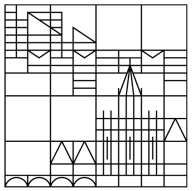
Universität Konstanz



Lecture 8 Deep Learning – Part 3 State-of-the-art CNNs

Dr. Hanhe Lin

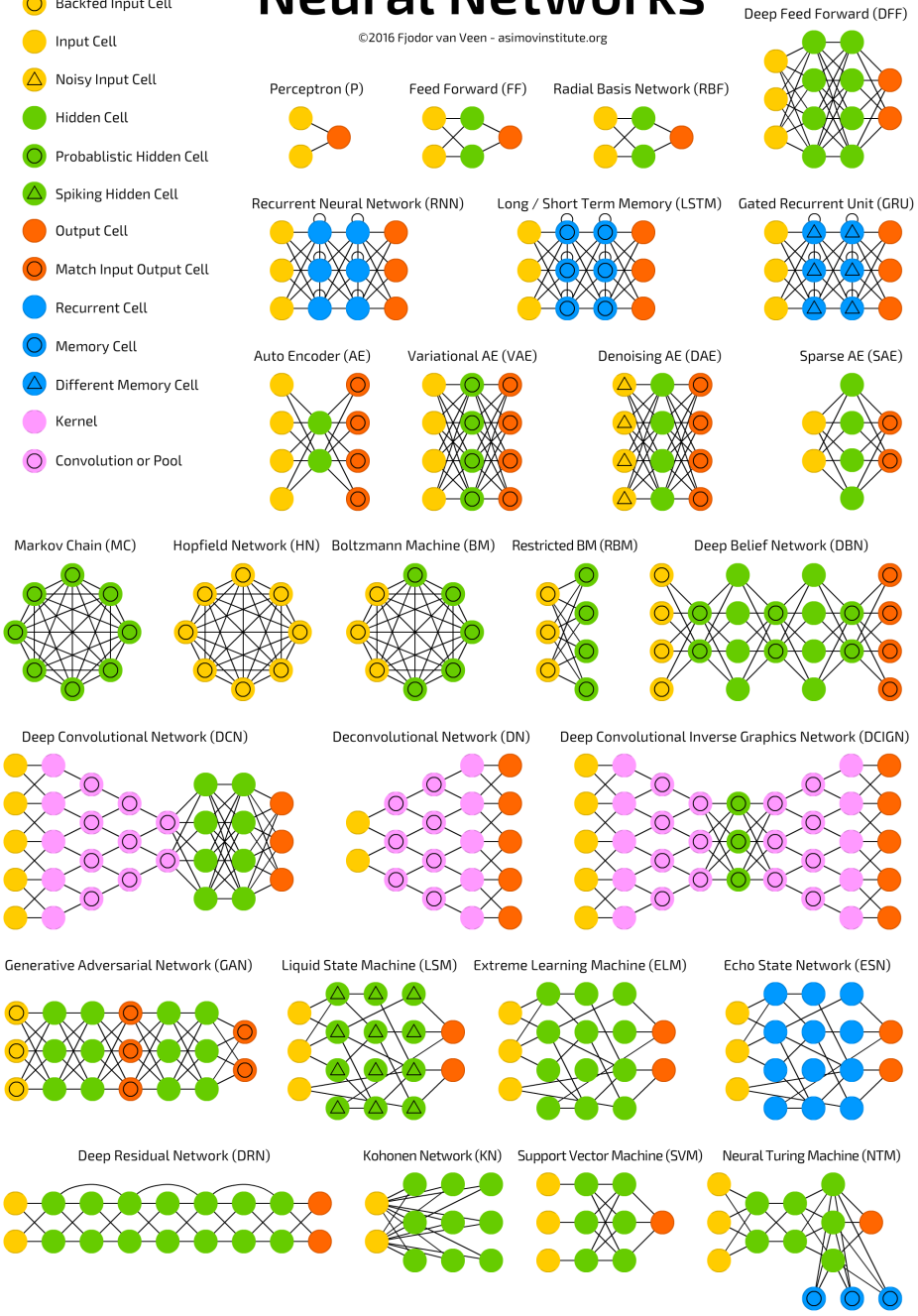
Dept. of Computer and Information Science

University of Konstanz

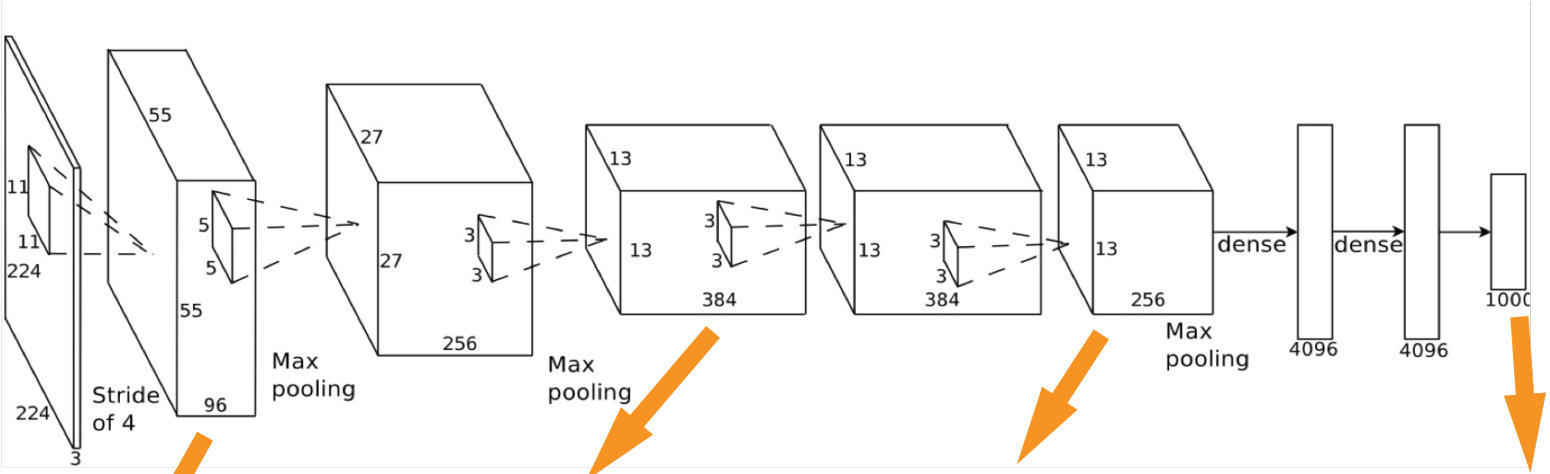
A mostly complete chart of

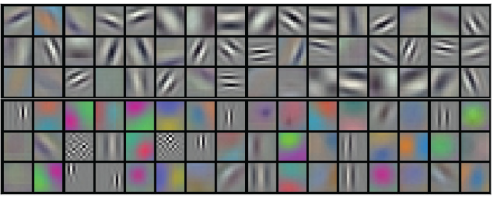
Neural Networks

- Backfed Input Cell
- Input Cell
- Noisy Input Cell
- Hidden Cell
- Probablistic Hidden Cell
- Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- Different Memory Cell
- Kernel
- Convolution or Pool

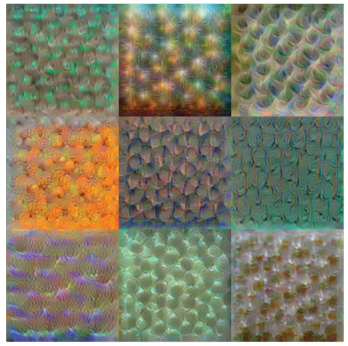


Hierarchical learning





Conv 1: Edge+Blob



Conv 3: Texture

Numerical

Data-driven

Conv 5: Object Parts

Fc8: Object Classes

Discussion

- More parameters, more powerful
  - Which one is better: more layers or more neurons?
  - Disadvantages?

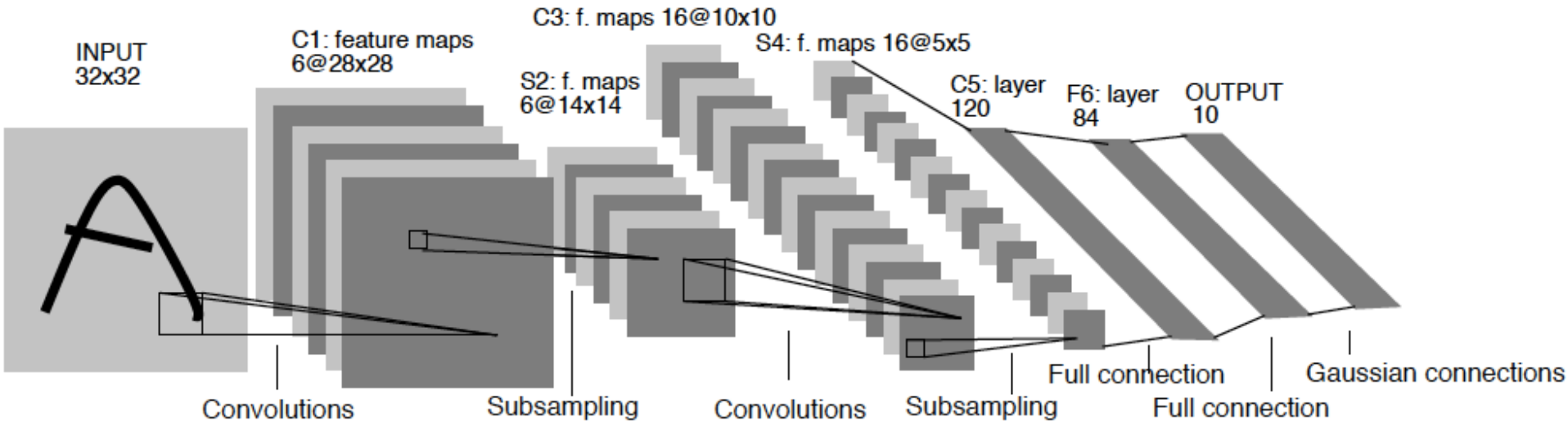
- Neural network is non-convex, gradient descent is susceptible to local optima; however, it works fairly well even though the optima is not global.

