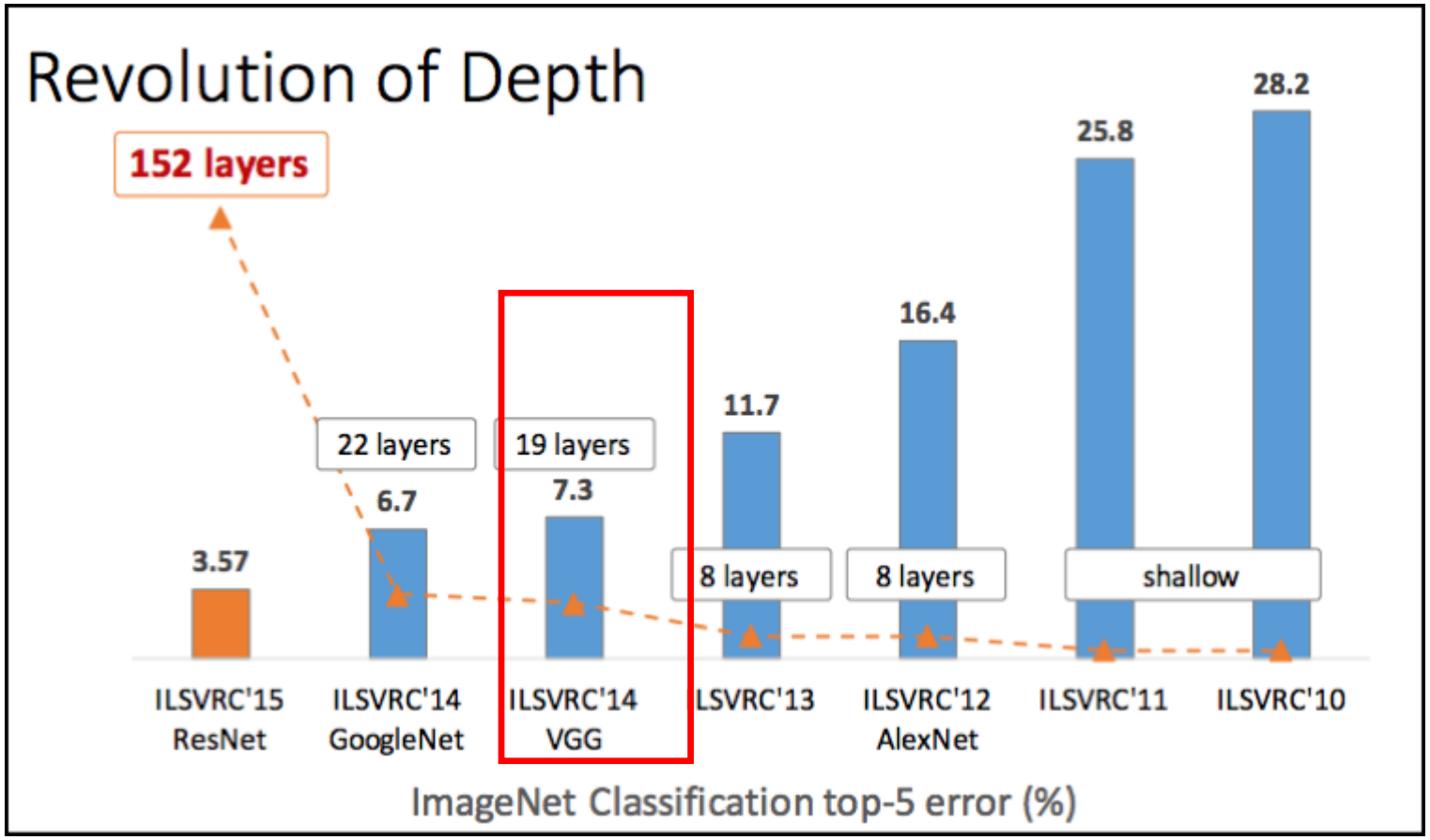


Differences with AlexNet:
The first CONV layer: 96 7*7 filters with Stride 2, output: 110*110*96

Large size of filters proved to be skipping a lot of relevant information, especially for the first layer

Error %	Val Top-1	Val Top-5	Test Top-5
Gunji *et al.* [12]	-	-	26.2
DeCAF [7]	-	-	19.2
Krizhevsky *et al.* [18], 1 convnet	40.7	18.2	−−
Krizhevsky *et al.* [18], 5 convnets	38.1	16.4	16.4
Krizhevsky *et al.* *[18], 1 convnets	39.0	16.6	−−
Krizhevsky *et al.* *[18], 7 convnets	36.7	15.4	15.3
Our replication of Krizhevsky *et al.* , 1 convnet	40.5	18.1	−−
1 convnet as per Fig. 3	38.4	16.5	−−
5 convnets as per Fig. 3 – (a)	36.7	15.3	15.3
1 convnet as per Fig. 3 but with layers 3,4,5: 512,1024,512 maps – (b)	37.5	16.0	16.1
6 convnets, (a) & (b) combined	36.0	14.7	14.8
Howard [15]	-	-	13.5
Clarifai [28]	-	-	11.7

CNN Architectures



Zeiler, M. D., & Fergus, R. (2014, September). Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818-833). Springer, Cham.

VGG Net

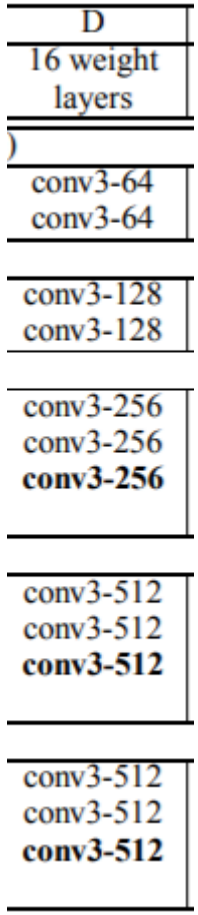
- **11.2% to 7.3%**
- Beautifully uniform: 3x3 conv, stride 1, pad 1, and 2x2 max pool with stride 2
- 16 layers
- 138 million parameters

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					

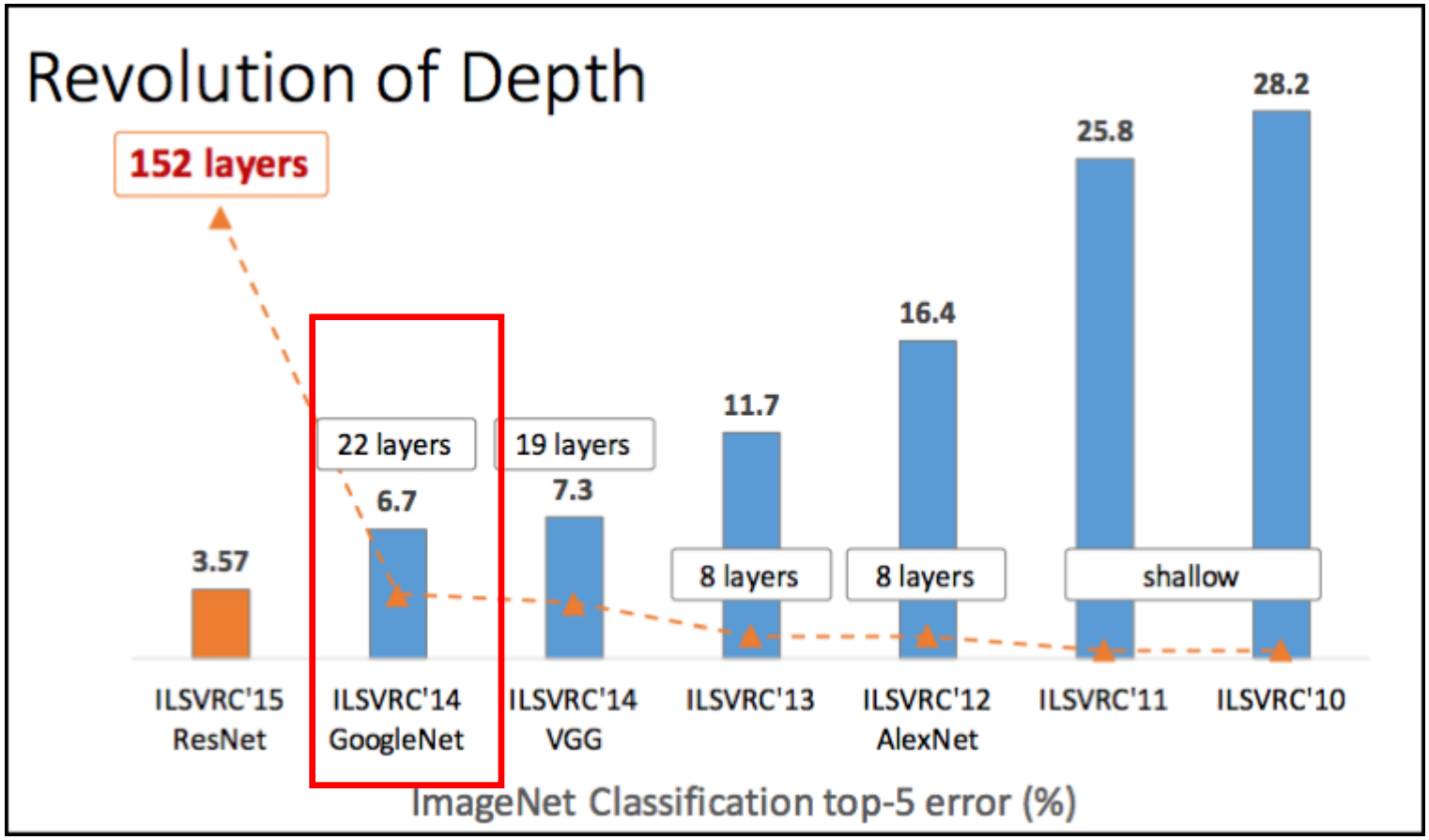
Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

VGG Net

- Why use smaller filters? (3*3)
  - Stack of two 3*3 has the same effective receptive field as one 5*5
  - Stack of three 3*3 ~ one 7*7
  - Deeper, more non-linearities
  - Fewer parameters
- ILSVRC2014 2nd in classification, 1st in localization
- The full layer features generalize well to other tasks



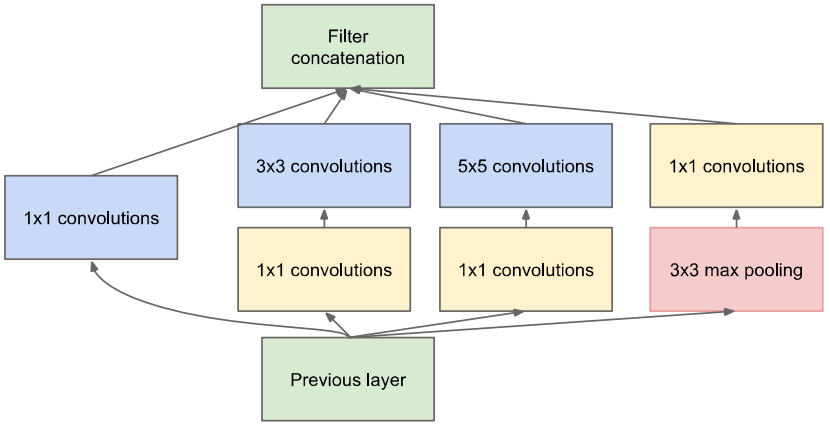
CNN Architectures

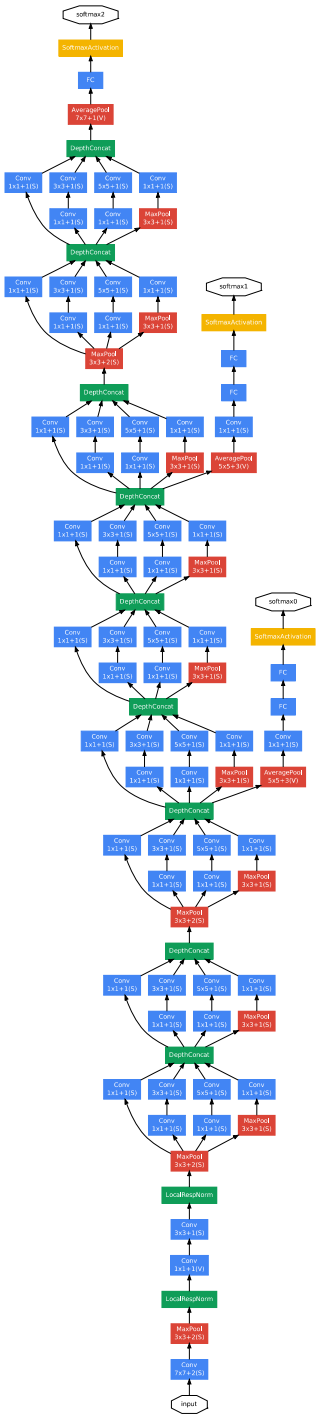


Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015, June). Going deeper with convolutions. Cvpr.

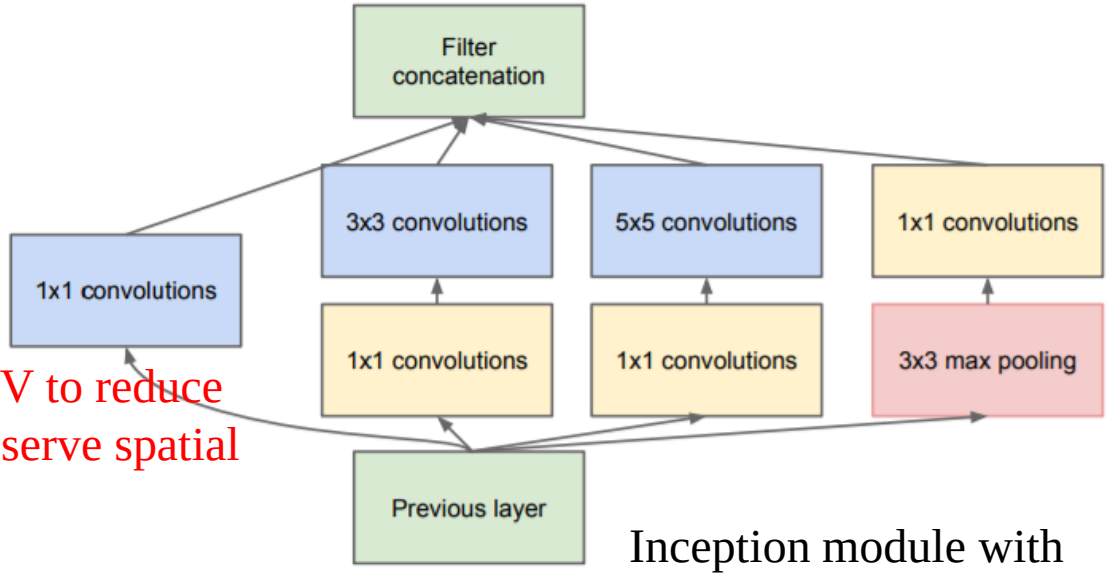
GoogLeNet

- 22 layers, 5 million parameters
- No FC layers
- 9 Inception modules
  - Operated at the fourth layer (memory efficiency)





