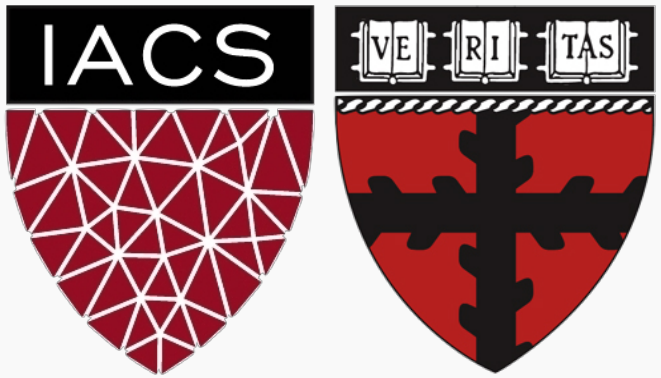
Lecture 9: Convolutional Neural Networks 2

CS109B Data Science 2

Pavlos Protopapas and Mark Glickman



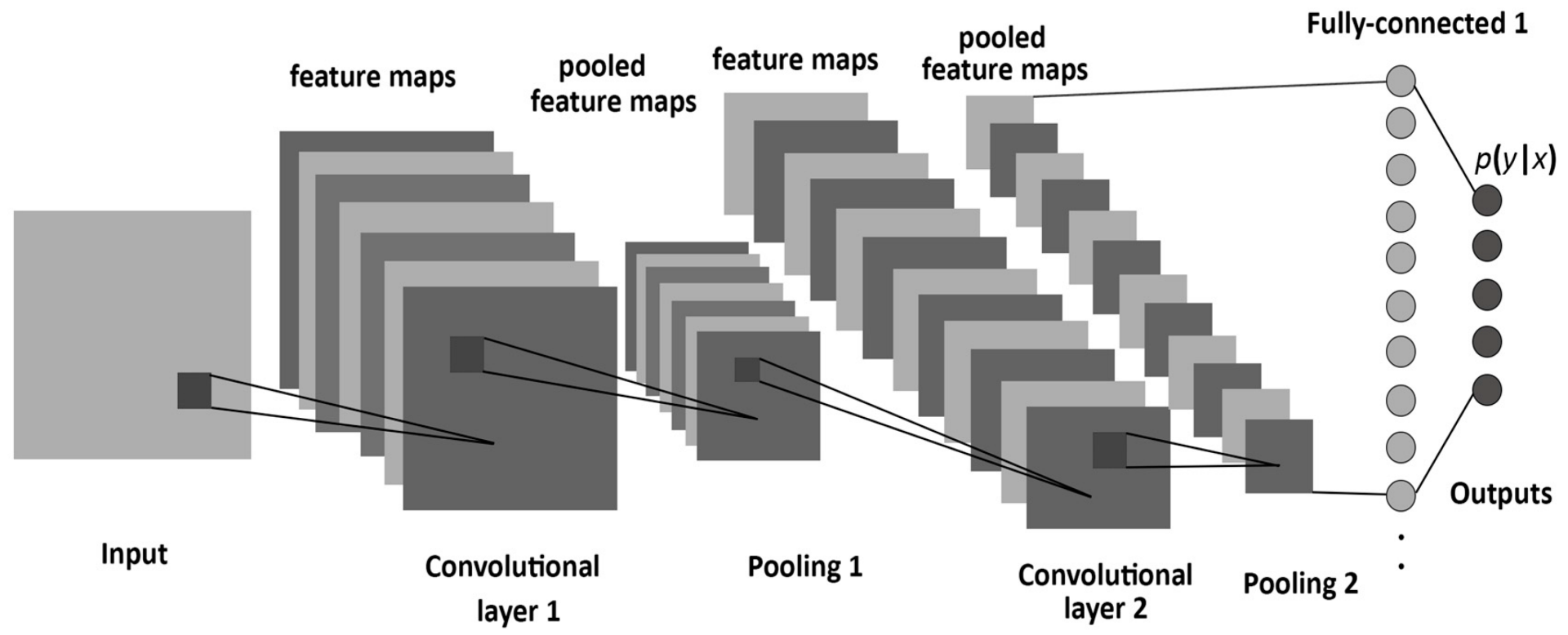
Outline

1. Review from last lecture
2. BackProp of MaxPooling layer
3. A bit of history
4. Layers Receptive Field
5. Saliency maps
6. Transfer Learning
7. CNN for text analysis (example)

Outline

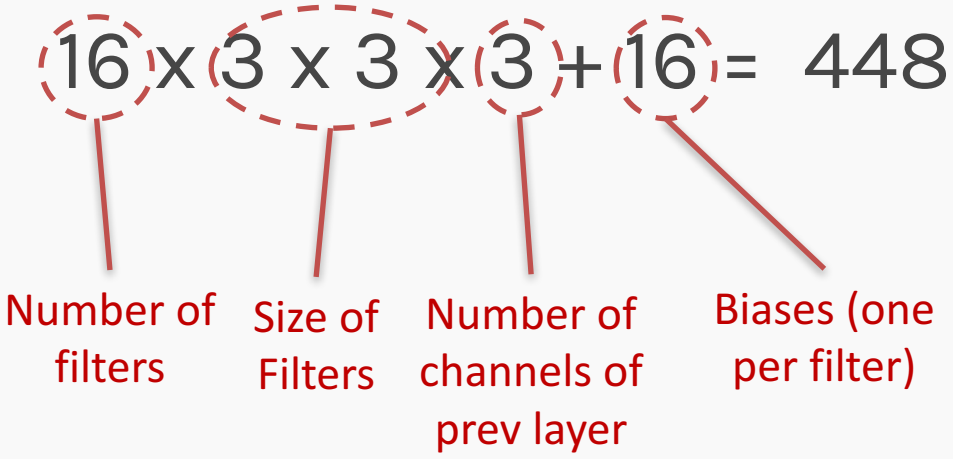
1. **Review from last lecture**
2. BackProp of MaxPooling layer
3. A bit of history
4. Layers Receptive Field
5. Saliency maps
6. Transfer Learning
7. CNN for text analysis (example)

From last lecture



Examples

- I have a convolutional layer with 16 3x3 filters that takes an RGB image as input.
  - What else can we define about this layer?
    - Activation function
    - Stride
    - Padding type
  - How many parameters does the layer have?



Examples

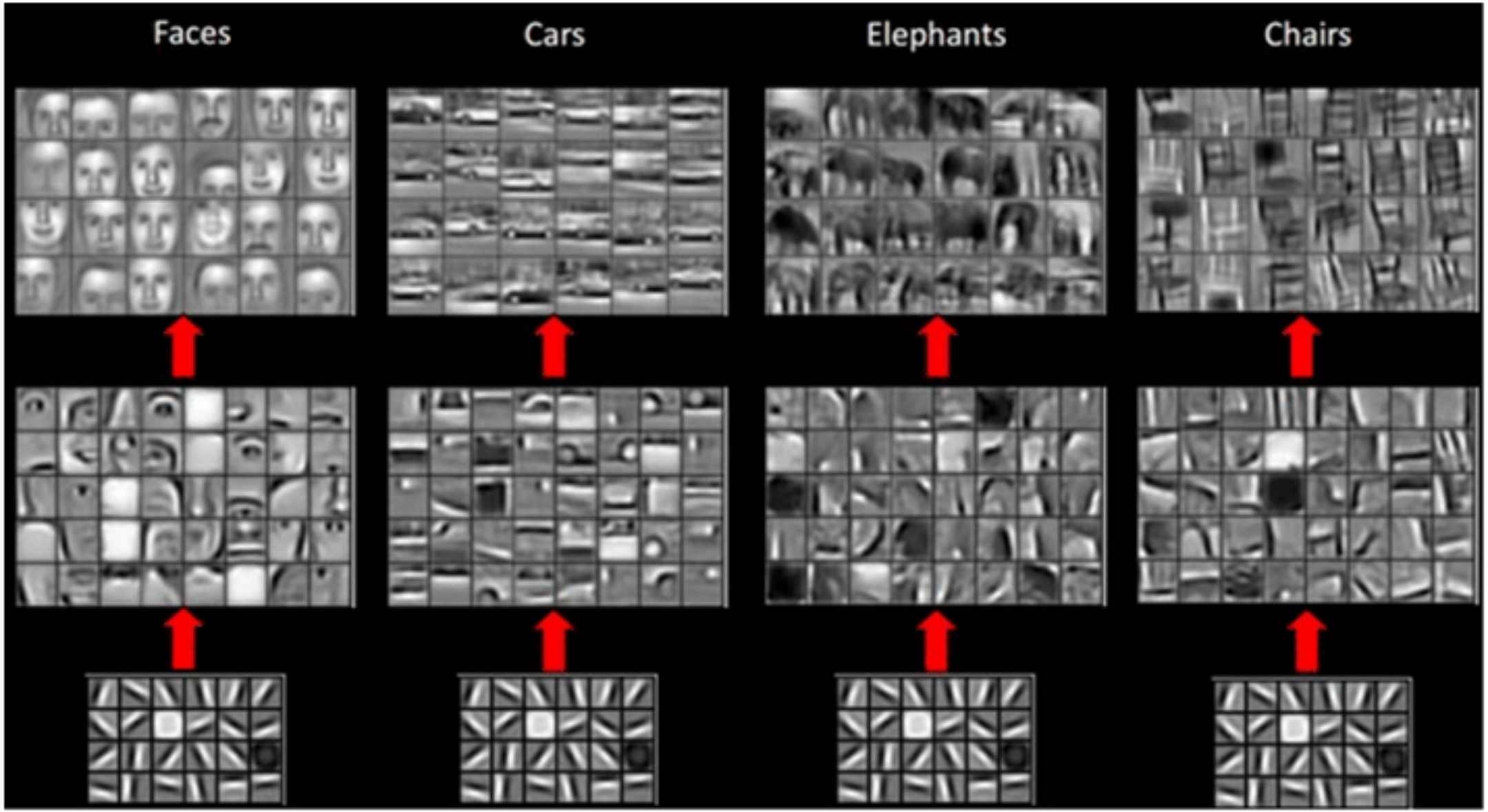
- Let C be a CNN with the following disposition:
  - Input: 32x32x3 images
  - Conv1: 8 3x3 filters, stride 1, padding=same
  - Conv2: 16 5x5 filters, stride 2, padding=same
  - Flatten layer
  - Dense1: 512 nodes
  - Dense2: 4 nodes
- How many parameters does this network have?

$(8 \times 3 \times 3 \times 3 + 8) + (16 \times 5 \times 5 \times 8 + 16) + (16 \times 16 \times 16 \times 512 + 512) + (512 \times 4 + 4)$

Conv1 Conv2 Dense1 Dense2

What do CNN layers learn?

- Each CNN layer learns filters of increasing complexity.
- The first layers learn basic feature detection filters: edges, corners, etc.
- The middle layers learn filters that detect parts of objects. For faces, they might learn to respond to eyes, noses, etc.
- The last layers have higher representations: they learn to recognize full objects, in different shapes and positions.


Faces
Cars
Elephants
Chairs

3D visualization of networks in action

http://scs.ryerson.ca/~aharley/vis/conv/

https://www.youtube.com/watch?v=3JQ3hYko51Y

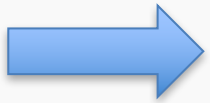
Outline

1. Review from last lecture
2. **BackProp of MaxPooling layer**
3. A bit of history
4. Layers Receptive Field
5. Saliency maps
6. Transfer Learning
7. CNN for text analysis (example)

Backward propagation of Maximum Pooling Layer

Forward mode, 3x3 stride 1

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



Backward propagation of Maximum Pooling Layer

Forward mode, 3x3 stride 1

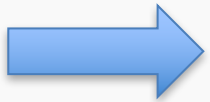
2	4	8	3	6
9	3	4	2	5
5	4	6	3	1
2	3	1	3	4
2	7	4	5	7

9		

Backward propagation of Maximum Pooling Layer

Forward mode, 3x3 stride 1

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

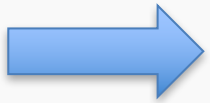


9	8	

Backward propagation of Maximum Pooling Layer

Forward mode, 3x3 stride 1

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



9	8	8

Backward propagation of Maximum Pooling Layer

Forward mode, 3x3 stride 1

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

9	8	8
9	6	6
7	7	7

Backward propagation of Maximum Pooling Layer

Backward mode. Large fonts represents the values of the derivatives of the current layer (max-pool) and small font the corresponding value of the previous layer.