- Black box model

LeNet

- LeCun, et al. (1990) developed a pioneer ConvNet for handwritten digits:
  - Many hidden layers
  - Many kernels in each layer
  - Pooling of the outputs of nearby replicated units
  - A wide net that can cope with several digits at once even if they overlap
- This net was used for reading 10% of the checks in North America

LeNet-5 Architecture



- The early layers are convolutional
- The last two layers are fully-connected
- Subsampling: average pooling
- Activation function: tanh

LeNet-5 vs. Human

- LeNet misclassified 82 test patterns
- Notice that most of the errors are cases that people find quite easy
- The human error rate is probably 20 to 30 errors but nobody has had the patience to measure it

LeNet-5: overview

- LeNet uses knowledge about the invariance to design:
  - Local connectivity
  - Weight sharing
  - Pooling
- It achieved 82 errors
  - It can be reduced to about 40 errors by creating a whole lot more training data
  - However, it may require a lot of work and may make learning take much longer
- It also proposed a benchmark dataset, MNIST, including 60,000 training data and 10,000 test data

From handwritten digits to objects

- Recognizing real objects in color images downloaded from the Internet is much more complicated than recognizing handwritten digits:
  - Hundred times as many classes (1000 vs. 10)
  - Hundred times as many pixels (256×256×3 color vs. 28×28 gray)
  - Cluttered scenes requiring segmentation
  - Multiple objects in each image

- Now the question is: will the same type of convolutional neural network work?

What is ILSVRC?

- The ImageNet is an image dataset, containing 14,197,122 annotated images organized by the semantic hierarchy of WordNet

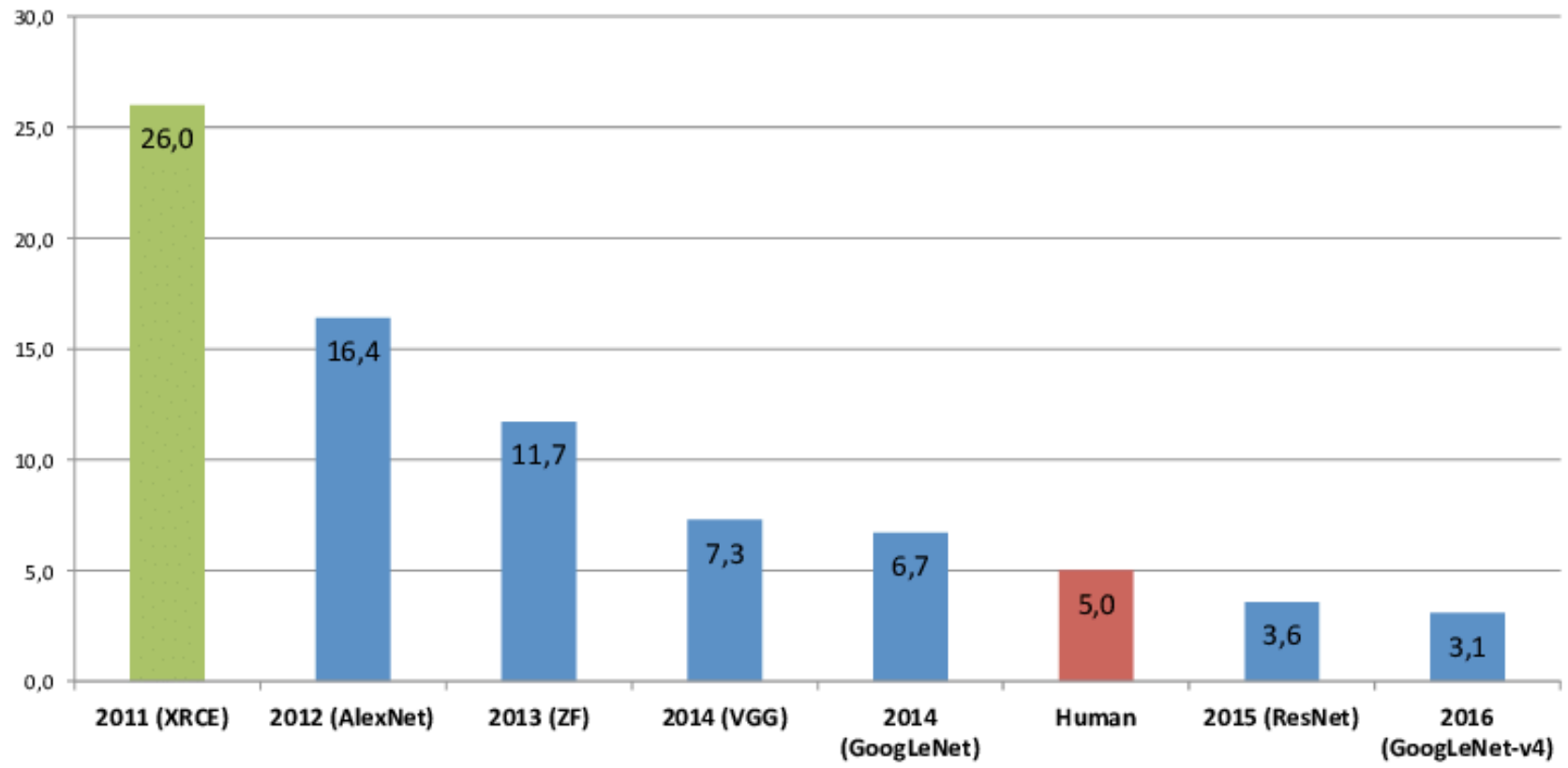
- ImageNet Large Scale Visual Recognition Challenge (ILSVRC) uses a subset of ImageNet images for training the algorithms and some of ImageNet's image collection protocols for annotating additional images for testing the algorithms

- ILSVRC over the years has consisted of one or more of the following tasks:
  - Image classification
  - Single-object localization
  - Object detection

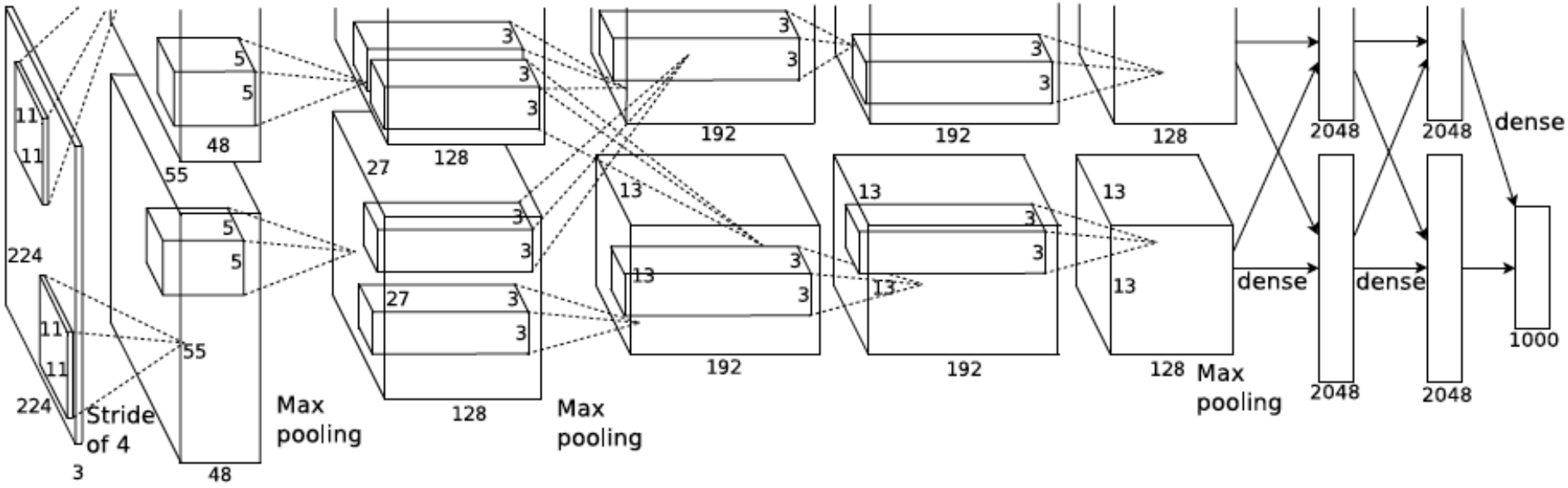
ILSVRC task

- Image classification
  - Each image contains one ground truth label of 1000 object categories
  - Get the "correct" class in the top 5 bets
- Single-object localization
  - Each image contains one ground truth label of 1000 object categories.
  - Additionally, every instance of this category is annotated with an axis-aligned bounding box
  - For each bet, put a box around the object. The correct localization must have at least 50% overlap with the ground truth bounding box
- Object detection
  - The images are annotated with axis-aligned bounding boxes indicating the position and scale of every instance of each target object category
  - Evaluation is similar to single-object localization, but with multiple objects

