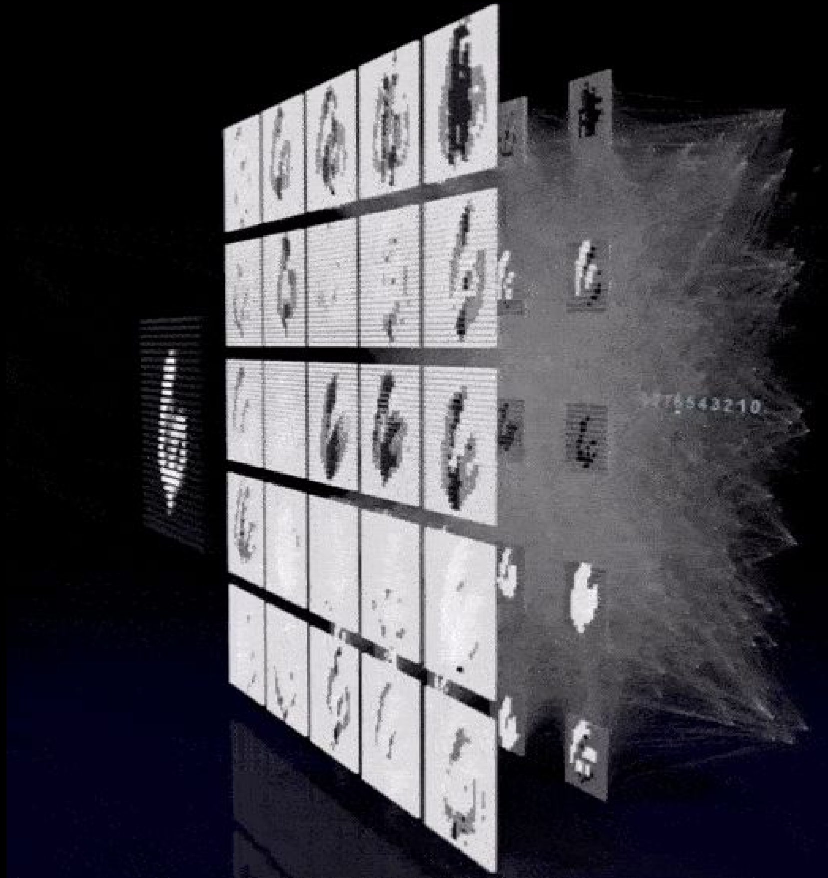
Convolutional Neural Network from Zero to Hero

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN



Speakers:

- Hermann Redich
- Patrick Börzel
- Isabella Galter
- Collin Leiber
- Ou Changkun

Seminar Deep Learning WiSe 2017/18
Institut für Informatik
Universtät München

Agenda

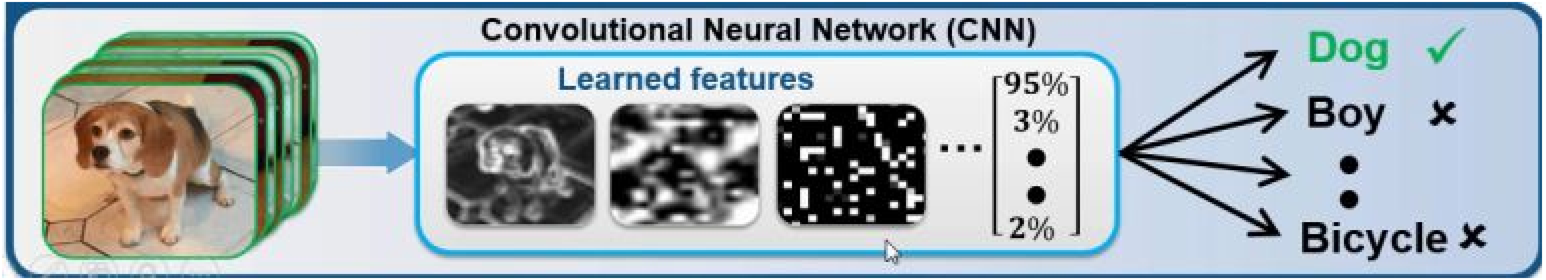
1. **Introduction and Convolutional Layer (HR)**
   - Whole picture of CNN
   - convolutional layer
2. **Pooling + Fully Connected Layers (PB)**
   - pooling layer
   - fully connected layer
   - Properties: Invariances
3. **ImageNet Case Study (CL + OC)**
   - AlexNet / VGG / GoogLeNet
   - ResNet / DenseNet
4. **Practical Tricks (IG)**
   - Transfer learning
   - Visualization techniques
5. **Outlook (OC)**
   - Limitations with CNNs
   - Capsule network / Dynamic routing

#1 Introduction & Convolutional Layers

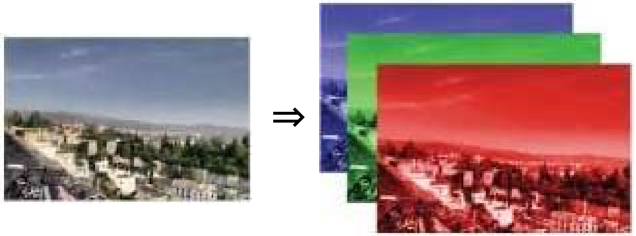
- CNN Overview
- CNN Structure
- Example
- Hyperparameters

CNN Overview

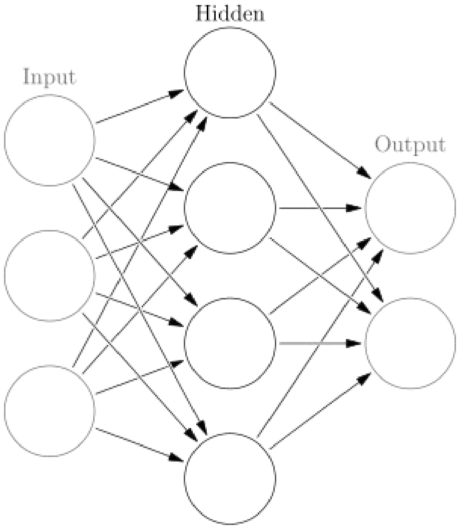
- Inspired by the visual cortex
- Is often used for image and video recognition
- Detects features in a image
- Classification



Images and Neural Networks



An image is represented by pixels in X and Y dimension and consists of several color channel (ex. RGB)



Recap: A neural network trains the weights to learn a good classification of labeld data

For an (RGB) image of 10x10 pixels we need 300 inputs. Each representing a pixel in the image

Disadvantages:

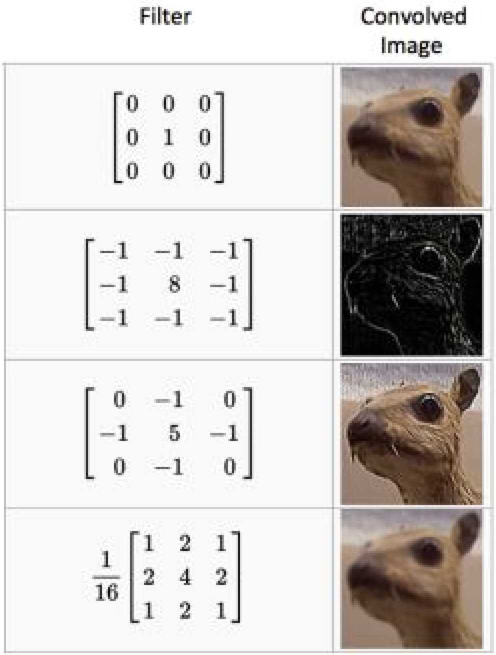
- super linear growth of weights
- locality is not mentioned

Feature Detection in Images

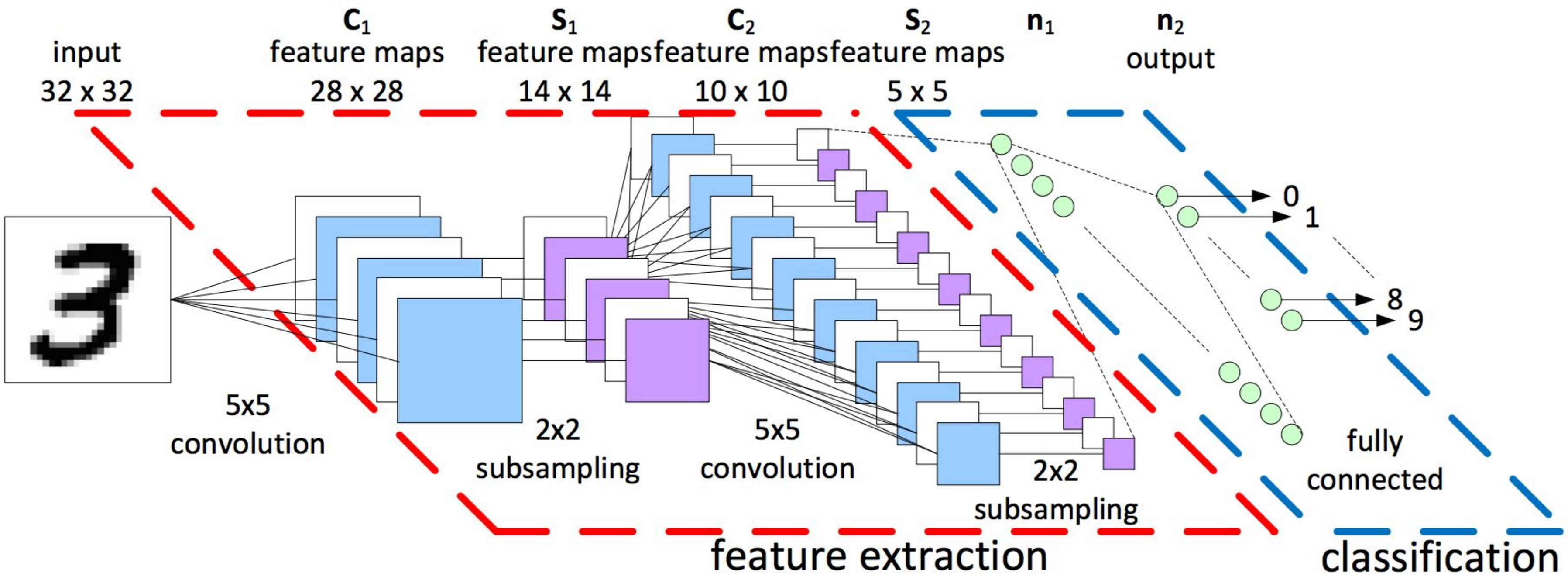
A filters is a square matrix. It is used to detect features in the original image. Therefore the filter slides over the image and outputs a value that says whether a feature was detected or not.

For every feature an own filter is applied.

We want this filters to be learned from a model.



CNN Structure



Convolutional Layer - Example

Input

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

1	0	-2
1	1	0
0	-1	1

Filter

3 * 1 = 3
2 * 0 = 0
2 * (-2) = -4
3 * 1 = 3
1 * 1 = 1
1 * 0 = 0
1 * 0 = 0
4 * (-1) = -4
3 * 1 = 3

Sum is ‘2’

Output

2			

Convolutional Layer - Example

Input

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

1	0	-2
1	1	0
0	-1	1

Filter

2 * 1 = 2
2 * 0 = 0
5 * (-2) = -10
1 * 1 = 1
1 * 1 = 1
2 * 0 = 0
4 * 0 = 0
3 * (-1) = -3
3 * 1 = 3

Sum is ‘-6’

Output

2	-6		

Convolutional Layer - Example

Input

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

1	0	-2
1	1	0
0	-1	1

Filter

2 * 1 = 2
5 * 0 = 0
5 * (-2) = -10
1 * 1 = 1
2 * 1 = 2
6 * 0 = 0
3 * 0 = 0
3 * (-1) = -3
1 * 1 = 1

Sum is ‘-7’

Output

2	-6	-7	

Convolutional Layer - Example

Input

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

1	0	-2
1	1	0
0	-1	1

Filter

5 * 1 = 5
5 * 0 = 0
3 * (-2) = -6
2 * 1 = 2
6 * 1 = 6
2 * 0 = 0
3 * 0 = 0
1 * (-1) = -1
4 * 1 = 4

Sum is '10'

Output

2	-6	-7	10

Convolutional Layer - Example

Input

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

1	0	-2
1	1	0
0	-1	1

Filter

3 * 1 = 3
1 * 0 = 0
1 * (-2) = -2
1 * 1 = 1
4 * 1 = 4
3 * 0 = 0
2 * 0 = 0
2 * (-1) = -2
3 * 1 = 3

Sum is '7'

Output

2	-6	-7	10
7			

Convolutional Layer - Example

Input

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

1	0	-2
1	1	0
0	-1	1

Filter

7 * 1 = 7
0 * 0 = 0
1 * (-2) = -2
0 * 1 = 0
3 * 1 = 3
4 * 0 = 0
4 * 0 = 0
4 * (-1) = -4
0 * 1 = 0

Sum is '4'

Output

2	-6	-7	10
7	8	-12	3
3	-1	14	3
4	-7	7	4

Convolutional Layer - Hyperparameters

filter:

- the dimensionality of the output space

kernel size:

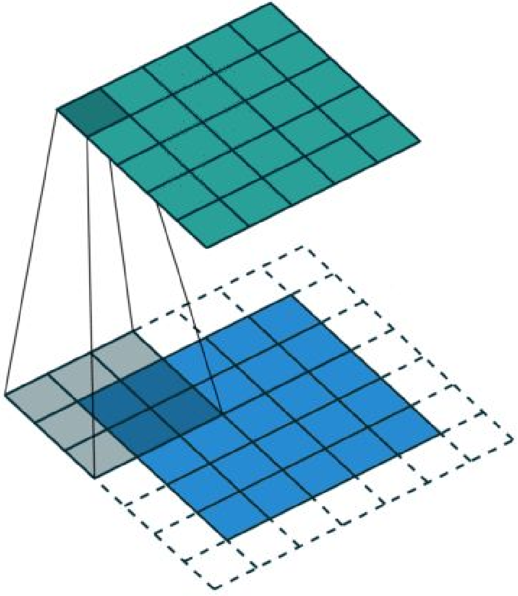
- describes the size of the filters

stride:

- amount of steps (pixels) the filter moves

padding:

- adds values on the border of the input (image)



#2 Pooling layers & FC layers

- Pooling Layers
    - How do they work?
    - Why are they used?
    - Disadvantages
- Fully Connected Layer
    - FC Layer in CNN
    - Purpose of FC - Layers in CNN
- Invariance of a CNN
    - Shift invariance
    - Distortion invariance

Pooling Layers - How do they work?