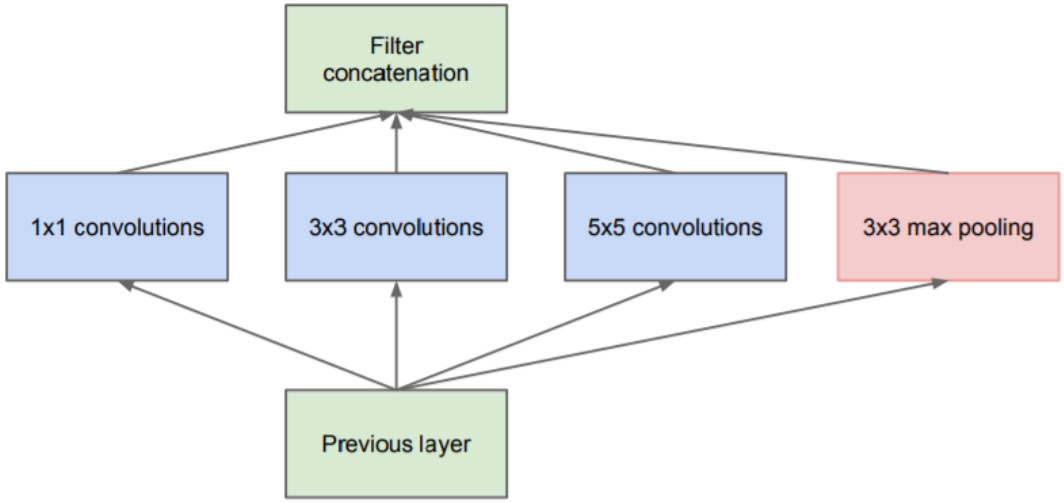
GoogLeNet



Use 1*1 CONV to reduce depth, and preserve spatial dimensions

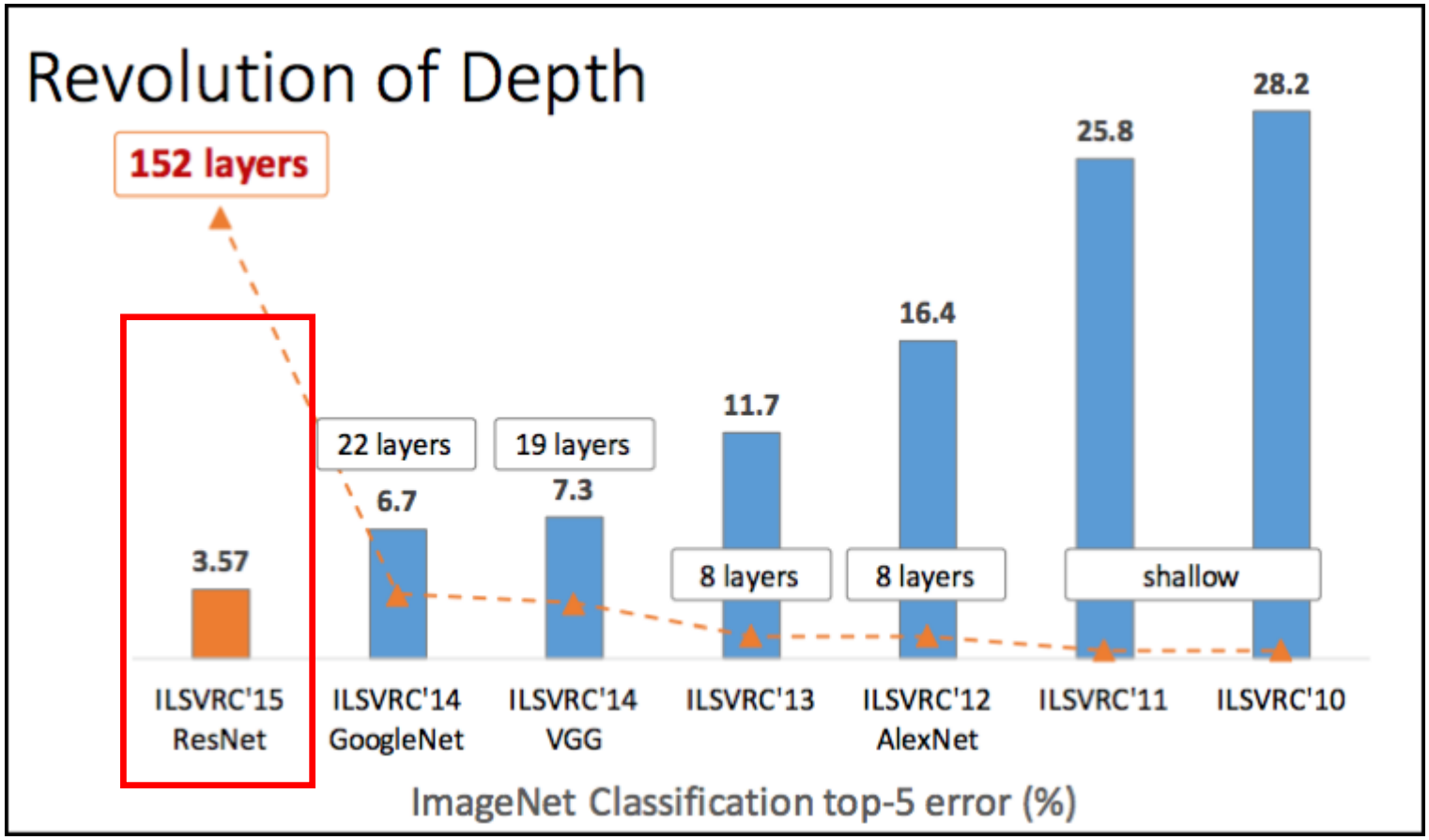
Inception module with dimension reduction

Computational complexity



Naïve Inception model

CNN Architectures



He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

ResNet

- Very deep networks using residual connections
- 152-layer model
  - The winners in all classification and detection competitions in 2015

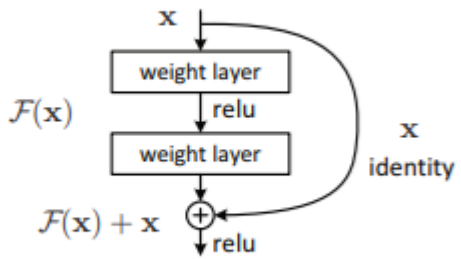
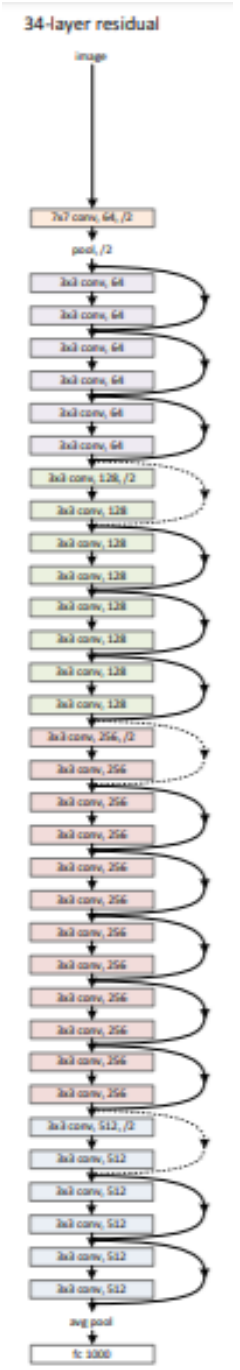


Figure 2. Residual learning: a building block.



ResNet

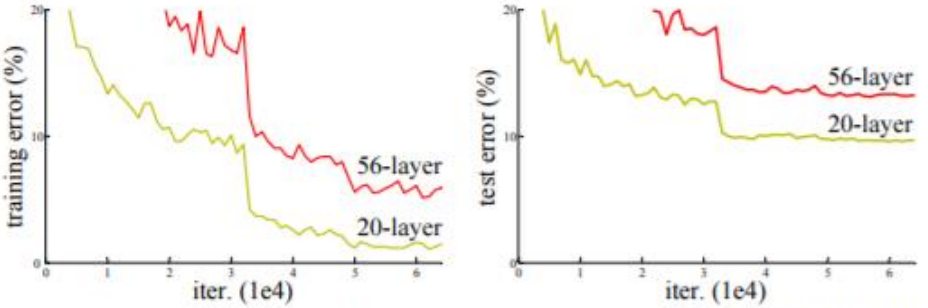
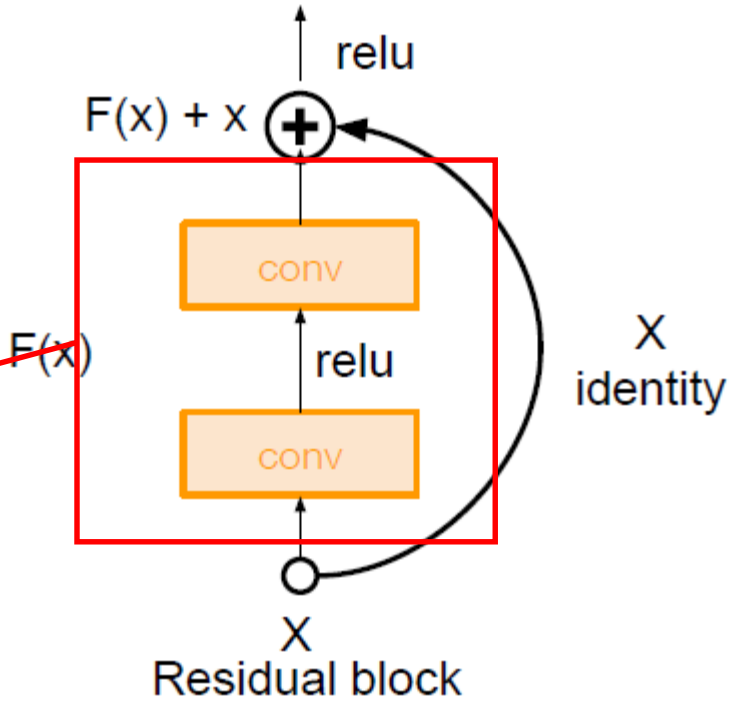


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

- Performance degradation problem caused not by overfitting, but the increased difficulty to optimize
  - Deeper models are harder to optimize
- solution



H(X)=F(X)+X;
Use these layers to fit residual F(X)=H(X)-X instead of H(x) directly

ResNet

- Stack residual blocks
- For deeper networks, use "bottleneck" layer to improve efficiency (similar to GoogleNet)

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

model	top-1 err.	top-5 err.
VGG-16 [41]	28.07	9.33
GoogLeNet [44]	-	9.15
PReLU-net [13]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

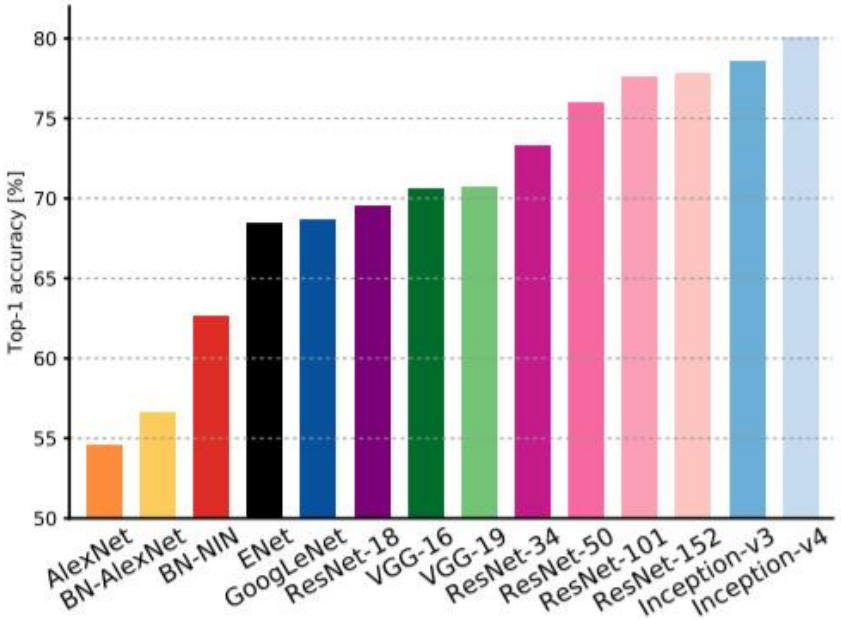
method	top-5 err. (**test**)
VGG [41] (ILSVRC'14)	7.32
GoogLeNet [44] (ILSVRC'14)	6.66
VGG [41] (v5)	6.8
PReLU-net [13]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	**3.57**

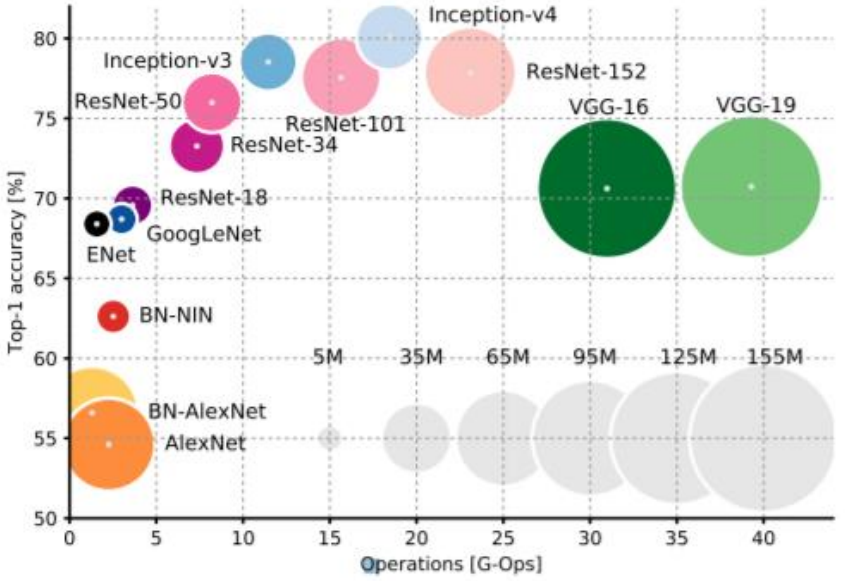
Veit, A., Wilber, M. J., & Belongie, S. (2016). Residual networks behave like ensembles of relatively shallow networks. In *Advances in neural information processing systems* (pp. 550-558). http://papers.nips.cc/paper/6556-residual-networks-behave-like-ensembles-of-relatively-shallow-networks.pdf

ResNet

- Training in practice
  - Batch Normalization after every CONV layer
  - Xavier/2 initialization
  - SGD + Momentum (0.9)
  - Learning rate: 0.1, divided by 10 when validation error plateaus
  - Mini-batch size 256
  - Weight decay of 1e-5
  - No dropout used

Performance and Complexity Comparisons

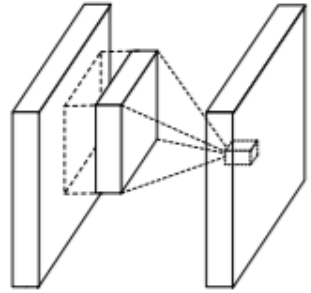




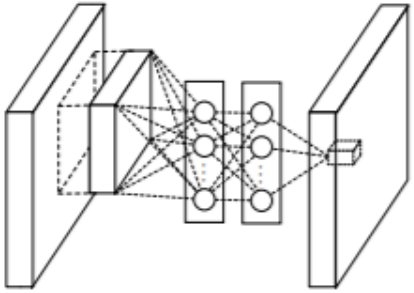
Canziani, A., Paszke, A., & Culurciello, E. (2016). An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*.

More Architectures

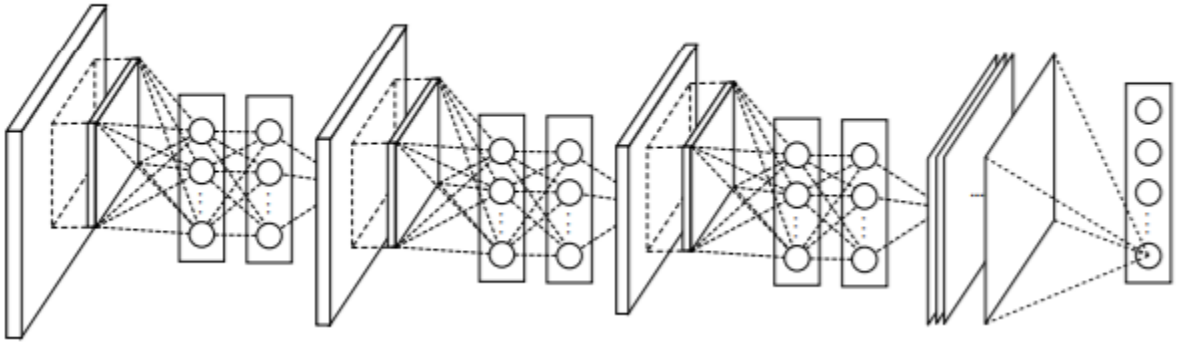
- Network in Network (NiN)
  - A small MLP to CONV the receptive field



(a) Linear convolution layer



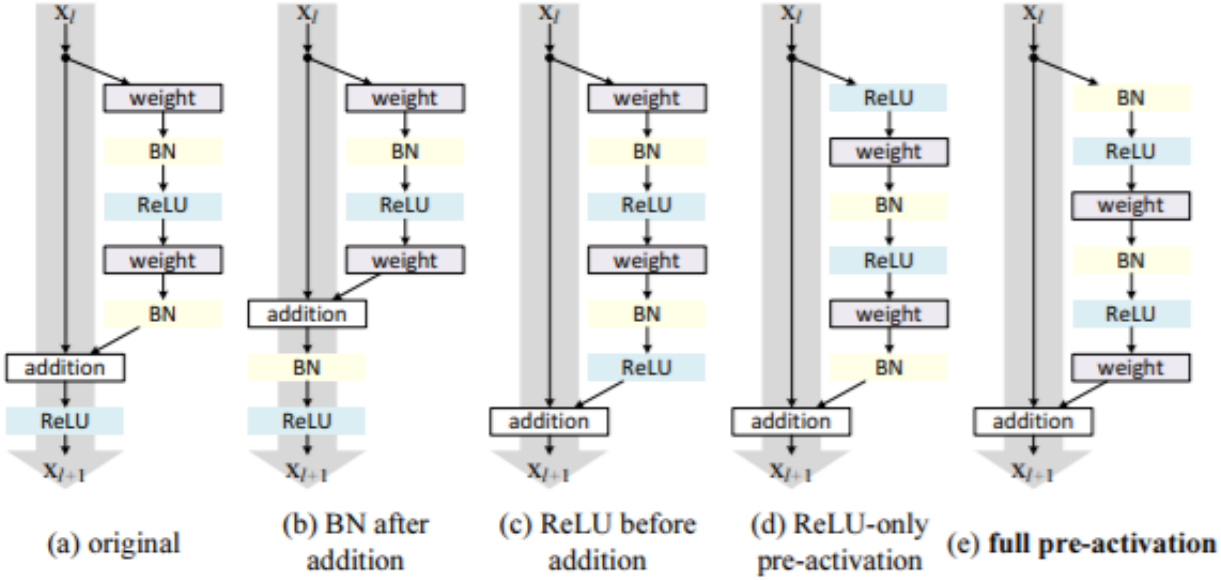
(b) Mlpconv layer



Lin, M., Chen, Q., & Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*.