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

1_{9}	3_{8}	1_{8}
1_{9}	4_{6}	2_{6}
6_{7}	2_{7}	1_{7}

Backward propagation of Maximum Pooling Layer

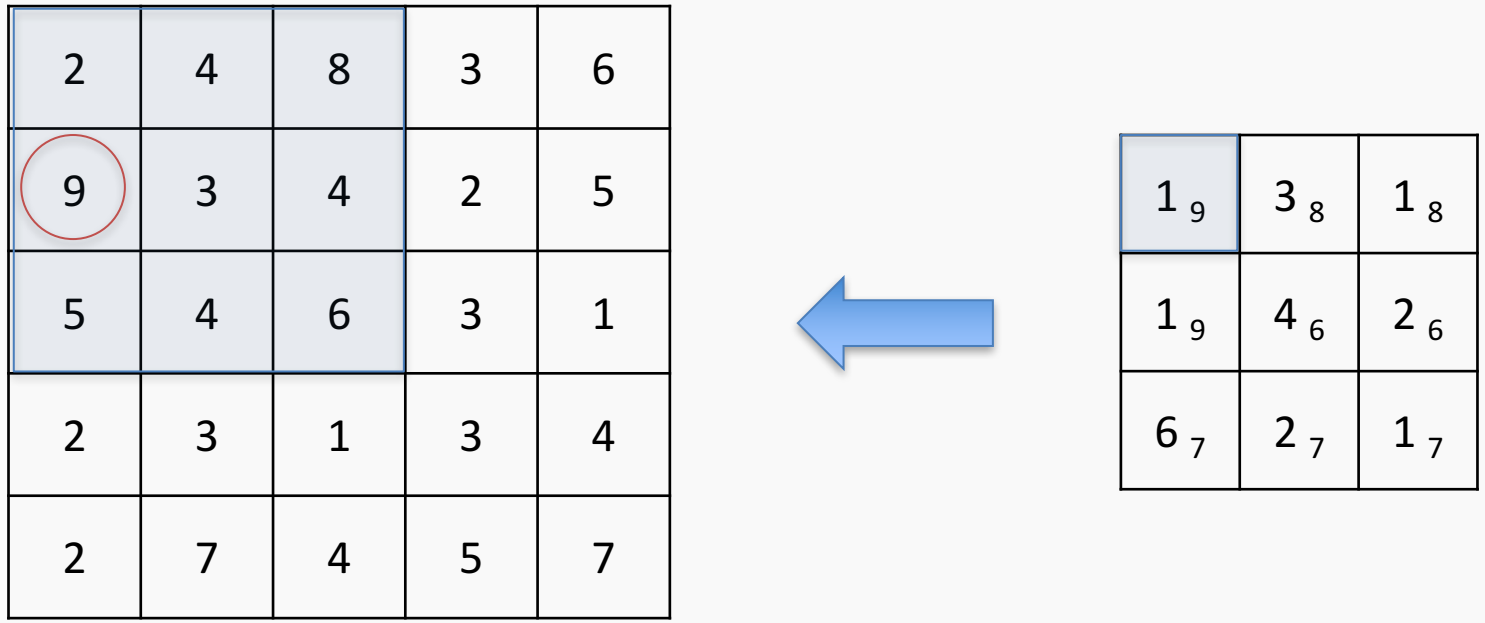
Backward mode. Large fonts represents the values of the derivatives of the current layer (max-pool) and small font the corresponding value of the previous layer.

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

1_{9}	3_{8}	1_{8}
1_{9}	4_{6}	2_{6}
6_{7}	2_{7}	1_{7}

Backward propagation of Maximum Pooling Layer

Backward mode. Large fonts represents the values of the derivatives of the current layer (max-pool) and small font the corresponding value of the previous layer.



Backward propagation of Maximum Pooling Layer

Backward mode. Large fonts represents the values of the derivatives of the current layer (max-pool) and small font the corresponding value of the previous layer.

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

1_{9}	3_{8}	1_{8}
1_{9}	4_{6}	2_{6}
6_{7}	2_{7}	1_{7}

Backward propagation of Maximum Pooling Layer

Backward mode. Large fonts represents the values of the derivatives of the current layer (max-pool) and small font the corresponding value of the previous layer.

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

1_{9}	3_{8}	1_{8}
1_{9}	4_{6}	2_{6}
6_{7}	2_{7}	1_{7}

Backward propagation of Maximum Pooling Layer

Backward mode. Large fonts represents the values of the derivatives of the current layer (max-pool) and small font the corresponding value of the previous layer.

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

1_{9}	3_{8}	1_{8}
1_{9}	4_{6}	2_{6}
6_{7}	2_{7}	1_{7}

Backward propagation of Maximum Pooling Layer

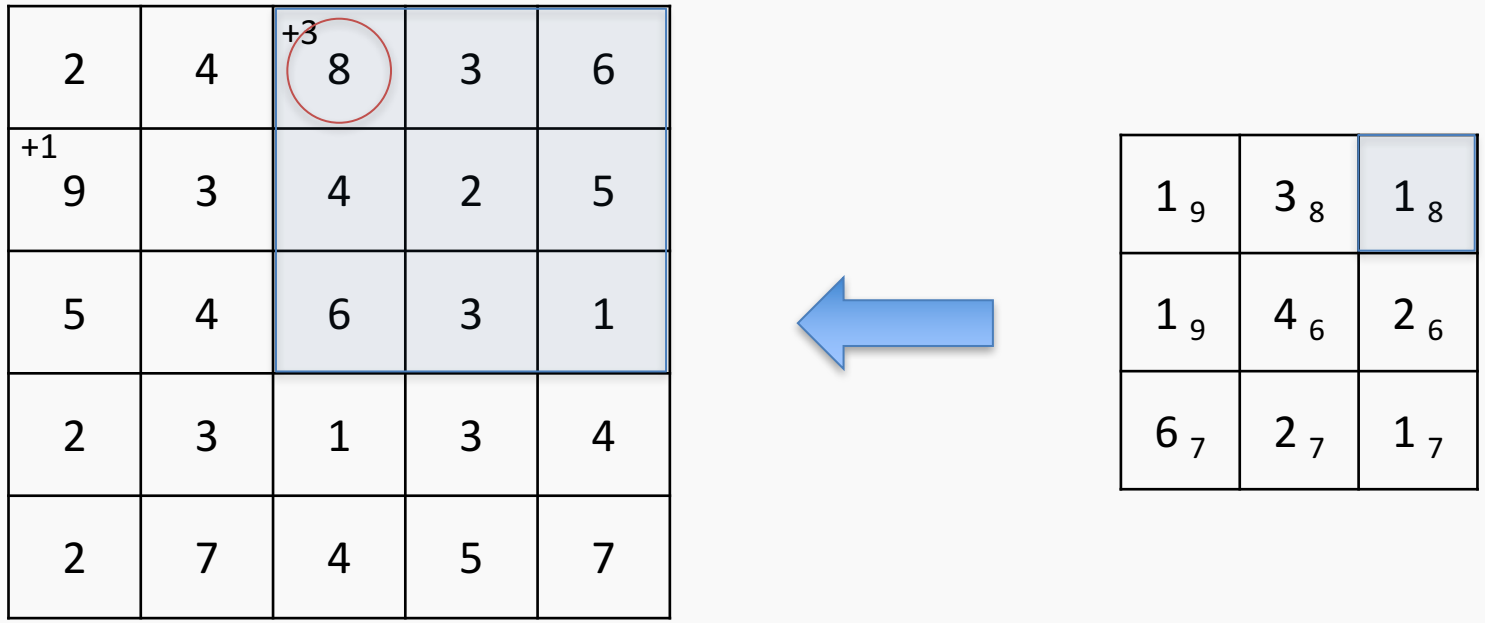
Backward mode. Large fonts represents the values of the derivatives of the current layer (max-pool) and small font the corresponding value of the previous layer.

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

1_{9}	3_{8}	1_{8}
1_{9}	4_{6}	2_{6}
6_{7}	2_{7}	1_{7}

Backward propagation of Maximum Pooling Layer

Backward mode. Large fonts represents the values of the derivatives of the current layer (max-pool) and small font the corresponding value of the previous layer.



Backward propagation of Maximum Pooling Layer

Backward mode. Large fonts represents the values of the derivatives of the current layer (max-pool) and small font the corresponding value of the previous layer.

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

1_{9}	3_{8}	1_{8}
1_{9}	4_{6}	2_{6}
6_{7}	2_{7}	1_{7}

Outline

1. Review from last lecture
2. BackProp of MaxPooling layer
3. **A bit of history**
4. Layers Receptive Field
5. Saliency maps
6. Transfer Learning
7. CNN for text analysis (example)

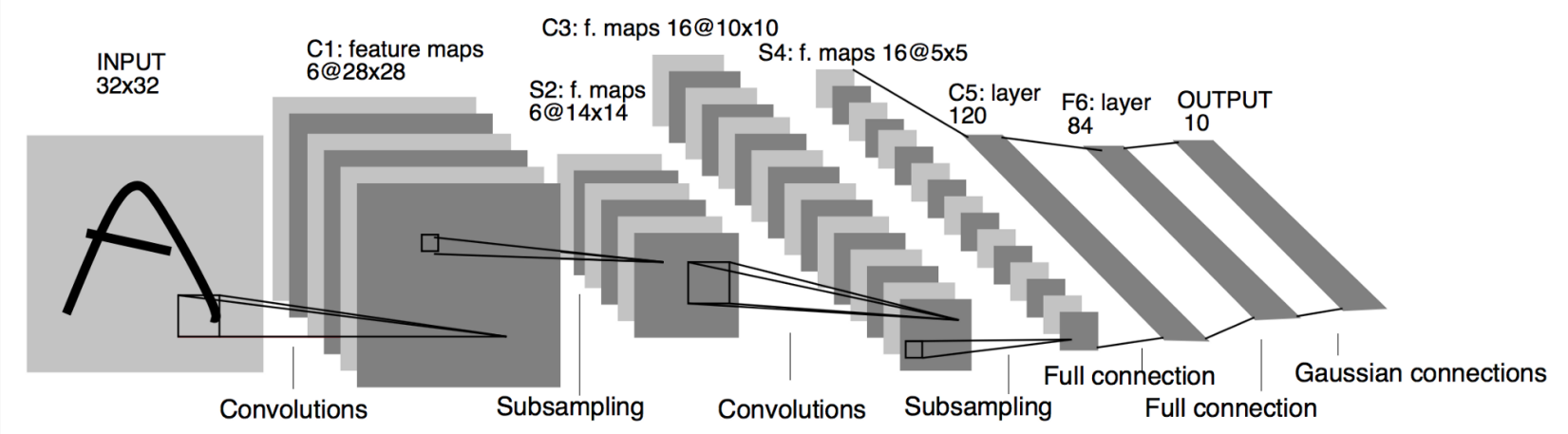
Initial ideas

- The first piece of research proposing something similar to a Convolutional Neural Network was authored by Kunihiko Fukushima in 1980, and was called the NeoCognitron[1].
- Inspired by discoveries on visual cortex of mammals.
- Fukushima applied the NeoCognitron to hand-written character recognition.
- End of the 80's: several papers advanced the field
  - Backpropagation published in French by Yann LeCun in 1985 (independently discovered by other researchers as well)
  - TDNN by Waiber et al., 1989 - Convolutional-like network trained with backprop.
  - Backpropagation applied to handwritten zip code recognition by LeCun et al., 1989

[1] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics, 36(4): 93-202, 1980.