Specified by following properties:

- Filter dimension. Not necessary to be a square.
- Stride, which defines the movement of the filter.
- Pooling algorithm. Most common are the max pooling filter.

Pooling Layers - How do they work?

Max Pooling Filter with Size 2 x 2 and Stride 2.

Input:

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

Output:

Pooling Layers - How do they work?

Max Pooling Filter with Size 2 x 2 and Stride 2.

Input:

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

Output:

5	

Pooling Layers - How do they work?

Max Pooling Filter with Size 2 x 2 and Stride 2.

Input:

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

$\longrightarrow$

Output:

5	3

Pooling Layers - How do they work?

Max Pooling Filter with Size 2 x 2 and Stride 2.

Input:

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

Output:

5	3
3	

Pooling Layers - How do they work?

Max Pooling Filter with Size 2 x 2 and Stride 2.

Input:

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

Output:

5	3
3	8

Pooling Layers - Why are they used?

- Reduce the size of the image and therefore the number of parameters and computational requirements.
- As countermeasure against overfitting.

Pooling Layers - Disadvantage and alternative

Limiting factor for the depth of the network.

But: Researches try to replace Pooling Layers by some Convolutional Layers with bigger Stride.

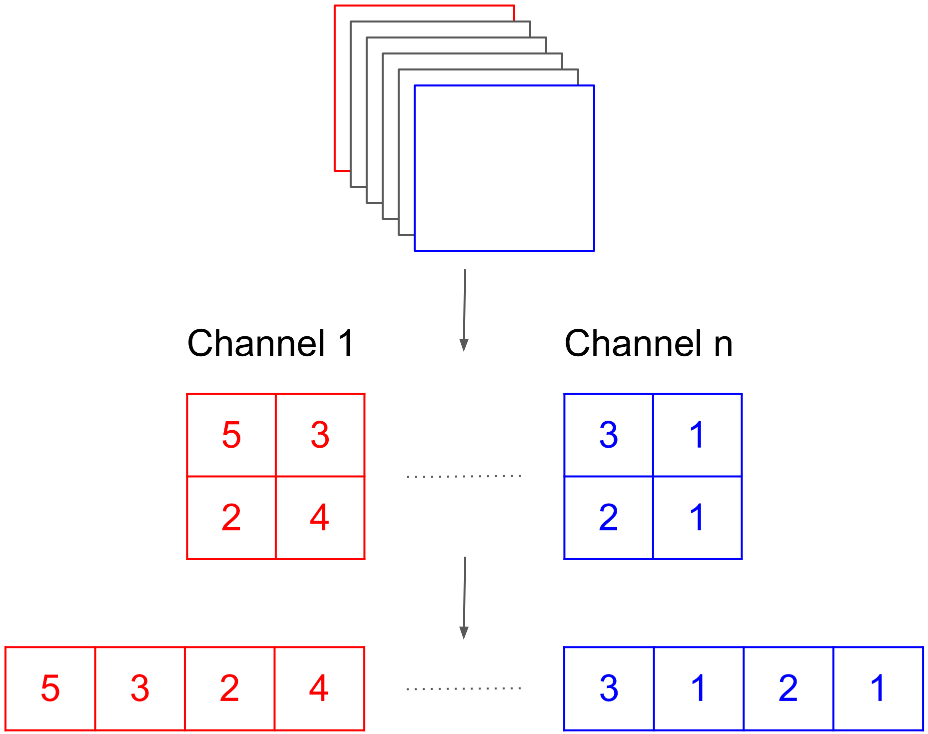
Fully Connected Layers - FC Layers in CNN

Convolutional and pooling layers normally use multiple filters on the same input. Therefore as output there are many channels, which are processed again by some layers.

On the other side: FC - Layer works on single vectors.

Fully Connected Layers - FC Layers in CNN

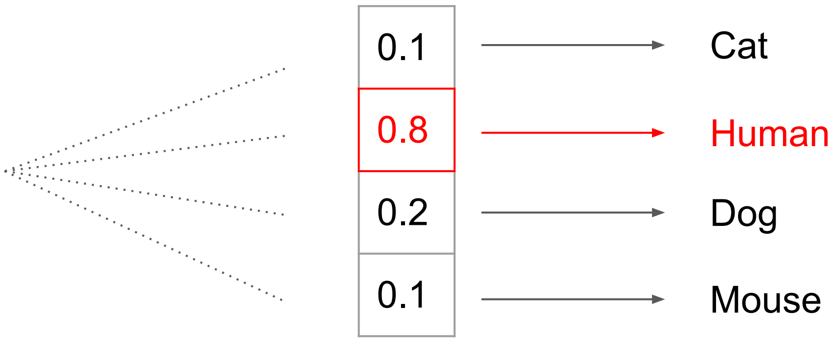
How to connect the FC Layer to the previous Layers?



Fully Connected Layers - Purpose of FC Layers in CNN

Analyzes the extracted features and performs a classification of the input based on those.

Outputs a vector with probabilities for each possible class and how likely the input belongs to one of those.



Invariance in CNN - Shift invariance

Each filter is learned to detect a single feature.

Convolutional Layers moves each filter over the input.

Therefore the filter detects its learned feature independent of its position.

Invariance in CNN - Distortion invariance

By using a pooling layer the image stays almost the same.

But it removes the importance of exact positions/values for a NN.

0	0	0	0	0.75	1
0	0	0	0.75	1	0.75
0	0	0.75	1	0.75	0
0	0.75	1	0.75	0	0
0.75	1	0.75	0	0	0
1	0.75	0	0	0	0

0	0.75	1
0.75	1	0.75
1	0.75	0

References of this Section

CS231n Convolutional Neural Networks for Visual Recognition. Retrieved November 27, 2017, from http://cs231n.github.io/convolutional-networks/#pool

LeCun, Y., & Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, *3361*(10), 1995.

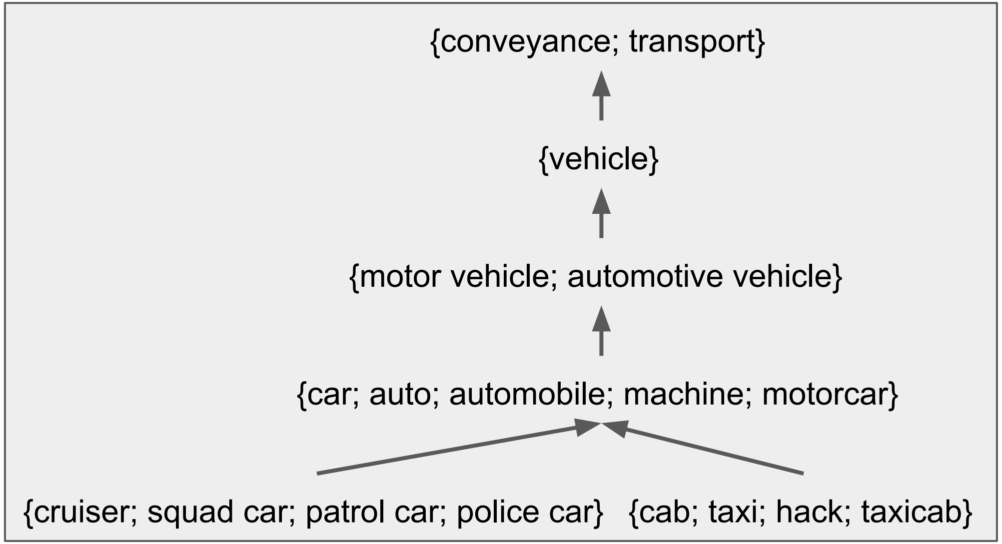
Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*.

#3 The ImageNet Competetion & Case Study

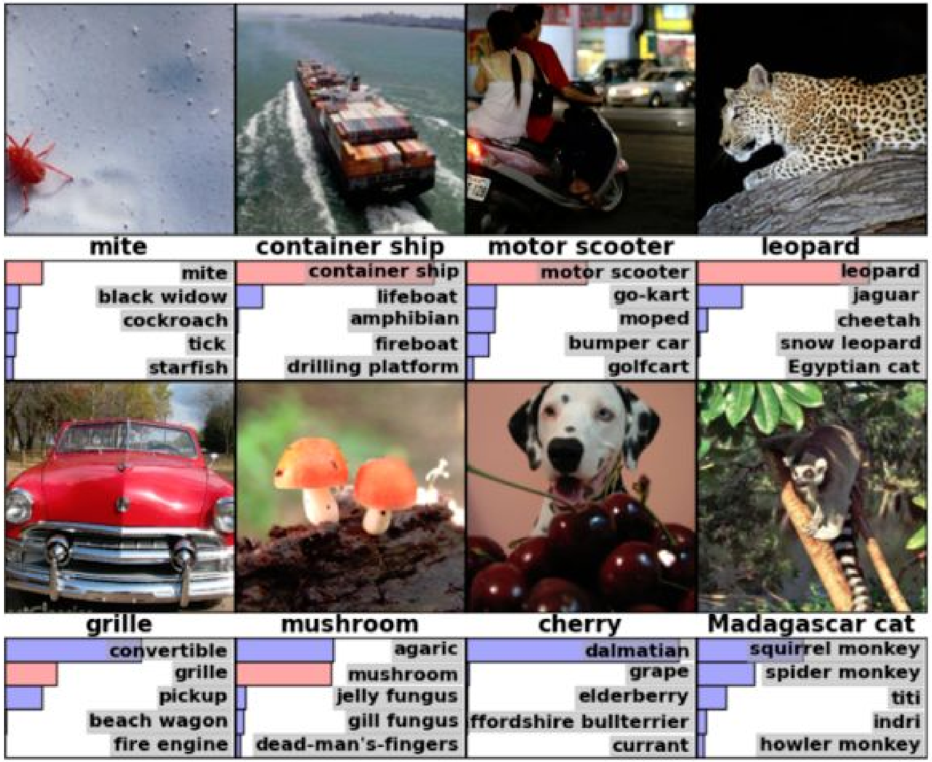
- ImageNet
- Classification Errors
- ILSVRC - Evolution
- ILSVRC - Winners
  - AlexNet
  - VGG
  - GoogLeNet
  - ResNet

ImageNet

- Lage scale image database
- Organized according to WordNet hirarchy
- 21,841 synsets
- 14,197,122 images
- Aim: provide average of 1000 images for each synset
- Hosts ILSVRC (*Large Scale Visual Recognition Challenge)*

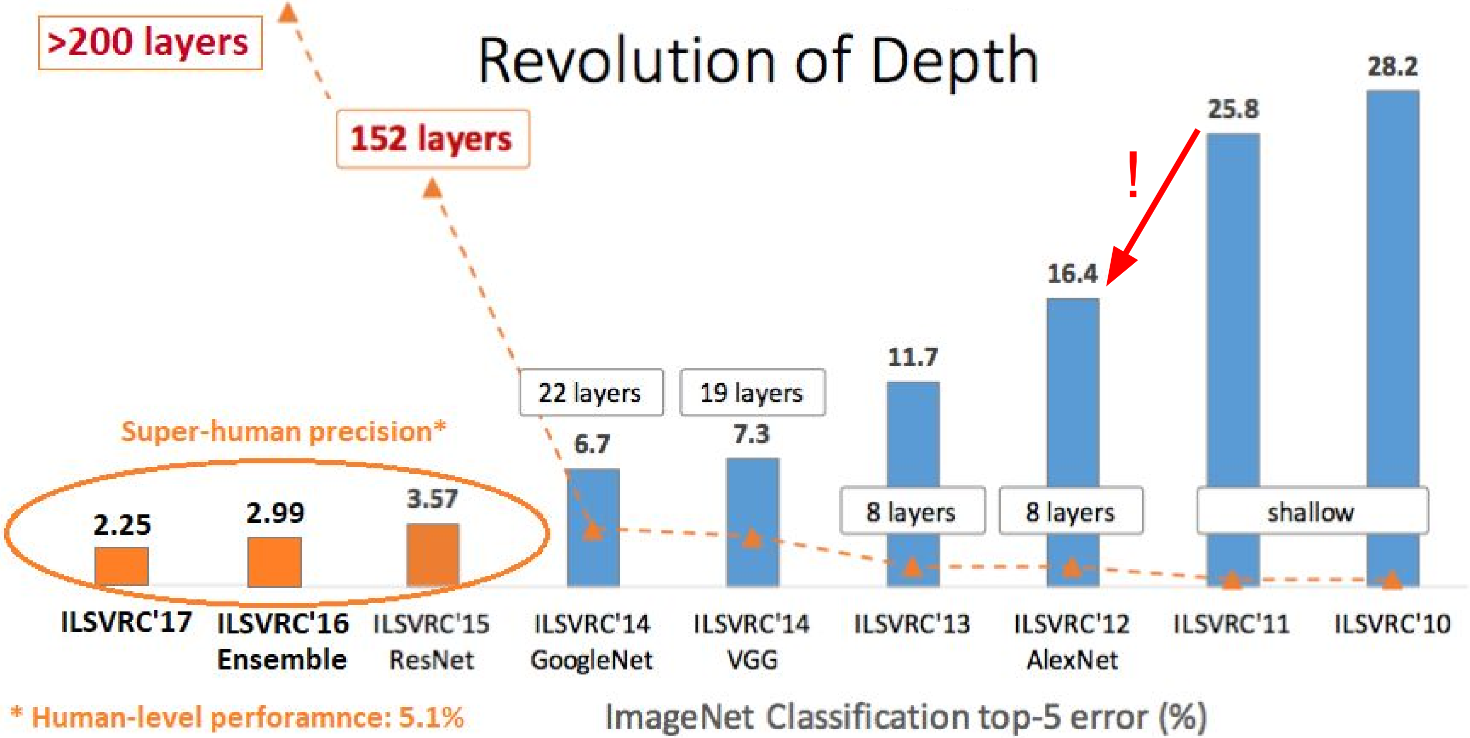


Classification Errors



[ImangeNet Classification with Deep Convolutional Neural Networks]