Improving ResNet- [He et. al, 2016]

- Ease of optimization
- Reduce over-fitting



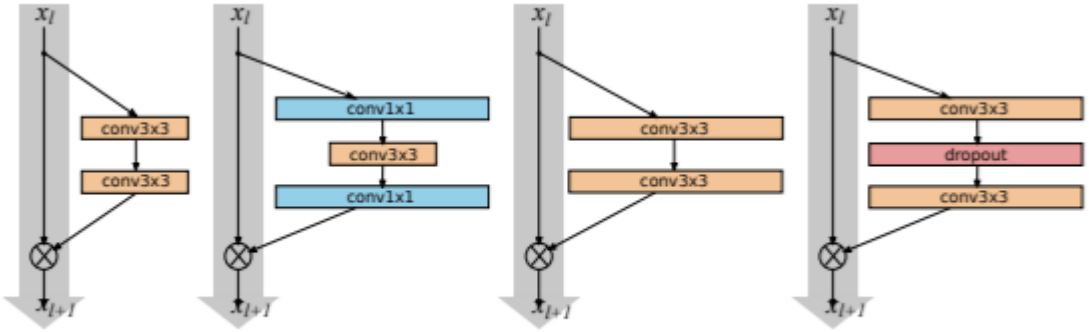
CIFAR-10	error (%)
NIN [15]	8.81
DSN [16]	8.22
FitNet [17]	8.39
Highway [7]	7.72
All-CNN [14]	7.25
ELU [12]	6.55
FitResNet, LSUV [18]	5.84
ResNet-110 [1] (1.7M)	6.61
ResNet-1202 [1] (19.4M)	7.93
ResNet-164 [ours] (1.7M)	5.46
ResNet-1001 [ours] (10.2M)	4.92 (4.89±0.14)
ResNet-1001 [ours] (10.2M)†	**4.62** (4.69±0.20)

CIFAR-100	error (%)
NIN [15]	35.68
DSN [16]	34.57
FitNet [17]	35.04
Highway [7]	32.39
All-CNN [14]	33.71
ELU [12]	24.28
FitNet, LSUV [18]	27.66
ResNet-164 [1] (1.7M)	25.16
ResNet-1001 [1] (10.2M)	27.82
ResNet-164 [ours] (1.7M)	24.33
ResNet-1001 [ours] (10.2M)	**22.71** (22.68±0.22)

He, K., Zhang, X., Ren, S., & Sun, J. (2016, October). Identity mappings in deep residual networks. In *European Conference on Computer Vision* (pp. 630-645). Springer, Cham.

Improving ResNet- [Zagoruyko et al. 2016]

- Wide Residual Networks
  - Motivated by diminishing feature reuse
  - Argue that residuals are the important factor, not depth
  - 50 times less layers and being more than 2 times faster



(a) basic (b) bottleneck (c) basic-wide (d) wide-dropout

group name	output size	block type = $B(3,3)$
conv1	32×32	[3×3, 16]
conv2	32×32	$\begin{bmatrix} 3\times3, 16\times k \\ 3\times3, 16\times k \end{bmatrix} \times N$
conv3	16×16	$\begin{bmatrix} 3\times3, 32\times k \\ 3\times3, 32\times k \end{bmatrix} \times N$
conv4	8×8	$\begin{bmatrix} 3\times3, 64\times k \\ 3\times3, 64\times k \end{bmatrix} \times N$
avg-pool	1×1	[8 × 8]

Zagoruyko, S., & Komodakis, N. (2016). Wide residual networks. *arXiv preprint arXiv:1605.07146*.

Improving ResNet- Inception-ResNet

- Varied Inception version
- Combine inception model with Residual learning

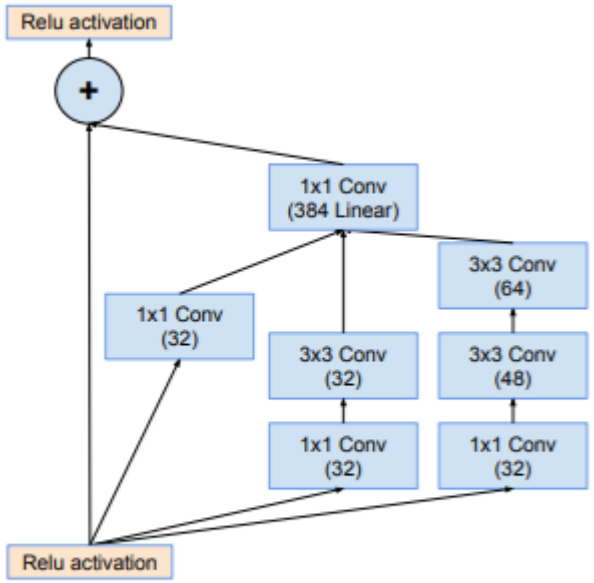


Figure 16. The schema for 35 × 35 grid (Inception-ResNet-A) module of the Inception-ResNet-v2 network.

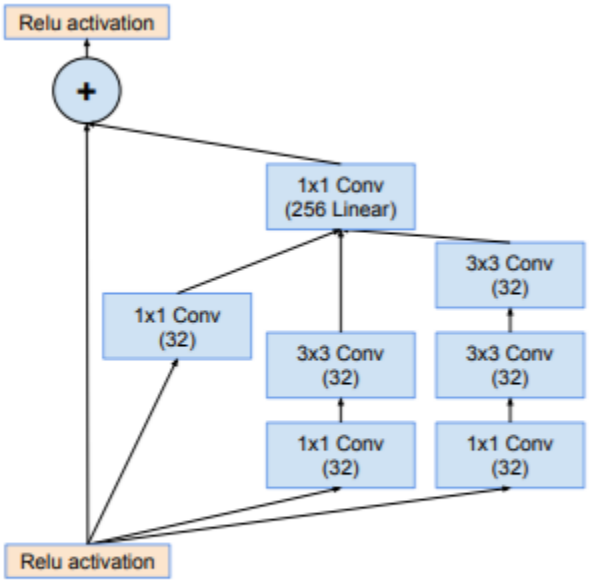
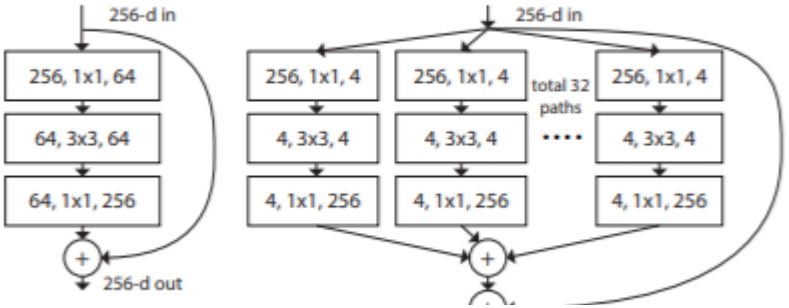


Figure 10. The schema for 35 × 35 grid (Inception-ResNet-A) module of Inception-ResNet-v1 network.

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017, February). Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI* (Vol. 4, p. 12).

Improving ResNet- ResNeXt

- Increasing cardinality is more effective than going deeper or wider



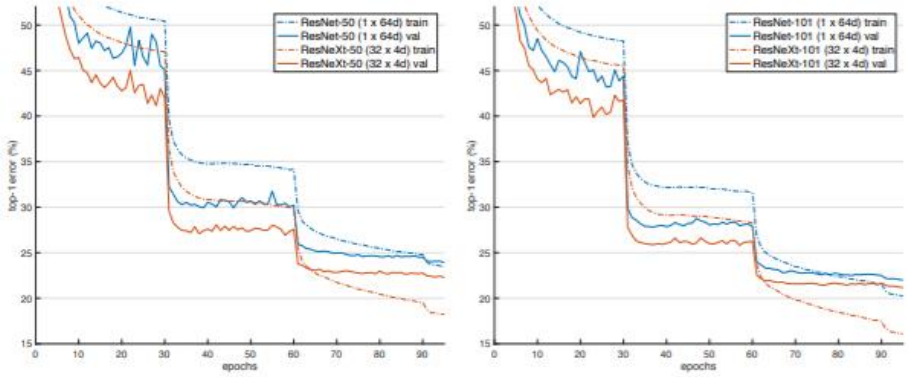


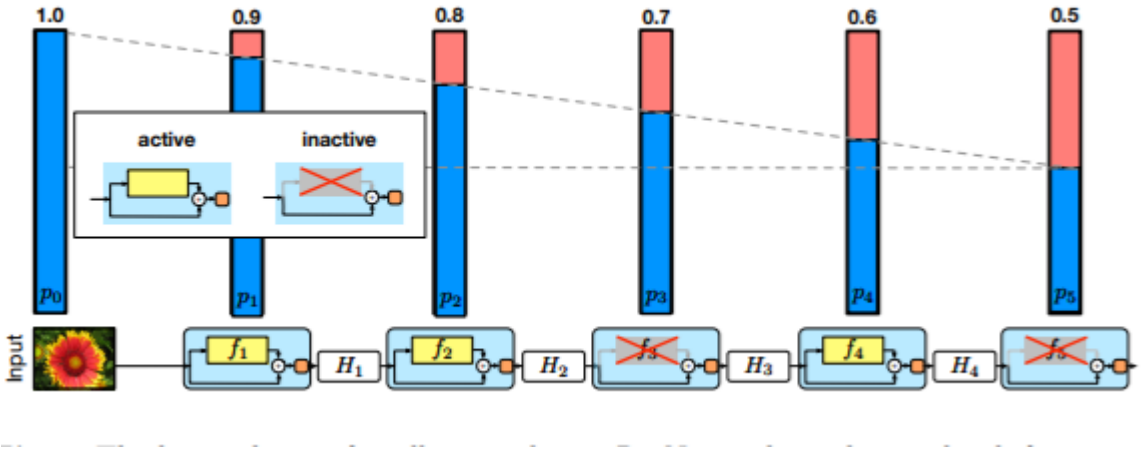
Figure 5. Training curves on ImageNet-1K. (**Left**): ResNet/ResNeXt-50 with preserved complexity (~4.1 billion FLOPs, ~25 million parameters); (**Right**): ResNet/ResNeXt-101 with preserved complexity (~7.8 billion FLOPs, ~44 million parameters).

stage	output	ResNet-50	**ResNeXt-50 (32×4d)**
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2	3×3 max pool, stride 2
		[1×1, 64 3×3, 64 1×1, 256] ×3	[1×1, 128 3×3, 128, C=32 1×1, 256] ×3
conv3	28×28	[1×1, 128 3×3, 128 1×1, 512] ×4	[1×1, 256 3×3, 256, C=32 1×1, 512] ×4
conv4	14×14	[1×1, 256 3×3, 256 1×1, 1024] ×6	[1×1, 512 3×3, 512, C=32 1×1, 1024] ×6
conv5	7×7	[1×1, 512 3×3, 512 1×1, 2048] ×3	[1×1, 1024 3×3, 1024, C=32 1×1, 2048] ×3
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		**25.5**×10^6	**25.0**×10^6
FLOPs		**4.1**×10^9	**4.2**×10^9

Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017, July). Aggregated residual transformations for deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on* (pp. 5987-5995). IEEE..

Improving ResNet- [Huang et al., 2016]

- Vanishing gradient problem in very deeper network
- Random drop some layers in training, but use full network in the testing



Huang, G., Sun, Y., Liu, Z., Sedra, D., & Weinberger, K. Q. (2016, October). Deep networks with stochastic depth. In *European Conference on Computer Vision* (pp. 646-661). Springer, Cham.

DesNet [Huang et al., 2017]

- Dense blocks where each layer is connected to every other layer in feedforward fashion
- Alleviates vanishing gradient, strengthens feature propagation, encourages feature reuse

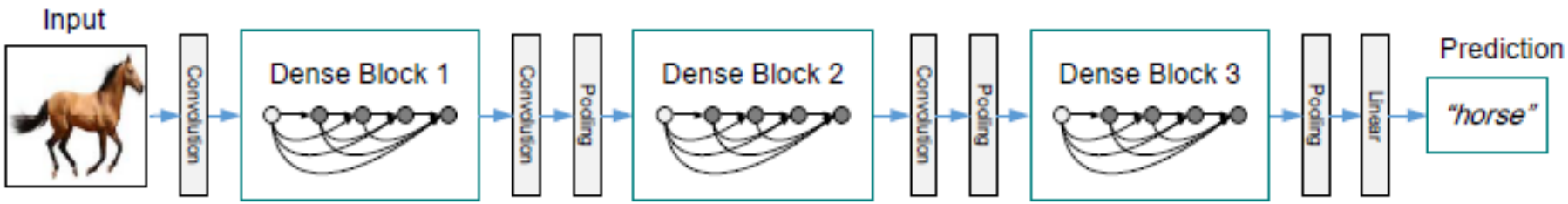
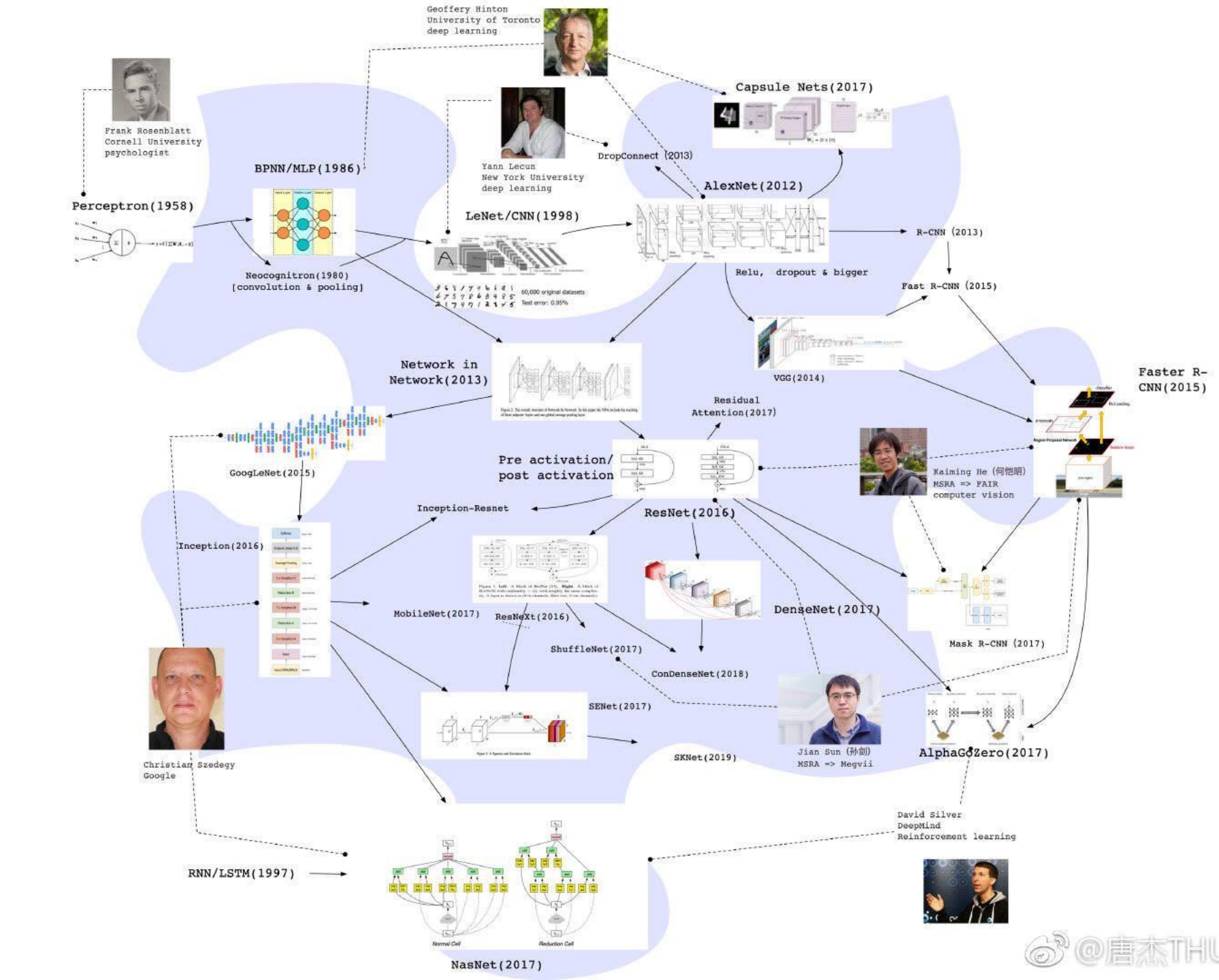


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4700-4708).