LeNet

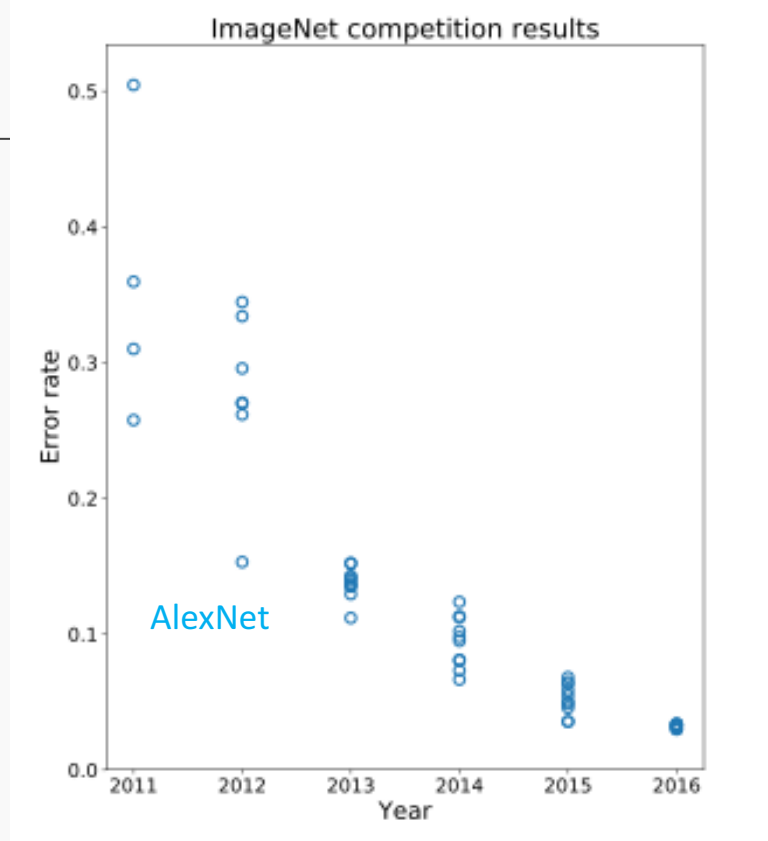
- November 1998: LeCun publishes one of his most recognized papers describing a “modern” CNN architecture for document recognition, called LeNet[1].
- Not his first iteration, this was in fact LeNet-5, but this paper is the commonly cited publication when talking about LeNet.

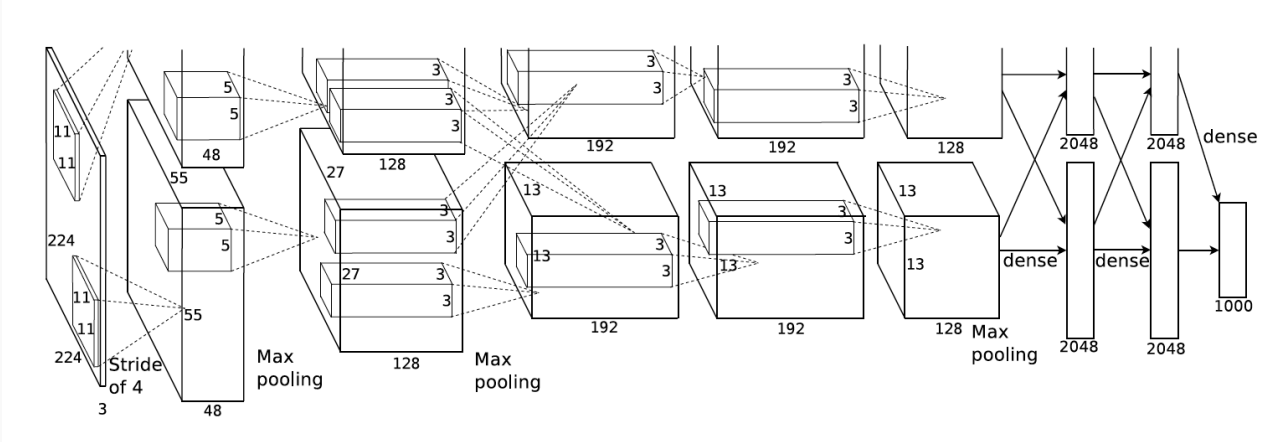


[1] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 (1998): 2278-2324.

AlexNet

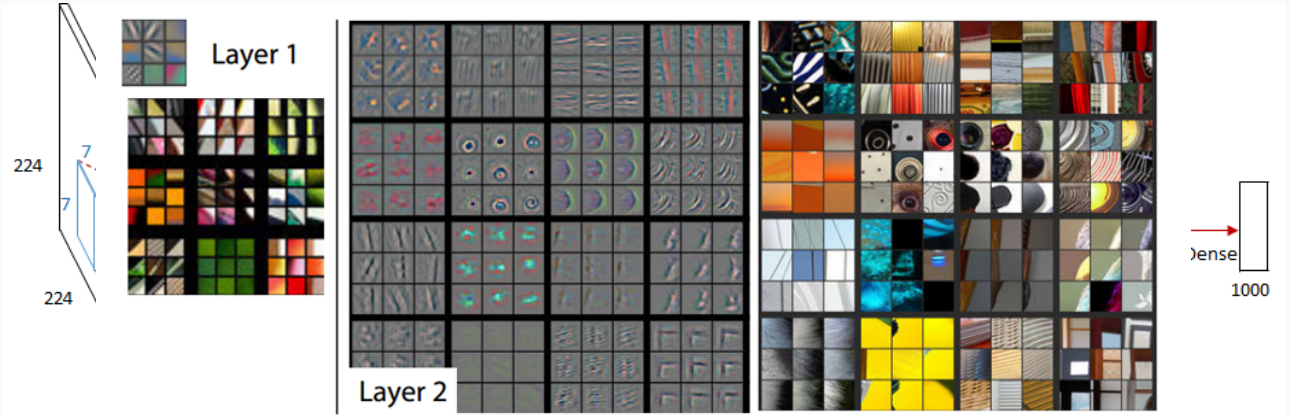
- Developed by Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton at Utoronto in 2012. More than 25000 citations.
- Destroyed the competition in the 2012 ImageNet Large Scale Visual Recognition Challenge. Showed benefits of CNNs and kickstarted AI revolution.
- top-5 error of 15.3%, more than 10.8 percentage points lower than runner-up.
- Main contributions:
  - Trained on ImageNet with data augmentation
  - Increased depth of model, GPU training (*five to six days*)
  - Smart optimizer and Dropout layers
  - ReLU activation!





ZFNet

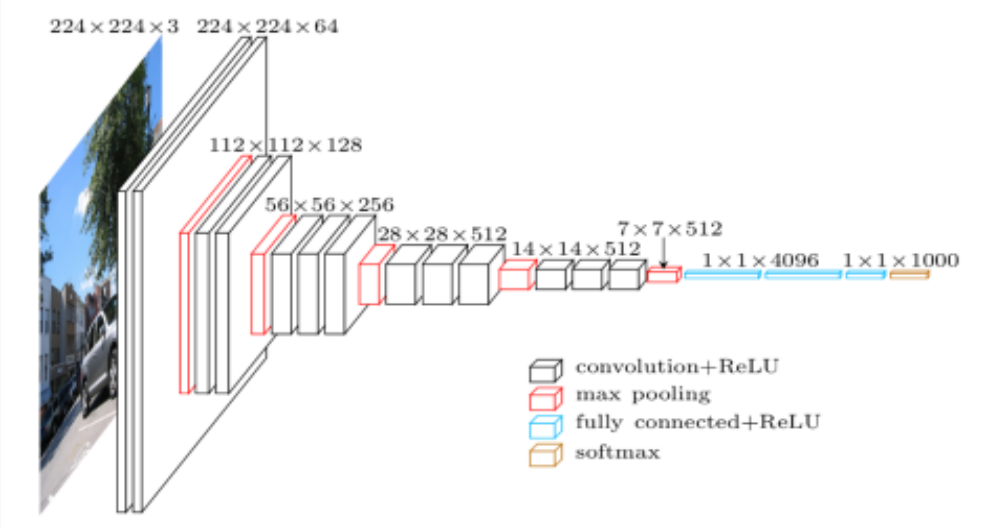
- Introduced by Matthew Zeiler and Rob Fergus from NYU, won ILSVRC 2013 with 11.2% error rate. Decreased sizes of filters.
- Trained for 12 days.
- Paper presented a visualization technique named **Deconvolutional Network**, which helps to examine different feature activations and their relation to the input space.



Visualizations of Layer 1 and 2. Each layer illustrates 2 pictures, one which shows the filters themselves and one that shows what part of the image are most strongly activated by the given filter. For example, in the space labled Layer 2, we have representations of the 16 different filters (on the left)

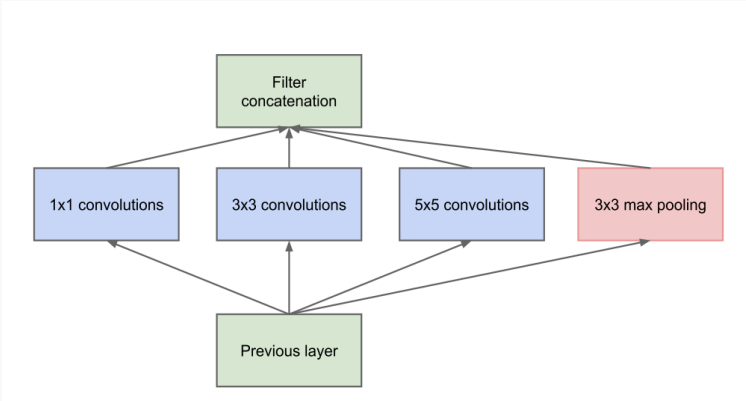
VGG

- Introduced by Simonyan and Zisserman (Oxford) in 2014
- Simplicity and depth as main points. Used 3x3 filters exclusively and 2x2 MaxPool layers with stride 2.
- Showed that two 3x3 filters have an effective receptive field of 5x5.
- As spatial size decreases, depth increases.
- Trained for *two to three weeks*.
- Still used as of today.

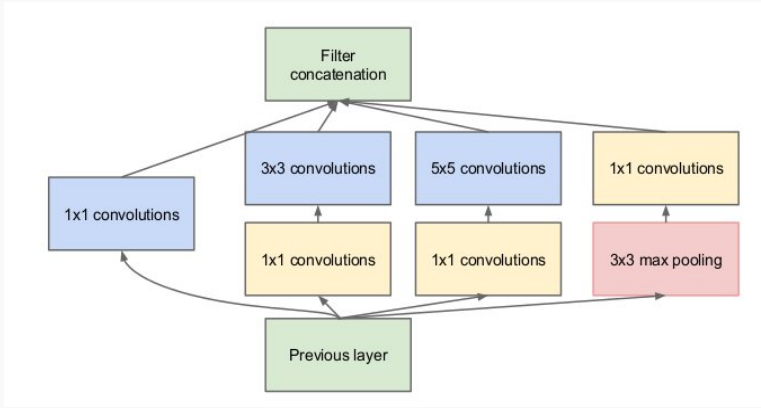


GoogLeNet (Inception-v1)

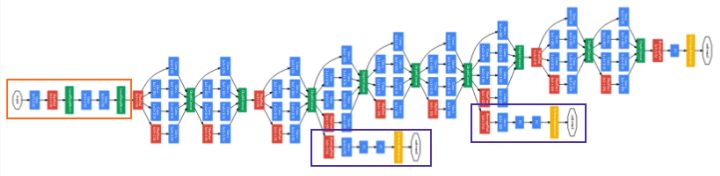
- Introduced by Szegedy et al. (Google), 2014. Winners of ILSVRC 2014.
- Introduces inception module: parallel conv. layers with different filter sizes. Motivation: we don't know which filter size is best – let the network decide. Key idea for future archs.
- No fully connected layer at the end. AvgPool instead. 12x fewer params than AlexNet.