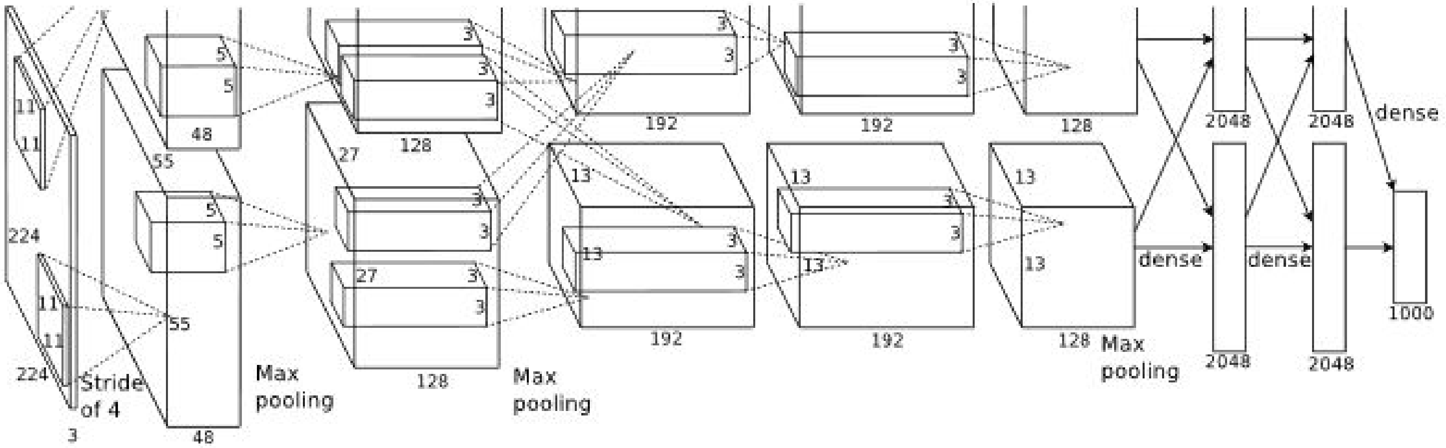
ILSVRC - Evolution



[Adaptation of: Semih Yagcioglu. *An Overview of Deep Residual Learning.* 2016]

ILSVRC - AlexNet

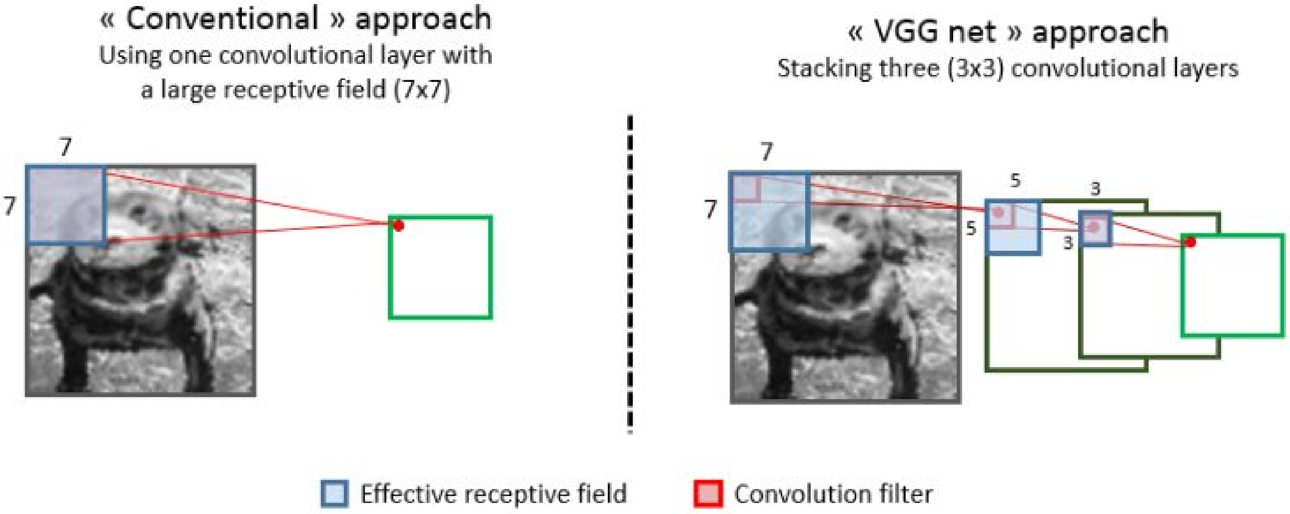
- Brought Deep Convolutional Neural Networks to the mainstream
- Significant improvement compared to 2nd place (26,2% error rate)
- Used ReLU instead of tanh function
- Implemented dropout layers to reduce overfitting



[ImangeNet Classification with Deep Convolutional Neural Networks]

ILSVRC - VGG

- “Keep it deep. Keep it simple”
- 19 layer CNN
- Only used 3x3 filters with stride and pad 1 and 2x2 maxpooling layers with stride 2



[https://guillaumebrg.wordpress.com/2016/02/13/adopting-the-vgg-net-approach-more-layers-smaller-filters/]

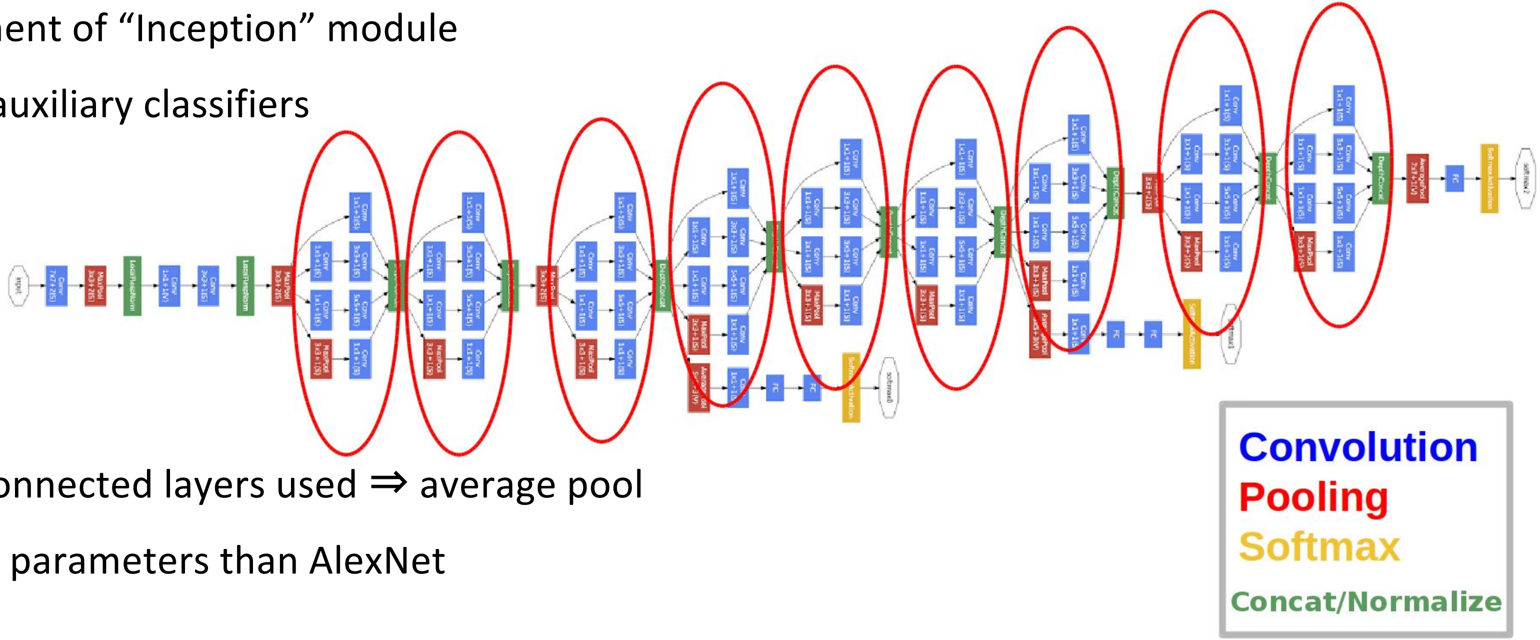
ILSVRC - VGG

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 imag)					
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					

[Very deep convolutional networks for large-scale image recognition]

ILSVRC - GoogLeNet

- Network does NOT follow a sequential structure of conv and pooling
- Development of “Inception” module
- Usage of auxiliary classifiers

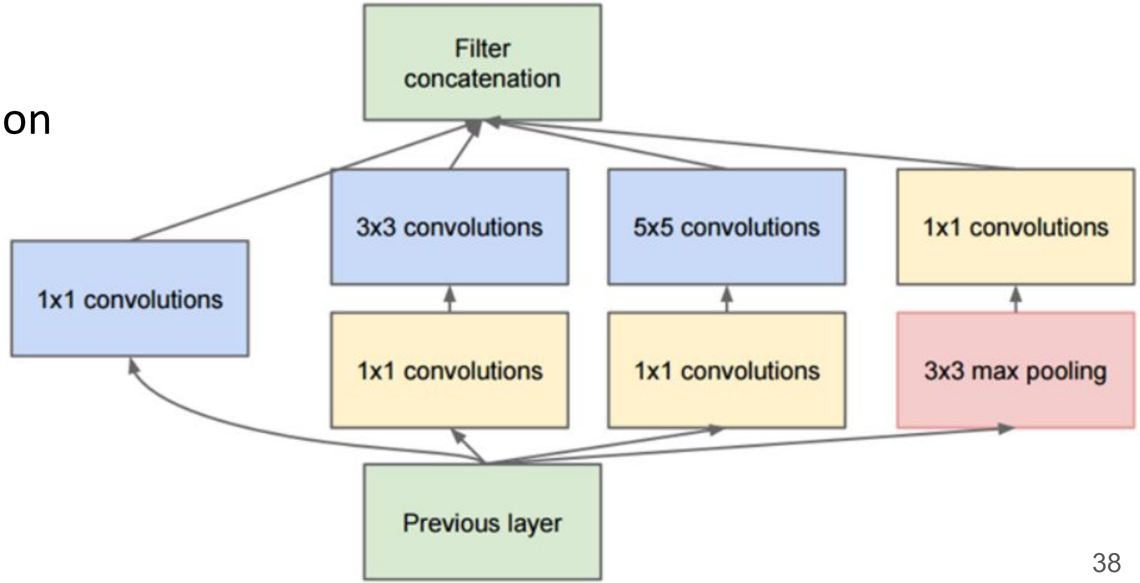


- No fully connected layers used ⇒ average pool
- 12x fewer parameters than AlexNet

[https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/googlenet.html]

ILSVRC - GoogLeNet - Inception module

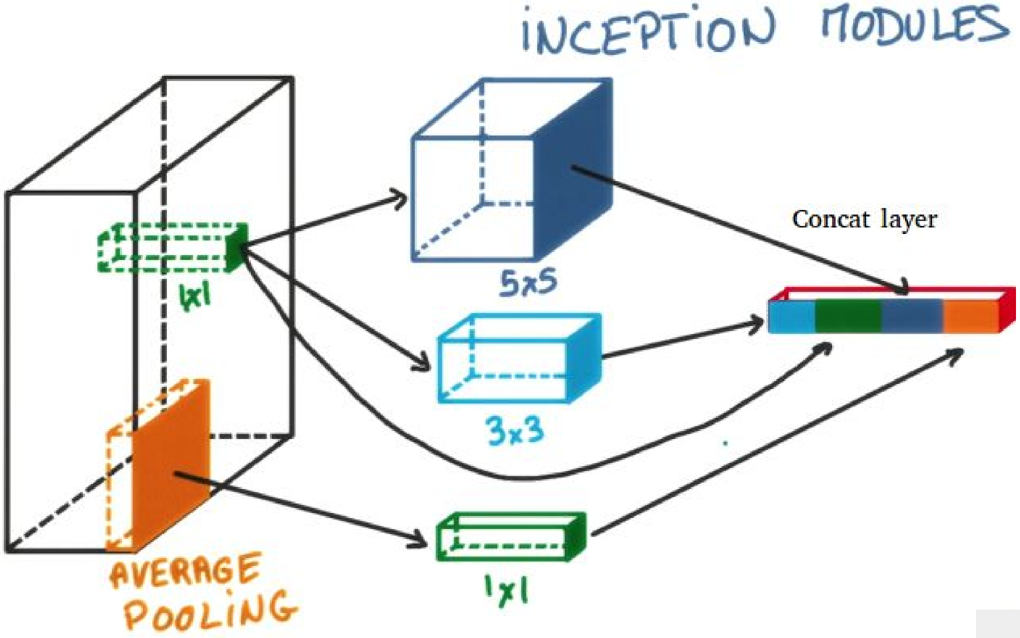
- No need to choose between pooling operation, conv operation and filter sizes
  ⇒ performe all in parallel
- Covers bigger area and keeps resolution for small information
- 1x1 convolutions to "pool the features"



[Going deeper with convolutions]

ILSVRC - GoogLeNet - Inception module

- No need to choose between pooling operation and conv operation / filter size ⇒ performe all in parallel
- Covers bigger area and keeps resolution for small information
- 1x1 convolutions to “pool the features”



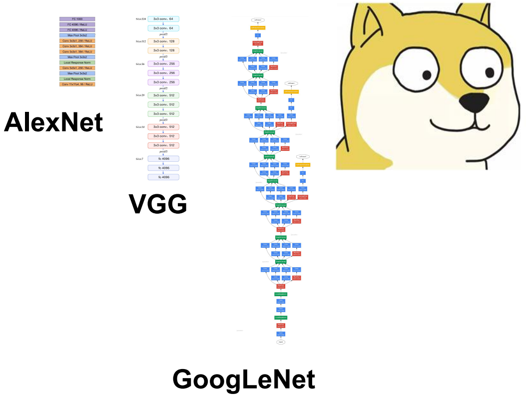
[https://www.youtube.com/watch?v=VxhSouuSZDY]