Geoffery Hinton
University of Toronto
deep learning
Capsule Nets(2017)
Frank Rosenblatt
Cornell University
psychologist
Yann Lecun
New York University
deep learning
DropConnect (2013)
BPNN/MLP(1986)
AlexNet(2012)
Perceptron(1958)
LeNet/CNN(1998)
R-CNN (2013)
Neocognitron(1980)
[convolution & pooling]
60,000 original datasets
Test error: 0.95%
Relu, dropout & bigger
Fast R-CNN (2015)
VGG(2014)
Faster R-CNN(2015)
Network in Network(2013)
Residual Attention(2017)
GoogLeNet(2015)
Pre activation/ post activation
Kaiming He (何恺明)
MSRA => FAIR
computer vision
Inception-Resnet
ResNet(2016)
Inception(2016)
MobileNet(2017)
ResNeXt(2016)
DenseNet(2017)
Mask R-CNN (2017)
ShuffleNet(2017)
ConDenseNet(2018)
SENet(2017)
SKNet(2019)
Jian Sun (孙剑)
MSRA => Megvii
AlphaGoZero(2017)
Christian Szedegy
Google
David Silver
DeepMind
Reinforcement learning
RNN/LSTM(1997)
NasNet(2017)
@唐杰THU

Summary of CNN Architectures

- VGG, GoogLeNet, ResNet all in wide use, available in model zoos
- ResNet current best default
- Trend towards extremely deep networks
- Significant research centers around design of layer/ skip connections and improving gradient flow
- Even more recent trend towards examining necessity of depth vs. width and residual connections

Transfer Learning Regarding CNN

- Common practice
  - Pretrain a ConvNet on a very large dataset (e.g. ImageNet)
  - Use the ConvNet as the initialization or fixed feature extractor for the task of interest
    - Remove the FC layer, and the rest of ConvNet is treated as the input of classifiers or other tasks

New dataset	Small	large
Similar	Train classifier only	Fine-tune the ConvNet
Different	Use low level features as input	Train a ConvNet from Scratch

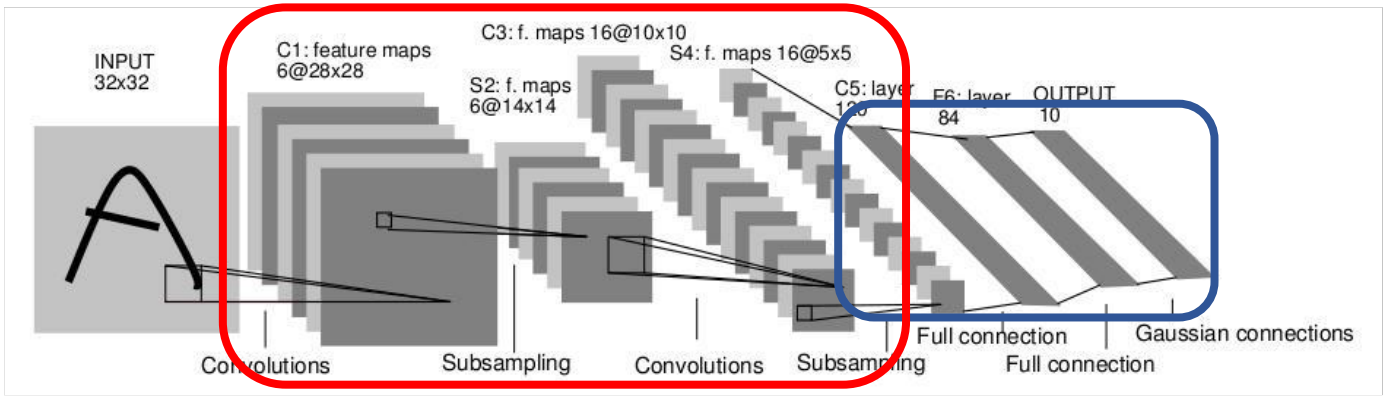
Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks?. In *Advances in neural information processing systems* (pp. 3320-3328).

Overview

- Image processing
- Convolutional neural networks
  - Convolution and pooling
  - CNN Architectures
  - ConvNets outside image classification

Beyond Image Classification

- Not only useful in classification, it can be extended for many other applications
  - + RNN -> image captioning
  - + bounding box -> object localization
  - + fully convolutional NN -> image segmentation
  - + text -> text classification, sentiment analysis, etc.



Beyond Image Classification

- Other CV tasks
  - Image captioning
  - Object localization
  - Image segmentation
  - Object detection
  - Image QA
- NLP
  - Sentence classification (e.g. sentiment analysis)
    - Character-level CNN
    - Topic Categorization
  - QA
  - Relation extraction

Image Captioning: CNN + RNN

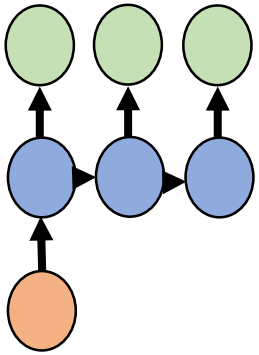
- CNN: feature representation learning
- RNN: language model

Input

Output

A man skiing down the snow covered mountain with a dark sky in the background

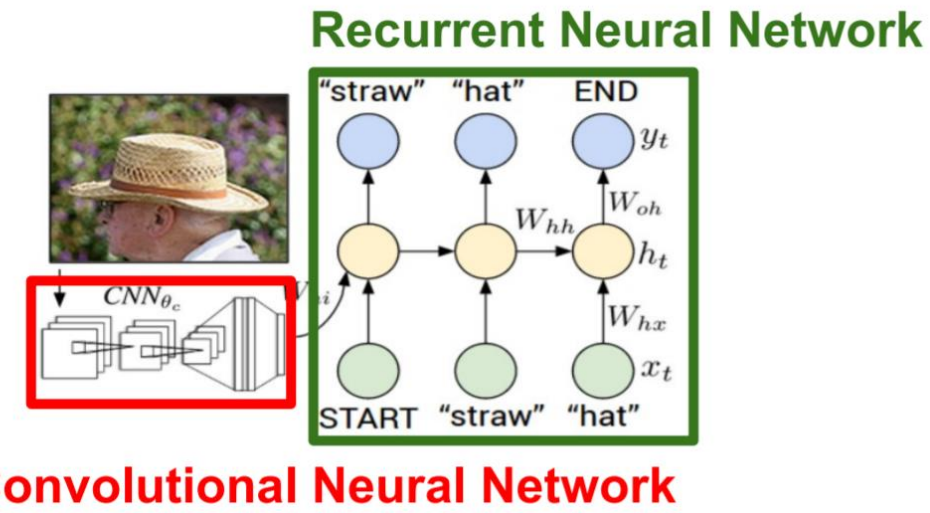
one to many



Picture from http://www.cs.cmu.edu/~rsalakhu/10807_2016/

Image Captioning

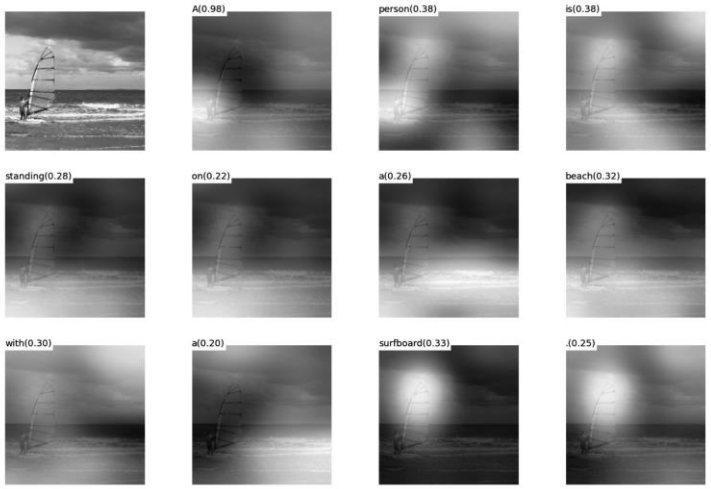
- CNN architectures: AlexNet, VGGNet, etc.
  - Express a single differentiable function from raw image pixel values to class probabilities



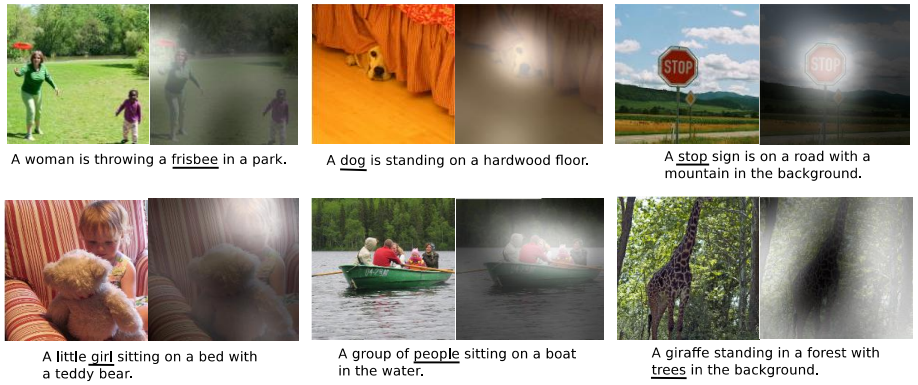
Karpathy A. and Li F. F., Deep Visual-Semantic Alignments for Generating Image Descriptions

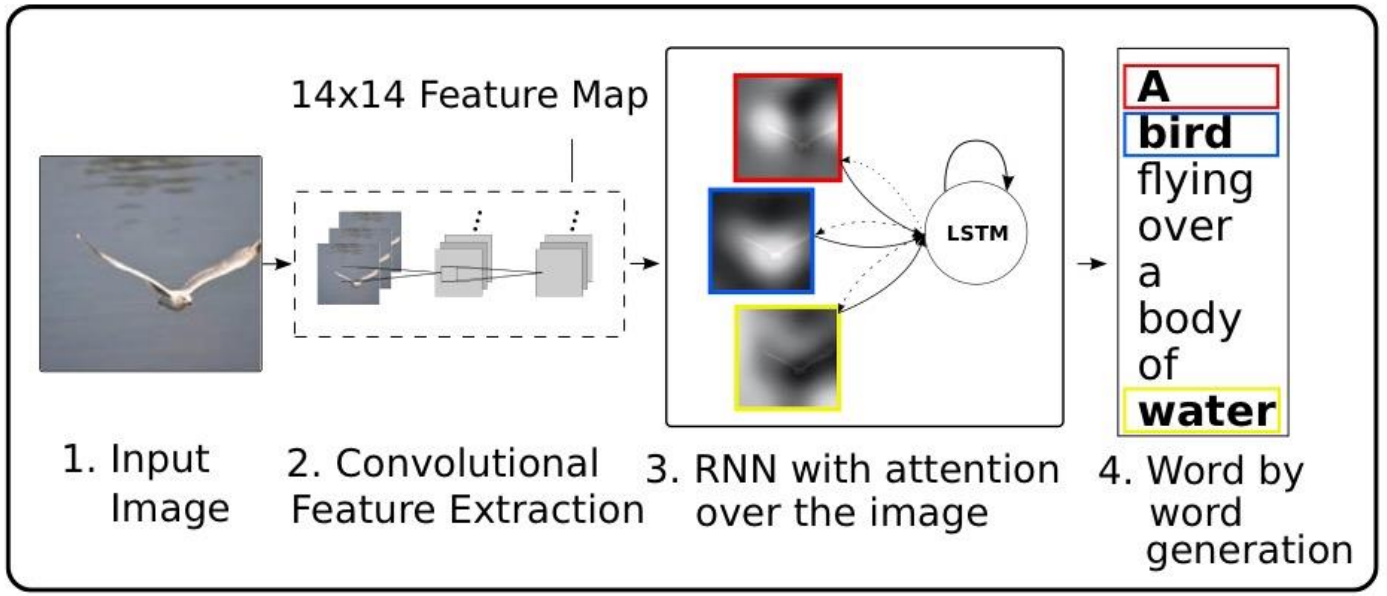
Automated Image Captioning with ConvNets and Recurrent Nets, https://cs.stanford.edu/people/karpathy/sfmltalk.pdf

Caption Generation with Visual Attention



(b) A person is standing on a beach with a surfboard.



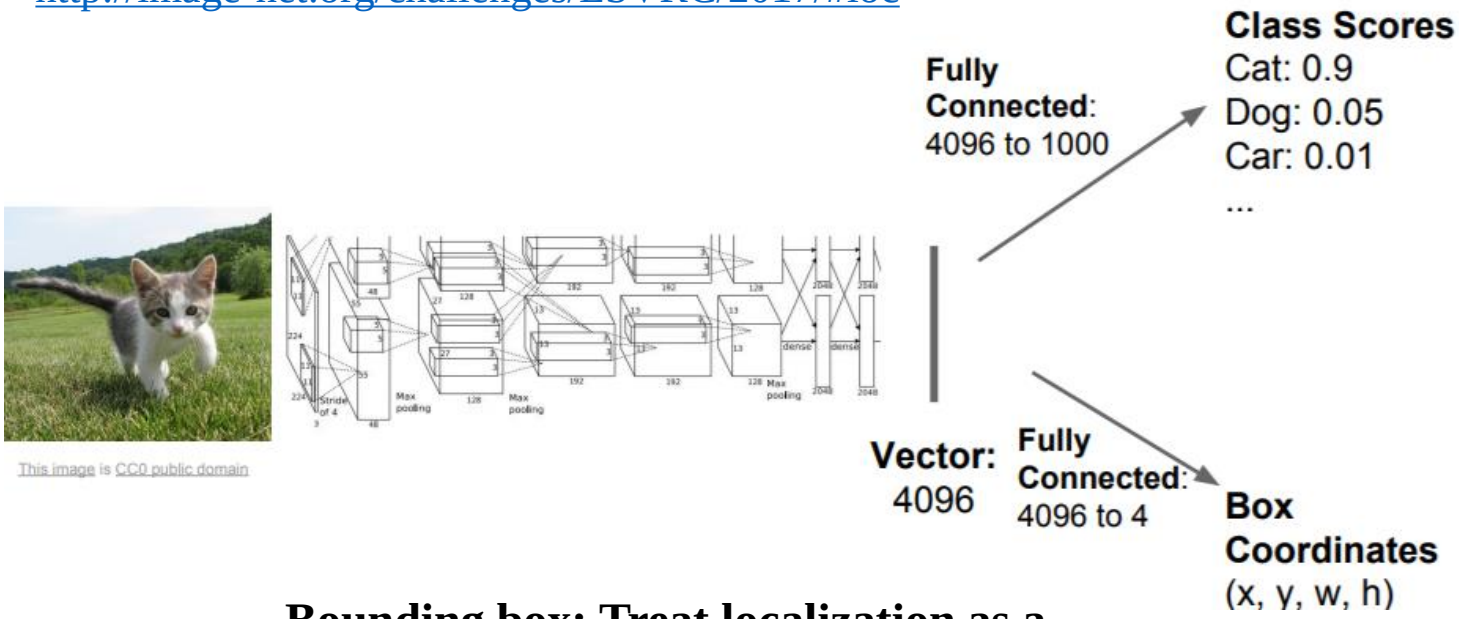


Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., ... & Bengio, Y. (2015, June). Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning* (pp. 2048-2057).