ILSVRC image classification winners (Top-5 error rate)



AlexNet Architecture



AlexNet: Review

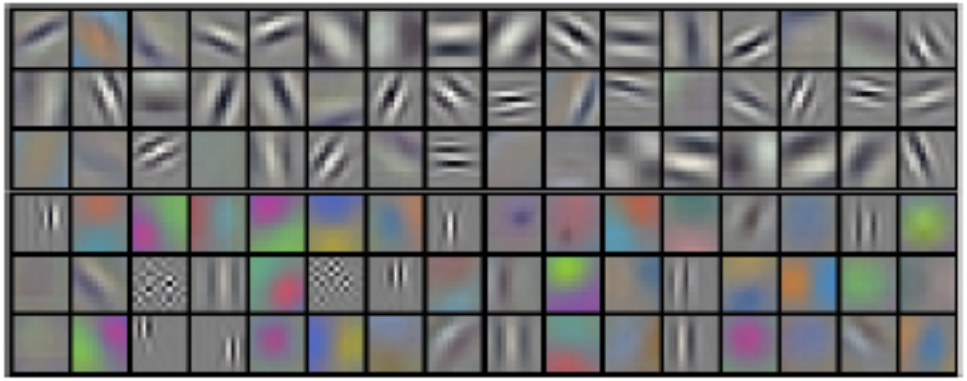
- The net contains eight layers with weights: the first five are convolutional and the remaining three are fully-connected
- Number of neurons: 150,528➔253,440➔186,624➔64,896➔64,896➔ 43,264➔4096➔4096➔1000
- Different from LeNet:
  - Bigger, deeper
  - ReLu: make training much faster and are more expressive than logistic units
  - Max pooling
  - Local response normalisation
  - Featured Convolutional Layers stacked on top of each other
- Training on two GPUs: half of the kernels (or neurons) on each GPU, with one additional trick: the GPUs communicate only in certain layers
- 90 epochs with five to six days

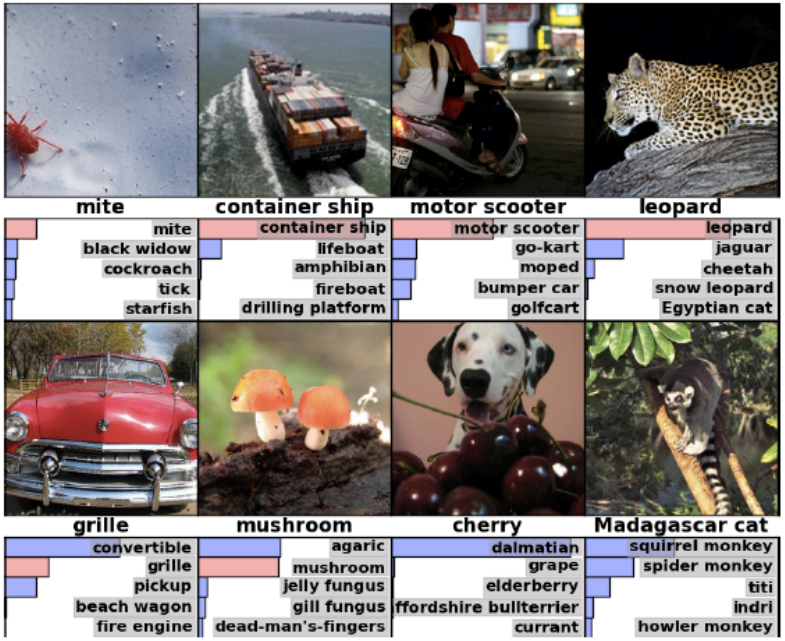
Tricks that reduce over-fitting

- Data augmentation:
  - Train on random 224×224 patches from the 256×256 images to get more data.
  - Also use left-right reflections of the images
  - At test time, combine the opinions from ten different patches: The four 224×224 corner patches plus the central 224×224 patch plus the reflections of those five patches

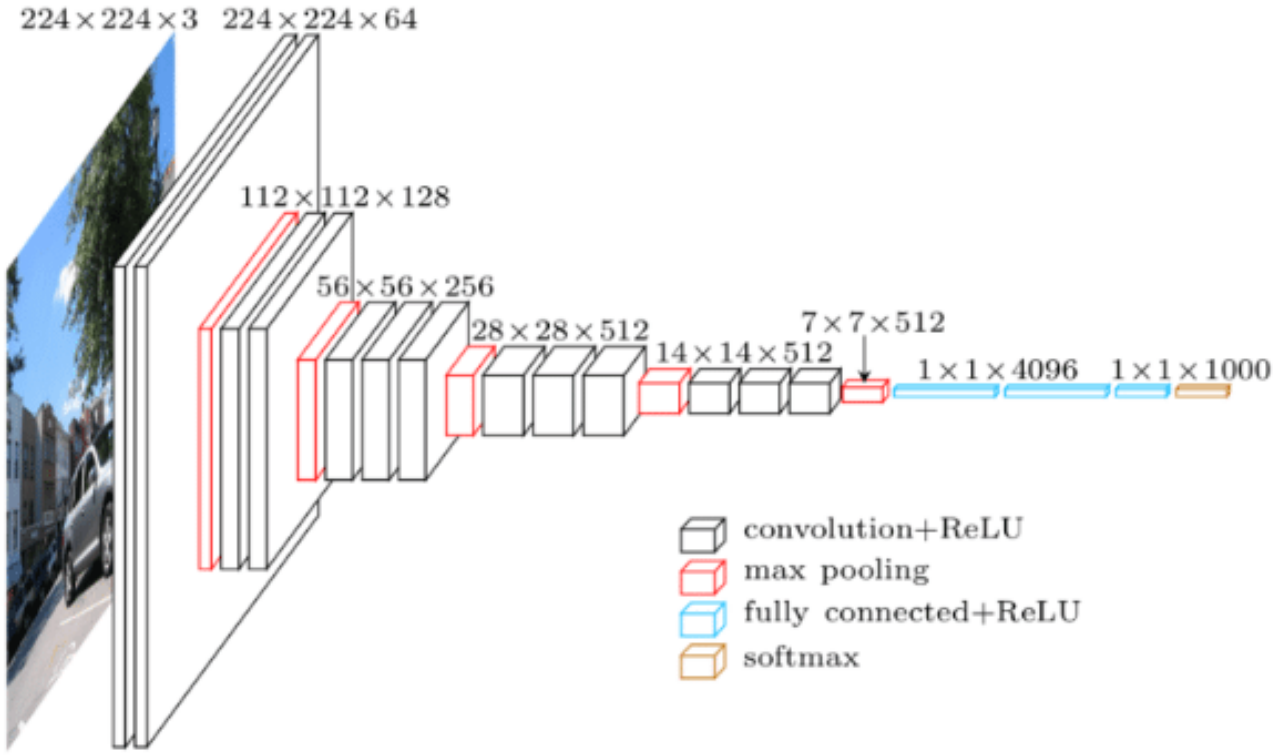
- Dropout:
  - Dropout in the first two fully-connected layers