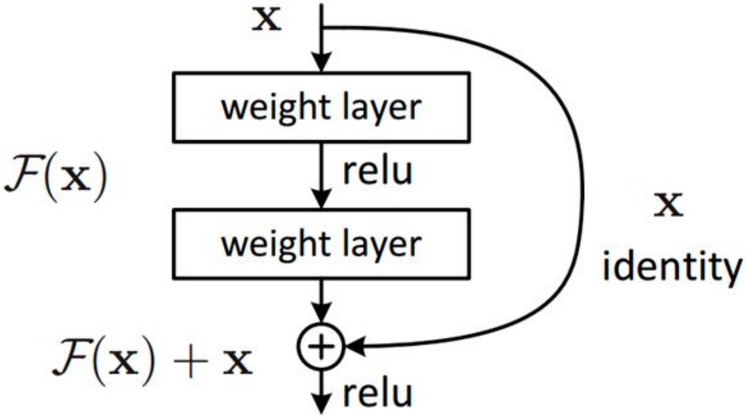
“We need to go deeper”

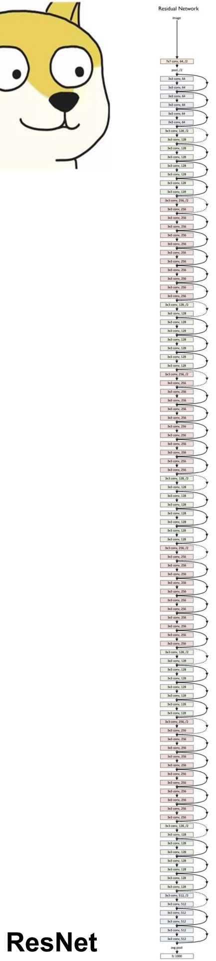
[Image source: http://knowyourmeme.com/memes/we-need-to-go-deeper]

Residual Network (ResNet)

- The problem is deeper models are harder to optimize
- Solution: Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping

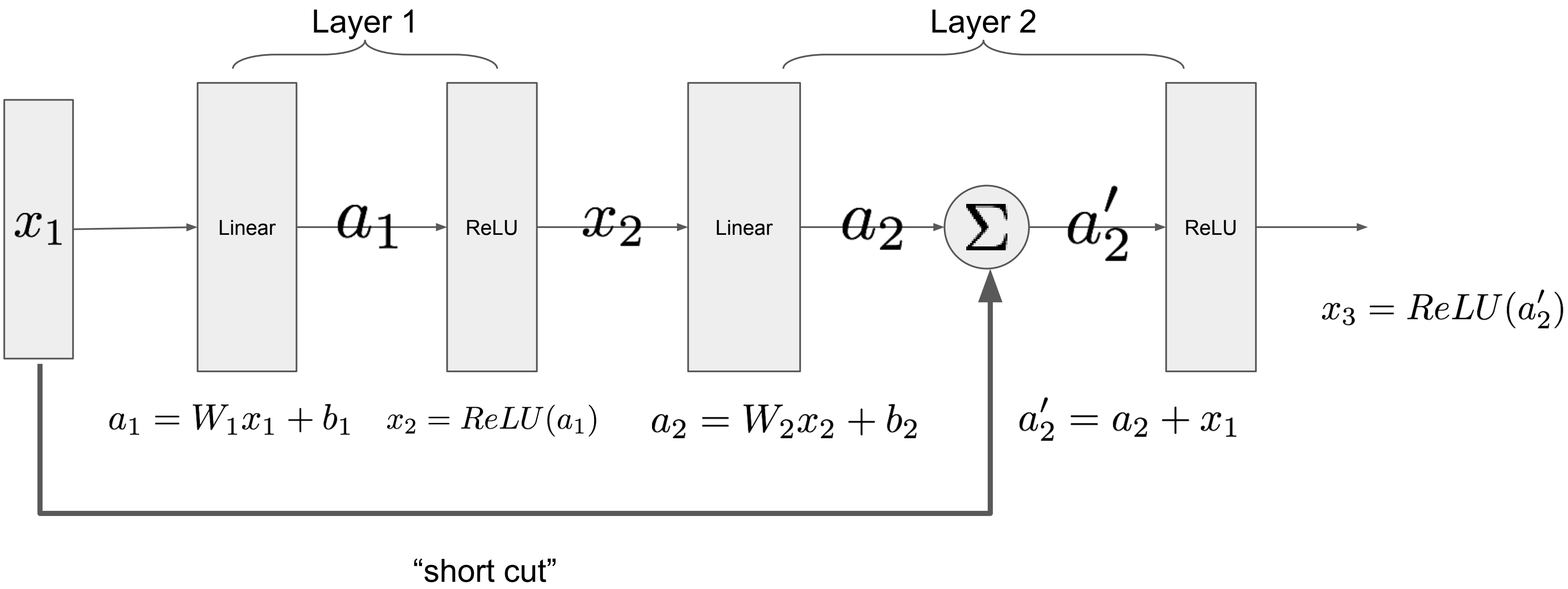






[He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition.]

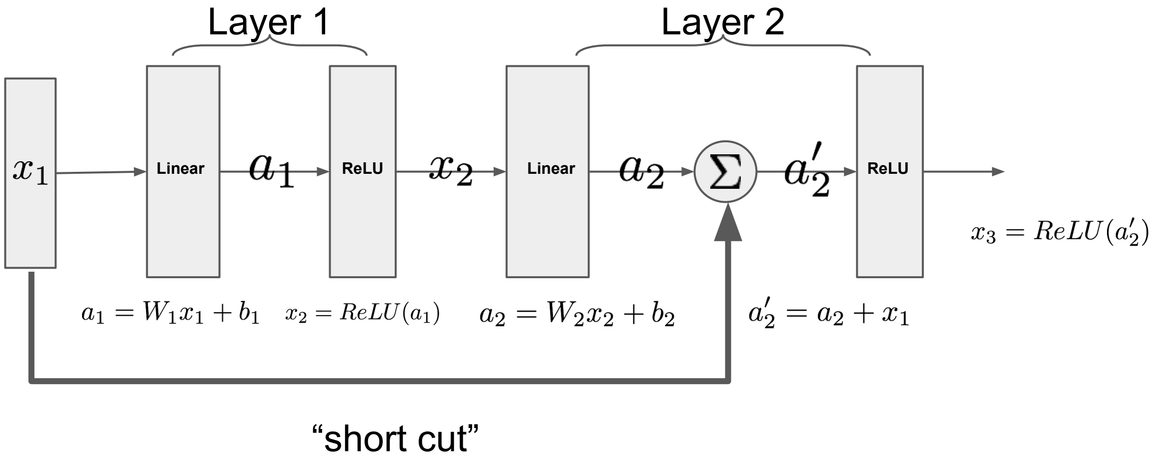
Residual Block



[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.]

Why Residual Block?

- Handling gradient vanishing (for sigmoid activation)
- Easy to learn identity function



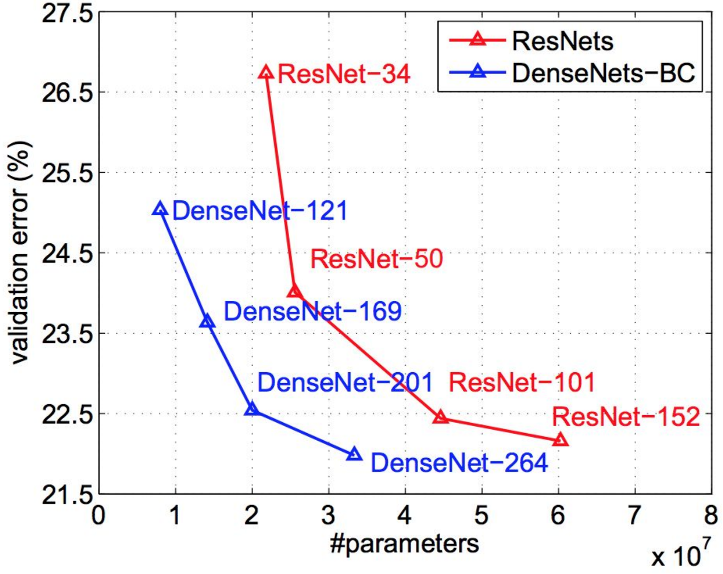
$$x_3 = ReLU(a_2 + x_1) = ReLU(W_2 x_2 + b_2 + x_1)$$

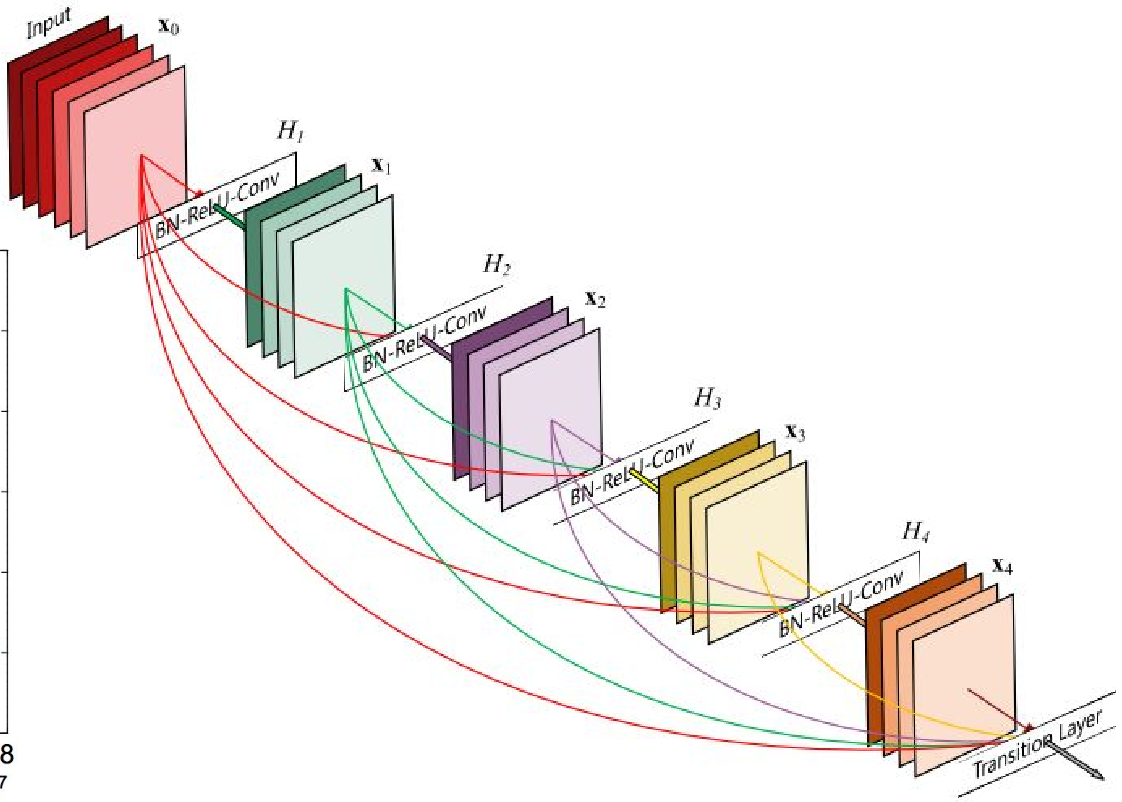
L2 regularization shrink weight

[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.]

A Glance of ResNet Application: DenseNet

- Densely connect every two layer





[Huang, G., Liu, Z., Weinberger, K. Q., & van der Maaten, L. (2016). Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*.]

Summary of ImageNet

- Architecture Evolution
  - **AlexNet**: Bring CNN to CV on board
  - **VGG**: “Keep deep, Keep simple”
  - **GoogLeNet**: Inception Module
  - **ResNet**: Residual Module

References of this Section

1. [http://www.image-net.org/about-overview, Accessed: 21.11.17]
2. [http://wordnet.princeton.edu/, Accessed: 21.11.17]
3. [http://image-net.org/about-stats, Accessed: 21.11.17]
4. [http://www.image-net.org/challenges/LSVRC/, Accessed: 21.11.17]
5. [https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/googlenet.html, Accessed: 22.11.17]
6. [https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/residual_net.html, Accessed: 22.11.17]
7. [https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html, Accessed 22.11.17]
8. [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).]
9. [Simonyan, K., & Zisserman, A. (2014). **Very deep convolutional networks for large-scale image recognition**. arXiv preprint arXiv:1409.1556.]
10. [Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). **Going deeper with convolutions (2014)**. arXiv preprint arXiv:1409.4842, 7.]
11. [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).]
12. [Huang, G., Liu, Z., Weinberger, K. Q., & van der Maaten, L. (2016). **Densely connected convolutional networks**. arXiv preprint arXiv:1608.06993.]