1x1 convs to Reduce number of parameters



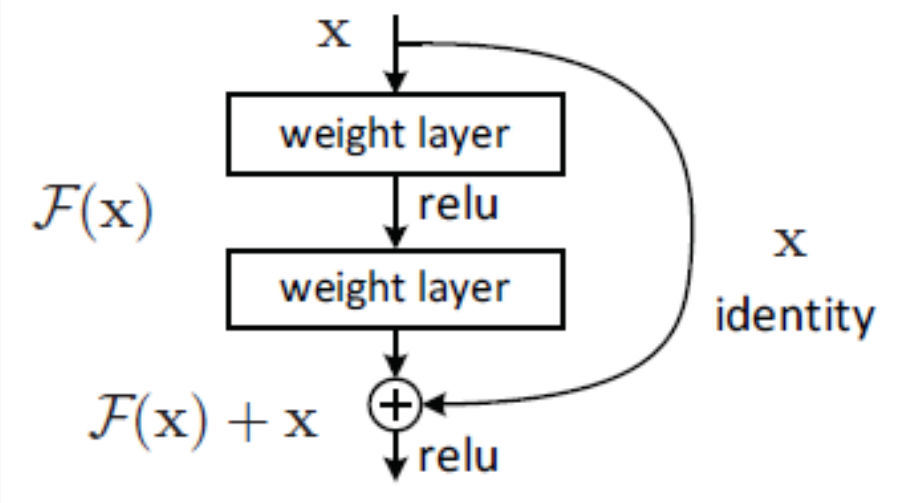
Proto Inception module



Inception module

ResNet

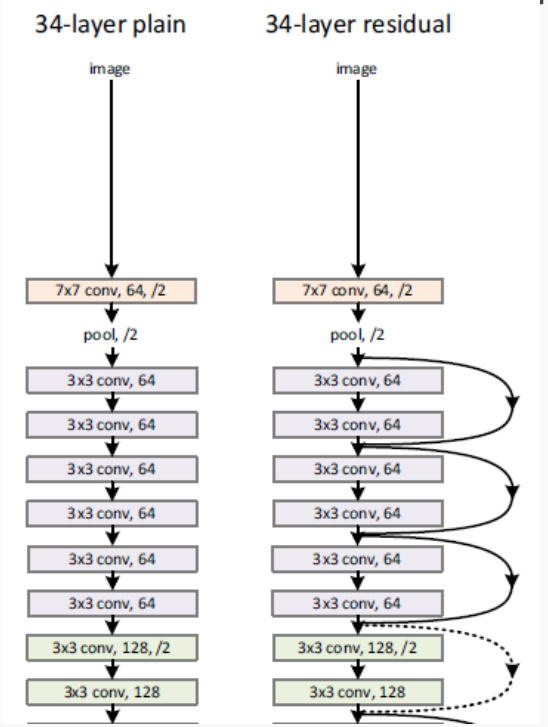
- Presented by He et al. (Microsoft), 2015. Won ILSVRC 2015 in multiple categories.
- Main idea: Residual block. Allows for extremely deep networks.
- Authors believe that it is easier to optimize the residual mapping than the original one. Furthermore, residual block can decide to "shut itself down" if needed.

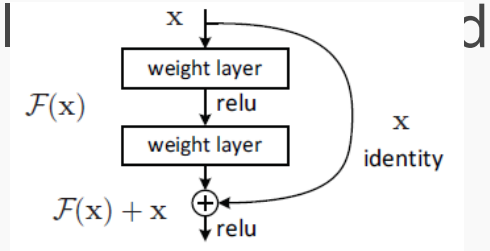


Residual Block

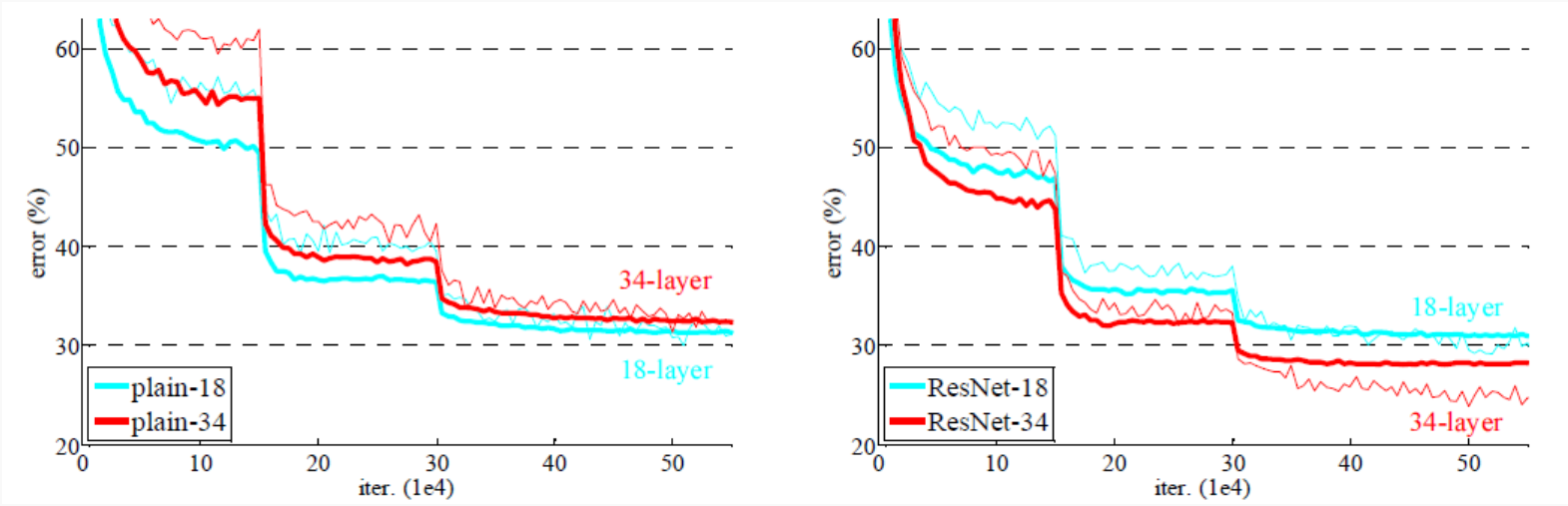
ResNet

- Presented by He et al. (Microsoft), 2015. Won ILSVRC 2015 in multiple categories.
- Main idea: Residual block. Allows for extremely deep networks.
- Authors believe that it is easier to optimize the residual mapping than the original one. Furthermore, residual [illegible]de to "shut itself down" if n[illegible]





Residual Block



DenseNet

- Proposed by Huang et al., 2016. Radical extension of ResNet idea.
- Each block uses every previous feature map as input.
- Idea: n computation of redundant features. All the previous information is available at each point.
- Counter-intuitively, it reduces the number of parameters needed.

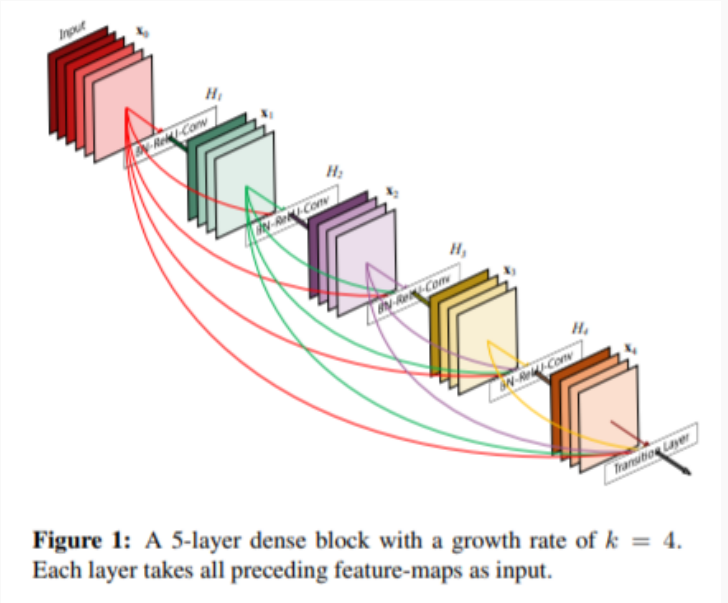


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

DenseNet

- Proposed by Huang et al., 2016. Radical extension of ResNet idea.
- Each block uses every previous feature map as input.
- Idea: n computation of redundant features. All the previous information is available at each point.
- Counter-intuitively, it reduces the number of parameters needed.

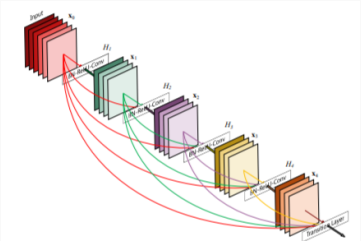
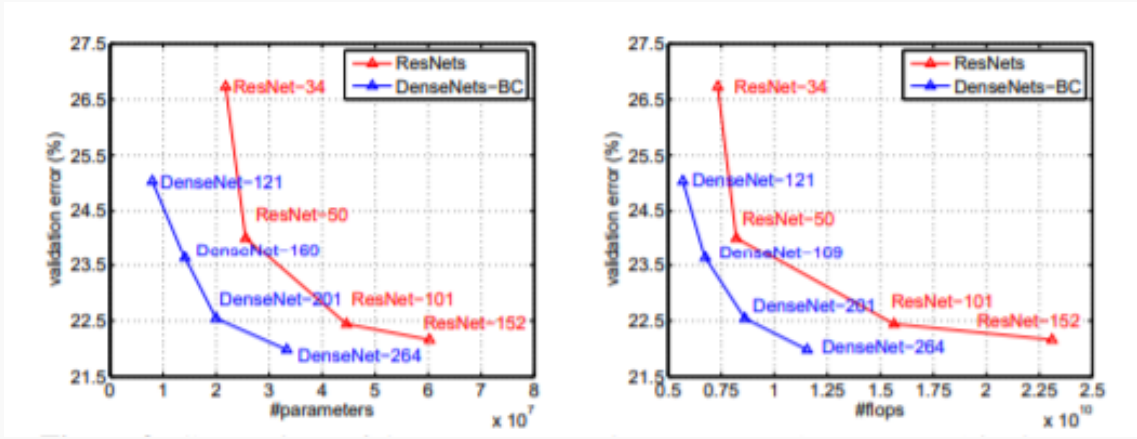
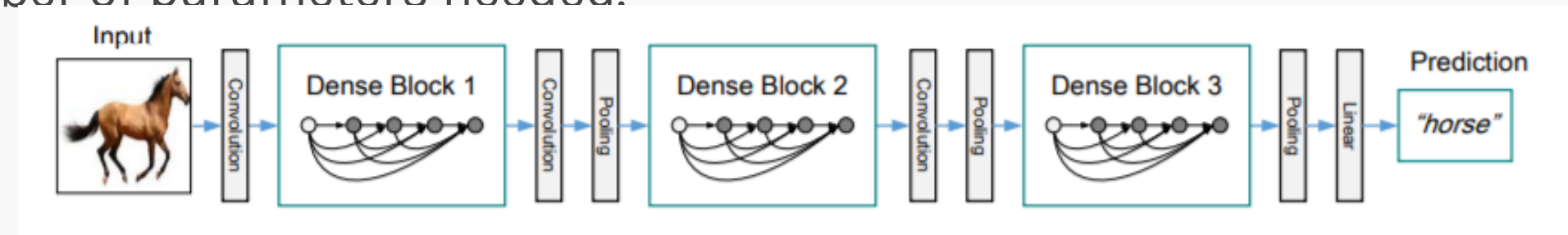


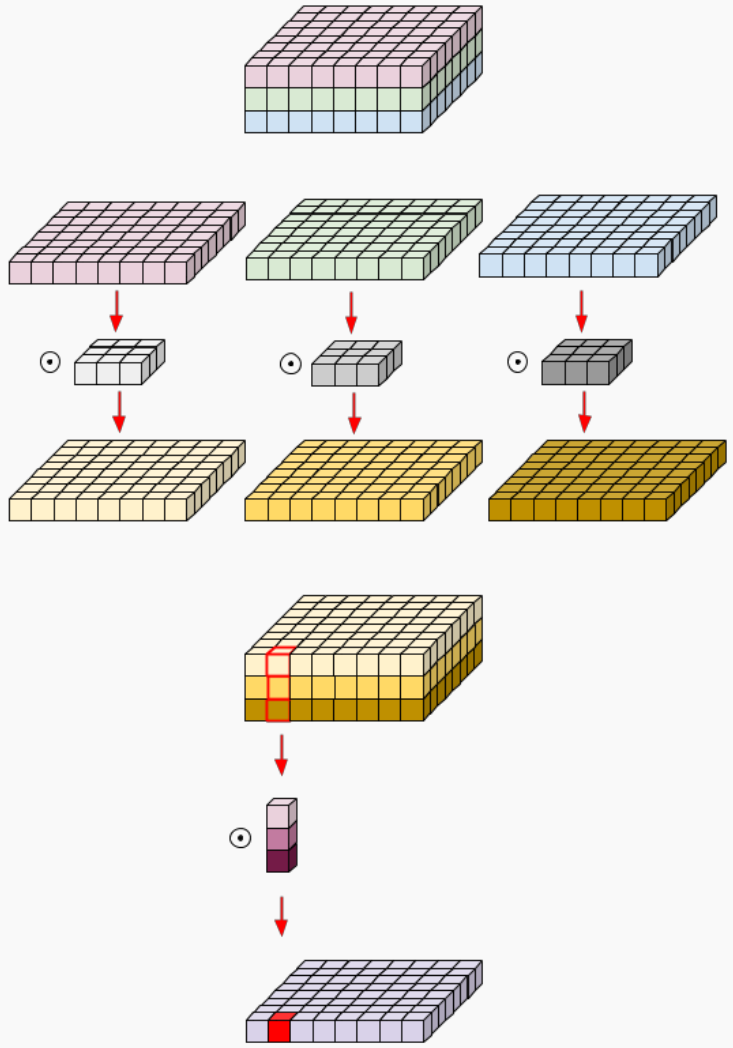
Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.