Object Localization

- Object localization competition

  http://image-net.org/challenges/LSVRC/2017/#loc



Bounding box: Treat localization as a regression problem

Image Segmentation

- Assign a class label to each pixel

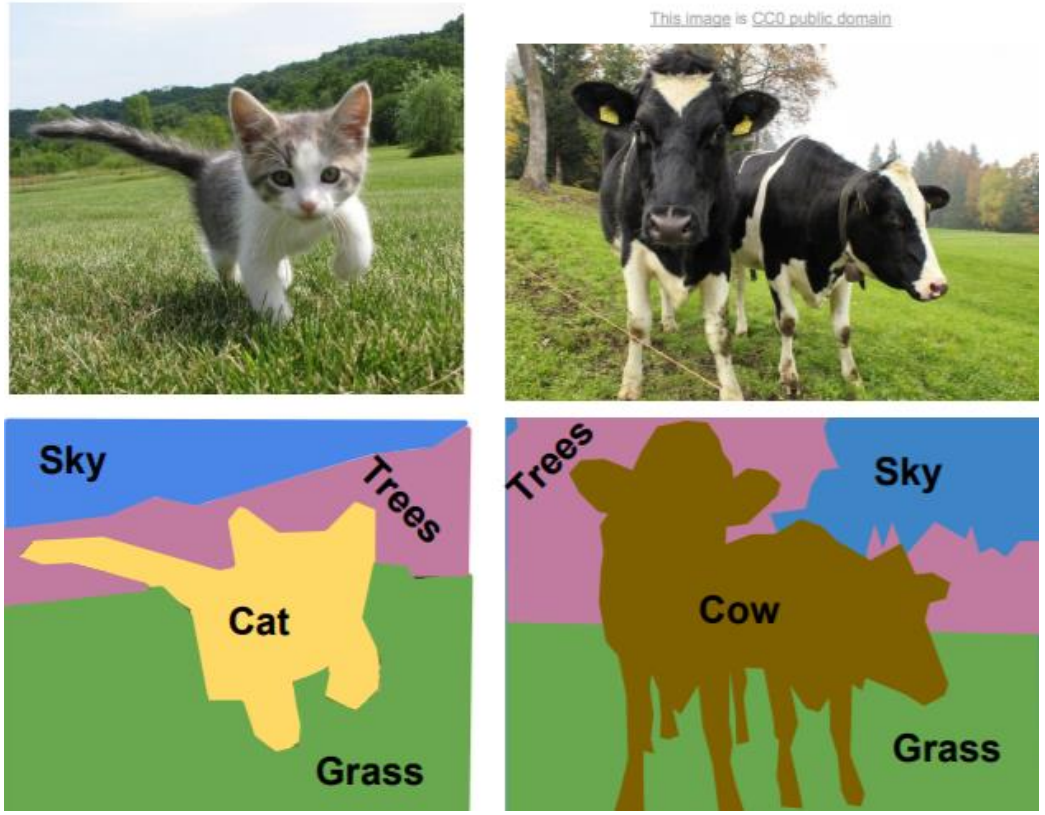
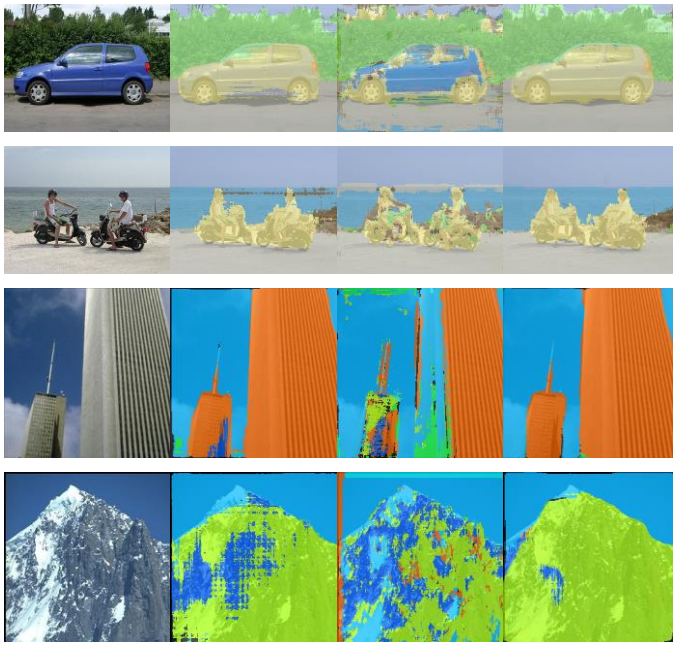
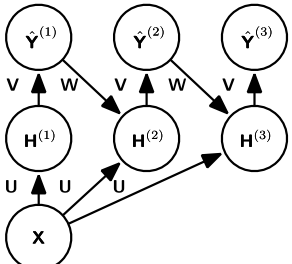


Image Segmentation: Pixel Scene Labeling

- Recurrent convolutional network
  - $\hat{Y}^{(t)}$ the current labeling
  - New estimate based on previous $\hat{Y}^{(t-1)}$ and RGB of input image

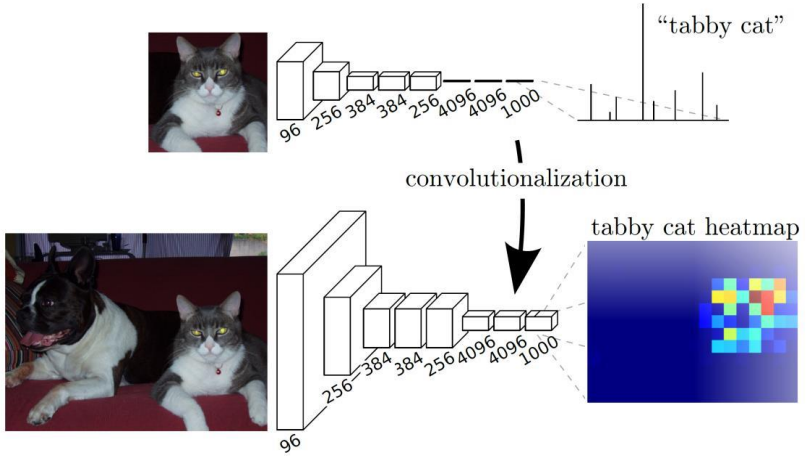


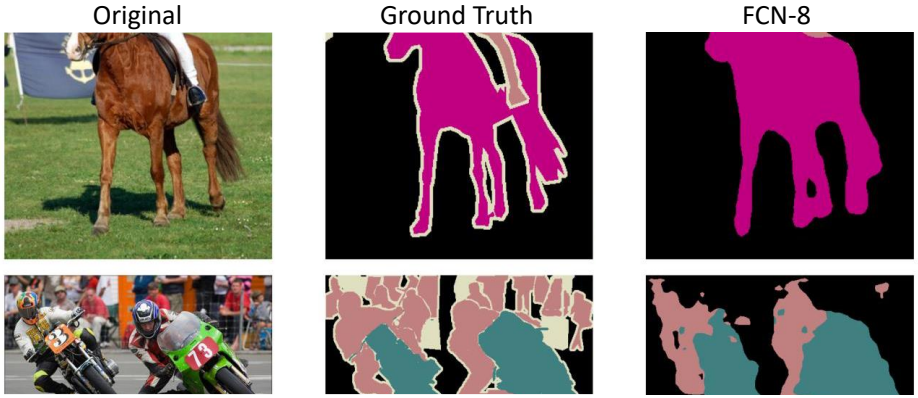
MEANS	CAPACITY CONTROL	SPEED
GRAPHICAL MODEL	–	SLOW
MULTISCALE	SCALE DOWN INPUT IMAGE	FAST
LARGE INPUT PATCHES	INCREASE POOLING	SLOW
	RECURRENT ARCHITECTURE	FAST



Pinheiro, P. H., & Collobert, R. (2014). Recurrent convolutional neural networks for scene labeling. In *31st International Conference on Machine Learning (ICML)* (No. EPFL-CONF-199822).

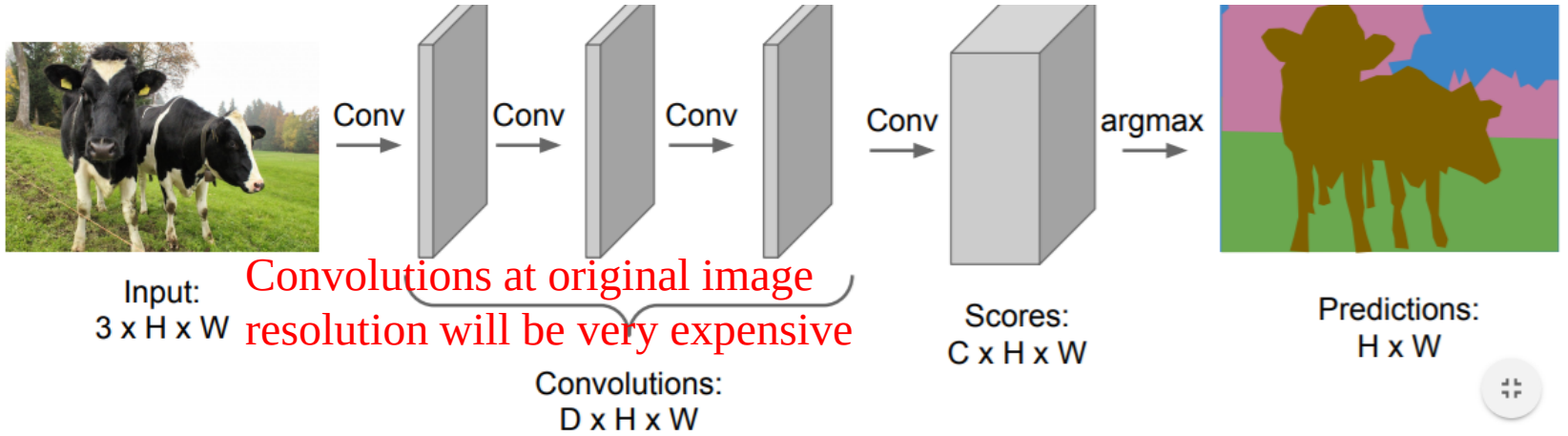
Image Segmentation: Fully Convolutional





Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3431-3440).

Image Segmentation: Fully Convolutional



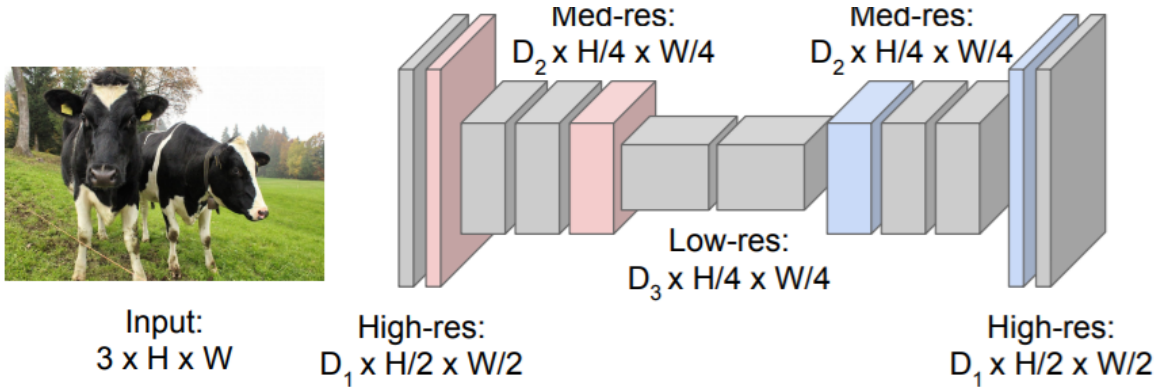
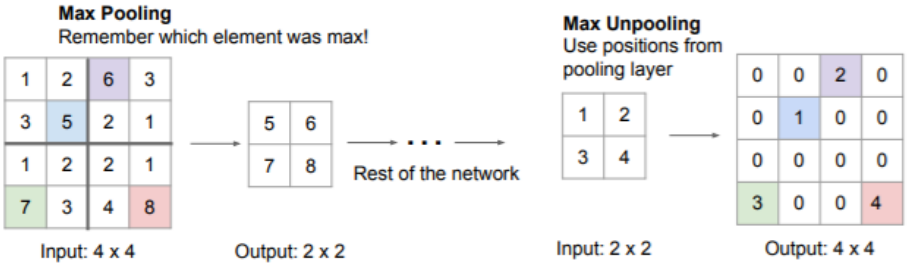


Image Segmentation: Fully Convolutional

- Downsampling
  - Pooling and stride convolutions
- Unsampling
  - Max unpooling
  - Transpose convolution



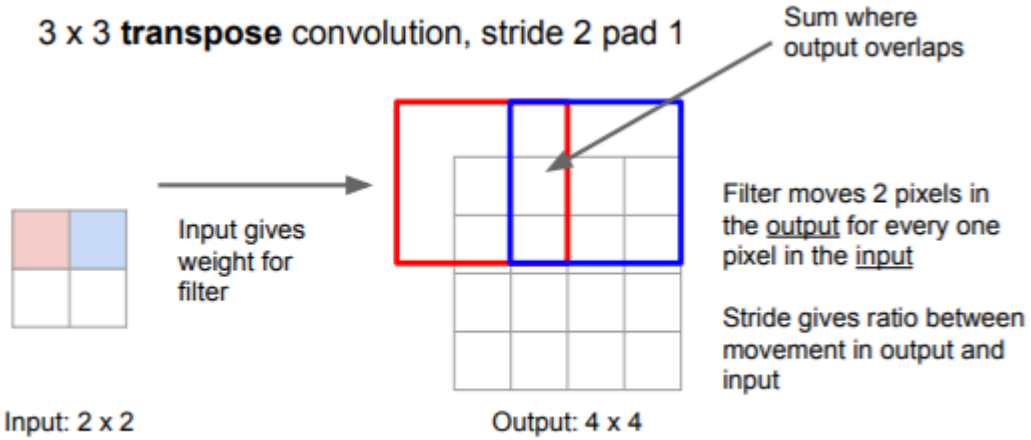
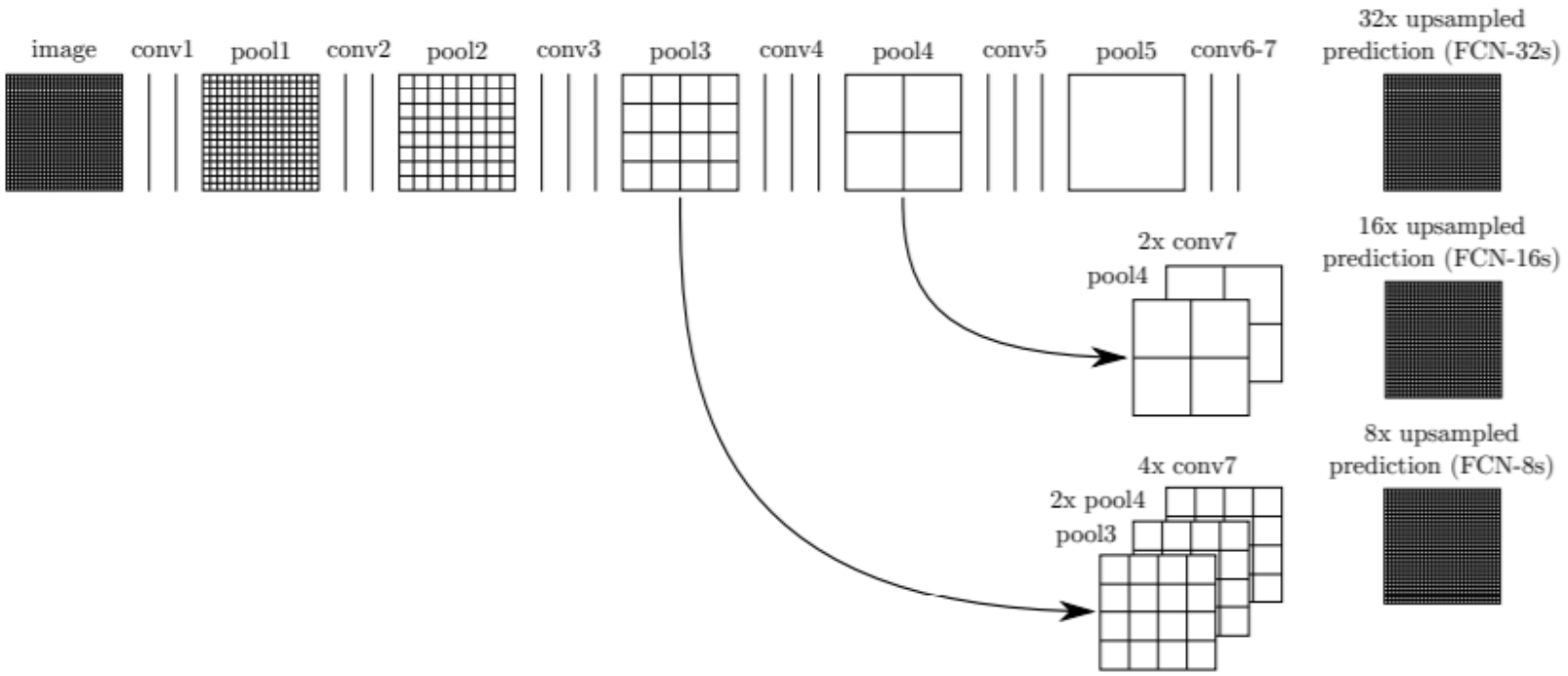
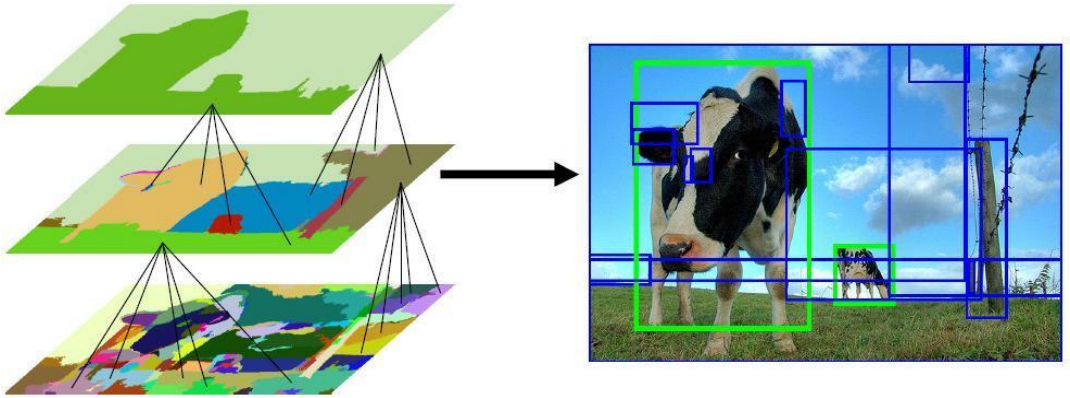