#4 Practical Tricks

- Transfer learning
  - What is it?
  - Why to use it?
  - Where to use it?
- Visualization techniques
  - Activations
  - Weights

Problems about CNNs

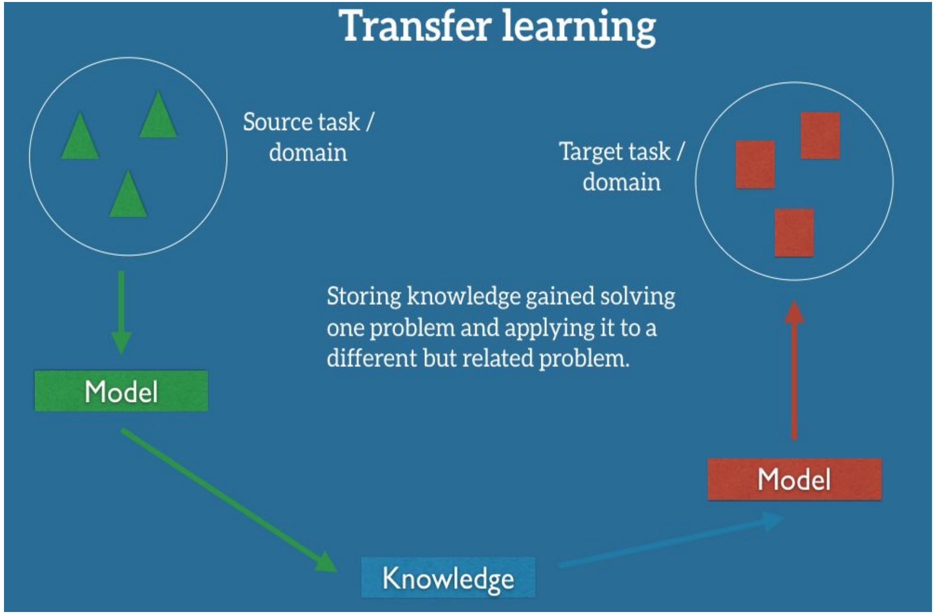
“We trained the network for roughly 90 cycles through the training set of 1.2 million images, which took **five to six days** on two NVIDIA GTX 580 3GB GPUs”, AlexNet

Reasons to do not reinvent the wheel

- training time is too long
- need of
    - powerful GPUs
    - need of datasets of sufficent size
- takes a lot of human resources

Transfer learning

- Idea:
  - Feature extraction
  - Use the architecture
  - Train some layers while freeze others
- How to:
  1. Decide what you want to learn
  2. Find an appropriate pre-trained model
  3. Fine-tuning

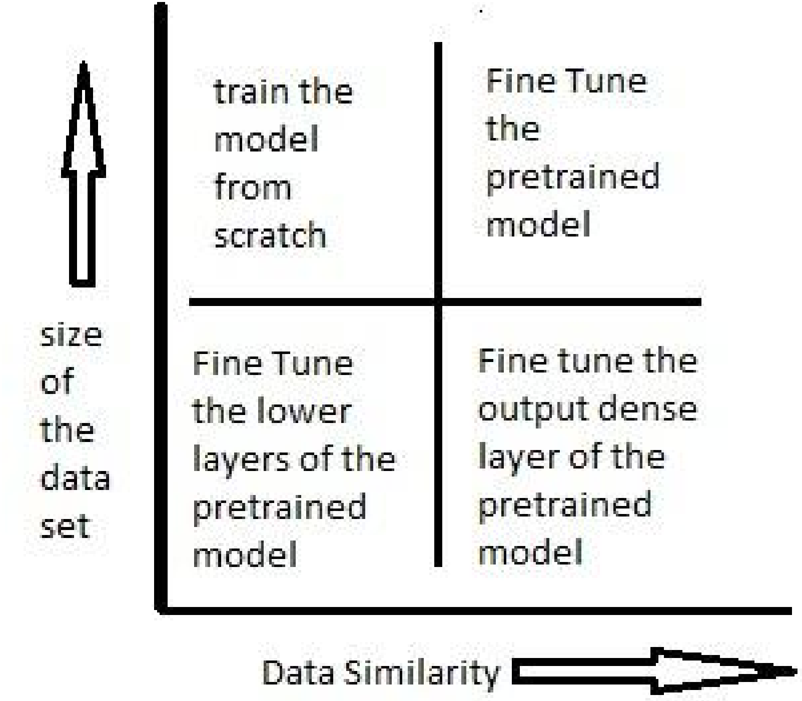


Finding the appropriate pre-trained model

- depends on
  - the task
  - input size
- similar task => more accuracy
- good news:

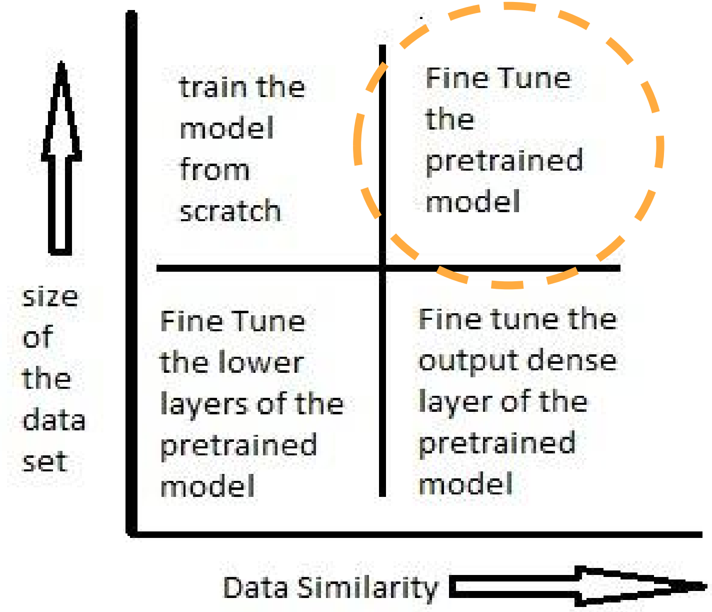
  => most of the pre-trained models are easy to access

Fine-tuning: Overview



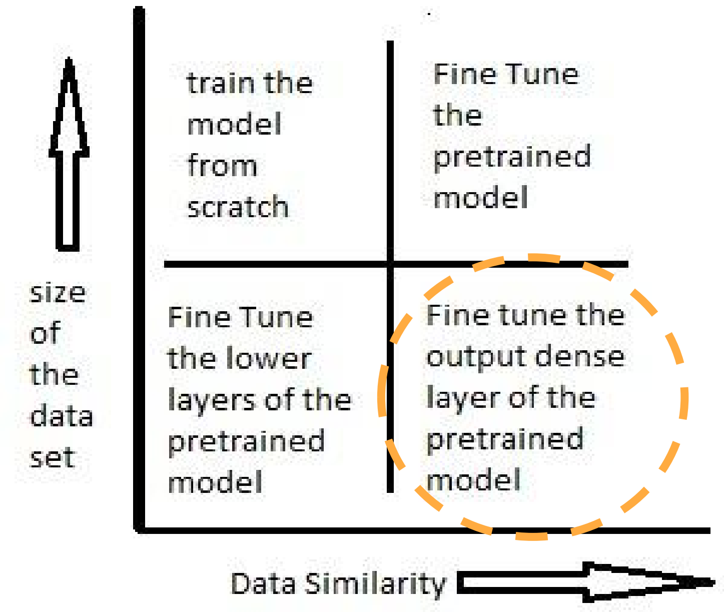
Transfer Learning - Case 1

- ideal situation
- use architecture + weights
- retrain with own data
- low learning rates to keep the knowledge



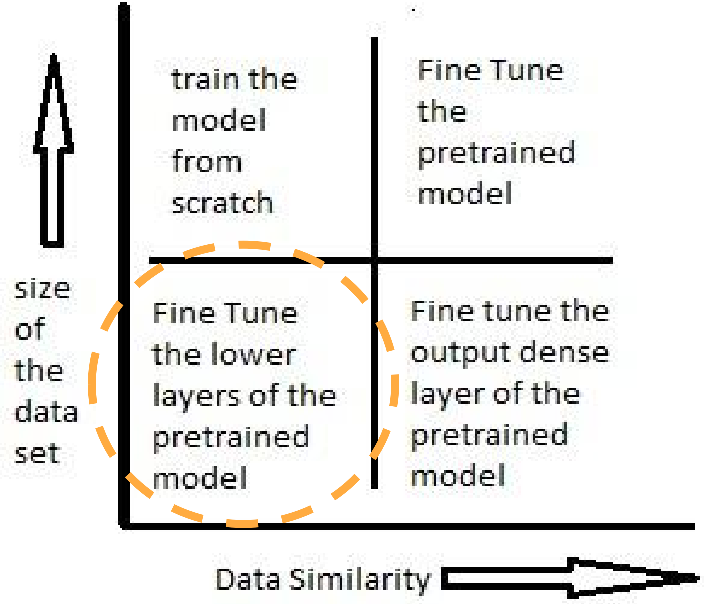
Transfer Learning - Case 2

- use it as a feature extractor
- modify the output layer
- do not retrain



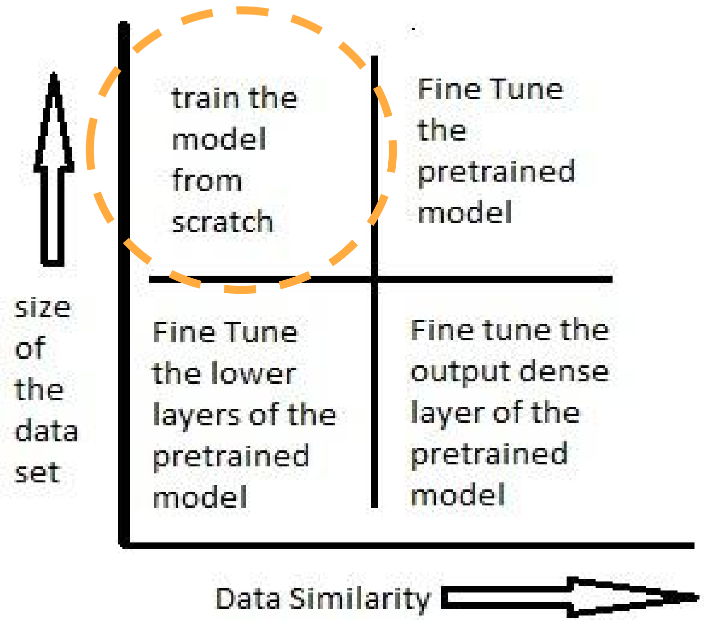
Transfer Learning - Case 3

- freeze the first k layers
  => only basic patterns
- retrain the others



Transfer Learning - Case 4

- “worst case”
- low similarity => usage of the model not effective
- build your own CNN



Common Criticism

The features learned by a

neural network are not interpretable!

Visualize the activations

- Why?

  Discover which features are learned by comparing of activation with the original image

- How to:
  1. Pass the image through the network
  2. Show the activations of a certain layer for all channels

Example: AlexNet - Recap

```
25x1 Layer array with layers:

   1   'data'     Image Input                   227x227x3 images with 'zerocenter' normalization
   2   'conv1'    Convolution                   96 11x11x3 convolutions with stride [4  4] and padding [0  0]
   3   'relu1'    ReLU                          ReLU
   4   'norm1'    Cross Channel Normalization   cross channel normalization with 5 channels per element
   5   'pool1'    Max Pooling                   3x3 max pooling with stride [2  2] and padding [0  0]
   6   'conv2'    Convolution                   256 5x5x48 convolutions with stride [1  1] and padding [2  2]
   7   'relu2'    ReLU                          ReLU
   8   'norm2'    Cross Channel Normalization   cross channel normalization with 5 channels per element
   9   'pool2'    Max Pooling                   3x3 max pooling with stride [2  2] and padding [0  0]
  10   'conv3'    Convolution                   384 3x3x256 convolutions with stride [1  1] and padding [1  1]
  11   'relu3'    ReLU                          ReLU
  12   'conv4'    Convolution                   384 3x3x192 convolutions with stride [1  1] and padding [1  1]
  13   'relu4'    ReLU                          ReLU
  14   'conv5'    Convolution                   256 3x3x192 convolutions with stride [1  1] and padding [1  1]
  15   'relu5'    ReLU                          ReLU
  16   'pool5'    Max Pooling                   3x3 max pooling with stride [2  2] and padding [0  0]
  17   'fc6'      Fully Connected               4096 fully connected layer
  18   'relu6'    ReLU                          ReLU
  19   'drop6'    Dropout                       50% dropout
  20   'fc7'      Fully Connected               4096 fully connected layer
  21   'relu7'    ReLU                          ReLU
  22   'drop7'    Dropout                       50% dropout
  23   'fc8'      Fully Connected               1000 fully connected layer
  24   'prob'     Softmax                       softmax
  25   'output'   Classification Output         crossentropyex with 'tench', 'goldfish', and 998 other classes
```

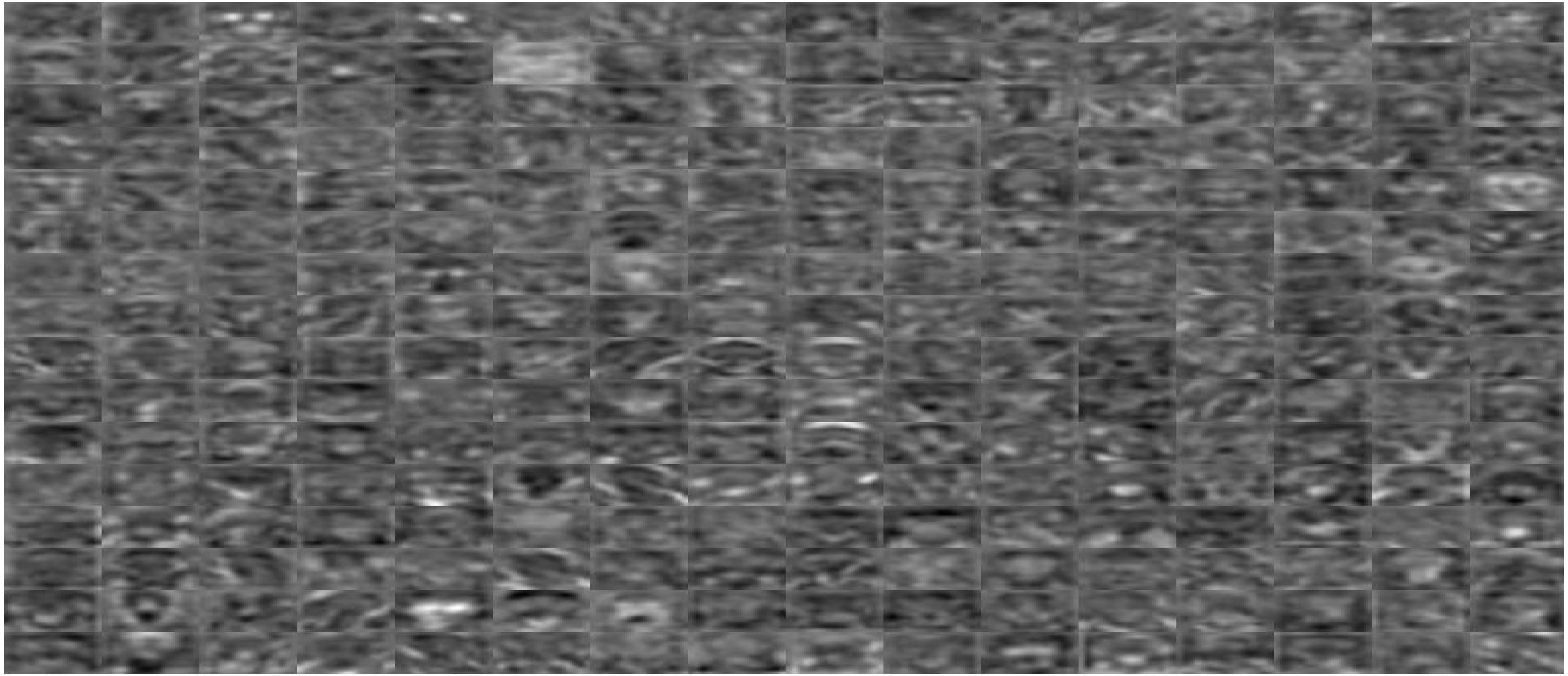
Example: AlexNet - Convolutional Layer 1