AlexNet: Results





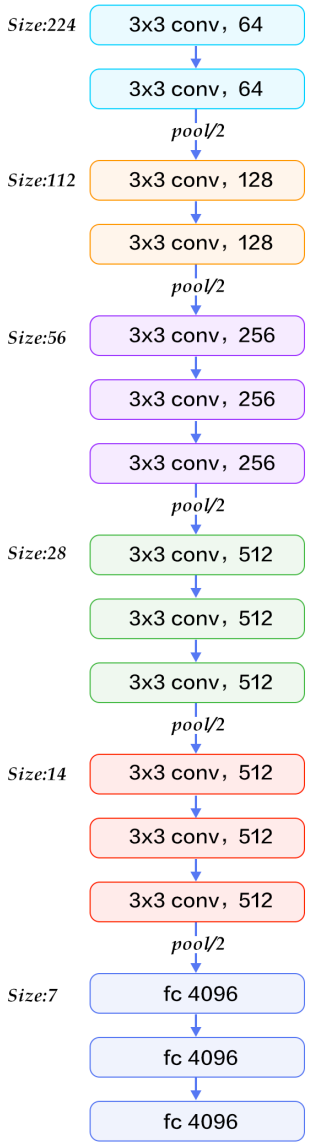
VGG16 Architecture



Gradually decrease spatial dimenions while increase depth

VGGNet

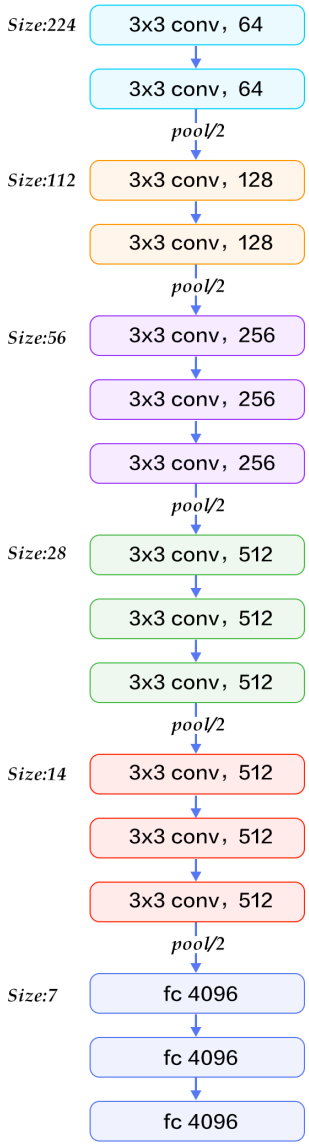
- VGG: Visual Geometry Group from University of Oxford
- 3×3 kernel stride 1, zero-padding 1 and 2×2 max pooling stride 2
- 16 or 19 layers



Smaller kernels, deeper network

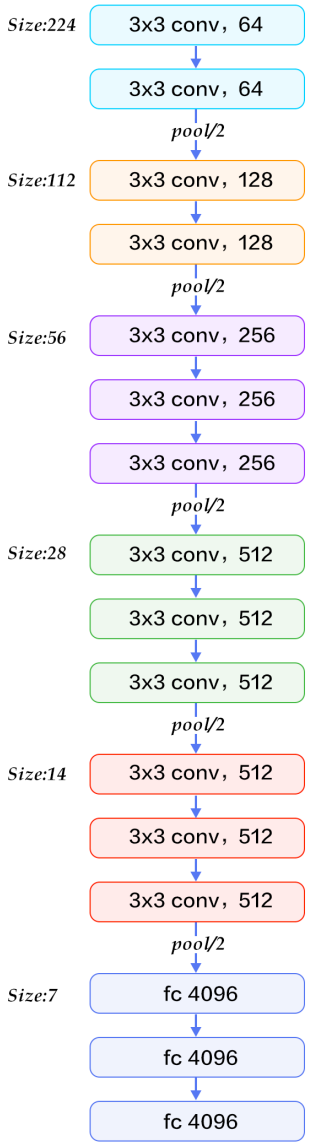
- What have we gained by using a stack of three 3×3 conv. layers instead of a single 7×7 layer?
  - Incorporate three non-linear rectification layers instead of a single one, which makes the decision function more discriminative

- Decrease the number of parameters from $(7^2C^2) = 49C^2$ to $3(3^2C^2)=27C^2$

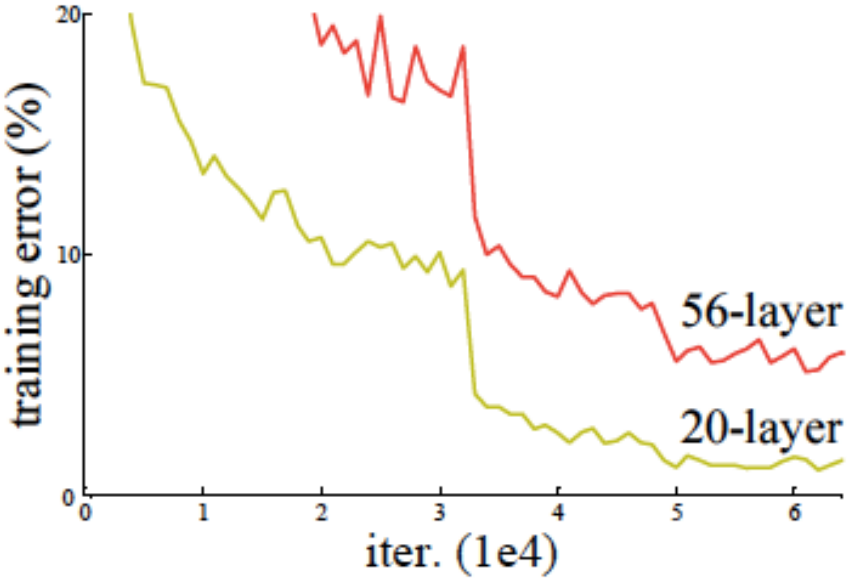


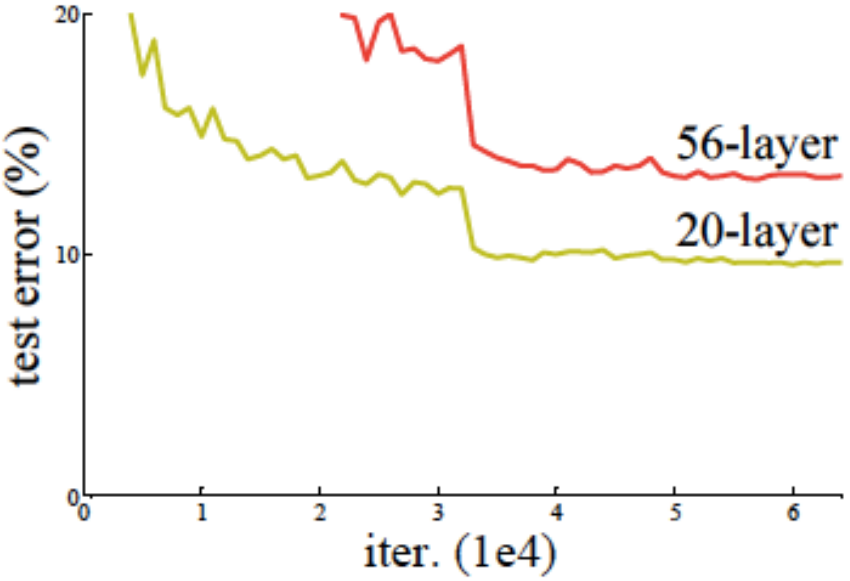
VGGNet: Review

- ILSVRC'14 2nd in classification, 1st in localization
  - Similar training procedure as AlexNet
  - VGG19 only slightly better than VGG16, but requires more memory
- FC4096 features generalize well to other tasks



ResNet: Motivation

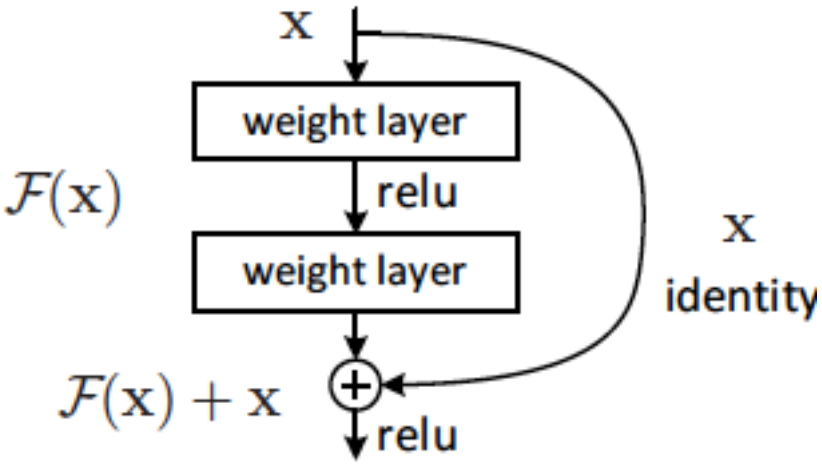




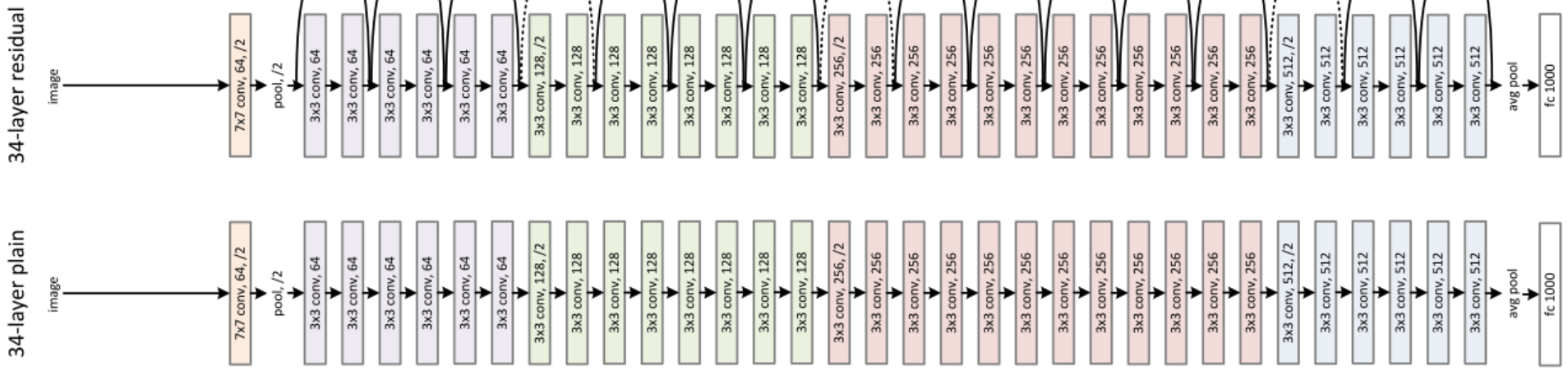
- Stacking more layers does not mean better performance
- With the network depth increasing, accuracy gets saturated and then degrades rapidly
- Such degradation is not caused by over-fitting