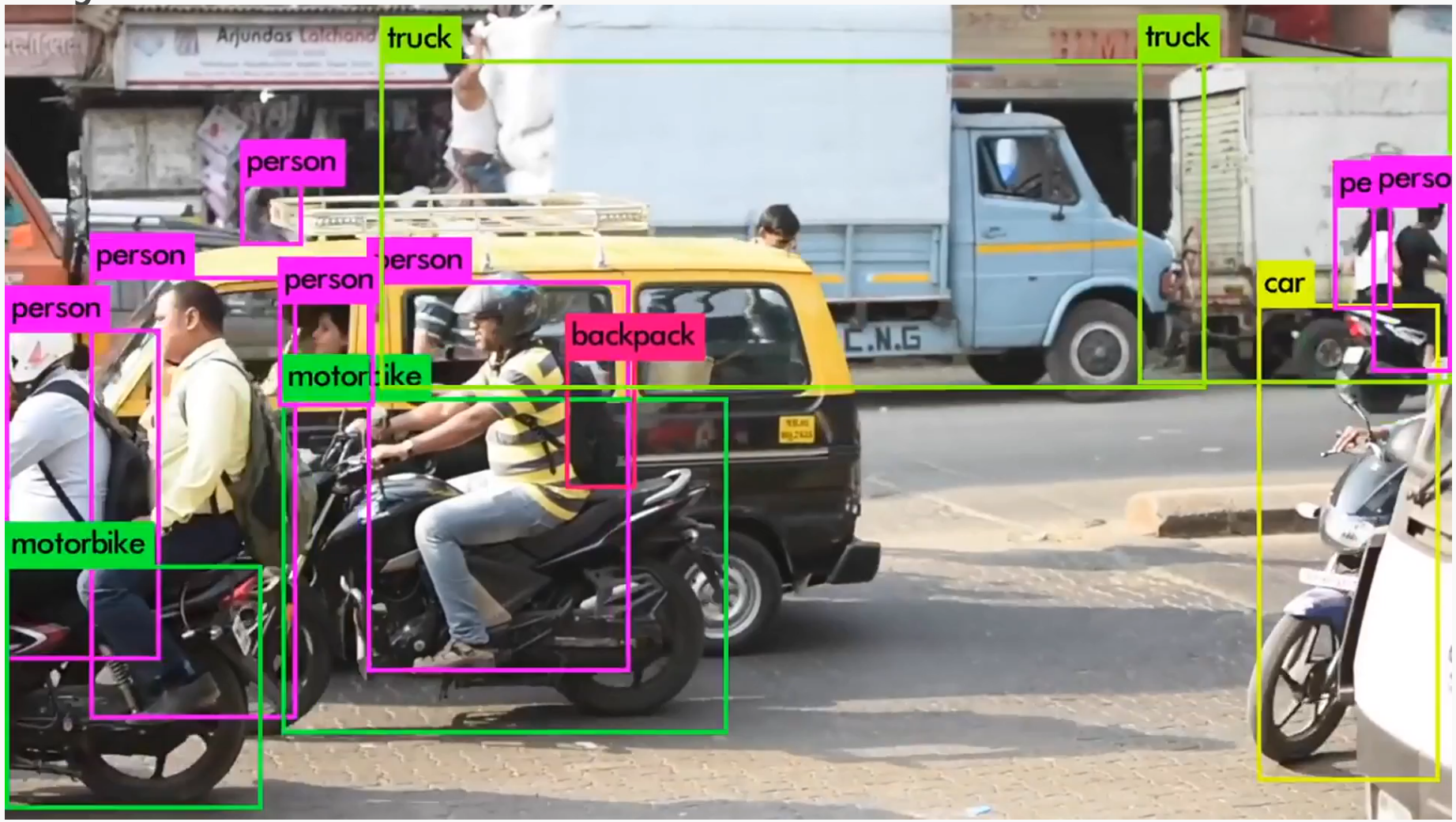
MobileNet

- Published by Howard et al., 2017.
- Extremely efficient network with decent accuracy.
- Main concept: depthwise-separable convolutions. Convolve each feature maps with a kernel, then use a 1x1 convolution to aggregate the result.
- This approximates vanilla convolutions without having to convolve large kernels through channels.



Latest events on Image Recognition

You Only Look Once (YOLO) - 2016



More on the greatest latest at a-sec later today

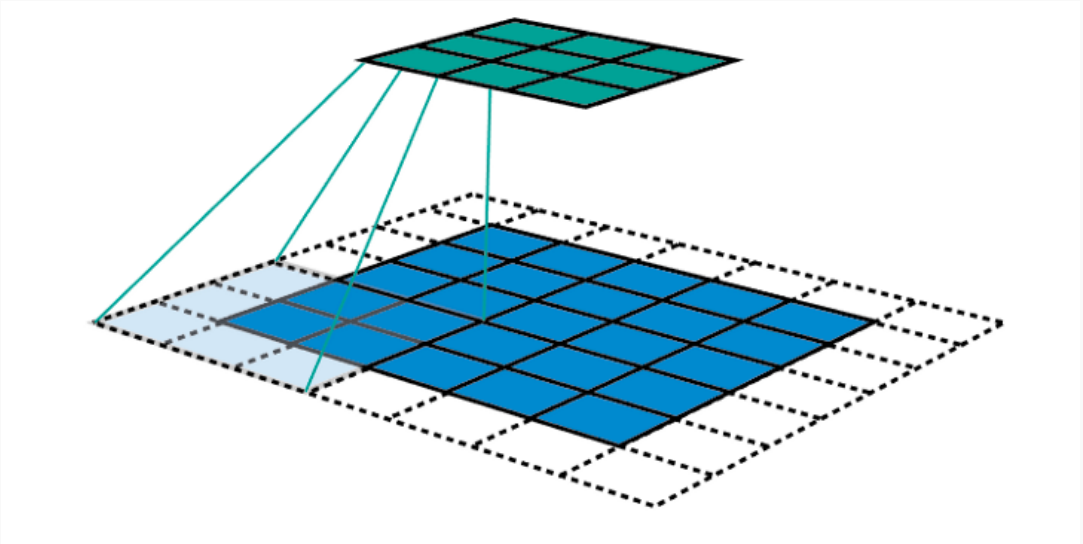
Outline

1. Review from last lecture
2. BackProp of MaxPooling layer
3. A bit of history
4. **Layers Receptive Field**
5. Saliency maps
6. Transfer Learning
7. CNN for text analysis (example)

Layer's Receptive Field

The **receptive field** is defined as the region in the input space that a particular CNN's feature is looking at (i.e. be affected by).

Apply a convolution C with kernel size ***k* = 3x3**, padding size ***p* = 1x1**, stride ***s* = 2x2** on an input map **5x5**, we will get an output feature map **3x3** (green map).

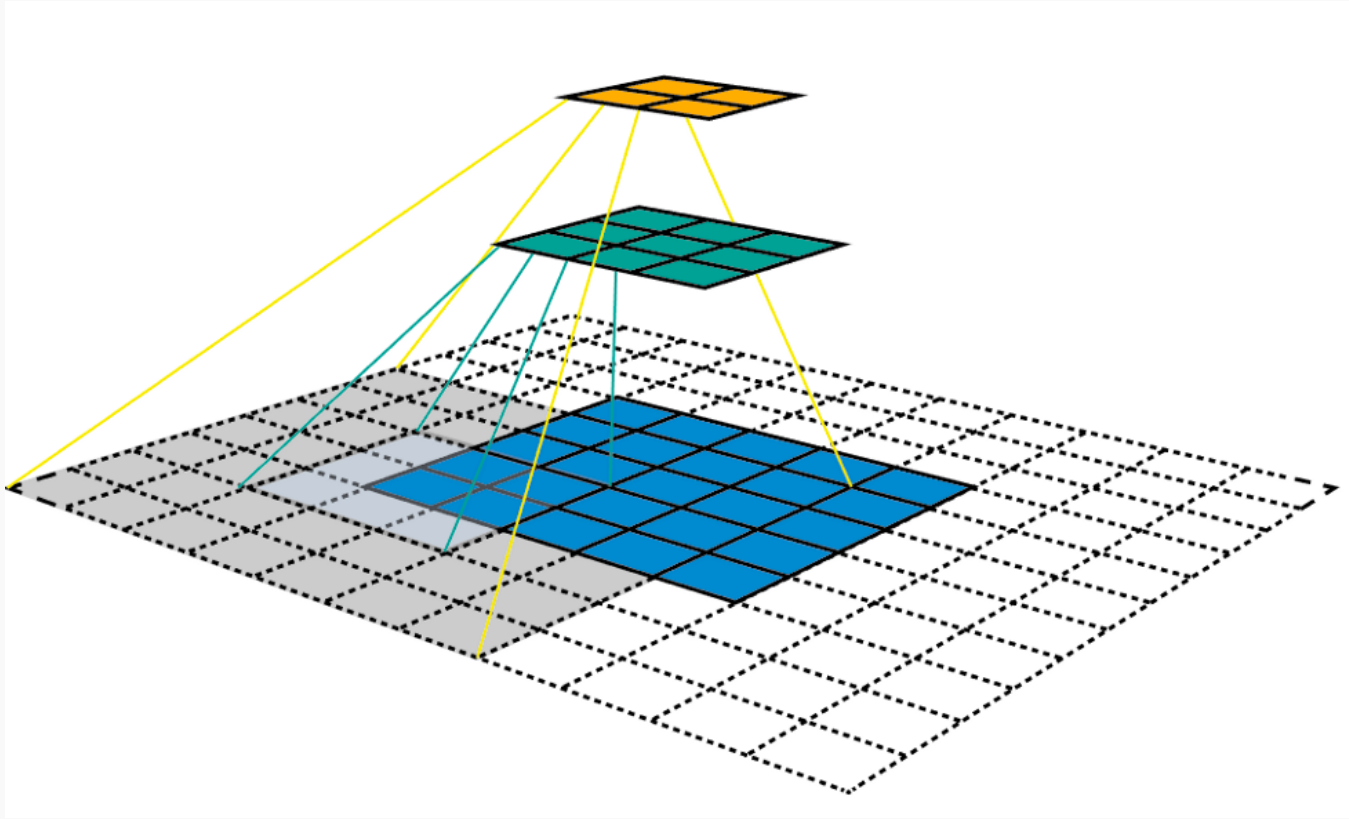


$$n_{out} = \left\lfloor \frac{n_{in} + 2p - k}{s} \right\rfloor + 1$$

n_{in}: number of input features
n_{out}: number of output features
k: convolution kernel size
p: convolution padding size
s: convolution stride size

Layer's Receptive Field

Applying the same convolution on top of the 3x3 feature map, we will get a **2x2** feature map (orange map)