- colorscale: grayscale 0-1
- 96 boxes for 96 filters
- white pixel: strong positive activation
- black pixel: strong negative activation

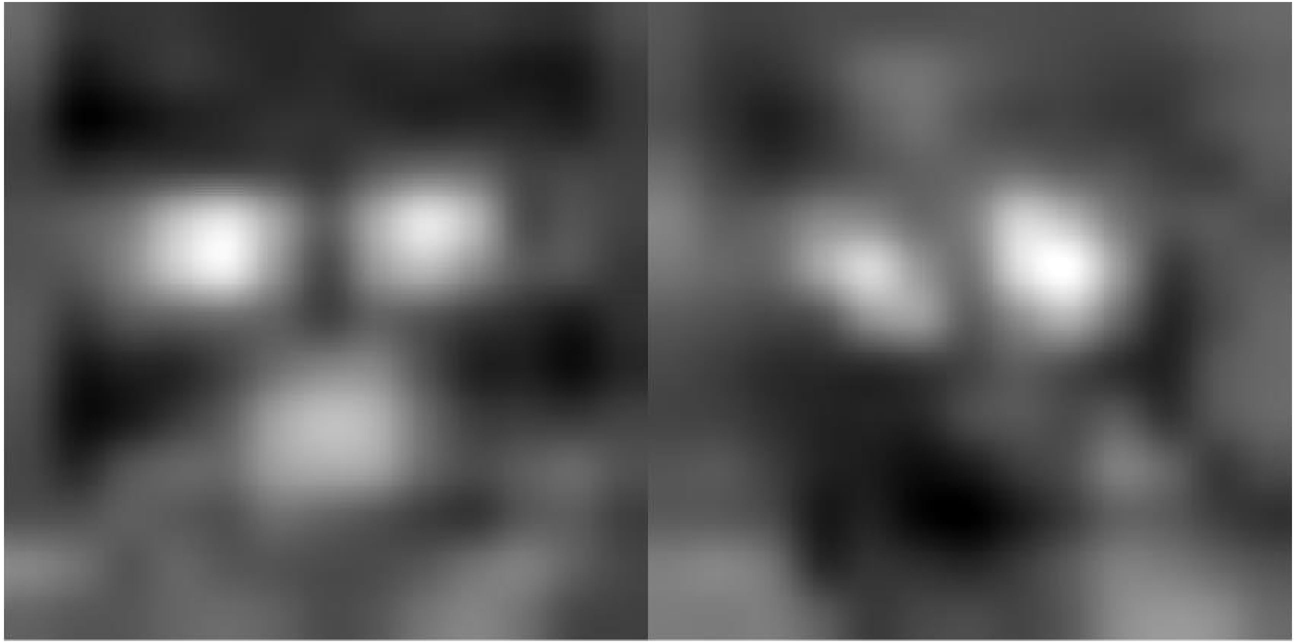
Filter 32

=> channel activates on red pixels

Example: AlexNet - Convolutional Layer 5

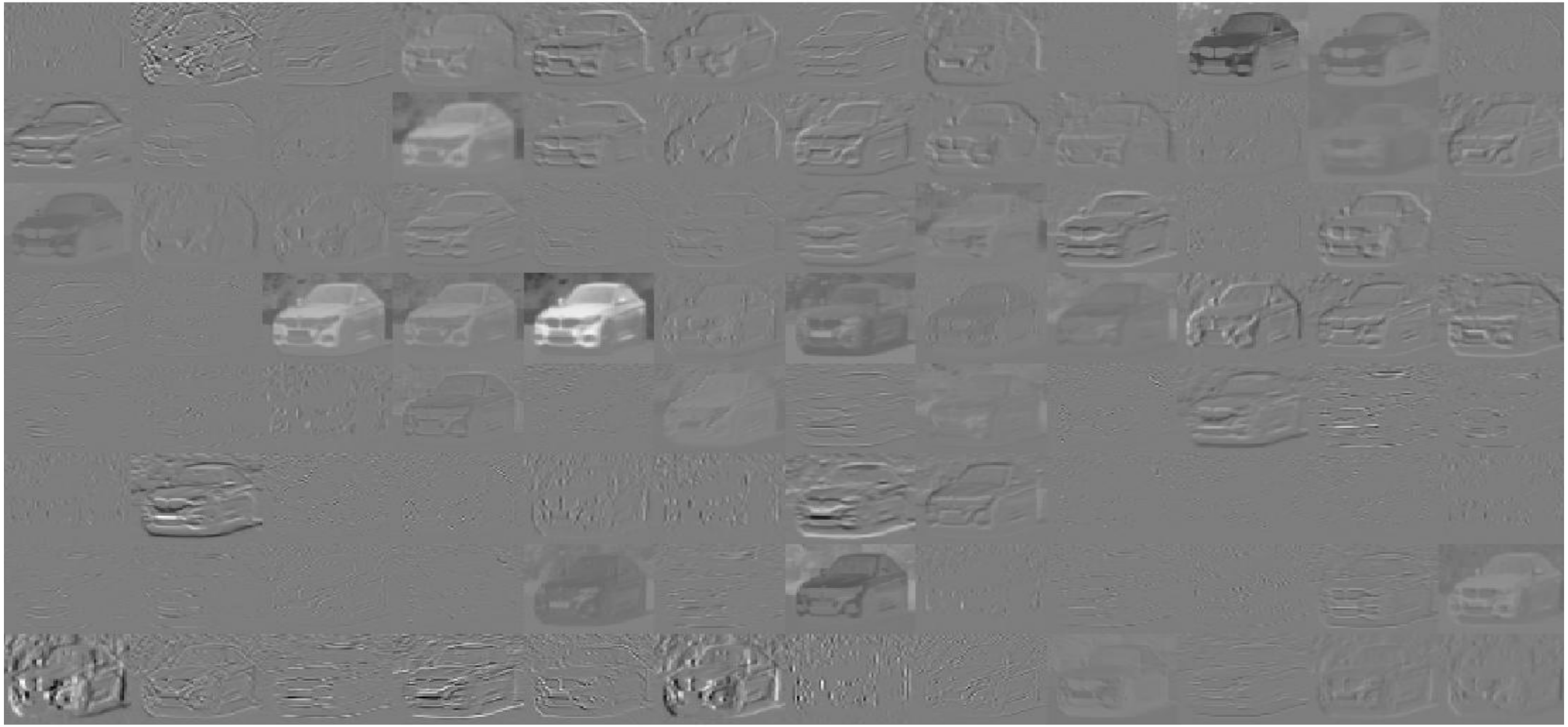


Convolutional Layer 5 Filter 3 and 5



=> activated on eyes

Another example: Conv Layer 1



Visualize the weights

- Why?

  Usually well-trained networks show nice and smooth filter patterns.

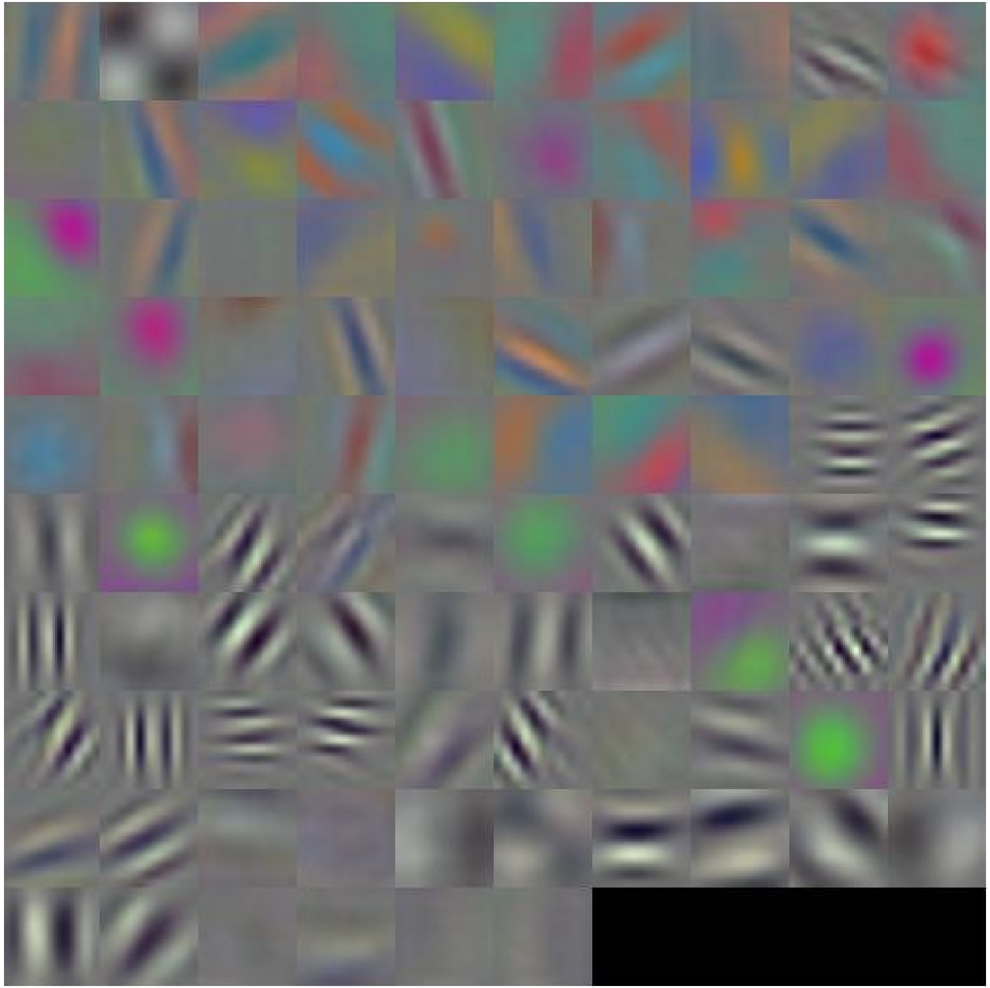
- How to?
  - Choose layer
  - Display the weights of all filter after the whole learning

Inspecting layer 1 of AlexNet

```
>> net.Layers(2)

ans =

  Convolution2DLayer with properties:

              Name: 'conv1'

   Hyperparameters
        FilterSize: [11 11]
       NumChannels: 3
        NumFilters: 96
            Stride: [4 4]
       PaddingMode: 'manual'
       PaddingSize: [0 0 0 0]

   Learnable Parameters
           Weights: [11×11×3×96 single]
              Bias: [1×1×96 single]
```

Conv Layer 1

- 96 filters
- size: 11x11
- 2 streams:
  - high-frequency grayscale features
  - low-frequency color features



Summary

Activation Visualization:

- channels in earlier layers:
  - edges
  - colors
- channels in later layers:
  - complex features
  - eyes, mouth

=> recognize dead channels

Weight Visualization:

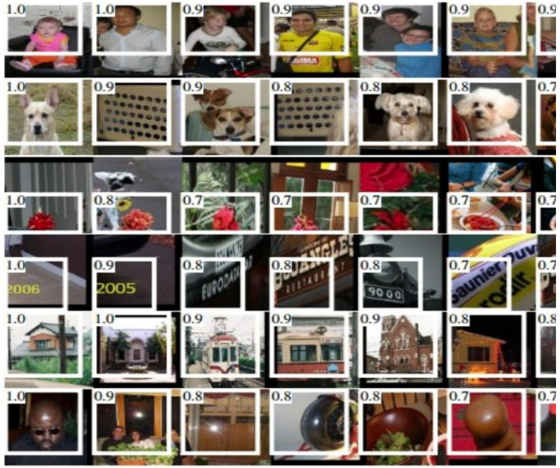
- patterns in early layers
  - smooth
  - well-formed
- patterns in later layers
  - less interpretable
  - too many

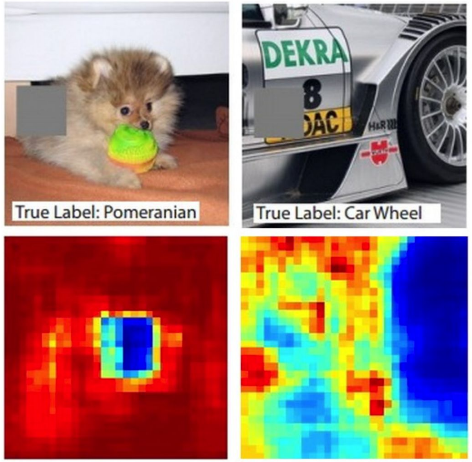
=> recognize “primitive” noisy patterns

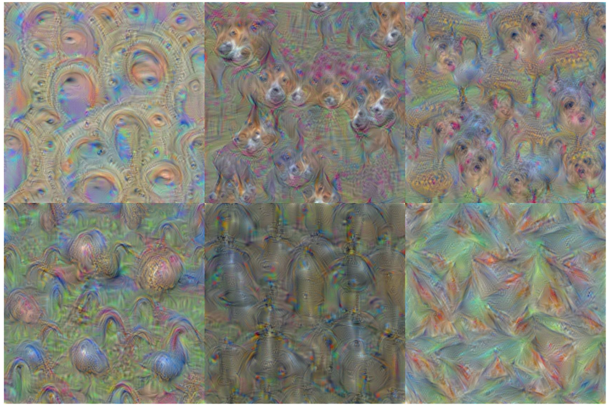
Advanced techniques

- Heatmaps
- Maximally activating images
- Reconstruct original images
- Deep Dream Images
- ….