Residual block

- Hypothesis: The problem is an optimization problem, deeper models are harder to optimize
- The deeper model should be able to perform at least as well as the shallower model
- The added layers are identity mapping, and the other layers are copied from the learned shallower model



ResNet Architecture



GoogLeNet: Motivation

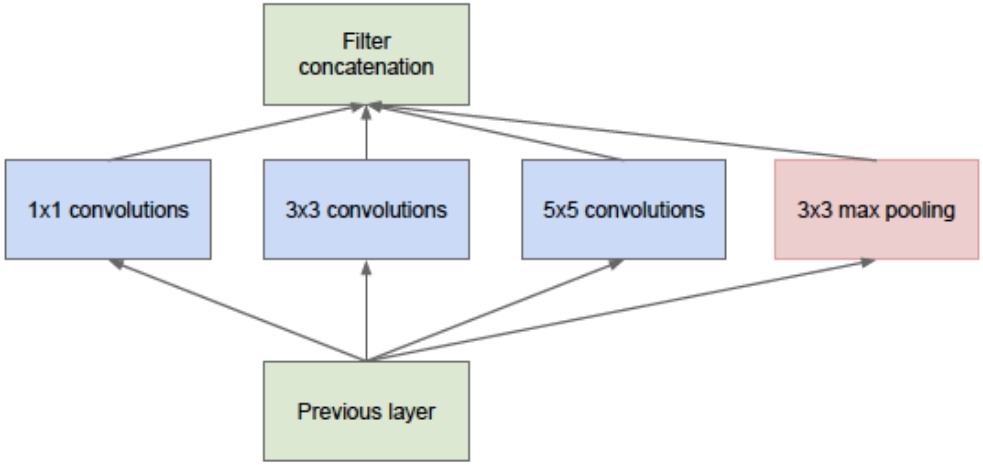
- The most straightforward way of improving the performance of deep neural networks is by increasing their size, both depth and width

- Increasing network size has two drawbacks:
  - A larger number of parameters! prove to over-fitting
  - The dramatically increased use of computational resources

- Increase the depth and width of the network while keeping the computational budget and number of parameters constant

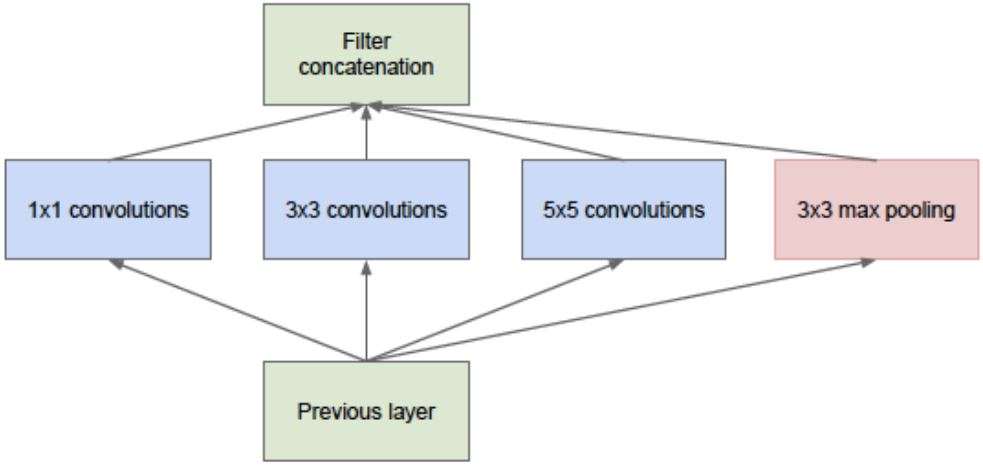
Inception module - naive

- Apply parallel operations on the input from from previous layer:
  - Multiple kernel size for convolution (1×1, 3×3, 5×5)
  - Pooling operation (3×3)
- Concatenate all filter outputs together depth-wise



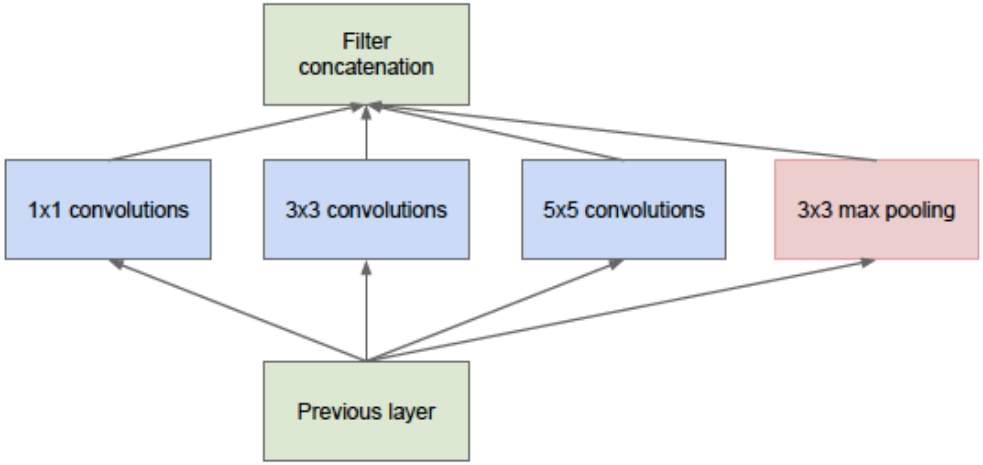
Inception module - naive

- Let assume the setup of inception module is:
  - Input size: 28×28×256
  - 1×1 convolutional kernels: 128 with stride 1
  - 3×3 convolutional kernels: 192 with stride 1, zero-padding 1
  - 5×5 convolutional kernels: 96 with stride 1, zero-padding 2
  - 3×3 max pooling: stride 1, zero-padding 1
- The output is 28×28×(128+192+96+256) = 28×28×672



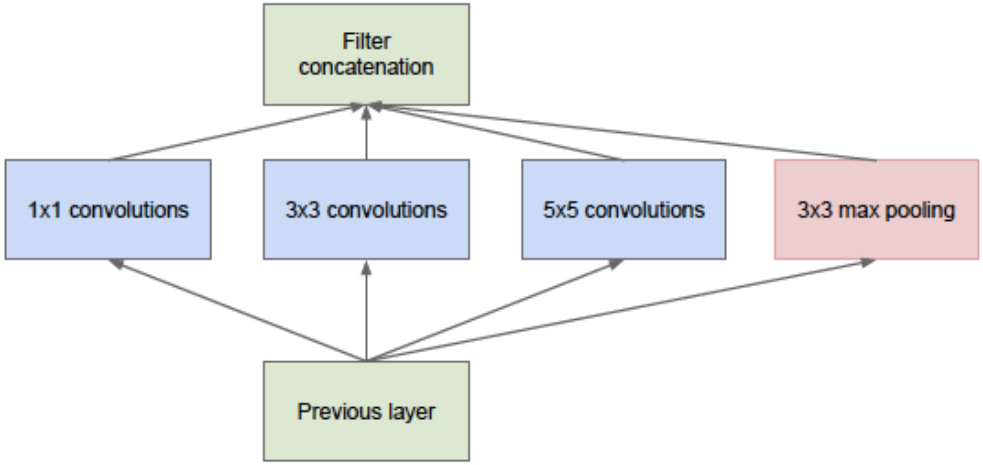
Inception module - naive

- **Conv Ops**:
  - 1×1 conv, 128: 28×28×128×1×1×256
  - 3×3 conv, 192: 28×28×192×3×3×256
  - 5×5 conv, 96: 28×28×96×5×5×256
  - **Total: 854M ops**
- What is the problem?