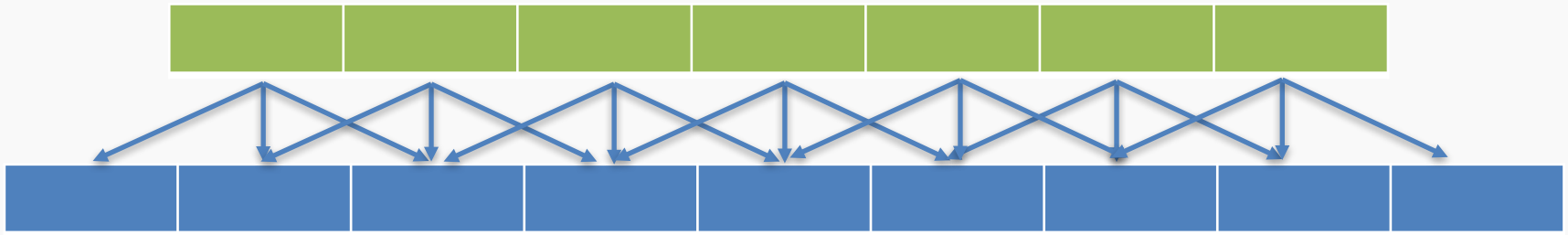


Dilated CNNs

Let's look at the receptive field again in 1D, no padding, stride 1 and kernel 3x1

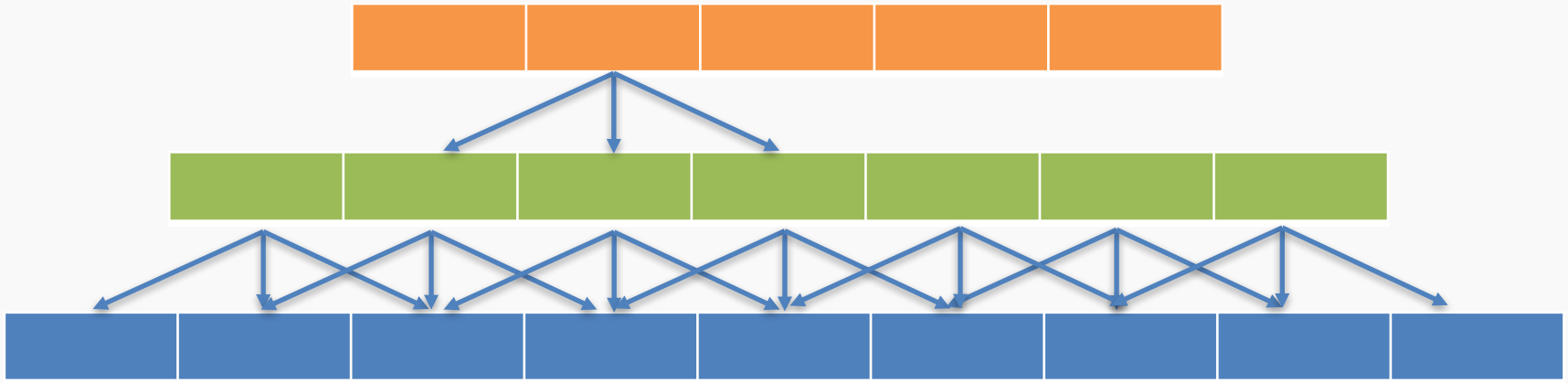
Dilated CNNs (cont)

Let's look at the receptive field again in 1D, no padding, stride 1 and kernel 3x1



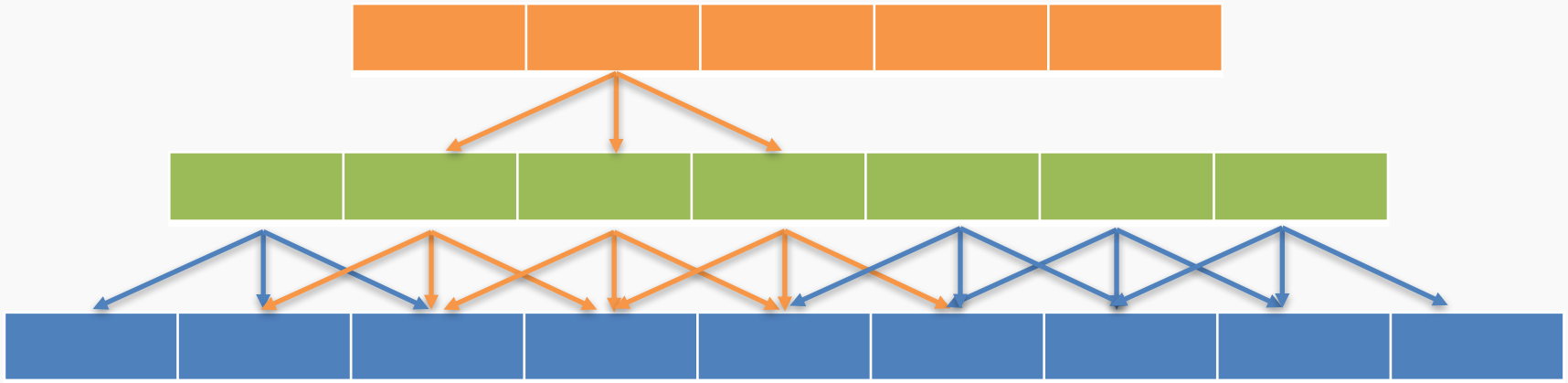
Dilated CNNs (cont)

Let's look at the receptive field again in 1D, no padding, stride 1 and kernel 3x1



Dilated CNNs (cont)

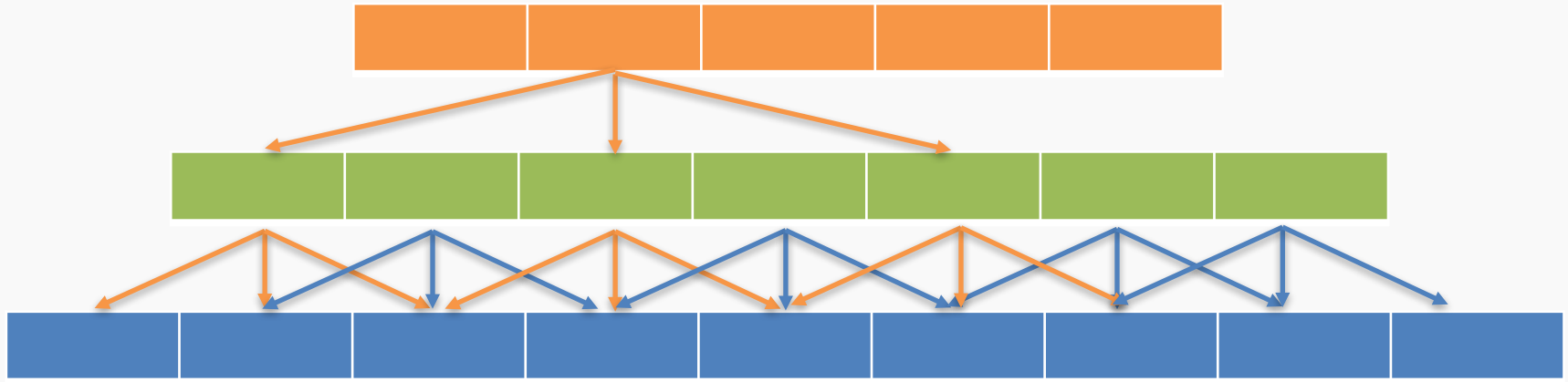
Let's look at the receptive field again in 1D, no padding, stride 1 and kernel 3x1



Dilated CNNs (cont)

Let's look at the receptive field again in 1D, no padding, stride 1 and kernel 3x1.

Skip some of the connections



Outline

1. Review from last lecture
2. BackProp of MaxPooling layer
3. A bit of history
4. Layers Receptive Field
5. **Saliency maps**
6. Transfer Learning
7. CNN for text analysis (example)

Saliency maps

Saliency maps (cont)

If you are given an image of a dog and asked to classify it. Most probably you will answer immediately – Dog! But your Deep Learning Network might not be as smart as you. It might classify it as a cat, a lion or Pavlos!

What are the reasons for that?

- bias in training data
- no regularization
- or your network has seen too many celebrities

Saliency maps (cont)

We want to understand what made my network give a certain class as output?

Saliency Maps, they are a way to measure the spatial support of a particular class in a given image.

"Find me pixels responsible for the class C having score S(C) when the image I is passed through my network".

Saliency maps (cont)

We want to understand what made my network give a certain class as output?

Saliency Maps, they are a way to measure the spatial support of a particular class in a given image.

"Find me pixels responsible for the class C having score S(C) when the image I is passed through my network".

Salience maps (cont)

Question: How do we do that?

We differentiate!

For any function $f(x, y, z)$, we can find the impact of variables x, y, z on $f$at any specific point $(x0, y0, z0)$ by finding its partial derivative w.r.t these variables at that point.

Similarly, to find the responsible pixels, we take the score function S, for class C and take the partial derivatives w.r.t **every pixel.**

Salience maps (cont)

Question: Easy Peasy?

Sort of! Auto-grad can do this!

1. Forwar pass of the image through the network
2. Calculate the scores for every class
3. Enforce derivative of score *S* at last layer for all classes except class *C* to be 0. For *C*, set it to 1
4. Backpropagate this derivative till the start
5. Render them and you have your Saliency Map!

Note: On step #2. Instead of doing softmax, we turn it to binary classification and use the probabilities.

Salience maps (cont)

Salience maps (cont)

Question: What do we do with color images?
Take the saliency map for each channel and either take the max or average or use all 3 channels.

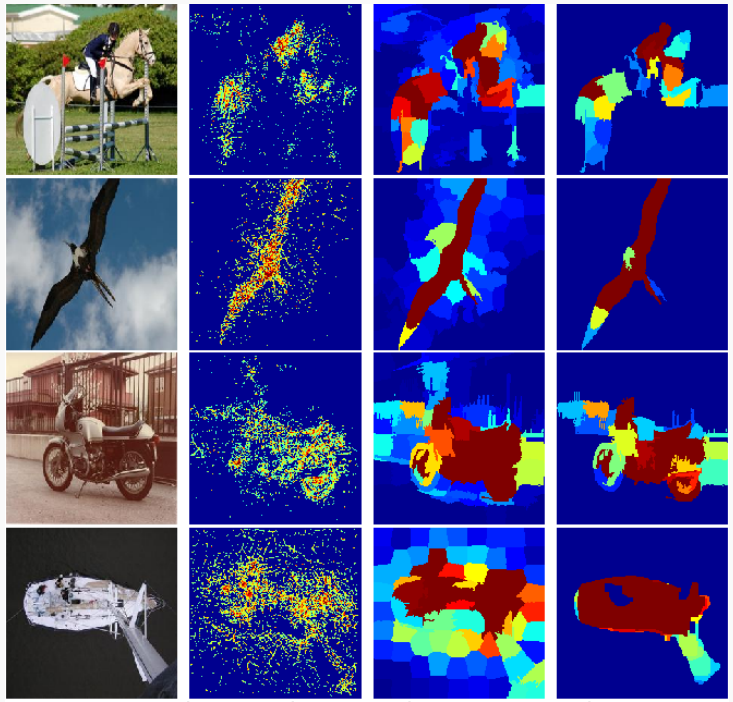


Figure 2. From left to right: original images, raw saliency maps,

[1]: Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps
[2]: Attention-based Extraction of Structured Information from Street View Imagery

Transposed Convolution

So far: convolution either maintain the size of their input or make it smaller.

We can use the same technique to also make the input tensor larger. This process is called **upsampling**.

When we do it inside of a convolution step, it's called **transposed convolution** or fractional striding.

Note: Some authors call upsampling while convolving **deconvolution**, but that name is already taken by a different idea [Zeiler 10, https://arxiv.org/pdf/1311.2901.pdf]..

Transposed Convolution (cont)

So far: convolution either maintain the size of their input or make it smaller.

We can use the same technique to also make the input tensor larger. This process is called **upsampling**.

When we do it inside of a convolution step, it's called **transposed convolution** or fractional striding.

Note: Some authors call upsampling while convolving **deconvolution**, but that name is already taken by a different idea [Zeiler 10, https://www.matthewzeiler.com/mattzeiler/deconvolutionalnetworks.pdf]

Transposed Convolution (cont)

Conv with no padding.