=> most of the frameworks have build in visualizations techniques







References of this Section

1. [CS231n Convolutional Neural Networks for Visual Recognition. Retrieved November 28, 2017, from http://cs231n.github.io/transfer-learning/]
2. [Zeiler, M. D., & Fergus, R. (2014, September). Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818-833). Springer, Cham.]
3. [CS231n Convolutional Neural Networks for Visual Recognition. Retrieved November 28, 2017, from http://cs231n.github.io/understanding-cnn/]
4. [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).]
5. [Transfer learning & the art of using Pre-trained Models in Deep Learning from. Retrieved Dezember 6, 2017, from https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/]
6. [Sebastian Ruder, What is transfer learning. Retrieved Dezember 6, 2017, from http://ruder.io/transfer-learning/index.html#whatistransferlearning/]
7. Note: Most of the pictures in the visualization part are taken from a self made matlab implementation.

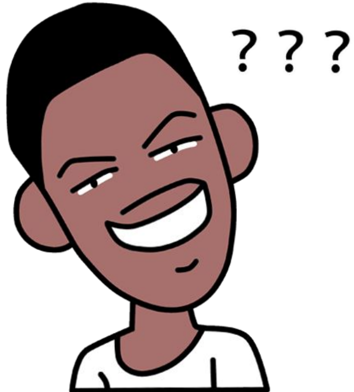
#5 Limitations & Outlooks

- Limitations of Convolutional Networks
- The Concept of the Capsule (Squashing Function)
- Dynamic *Routing* (*by agreement*)
- Case Study: Capsule Network

Limitations of CNN

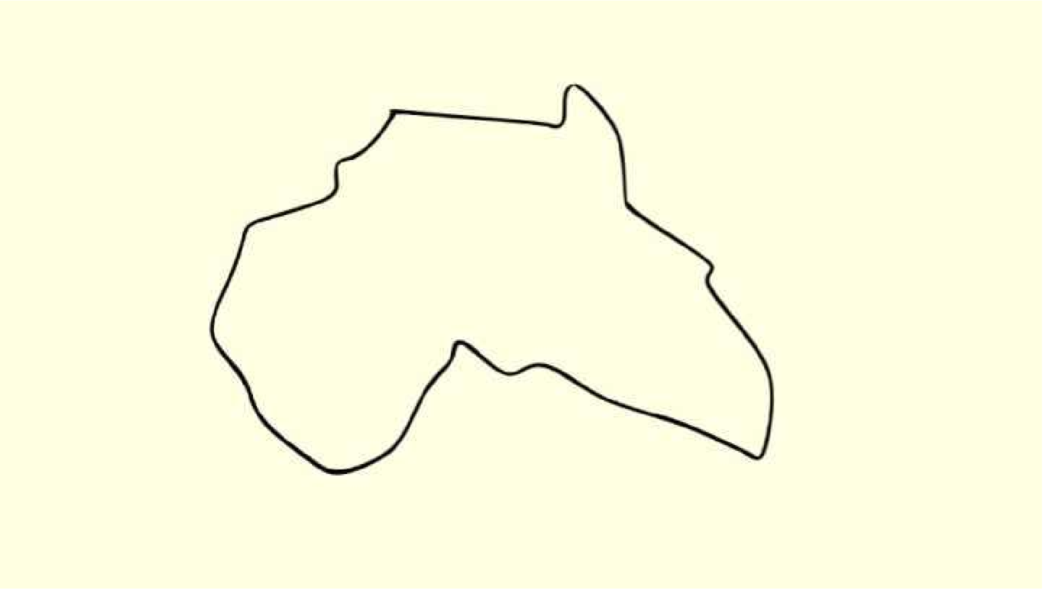
person	0.88
reddish orange color	0.78
light brown color	0.78
starlet	0.66
entertainer	0.66
female	0.60
woman	0.59
young lady (heroine)	0.59

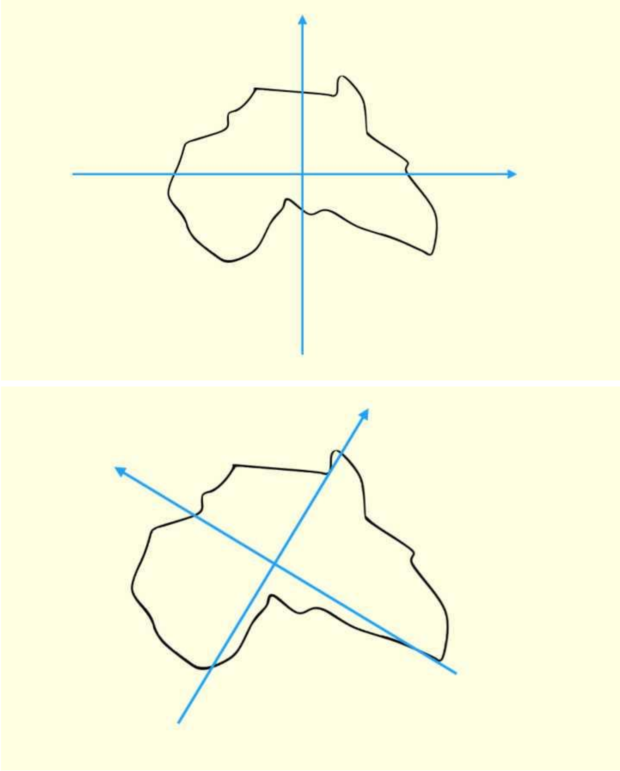
coal black color	0.79
hairpiece (hair)	0.71
dress	0.71
maroon color	0.71
person	0.58
toupee (hairpiece)	0.58
woman	0.56
Earrings	0.55
female	0.50



person	0.90
light brown color	0.84
starlet	0.77
entertainer	0.77
female	0.65
woman	0.64
young lady (heroine)	0.64
reddish orange color	0.64
newsreader	0.50

“Coordinate Frame” in Human Vision

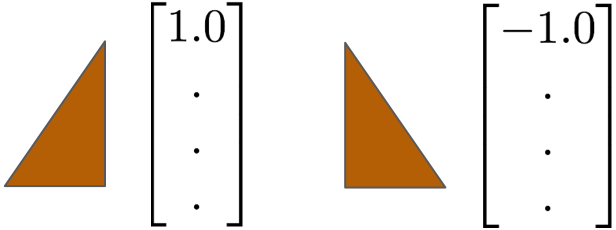


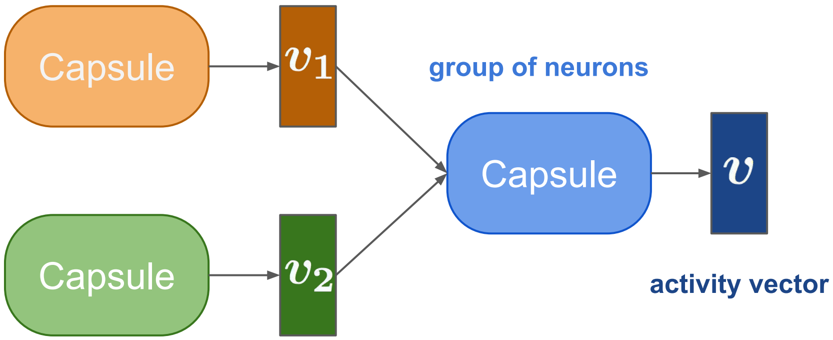


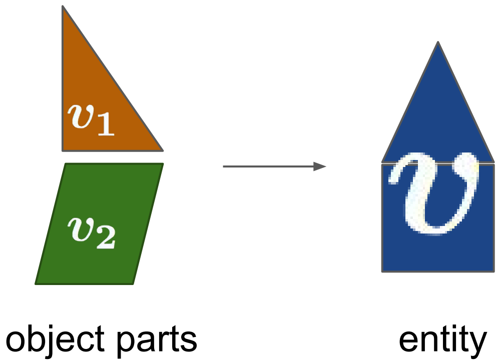
[Geoffery Hinton (2017). What’s wrong with convolutional neural nets. *Youtube:* https://www.youtube.com/watch?v=Mqt8fs6ZbHk&t=562s

What is a ***Capsule***?

*"A **capsule** is a **group of neurons** whose **activity vector represents** the instantiation parameters of a specific type of entity such as an object or **an object part**."*





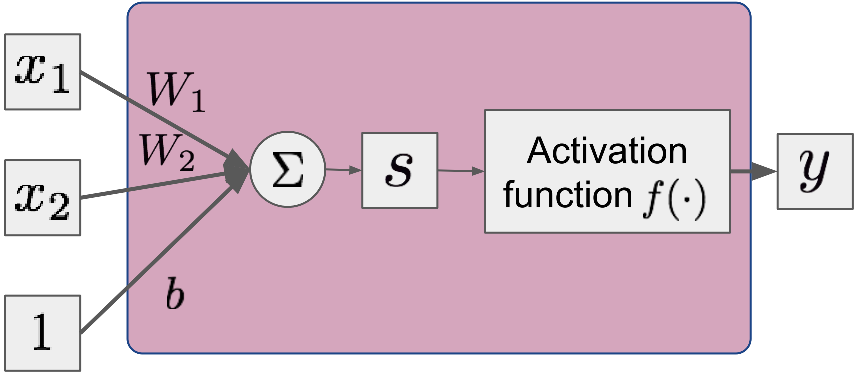


General ideas:

- ➢ Each dimension of v represents the characteristic of pattern;
- ➢ **The norm of v represents the exsistence (confidence). !!!**

[Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic Routing Between Capsules. *arXiv preprint arXiv:1710.09829*.]

Traditional v.s. Capsule Neuron

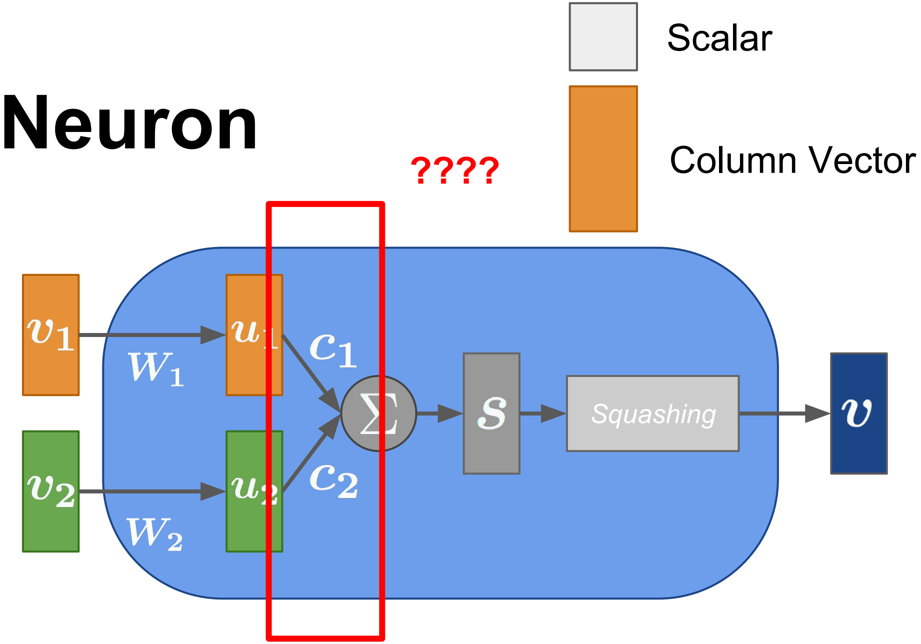


Traditional Neuron: Scalar → Scalar

$$s = b + \sum W_i x_i$$

$$y = f(s)$$

$$f(\cdot): \mathrm{Sigmoid}, \mathrm{ReLU}, \mathrm{Maxout}, etc.$$



Capsule Neuron: Vector → Vector

$$\boldsymbol{u_i} = \boldsymbol{W_i v_i} \quad \boldsymbol{s} = \sum c_i \boldsymbol{u_i}$$????

$$\boldsymbol{v} = Squashing(\boldsymbol{s}) = \boxed{\frac{\|\boldsymbol{s}\|^2}{1+\|\boldsymbol{s}\|^2} \frac{\boldsymbol{s}}{\|\boldsymbol{s}\|}}$$

[Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic Routing Between Capsules. *arXiv preprint arXiv:1710.09829*.]

Dynamic *Routing* *(by Agreement)*

$\|v\|$ is confidence

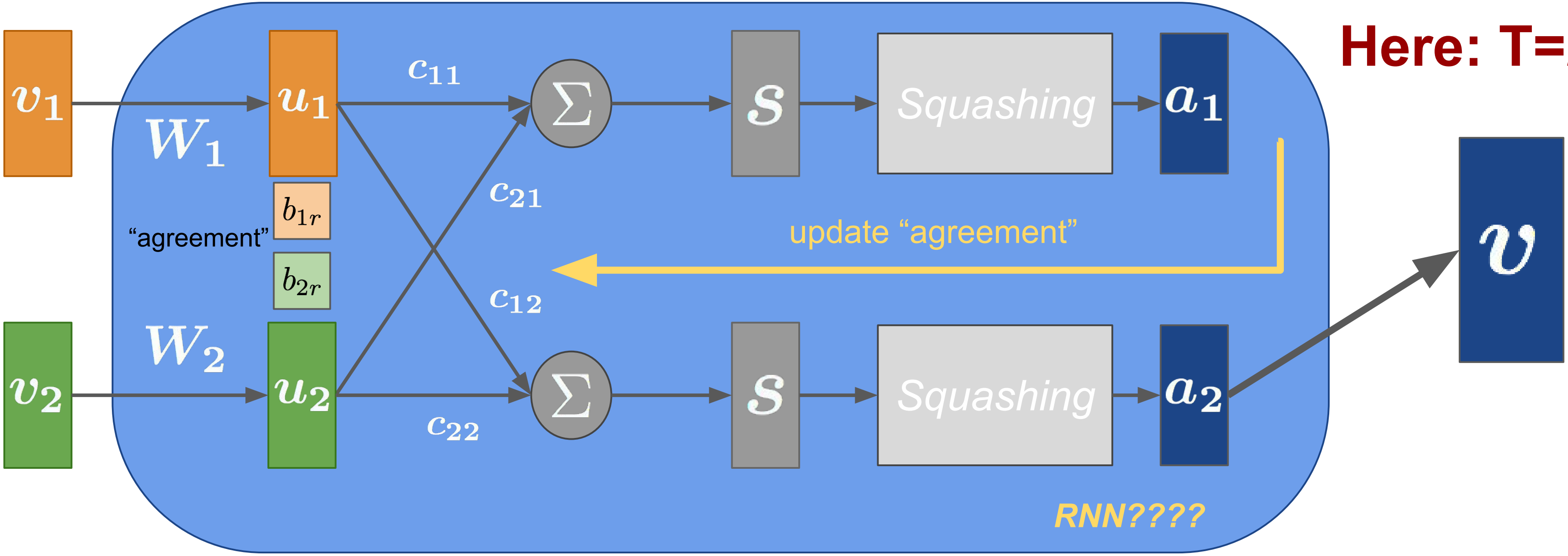
Initialize $b_{11}, b_{21} = 0$

***for** r **in** range(1…T)*

$$c_{1r}, c_{2r} = \mathrm{softmax}(b_{1r}, b_{2r})$$
$$a_r = squashing(c_{1r}u_1 + c_{2r}u_2)$$
$$b_{1(r+1)} = b_{1r} + a_r \cdot u_1$$
$$b_{2(r+1)} = b_{2r} + a_r \cdot u_2$$