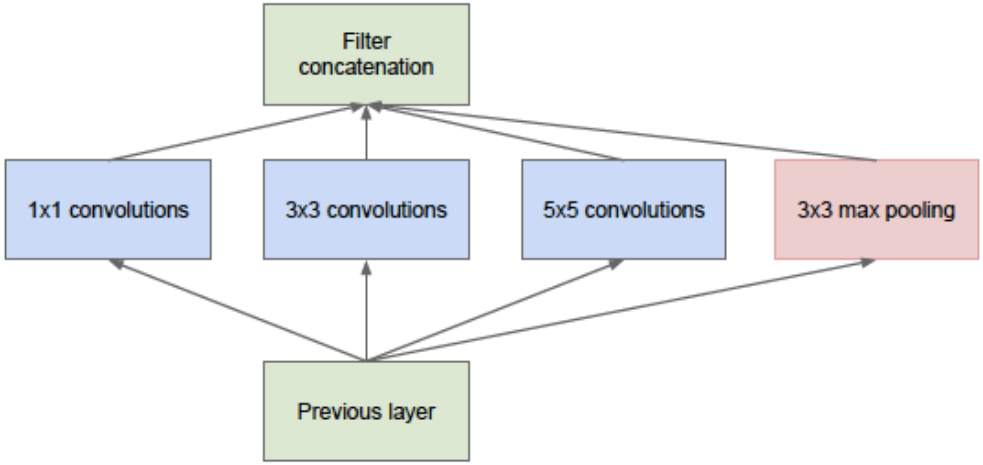
Inception module - naive

- **Conv Ops**:
  - 1×1 conv, 128: 28×28×128×1×1×256
  - 3×3 conv, 192: 28×28×192×3×3×256
  - 5×5 conv, 96: 28×28×96×5×5×256
  - **Total: 854M ops**
- What is the problem?
  - Very expensive to compute



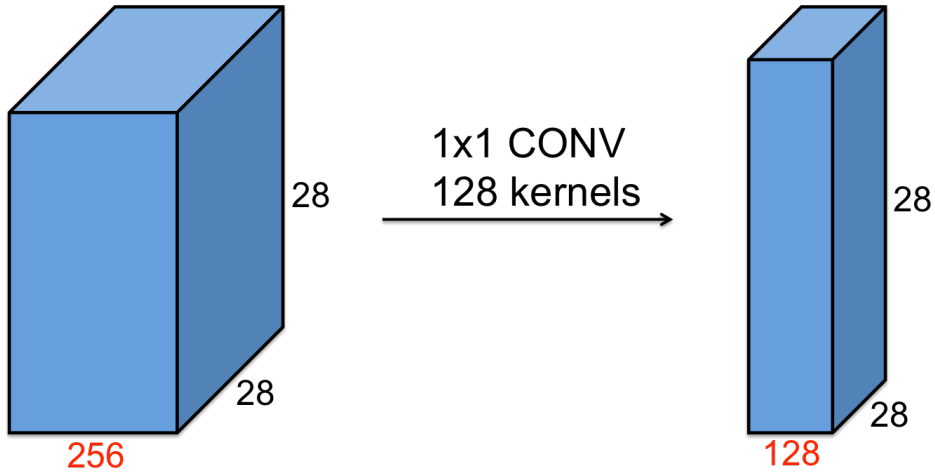
Inception module - naive

- **Conv Ops**:
  - 1×1 conv, 128: 28×28×128×1×1×256
  - 3×3 conv, 192: 28×28×192×3×3×256
  - 5×5 conv, 96: 28×28×96×5×5×256
  - **Total: 854M ops**
- What is the problem?
  - Very expensive to compute
- Solution: "bottleneck" layers that use 1×1 convolutions to reduce feature depth

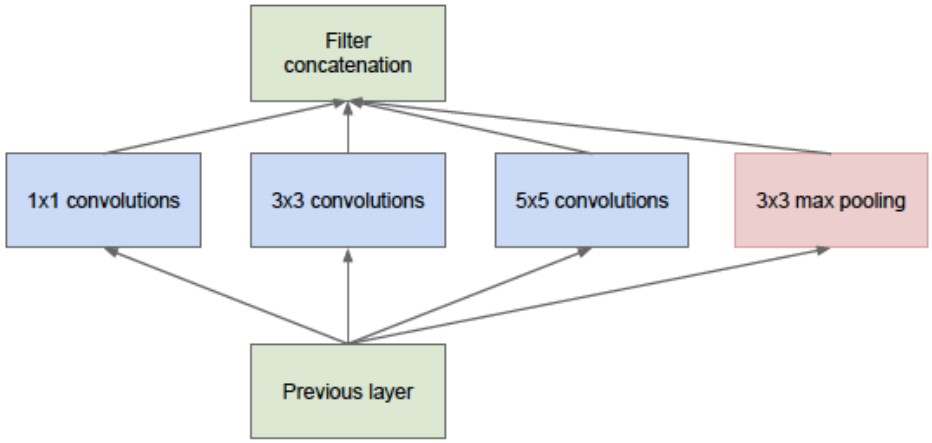


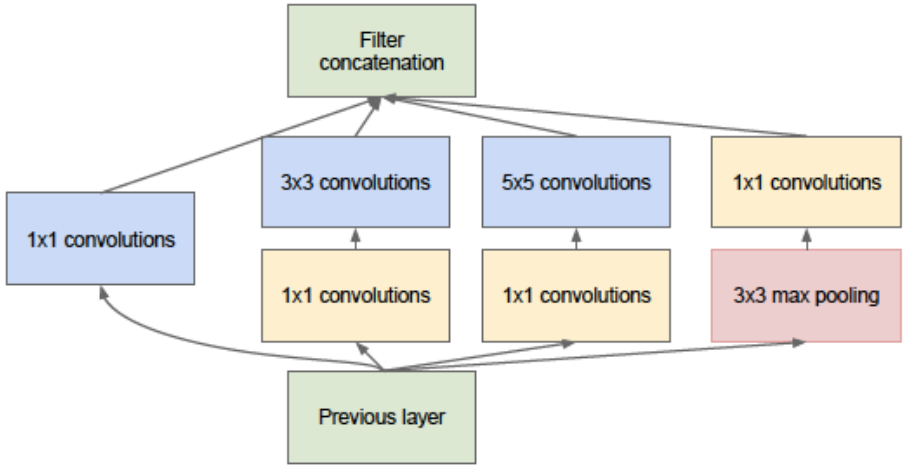
1×1 convolutions

- Preserve spatial dimensions, reduces depth
- Projects depth to lower dimension (combination of feature maps)



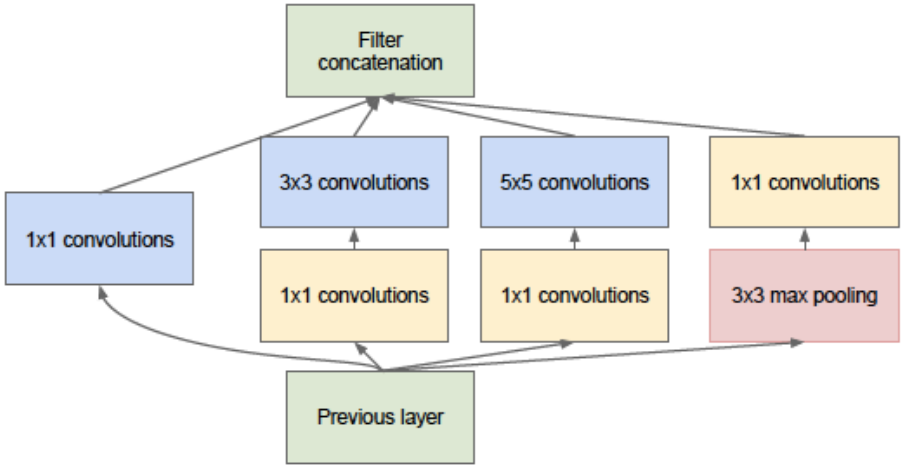
Inception module with dimensionality reduction