Original image:5x5

After conv: 3x3

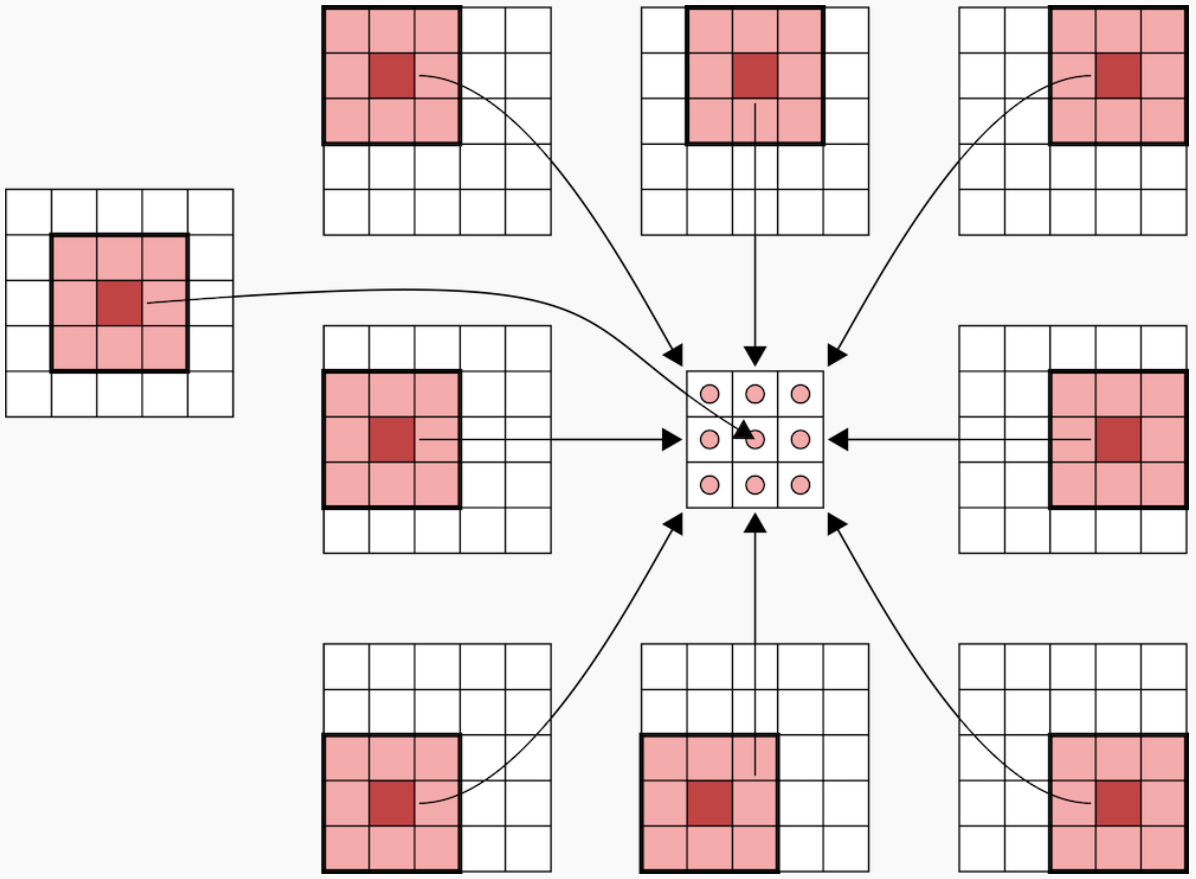


Image taken from A. Glassner, Deep Learning, Vol. 2: From Basics to Practice

Transposed Convolution (cont)

Conv with padding.

Original image:5x5

After conv: 5x5

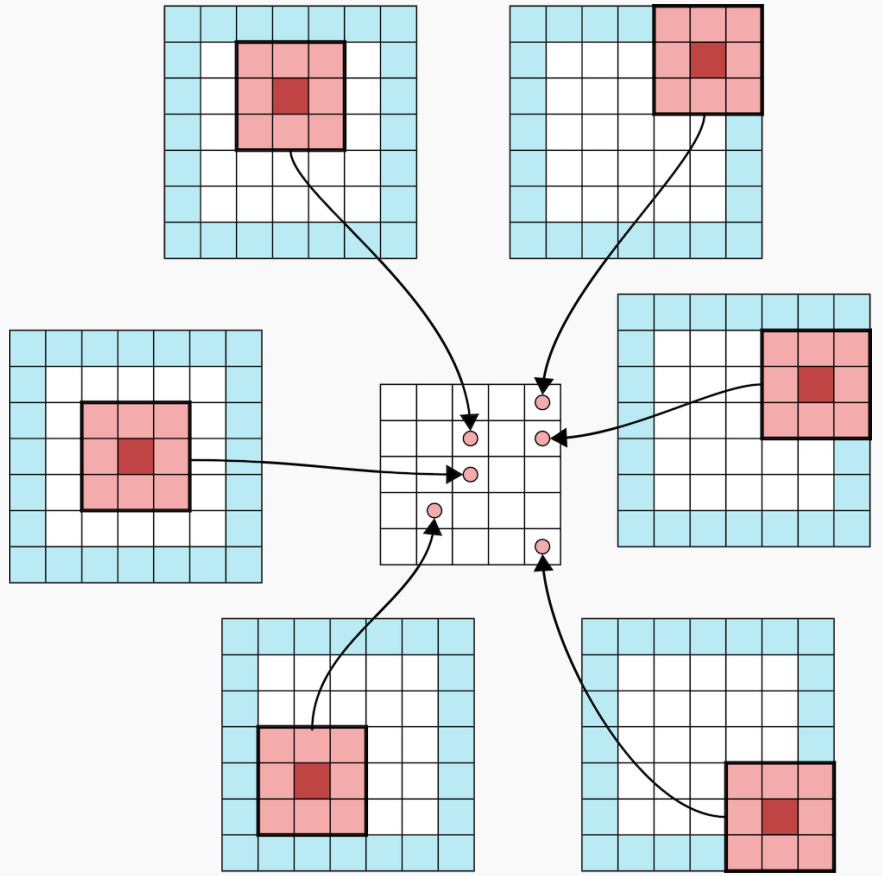


Image taken from A. Glassner, Deep Learning, Vol. 2: From Basics to Practice

Transposed Convolution (cont)

Conv with padding 2.

Original image:3x3

After conv: 5x5

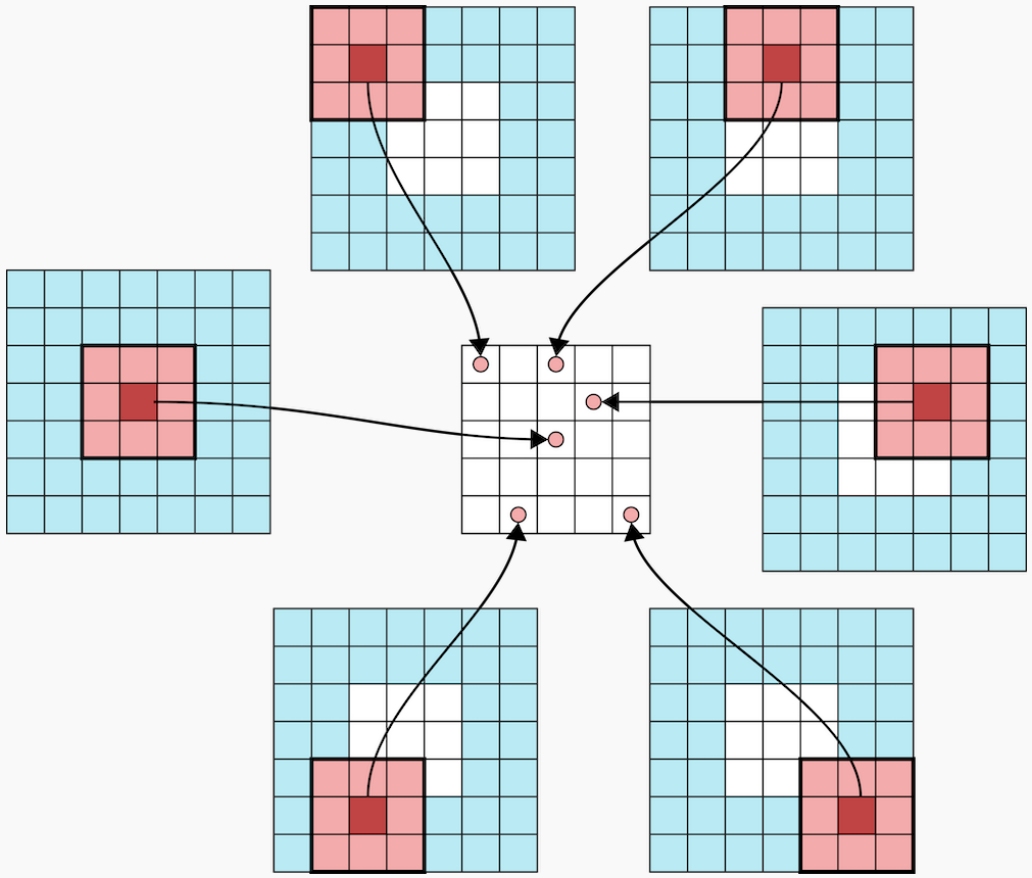


Image taken from A. Glassner, Deep Learning, Vol. 2: From Basics to Practice

Outline

1. Review from last lecture
2. BackProp of MaxPooling layer
3. A bit of history
4. Layers Receptive Field
5. Saliency maps
6. **Transfer Learning**
7. CNN for text analysis (example)

Transfer Learning

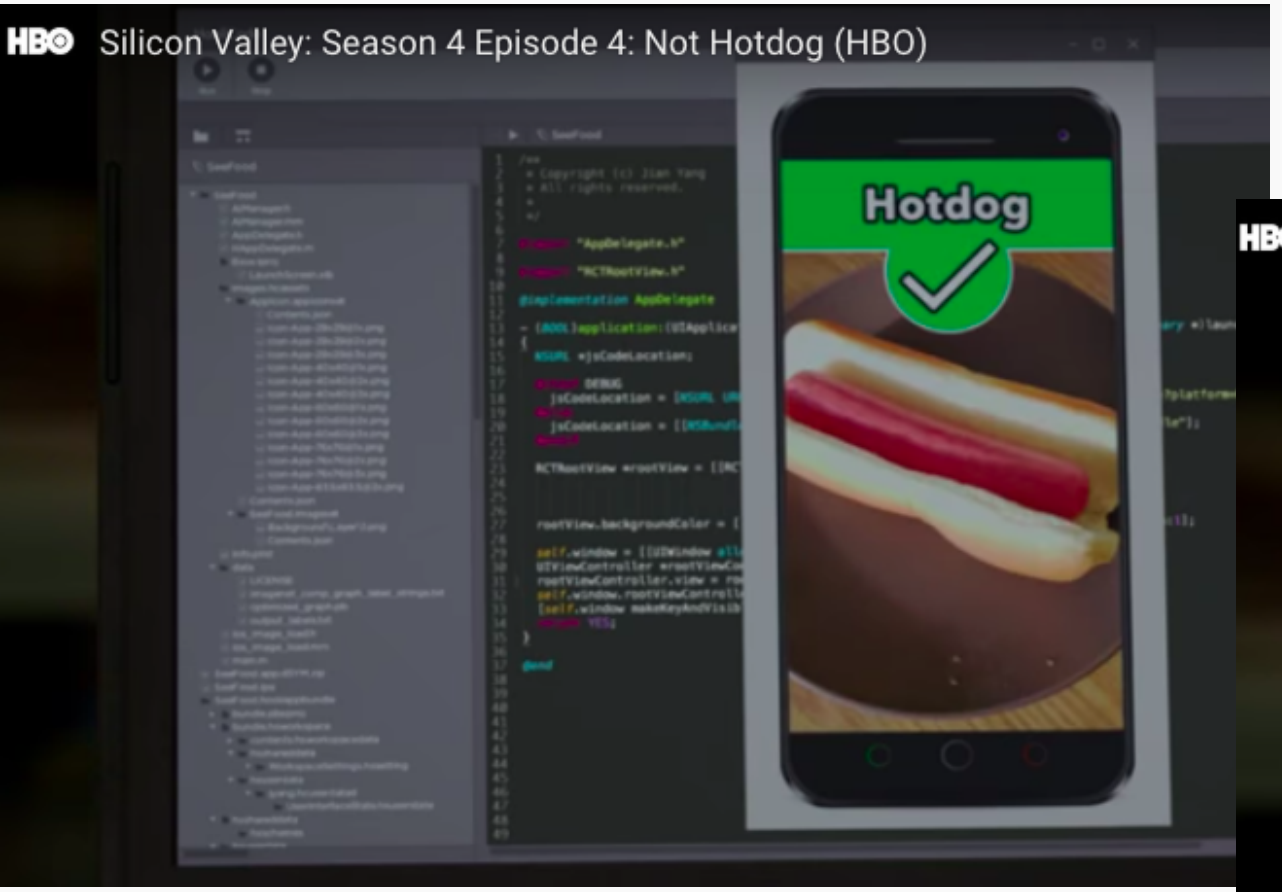
How do you make an image classifier that can be trained in a few hours (minutes) on a CPU?