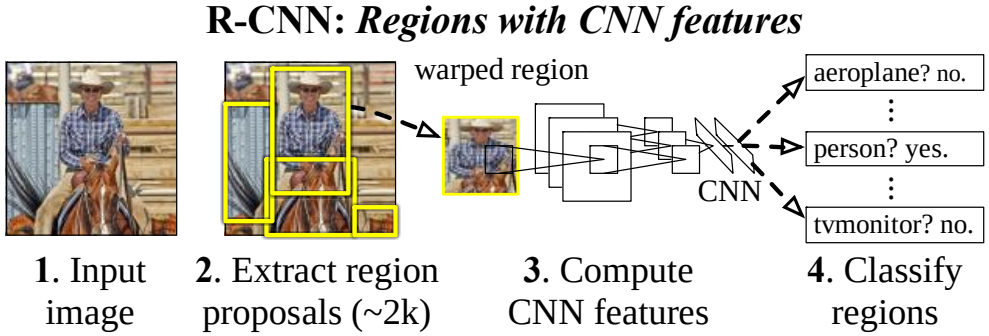
Image Segmentation: Fully Convolutional



Object Detection

- Regional-CNN
  - Build ConvNet for each region



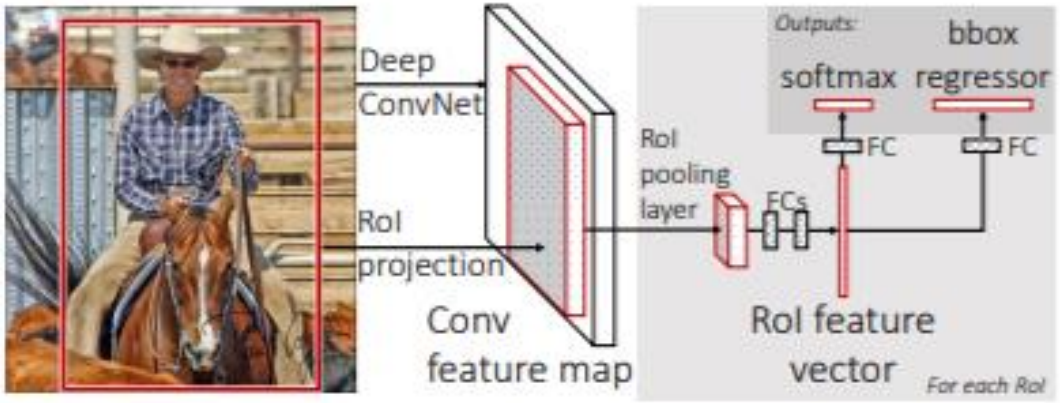


- Training is a multi-stage pipeline, and expensive in space and time
- Inference is slow

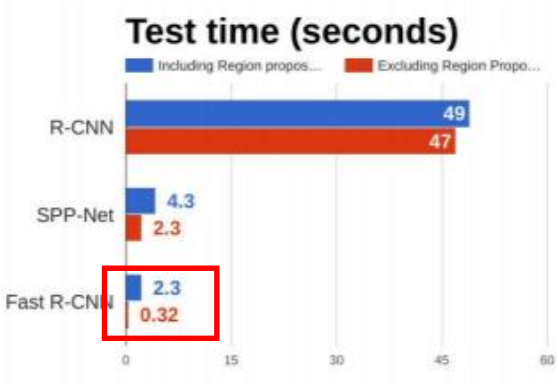
Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580-587).

Object Detection: Fast R-CNN

- Build Deep ConvNet on the whole input image



	Fast R-CNN			R-CNN			SPPnet
	S	**M**	**L**	**S**	**M**	**L**	†**L**
train time (h)	**1.2**	2.0	9.5	22	28	84	25
train speedup	**18.3**×	14.0×	8.8×	1×	1×	1×	3.4×
test rate (s/im)	0.10	0.15	0.32	9.8	12.1	47.0	2.3
▷ with SVD	**0.06**	0.08	0.22	-	-	-	-
test speedup	98×	80×	146×	1×	1×	1×	20×
▷ with SVD	169×	150×	**213**×	-	-	-	-
VOC07 mAP	57.1	59.2	**66.9**	58.5	60.2	66.0	63.1
▷ with SVD	56.5	58.7	66.6	-	-	-	-



Region proposal take too much time in reference

Object Detection: Faster R-CNN

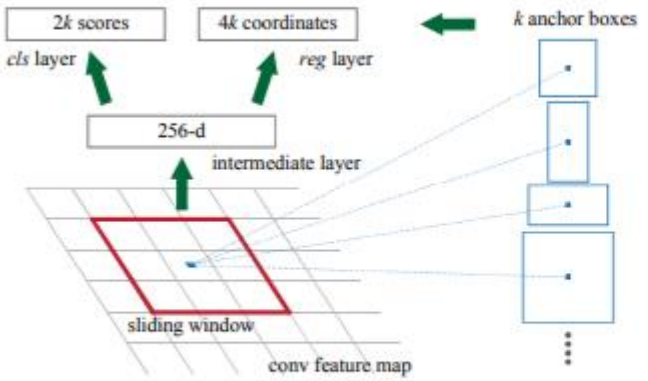
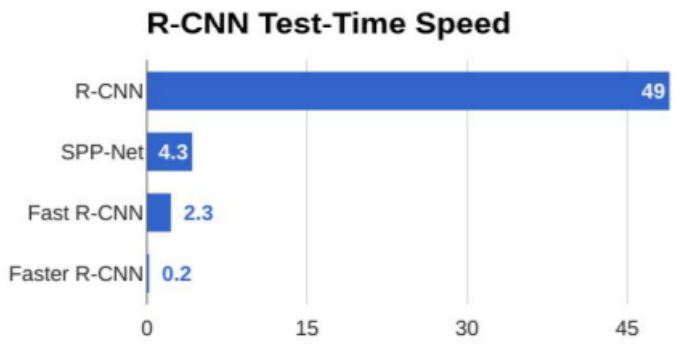


Figure 1: **Left**: Region Proposal Network
on PASCAL VOC 2007 test. Our method



Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, *39*(6), 1137-1149.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91-99).

Image Question Answering

- Use visual semantic embedding to combine CNN and RNN

DAQUAR 1553
What is there in front of the sofa?
Ground truth: table
IMG+BOW: table (0.74)
2-VIS+BLSTM: table (0.88)
LSTM: chair (0.47)

COCOQA 5078
How many leftover donuts is the red bicycle holding?
Ground truth: three
IMG+BOW: two (0.51)
2-VIS+BLSTM: three (0.27)
BOW: one (0.29)

COCOQA 1238
What is the color of the tee-shirt?
Ground truth: blue
IMG+BOW: blue (0.31)
2-VIS+BLSTM: orange (0.43)
BOW: green (0.38)

COCOQA 26088
Where is the gray cat sitting?
Ground truth: window
IMG+BOW: window (0.78)
2-VIS+BLSTM: window (0.68)
BOW: suitcase (0.31)

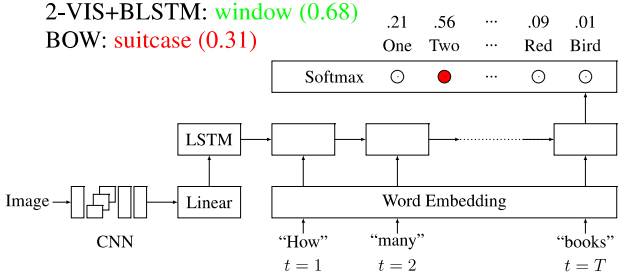


Figure 2: VIS+LSTM Model

Ren, M., Kiros, R., & Zemel, R. (2015). Image question answering: A visual semantic embedding model and a new dataset. *Proc. Advances in Neural Inf. Process. Syst*, *1*(2), 5.

Beyond Image Classification

- Other CV tasks
  - Image captioning
  - Object localization
  - Image segmentation
  - Object detection
  - Image QA
- NLP
  - Sentence classification (e.g. sentiment analysis)
    - Character-level CNN
    - Topic Categorization
  - QA
  - Relation extraction

Sentence Classification

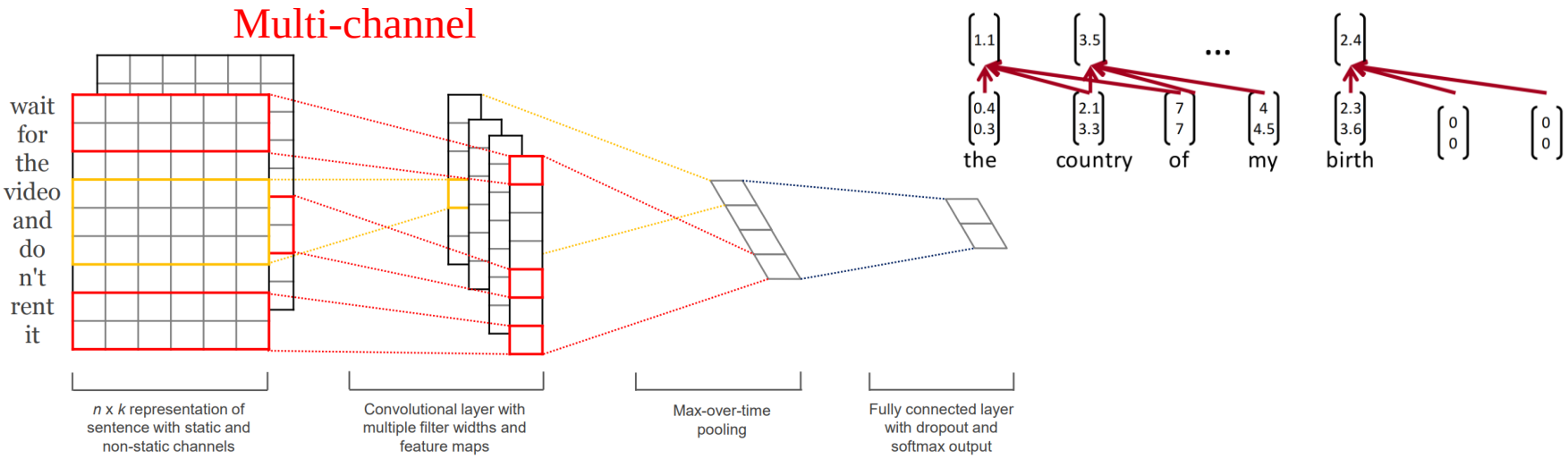


Figure 1: Model architecture with two channels for an example sentence.

A sentence of length n: $\boldsymbol{x}_{1:n} = x_1 \oplus x_2 \oplus \ldots x_n, x_i$ is a word vector of k dimensions

Convolutional filter: $\boldsymbol{W} \in \mathbb{R}^{hk}$, goes over window of h words (h=2 or 3） Multiple filters with different h

Feature map of the CONV layer: $c_i = f(\mathrm{W}^T \boldsymbol{x}_{\mathrm{i:i+h-1}} + \mathrm{b})$

Dropout: 2 – 4% improved accuracy and ability to use very large networks without overfitting

Kim, Yoon. "Convolutional neural networks for sentence classification." *arXiv preprint arXiv:1408.5882* (2014).

All hyperparameters in Kim (2014)

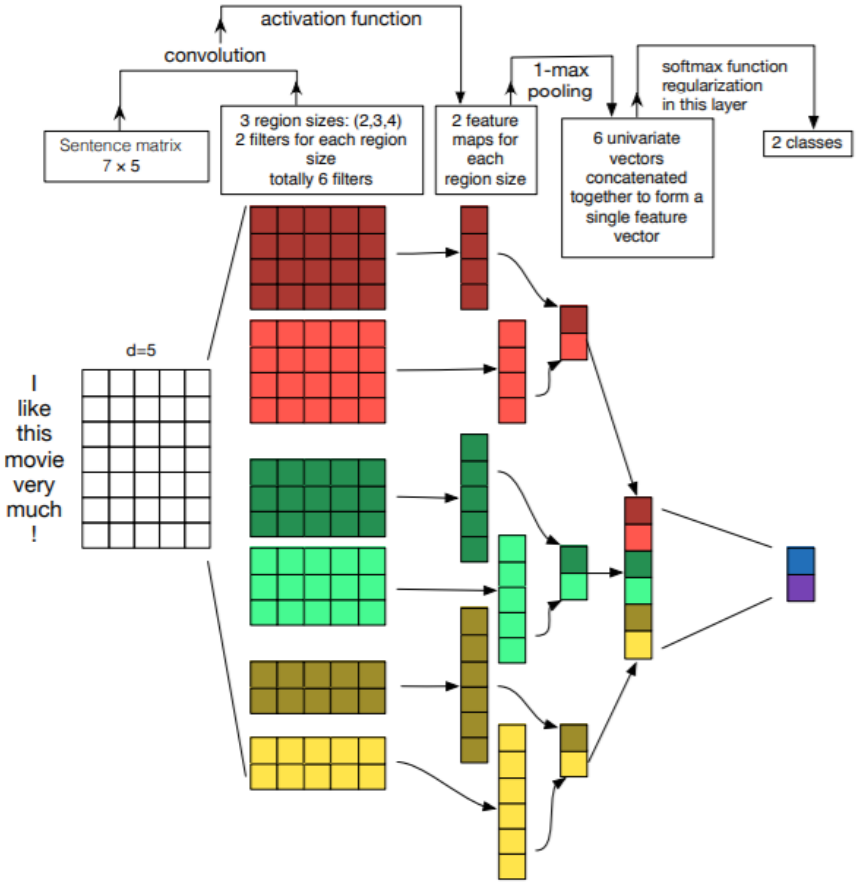
- Find hyperparameters based on dev set
- Nonlinearity: ReLU
- Window filter sizes h = 3,4,5, and each filter size has 100 feature maps
- Dropout p = 0.5
- L2 constraint s for rows of softmax s = 3
- Mini batch size for SGD training: 50
- Word vectors: pre-trained with word2vec, k = 300
- During training, keep checking performance on dev set and pick highest accuracy weights for final evaluation

Sentence Classification

Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-rand	76.1	45.0	82.7	89.6	91.2	79.8	83.4
CNN-static	81.0	45.5	86.8	93.0	92.8	84.7	**89.6**
CNN-non-static	**81.5**	48.0	87.2	93.4	93.6	84.3	89.5
CNN-multichannel	81.1	47.4	**88.1**	93.2	92.2	**85.0**	89.4
RAE (Socher et al., 2011)	77.7	43.2	82.4	–	–	–	86.4
MV-RNN (Socher et al., 2012)	79.0	44.4	82.9	–	–	–	–
RNTN (Socher et al., 2013)	–	45.7	85.4	–	–	–	–
DCNN (Kalchbrenner et al., 2014)	–	48.5	86.8	–	93.0	–	–
Paragraph-Vec (Le and Mikolov, 2014)	–	**48.7**	87.8	–	–	–	–
CCAE (Hermann and Blunsom, 2013)	77.8	–	–	–	–	–	87.2
Sent-Parser (Dong et al., 2014)	79.5	–	–	–	–	–	86.3
NBSVM (Wang and Manning, 2012)	79.4	–	–	93.2	–	81.8	86.3
MNB (Wang and Manning, 2012)	79.0	–	–	**93.6**	–	80.0	86.3
G-Dropout (Wang and Manning, 2013)	79.0	–	–	93.4	–	82.1	86.1
F-Dropout (Wang and Manning, 2013)	79.1	–	–	**93.6**	–	81.9	86.3
Tree-CRF (Nakagawa et al., 2010)	77.3	–	–	–	–	81.4	86.1
CRF-PR (Yang and Cardie, 2014)	–	–	–	–	–	82.7	–
SVM_S (Silva et al., 2011)	–	–	–	–	**95.0**	–	–