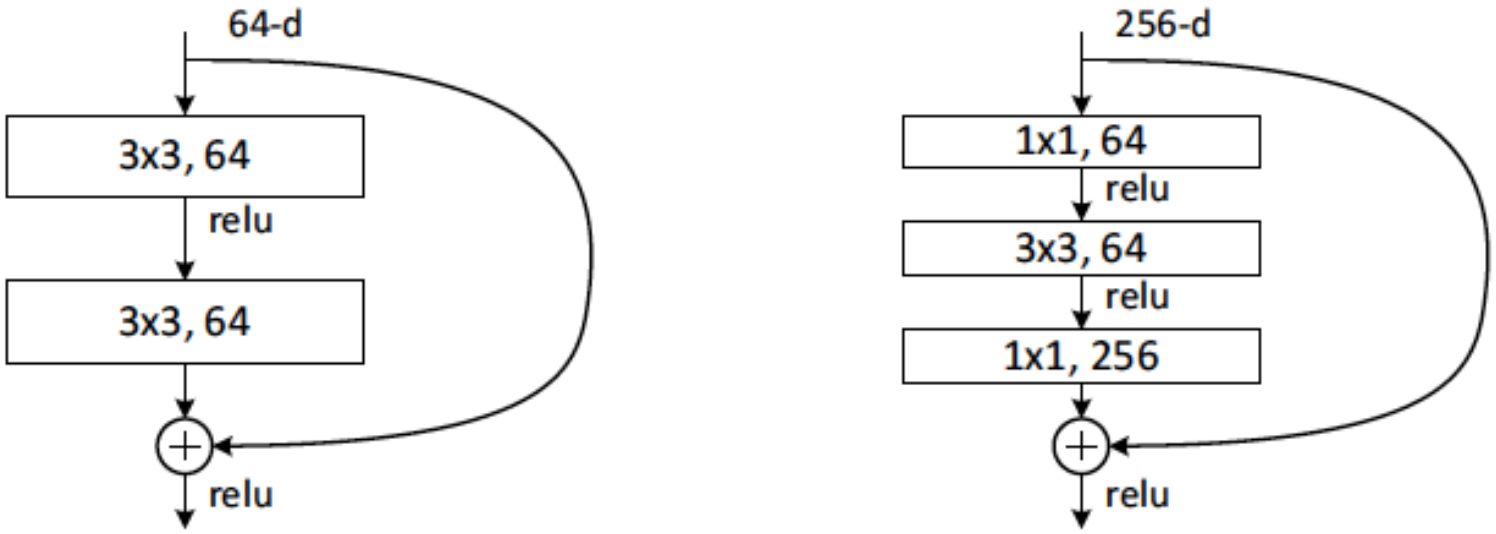


Inception module with dimensionality reduction

- If we adding three layers with“1×1 conv, 64 kernels”, then
- **Conv Ops:**
  - 1×1 conv, 128:
  - 28×28×128×1×1×256
  - 1×1 conv, 64: 28×28×64×1×1×256
  - 3×3 conv, 192: 28×28×192×3×3×64
  - 1×1 conv, 64: 28×28×64×1×1×256
  - 5×5 conv, 96: 28×28×96×5×5×64
  - 1×1 conv, 64: 28×28×64×1×1×256
  - **Total: 358M ops**

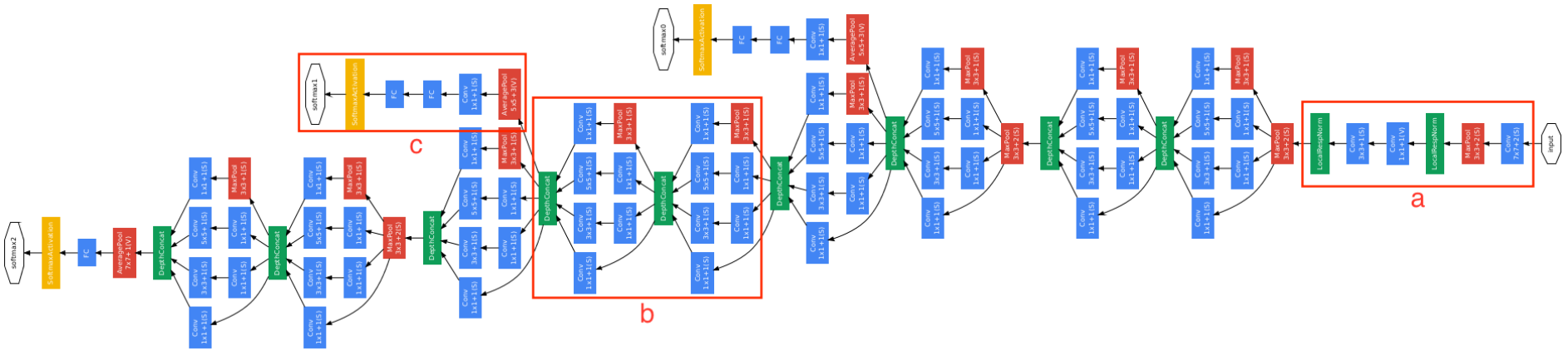


Residual block with dimensionality reduction

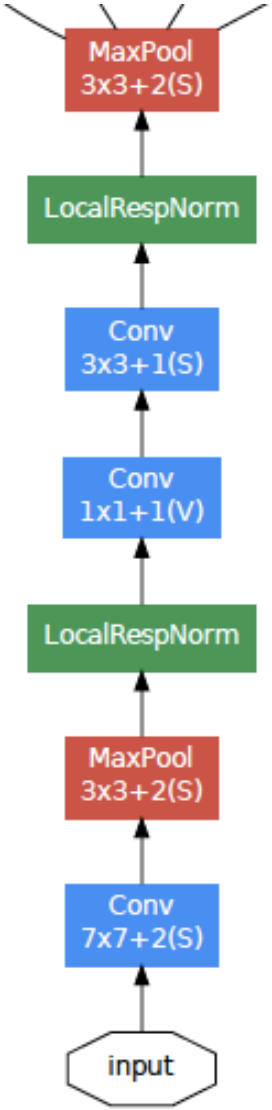


- Similarly, ResNet also uses “bottleneck” layer to improve efficiency for deeper networks