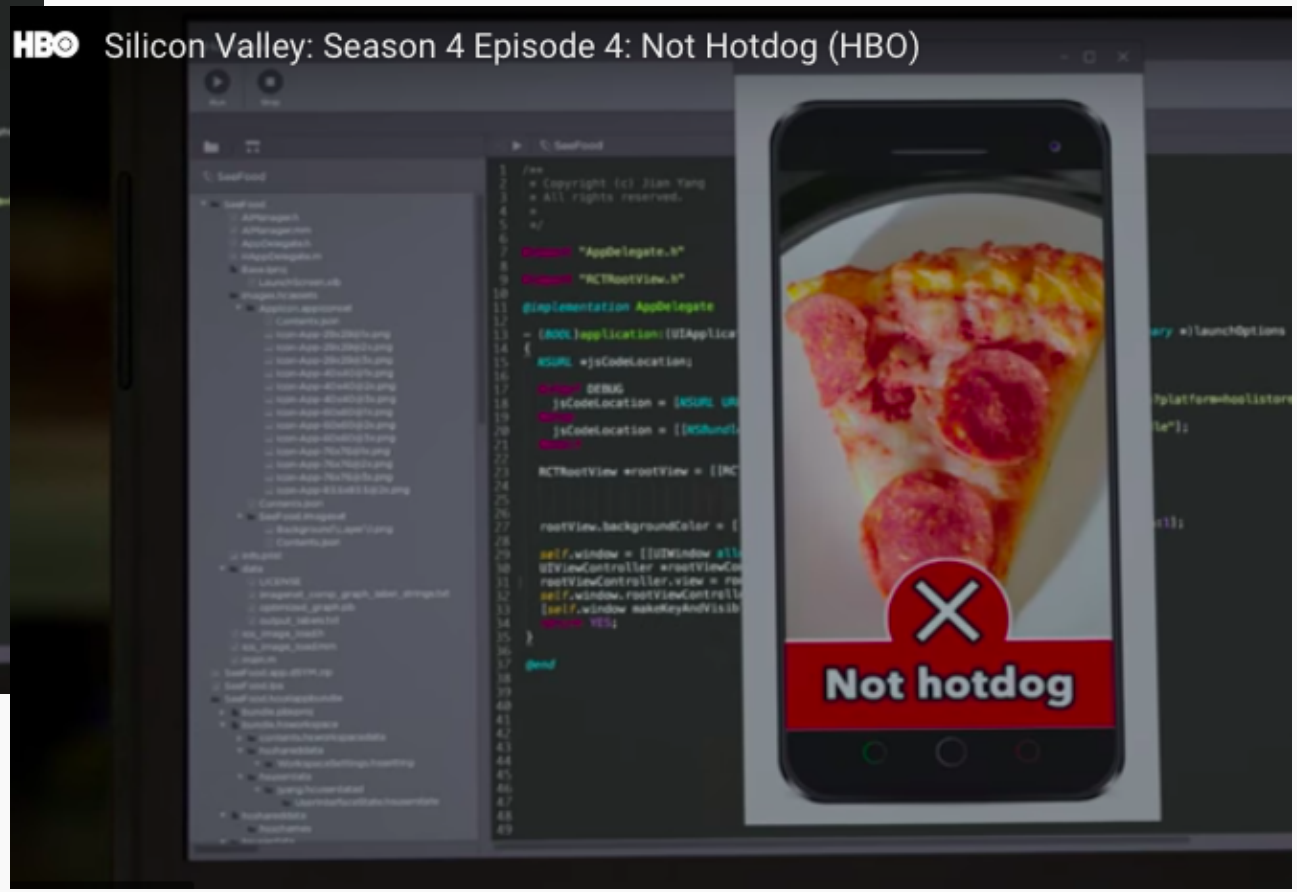
Use pre-trained models, i.e., models with known weights.

Main Idea: earlier layers of a network learn low level features, which can be adapted to new domains by changing weights at later and fully-connected layers.

Example: use Imagenet trained with any sophisticated huge network. Then retrain it on a few thousand hotdog images and you get...

Hotdog or NotHotDog: https://youtu.be/ACmydtFDTGs
(offensive language and tropes alert)

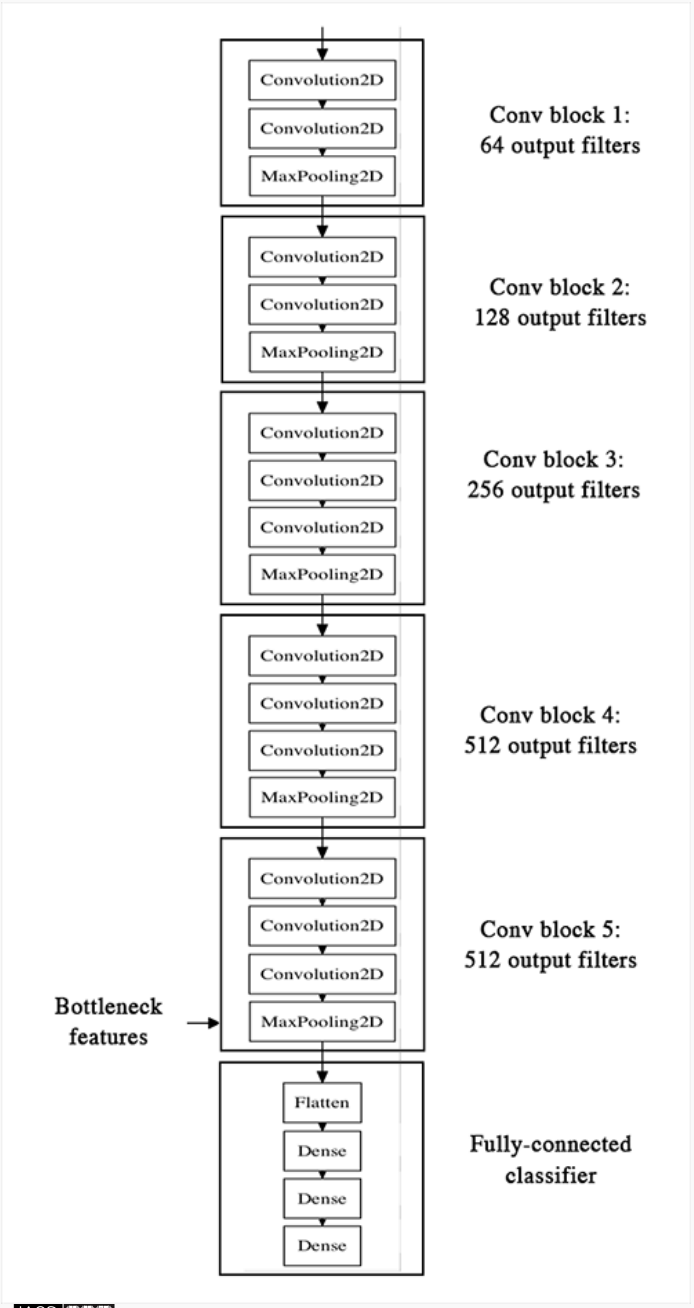


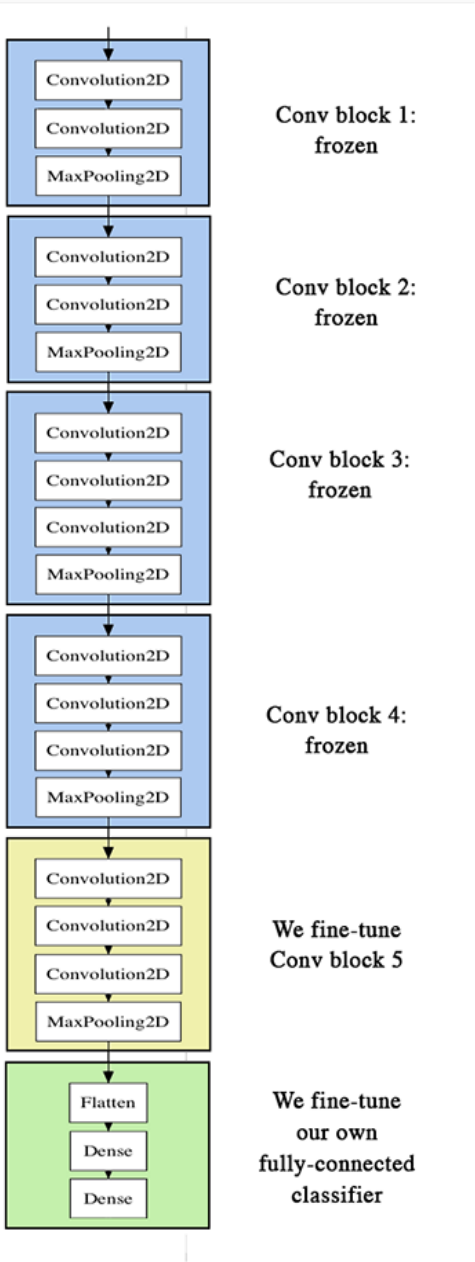


Transfer Learning (cont)

1. Get existing network weights
2. Un-freeze the “head” fully connected layers and train on your new images
3. Un-freeze the latest convolutional layers and train at a very low learning rate starting with the weights from the previously trained weights. This will change the latest layer convolutional weights without triggering large gradient updates which would have occurred had we not done 2.

See https://medium.com/@timanglade/how-hbos-silicon-valley-built-not-hotdog-with-mobile-tensorflow-keras-react-native-ef03260747f3and https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html for some details


Convolution2D
Convolution2D
MaxPooling2D
Conv block 1:
64 output filters
Convolution2D
Convolution2D
MaxPooling2D
Conv block 2:
128 output filters
Convolution2D
Convolution2D
Convolution2D
MaxPooling2D
Conv block 3:
256 output filters
Convolution2D
Convolution2D
Convolution2D
MaxPooling2D
Conv block 4:
512 output filters
Convolution2D
Convolution2D
Convolution2D
MaxPooling2D
Conv block 5:
512 output filters
Bottleneck features
Flatten
Dense
Dense
Dense
Fully-connected classifier


Convolution2D
Convolution2D
MaxPooling2D
Conv block 1:
frozen
Convolution2D
Convolution2D
MaxPooling2D
Conv block 2:
frozen
Convolution2D
Convolution2D
Convolution2D
MaxPooling2D
Conv block 3:
frozen
Convolution2D
Convolution2D
Convolution2D
MaxPooling2D
Conv block 4:
frozen
Convolution2D
Convolution2D
Convolution2D
MaxPooling2D
We fine-tune
Conv block 5
Flatten
Dense
Dense
We fine-tune
our own
fully-connected
classifier

Outline

1. Review from last lecture
2. BackProp of MaxPooling layer
3. A bit of history
4. Layers Receptive Field
5. Saliency maps
6. Transfer Learning
7. **CNN for text analysis (example)**

Convolutional Neural Networks for Text Classification

When applied to text instead of images, we have an 1 dimensional array representing the text.

Here the architecture of the ConvNets is changed to 1D convolutional-and-pooling operations.

One of the most typically tasks in NLP where ConvNet are used is sentence classification, that is, classifying a sentence into a set of pre-determined categories by considering ***n***-grams, i.e. it's words or sequence of words, or also characters or sequence of characters.

LETS SEE THIS THROUGH AN EXAMPLE

Beyond

- MobileNetV2 (https://arxiv.org/abs/1801.04381)
- Inception-Resnet, v1 and v2 (https://arxiv.org/abs/1602.07261)
- Wide-Resnet (https://arxiv.org/abs/1605.07146)
- Xception (https://arxiv.org/abs/1610.02357)
- ResNeXt (https://arxiv.org/pdf/1611.05431)
- ShuffleNet, v1 and v2 (https://arxiv.org/abs/1707.01083)
- Squeeze and Excitation Nets (https://arxiv.org/abs/1709.01507)