Routing Algorithm

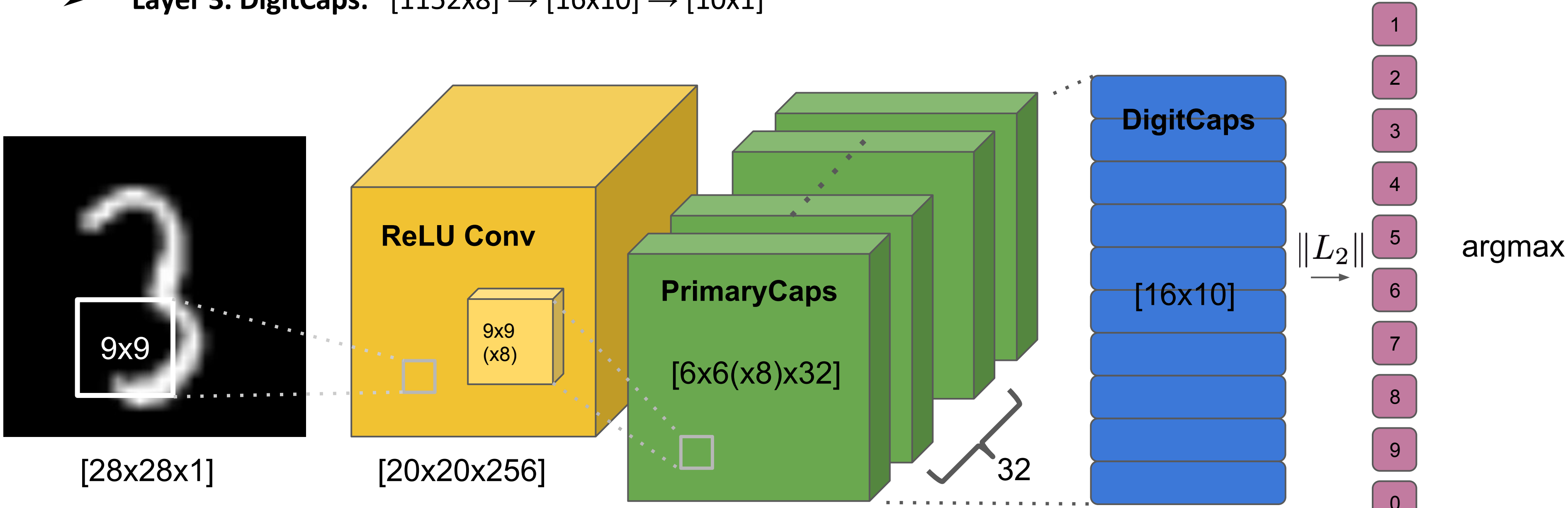
Here: T=2



[Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic Routing Between Capsules. *arXiv preprint arXiv:1710.09829*.]

A Capsule Network (CapsNet) for MNIST

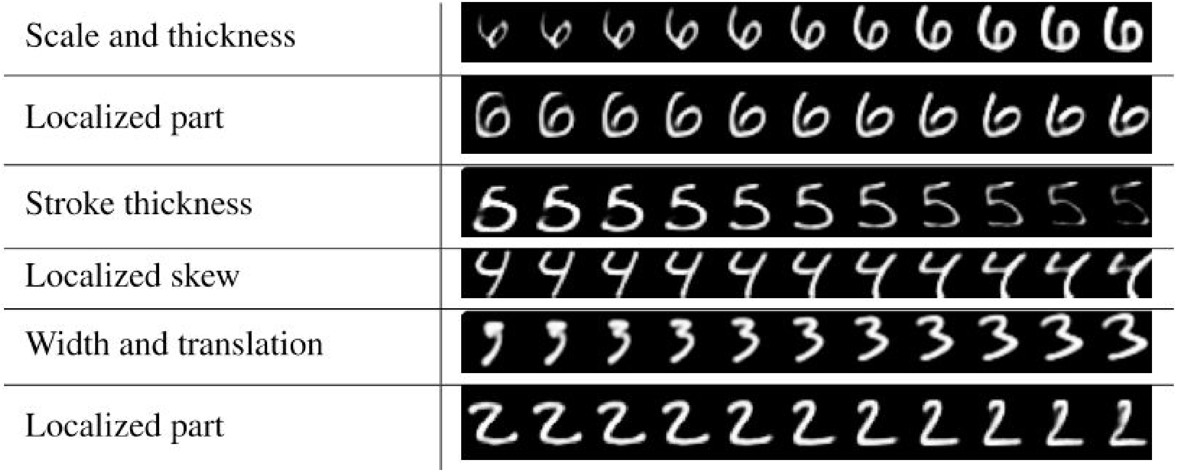
- **Layer 1. ReLU Conv:** [28x28x1] → [20x20x256]
- **Layer 2. PrimaryCaps:** [20x20x256] → [6x6(x8)x32] → [1152x8]
- **Layer 3. DigitCaps:** [1152x8] → [16x10] → [10x1]

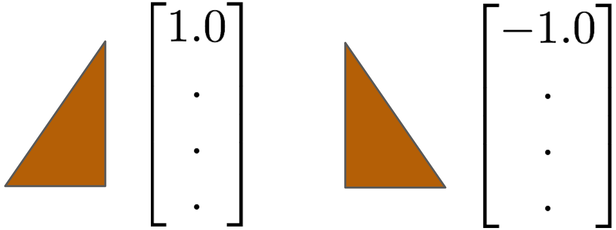


[Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic Routing Between Capsules. *arXiv preprint arXiv:1710.09829*.]

Interpretable Activity Vector

➢ Each dimension contains a specific information (pattern)





[Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic Routing Between Capsules. *arXiv preprint arXiv:1710.09829*.]

Summary of CapsNet

- Keypoints of Capsule:
    - **Vector** → **Vector** (**Tensor** → **Tensor**)
    - **Encapsulate** entity or its **pattern**
    - **Routing** by *agreement*
    - **Invariance** v.s. **Equivariance**

- Future works:
    - Other squashing
    - Improving routing process
    - …

[Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic Routing Between Capsules. *arXiv preprint arXiv:1710.09829*.]

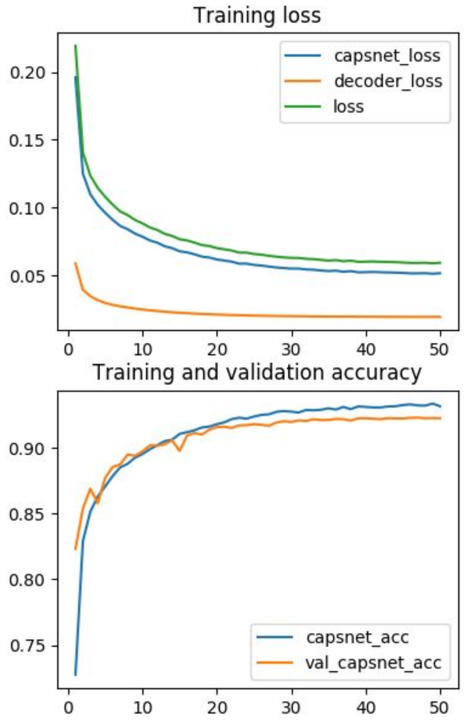
References of this Section

1. [Hinton, G. E., Krizhevsky, A., & Wang, S. D. (2011, June). **Transforming autoencoders**. In International Conference on Artificial Neural Networks (pp. 44-51). Springer, Berlin, Heidelberg.]
2. [Su, J., Vargas, D. V., & Kouichi, S. (2017). One pixel attack for fooling deep neural networks. *arXiv:1710.08864*.]
3. [Hinton, G (2017). **What's wrong with convolutional neural nets**. https://www.youtube.com/watch?v=Mqt8fs6ZbHk&t=562s]
4. [Sabour, S., Frosst, N., & Hinton, G. E. (2017). **Dynamic Routing Between Capsules**. *arXiv:1710.09829*.]
5. [**Under double-blink review (ICLR 2018). Matrix Capsules with EM Routing.**] Rating results: 4, 6, 7
6. [Sukhbaatar, S., Weston, J., & Fergus, R. (2015). End-to-end memory networks. In Advances in neural information processing systems (pp. 2440-2448).]

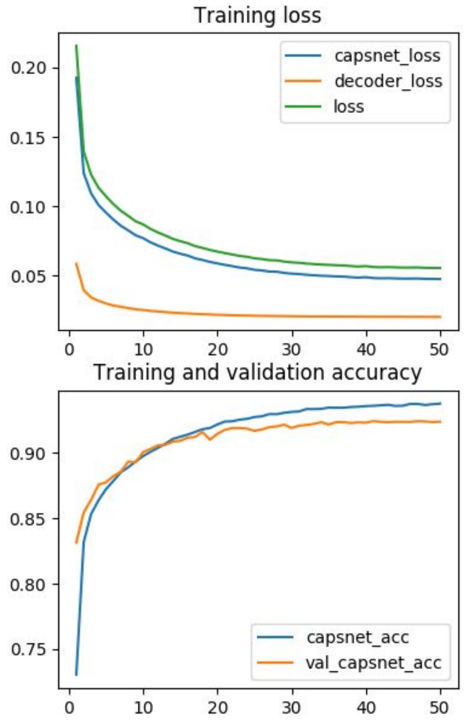
Q&A

BACKSTAGE SLIDES

Routing v.s. Backpropagation (on Fashion MNIST)

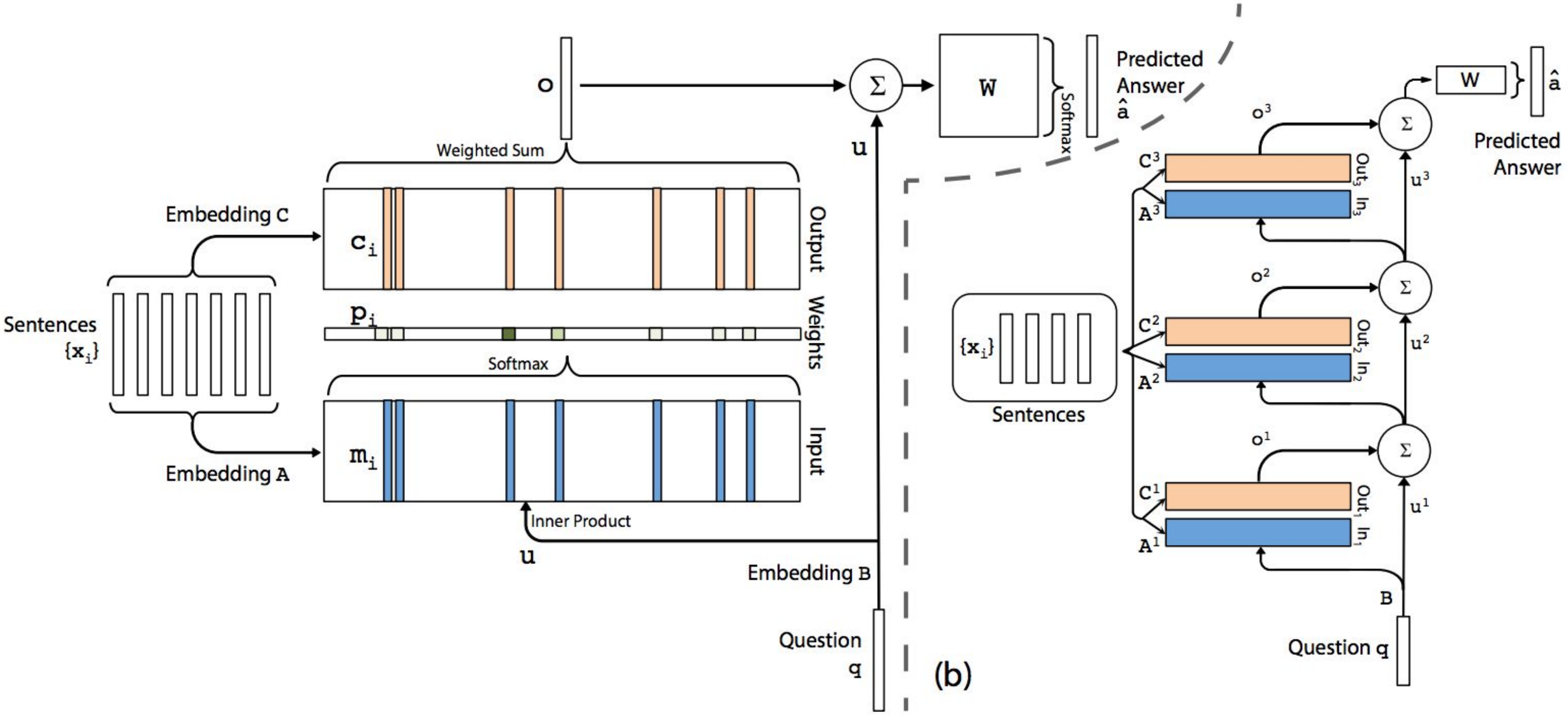


without backpropagation



with backpropagation

Why Routing? Memory Network (Hopping Machanism)



[Sukhbaatar, S., Weston, J., & Fergus, R. (2015). End-to-end memory networks. In Advances in neural information processing systems (pp. 2440-2448).]

Interpretable Activity Vector: Fasion MNIST