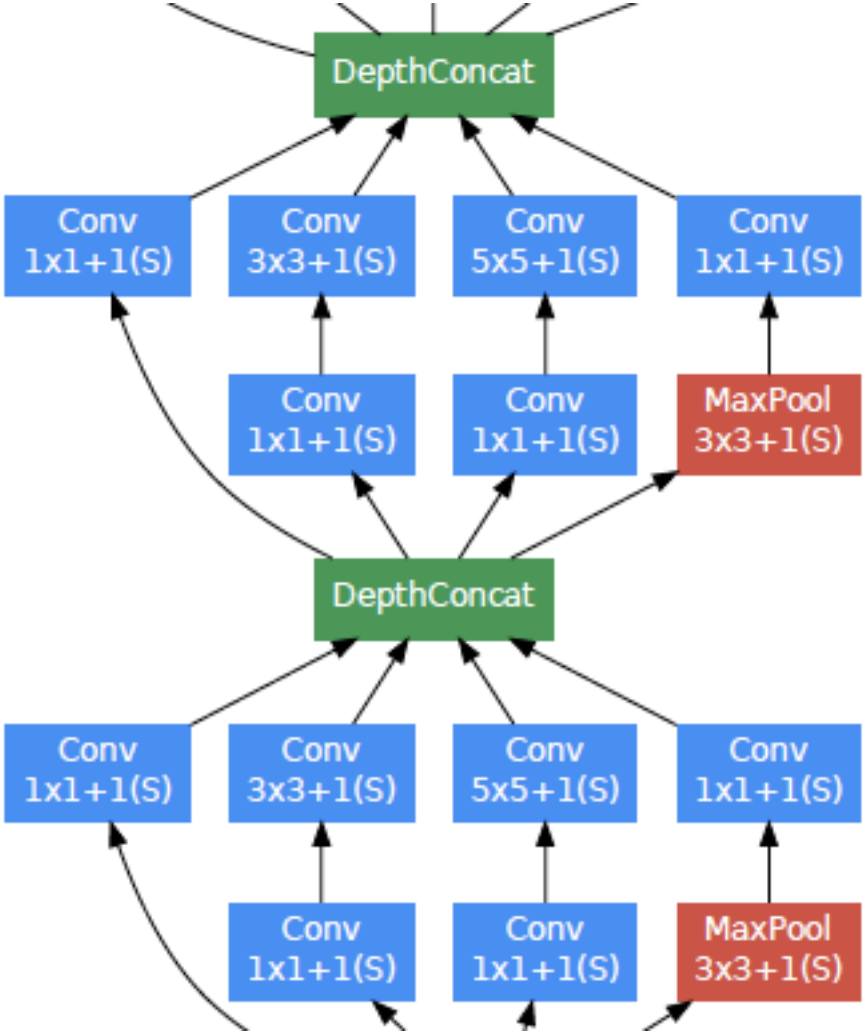
GoogLeNet Architecture



Part a: Stem network

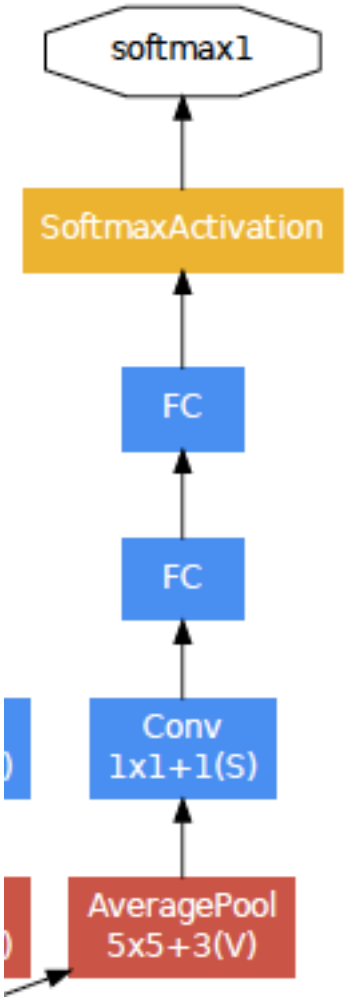


Part b: Stacked inception modules



Part c: Auxiliary classifiers

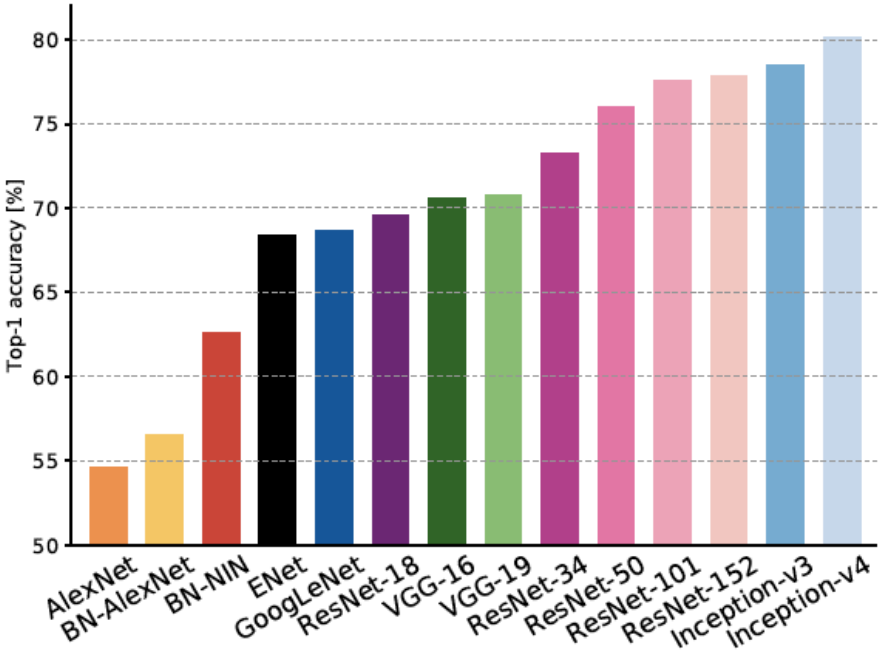
- Features produced by the layers in the middle of the network should be very discriminative
- Auxiliary classifiers connected to these intermediate layers, discrimination in the lower stages in the classifier was expected
- During training, their loss gets added to the total loss of the network with a discount weight (the losses of the auxiliary classifiers were weighted by 0.3).
- At inference (prediction) time, these auxiliary networks are discarded

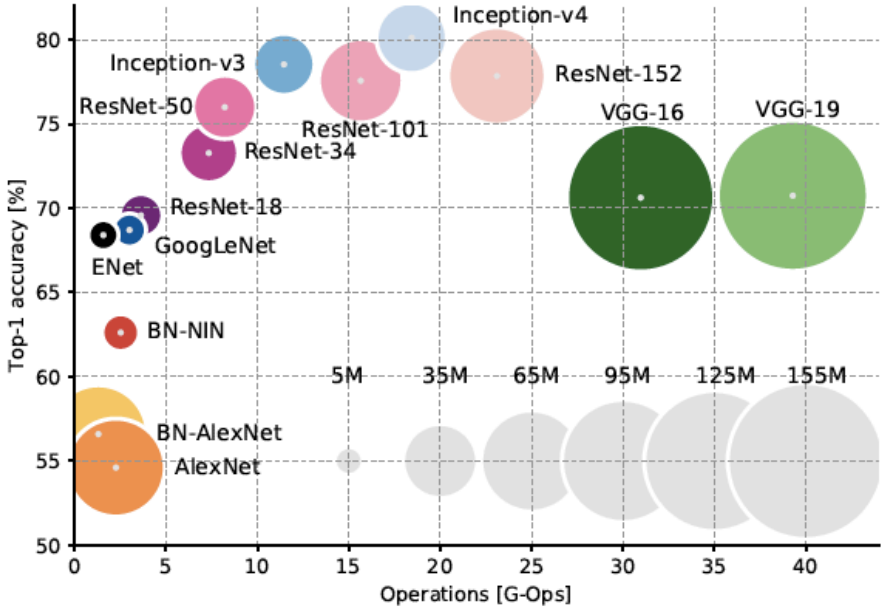


GoogLeNet: review

- GoogLeNet is deeper with computational efficiency
  - 22 layers
  - Efficient “Inception” module
  - Only 5 million parameters, 12x less than AlexNet
  - ILSVRC’14 image classification winner (6.7% top 5 error)

Performance comparison



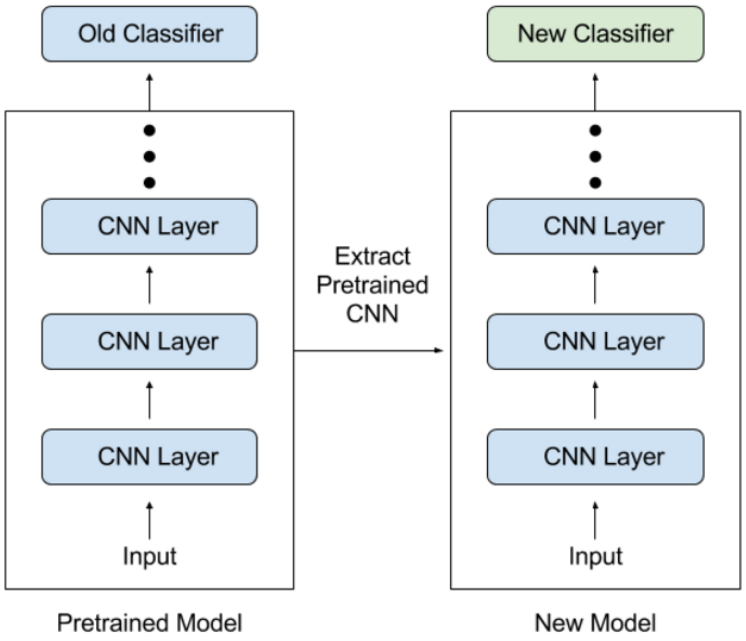


- LeNet: pioneer net for digit recognition
- AlexNet: smaller compute, still memory heavy, lower accuracy
- VGG: Highest memory, most operations
- ResNet: moderate efficiency depending on model, better accuracy
- GoogLeNet: most efficient
- Inception-v4: hybrid of ResNet and Inception, highest accuracy

Transfer learning scenarios

- Motivation: although one deep learning problem is different from another, they may have some properties in common

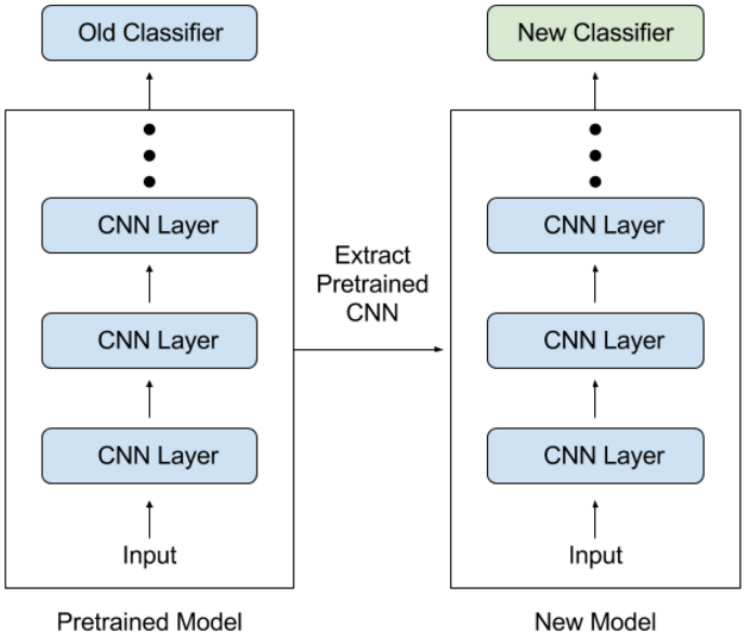
- Definition: storing knowledge gained while solving one problem and applying it to a different but related problem



Visualization of transfer learning

Transfer learning scenarios

- Pretrained models
  - People release their final ConvNet
  - Checkpoints for the benefit of others
- ConvNet as fixed feature extractor
- Fine-tuning the ConvNet



Visualization of transfer learning

ConvNet as fixed feature extractor

- Take a ConvNet pretrained on ImageNet, remove the last fully-connected layer, then treat the rest of the ConvNet as a fixed feature extractor for the new dataset

- Features are called CNN codes

- Train a linear classifier (e.g. Linear SVM or Softmax classifier) for the new dataset CNN codes

Fine-tuning the ConvNet

- Remove the last fully-connected layer, add one or more layers for your task, change output, loss function if necessary

- Fine-tune all the layers of the ConvNet or only fine-tune some higher-level portion of the network while keep some of the earlier layers fixed keeping

- The earlier features of a ConvNet contain more generic features that should be useful to many tasks, but later layers of the ConvNet becomes progressively more specific to the details of the classes contained in the original dataset

When and how to transfer learning?

- Factors to decide the transfer learning scenario: the size of the new dataset and its similarity to the original dataset
  - New dataset is small and similar to original dataset: train a linear classifier on the CNN codes
  - New dataset is large and similar to the original dataset: fine-tune through the full network
  - New dataset is small but very different from the original dataset: train a linear classifier from activations somewhere earlier in the network
  - New dataset is median but very different from the original dataset: fine-tune some top layers while keeping the lower layers fixed
  - New dataset is large and very different from the original dataset: fine-tune through the entire network

Practical advice

- Constraints from pretrained models
  - We cannot arbitrarily take out Conv layers from the pretrained network
  - Due to parameter sharing, we can easily run a pretrained network on images of different spatial size

- Learning rate
  - Use a smaller learning rate for ConvNet weights that are being fine-tune
  - The ConvNet weights are relatively good, so we don't wish to distort them too quickly and too much