Sentence classification

- Sentences as “images”
- Features can be taken from word2vec or one-hot coding
- Filters with variable receptive fields
  - Local in length but full in depth
- 1-max pooling to pick the largest activation of each filter
- Concatenate the feature maps for soft-max layers



Zhang, Y., & Wallace, B. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

Sentence Classification: Effect of Input Word Vectors

Dataset	Non-static word2vec-CNN	Non-static GloVe-CNN	Non-static GloVe+word2vec CNN
MR	81.24 (80.69, 81.56)	81.03 (80.68,81.48)	81.02 (80.75,81.32)
SST-1	47.08 (46.42,48.01)	45.65 (45.09,45.94)	45.98 (45.49,46.65)
SST-2	85.49 (85.03, 85.90)	85.22 (85.04,85.48)	85.45 (85.03,85.82)
Subj	93.20 (92.97, 93.45)	93.64 (93.51,93.77)	93.66 (93.39,93.87)
TREC	91.54 (91.15, 91.92)	90.38 (90.19,90.59)	91.37 (91.13,91.62)
CR	83.92 (82.95, 84.56)	84.33 (84.00,84.67)	84.65 (84.21,84.96)
MPQA	89.32 (88.84, 89.73)	89.57 (89.31,89.78)	89.55 (89.22,89.88)
Opi	64.93 (64.23,65.58)	65.68 (65.29,66.19)	65.65 (65.15,65.98)
Irony	67.07 (65.60,69.00)	67.20 (66.45,67.96)	67.11 (66.66,68.50)

One hot encoding performs poorly for sentence classification (but good in document classification)

Sentence Classification: Effect of Filter Region Size (h)

Region size	MR
1	77.85 (77.47,77.97)
3	80.48 (80.26,80.65)
5	81.13 (80.96,81.32)
7	**81.65 (81.45,81.85)**
10	81.43 (81.28,81.75)
15	81.26 (81.01,81.43)
20	81.06 (80.87,81.30)
25	80.91 (80.73,81.10)
30	80.91 (80.72,81.05)

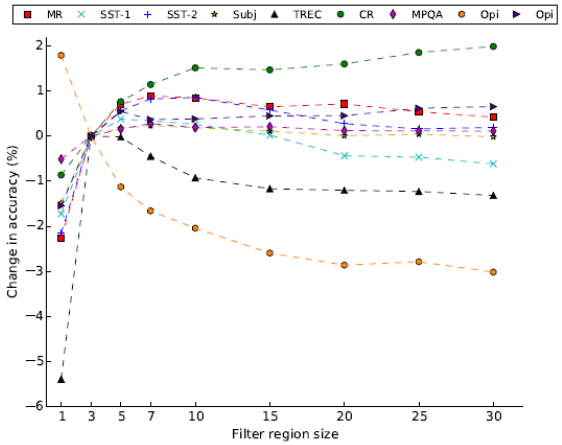


Figure 3: Effect of the region size (using only one).

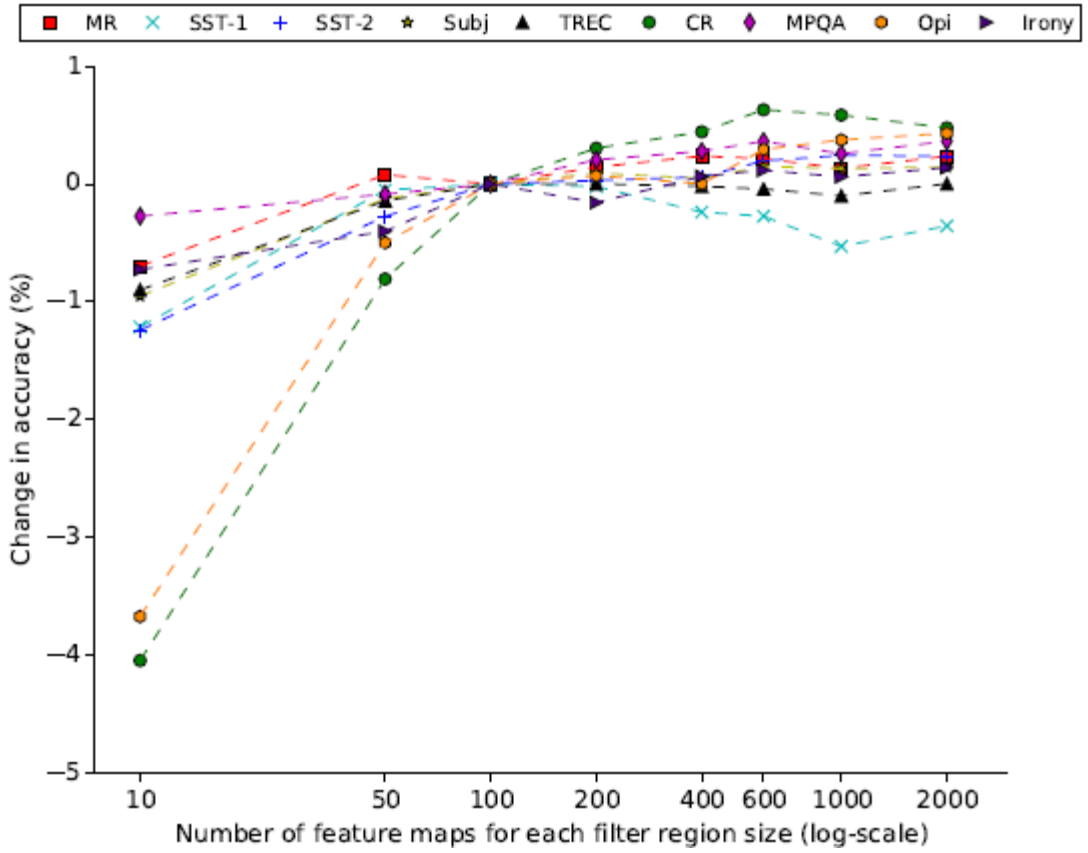
Multiple region size	Accuracy (%)
(7)	81.65 (81.45,81.85)
(3,4,5)	81.24 (80.69, 81.56)
(4,5,6)	81.28 (81.07,81.56)
(5,6,7)	81.57 (81.31,81.80)
(7,8,9)	81.69 (81.27,81.93)
(10,11,12)	81.52 (81.27,81.87)
(11,12,13)	81.53 (81.35,81.76)
(3,4,5,6)	81.43 (81.10,81.61)
(6,7,8,9)	81.62 (81.38,81.72)
(7,7,7)	81.63 (81.33,82.08)
(7,7,7,7)	**81.73 (81.33,81.94)**

Hints for applications:

-first perform a coarse line-search over a single filter region size to find the 'best' size for the dataset under consideration

-then explore the combination of several region sizes nearby this single best size, including combining both different region sizes and copies of the optimal sizes

Sentence Classification: Effect of No. of Feature Maps for Each filter region size



Sentence Classification: Effect of Activation Functions

Dataset	tanh	Softplus	Iden	ReLU
MR	81.28 (81.07, 81.52)	80.58 (80.17, 81.12)	**81.30 (81.09, 81.52)**	81.16 (80.81, 83.38)
SST-1	47.02 (46.31, 47.73)	46.95 (46.43, 47.45)	46.73 (46.24,47.18)	**47.13 (46.39, 47.56)**
SST-2	**85.43 (85.10, 85.85)**	84.61 (84.19, 84.94)	85.26 (85.11, 85.45)	85.31 (85.93, 85.66)
Subj	**93.15 (92.93, 93.34)**	92.43 (92.21, 92.61)	93.11 (92.92, 93.22)	93.13 (92.93, 93.23)
TREC	91.18 (90.91, 91.47)	91.05 (90.82, 91.29)	91.11 (90.82, 91.34)	**91.54 (91.17, 91.84)**
CR	84.28 (83.90, 85.11)	83.67 (83.16, 84.26)	**84.55 (84.21, 84.69)**	83.83 (83.18, 84.21)
MPQA	89.48 (89.16, 89.84)	**89.62 (89.45, 89.77)**	89.57 (89.31, 89.88)	89.35 (88.88, 89.58)
Opi	**65.69 (65.16,66.40)**	64.77 (64.25,65.28)	65.32 (64.78,66.09)	65.02 (64.53,65.45)
Irony	**67.62 (67.18,68.27)**	66.20 (65.38,67.20)	66.77 (65.90,67.47)	66.46 (65.99,67.17)

Hints for applications:
-sigmoid, cube, tanh cube perform worse (might not consider in sentence classification)
- Can try tanh, identity (for shallow network), and ReLU

21 types of activation functions

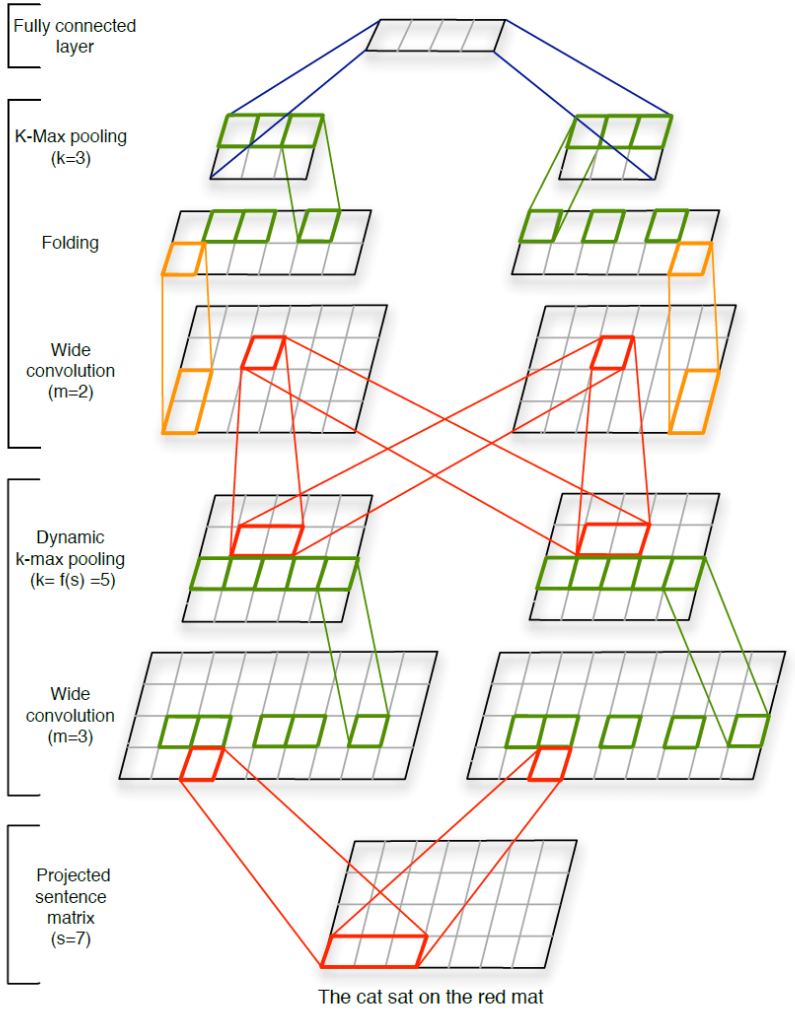
Eger, S., Youssef, P., & Gurevych, I. (2019). Is it Time to Swish? Comparing Deep Learning Activation Functions Across NLP tasks. *arXiv preprint arXiv:1901.02671*.

Sentence Classification: Other Hints

- 1-max pooling consistently perform better than other strategies
- Regularization has relatively little effect on CNN sentence classification

Sentence Classification

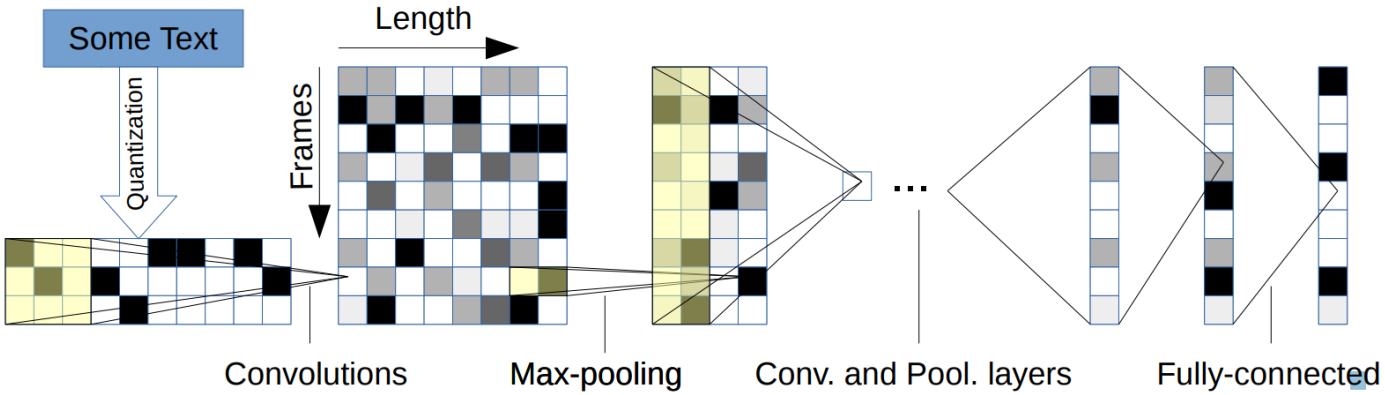
- Dynamic k-Max Pooling
  - k be a function of the length of the sentence and the depth of the network



Kalchbrenner, Nal, Edward Grefenstette, and Phil Blunsom. "A convolutional neural network for modelling sentences." *arXiv preprint arXiv:1404.2188* (2014).

Character Level CNN

- Character quantization method
  - A sequence of characters as input
  - Alphabet of size m, 1 of m encoding
- 1-d convolution
- Data augmentation with English thesaurus
  - replace words or phrases with their synonyms



Zhang, X., & LeCun, Y. (2015). Text understanding from scratch. *arXiv preprint arXiv:1502.01710*.

Char-CNN for Text Classification

Table 4: Testing errors of all the models. Numbers are in percentage. "Lg" stands for "large" and "Sm" stands for "small". "w2v" is an abbreviation for "word2vec", and "Lk" for "lookup table". "Th" stands for thesaurus. ConvNets labeled "Full" are those that distinguish between lower and upper letters

Model	AG	Sogou	DBP.	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
BoW	11.19	7.15	3.39	7.76	42.01	31.11	45.36	9.60
BoW TFIDF	10.36	6.55	2.63	6.34	40.14	28.96	44.74	9.00
ngrams	7.96	2.92	1.37	**4.36**	43.74	31.53	45.73	7.98
ngrams TFIDF	**7.64**	**2.81**	**1.31**	4.56	45.20	31.49	47.56	8.46
Bag-of-means	**16.91**	**10.79**	**9.55**	**12.67**	**47.46**	**39.45**	**55.87**	**18.39**
LSTM	13.94	4.82	1.45	5.26	41.83	29.16	40.57	6.10
Lg. w2v Conv.	9.92	4.39	1.42	4.60	40.16	31.97	44.40	5.88
Sm. w2v Conv.	11.35	4.54	1.71	5.56	42.13	31.50	42.59	6.00
Lg. w2v Conv. Th.	9.91	-	1.37	4.63	39.58	31.23	43.75	5.80
Sm. w2v Conv. Th.	10.88	-	1.53	5.36	41.09	29.86	42.50	5.63
Lg. Lk. Conv.	8.55	4.95	1.72	4.89	40.52	29.06	45.95	5.84
Sm. Lk. Conv.	10.87	4.93	1.85	5.54	41.41	30.02	43.66	5.85
Lg. Lk. Conv. Th.	8.93	-	1.58	5.03	40.52	28.84	42.39	5.52
Sm. Lk. Conv. Th.	9.12	-	1.77	5.37	41.17	28.92	43.19	5.51
Lg. Full Conv.	9.85	8.80	1.66	5.25	38.40	29.90	40.89	5.78
Sm. Full Conv.	11.59	8.95	1.89	5.67	38.82	30.01	40.88	5.78
Lg. Full Conv. Th.	9.51	-	1.55	4.88	38.04	29.58	40.54	5.51
Sm. Full Conv. Th.	10.89	-	1.69	5.42	**37.95**	29.90	40.53	5.66
Lg. Conv.	12.82	4.88	1.73	5.89	39.62	29.55	41.31	5.51
Sm. Conv.	15.65	8.65	1.98	6.53	40.84	29.84	40.53	5.50
Lg. Conv. Th.	13.39	-	1.60	5.82	39.30	**28.80**	40.45	**4.93**
Sm. Conv. Th.	14.80	-	1.85	6.49	40.16	29.84	**40.43**	5.67

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Advances in neural information processing systems* (pp. 649-657).

Char-CNN for Text Classification

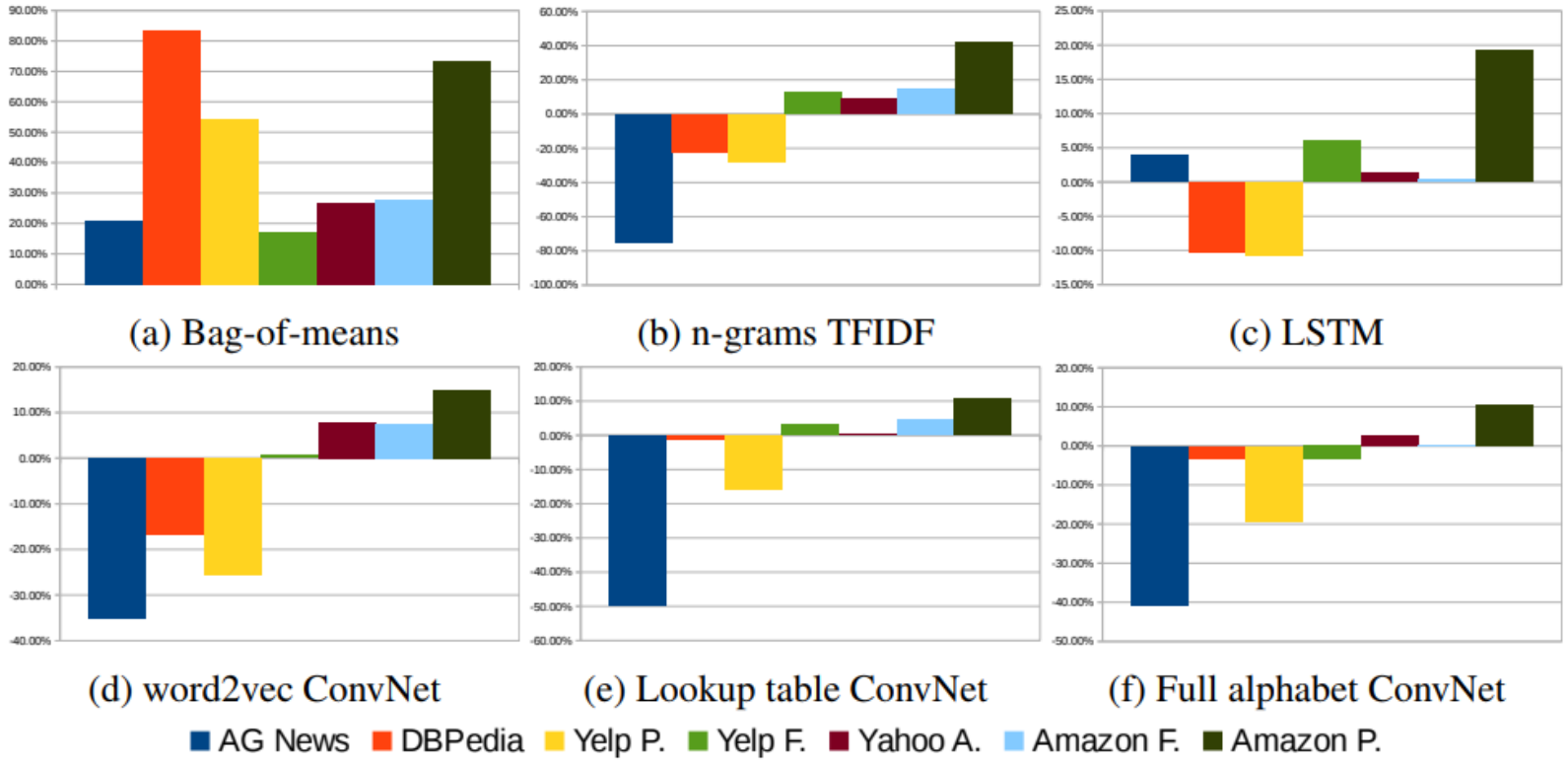


Figure 3: Relative errors with comparison models

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Advances in neural information processing systems* (pp. 649-657).

Sentiment Analysis with CNN

- Initial layer
  - embedding on both word-level and character level
  - convolution on the embedded char features
- Second layer
  - convolution on the sentence level representation
- Fully connected layers

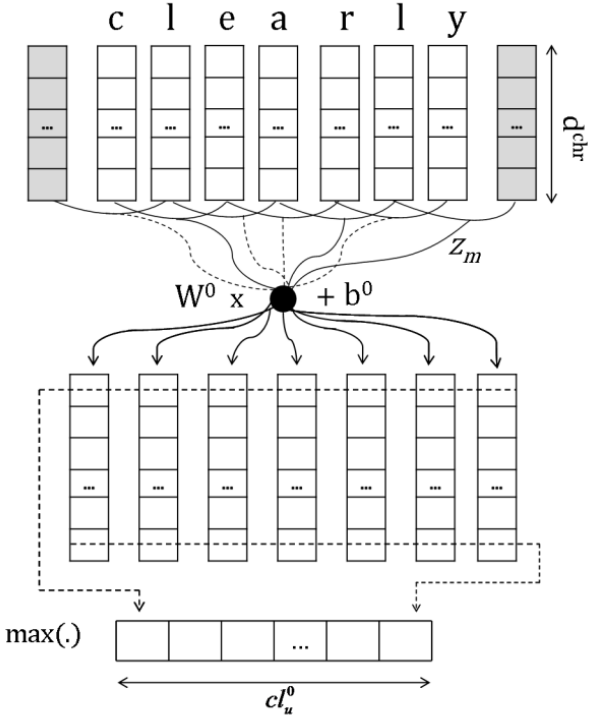


Figure 1: Convolutional approach to character-level feature extraction.

dos Santos, C., & Gatti, M. (2014). Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers* (pp. 69-78).

Model	Phrases	Fine-Grained	Positive/Negative
CharSCNN	yes	**48.3**	**85.7**
SCNN	yes	**48.3**	85.5
CharSCNN	no	43.5	82.3
SCNN	no	43.5	82.0
RNTN (Socher et al., 2013b)	yes	45.7	85.4
MV-RNN (Socher et al., 2013b)	yes	44.4	82.9
RNN (Socher et al., 2013b)	yes	43.2	82.4
NB (Socher et al., 2013b)	yes	41.0	81.8
SVM (Socher et al., 2013b)	yes	40.7	79.4

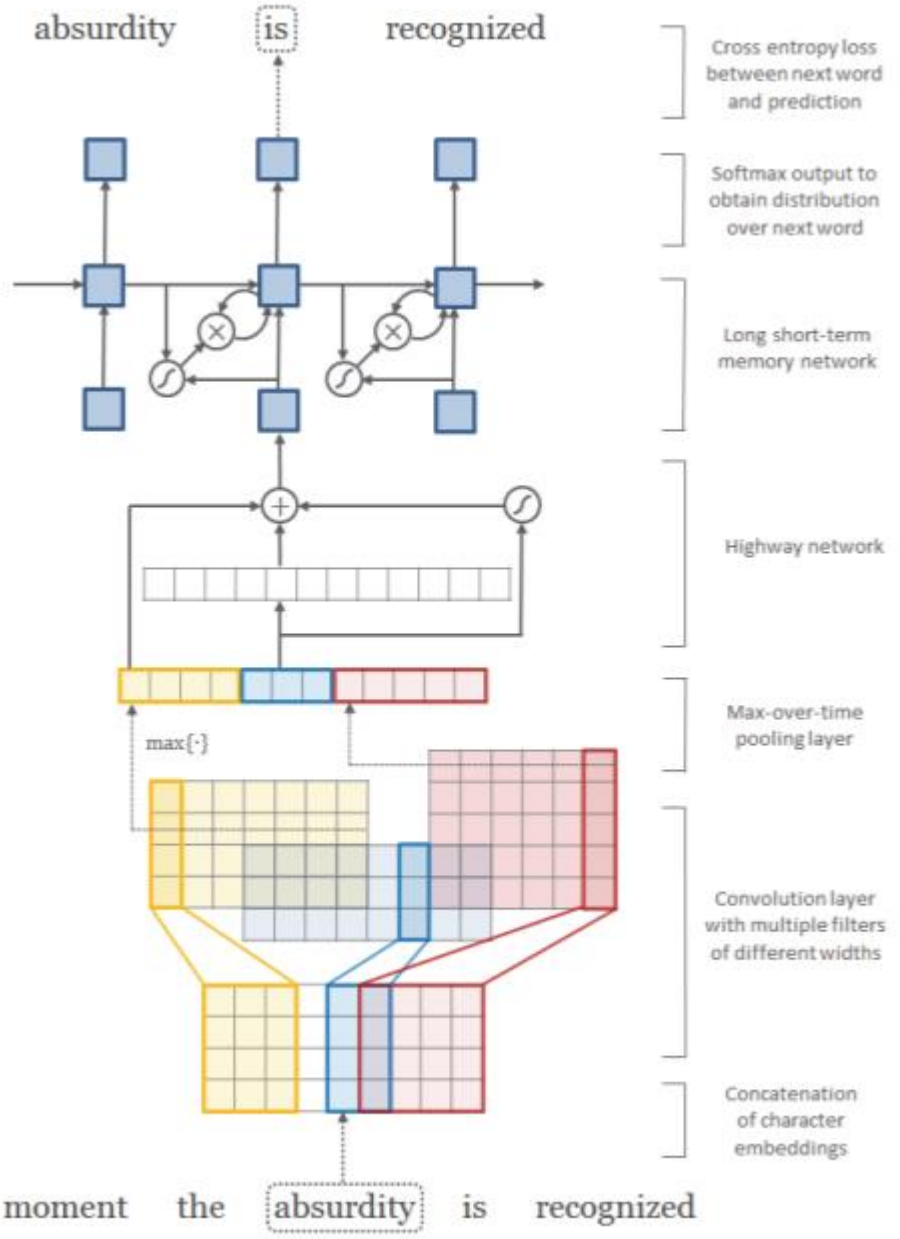
Accuracy comparison on SSTb

Model	Accuracy (unsup. pre-training)	Accuracy (random word embeddings)
CharSCNN	**86.4**	81.9
SCNN	85.2	82.2
LProp (Speriosu et al., 2011)	84.7	
MaxEnt (Go et al., 2009)	83.0	
NB (Go et al., 2009)	82.7	
SVM (Go et al., 2009)	82.2	

Accuracy comparison on STS

CNN+RNN for NLP

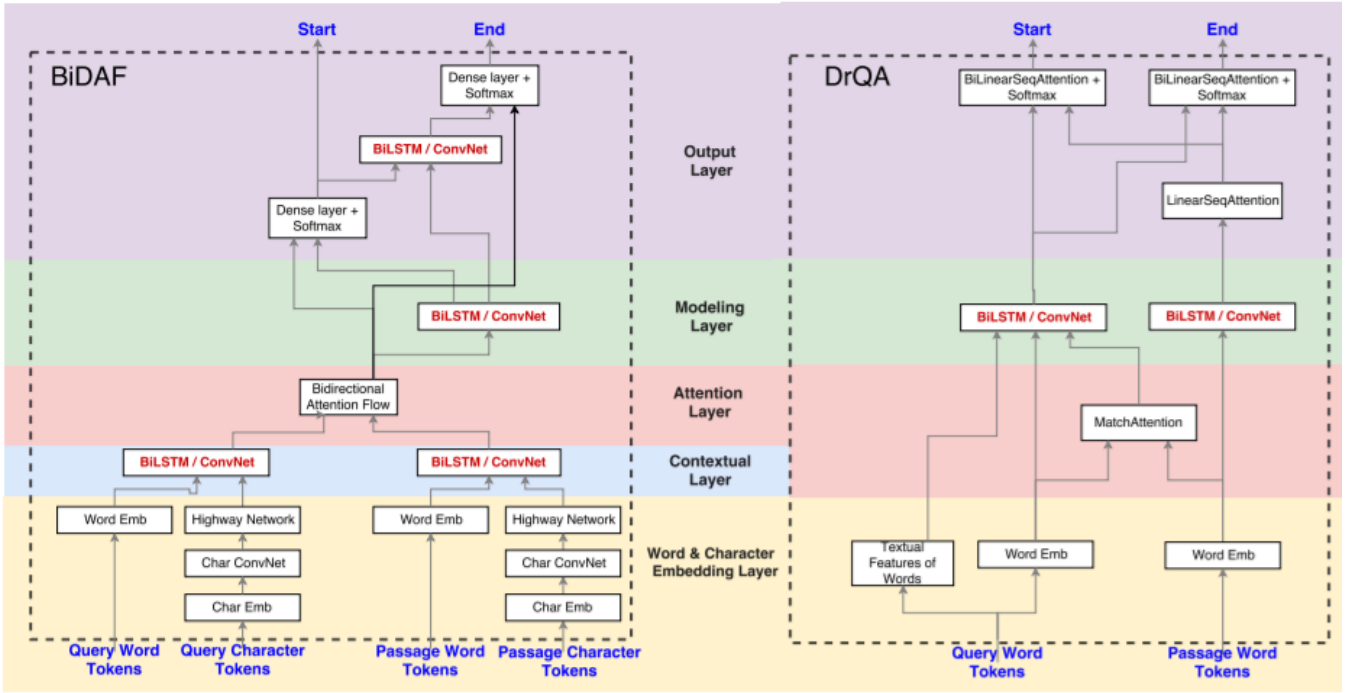
- CNN features given to LSTM
- Using character-level CNN



Kim, Y., Jernite, Y., Sontag, D., & Rush, A. M. (2016, February). Characte

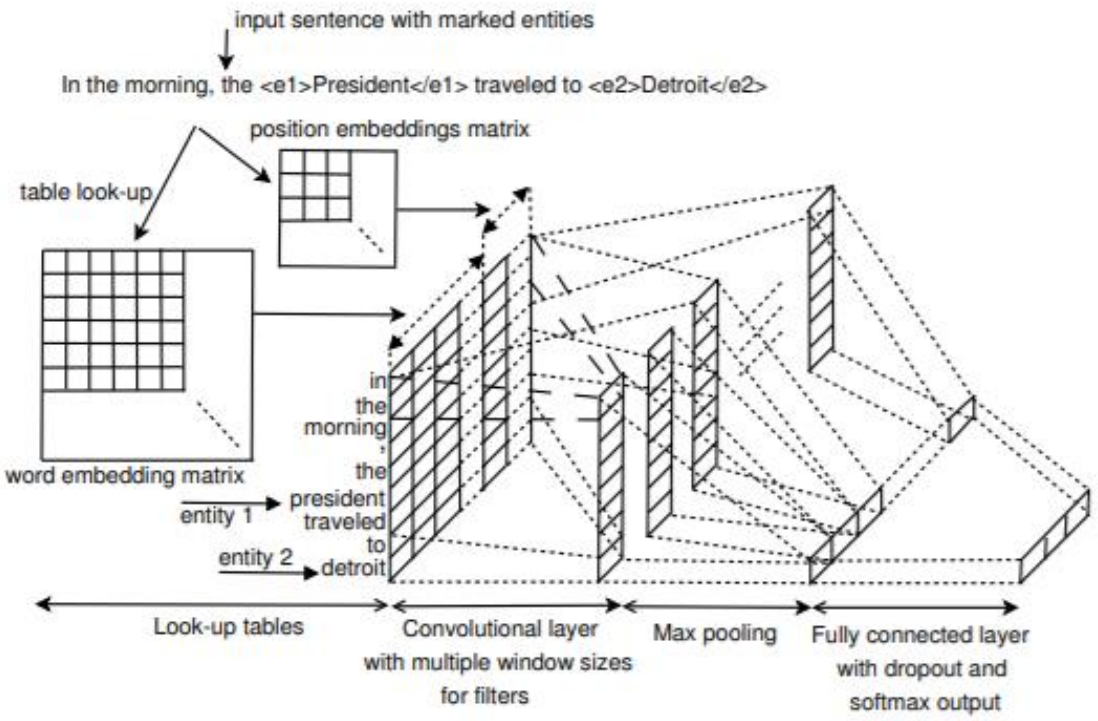
QA

- Replace RNN with ConvNet



Wu, F., Lao, N., Blitzer, J., Yang, G., & Weinberger, K. (2017). Fast Reading Comprehension with ConvNets. *arXiv preprint arXiv:1711.04352*.

Relation Extraction



Nguyen, T. H., & Grishman, R. (2015). Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing* (pp. 39-48).

Summary

- CNN has wide applications
- Powerful for feature representation in both images and text

Summary of the Lecture

- Image processing
- Convolutional neural networks
  - Convolution and pooling
  - CNN Architectures
  - ConvNets outside image processing