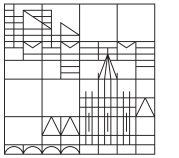
Universität Konstanz



# Deep Learning Programming
# Lecture 8: State-of-the-art Convolutional Neural Networks

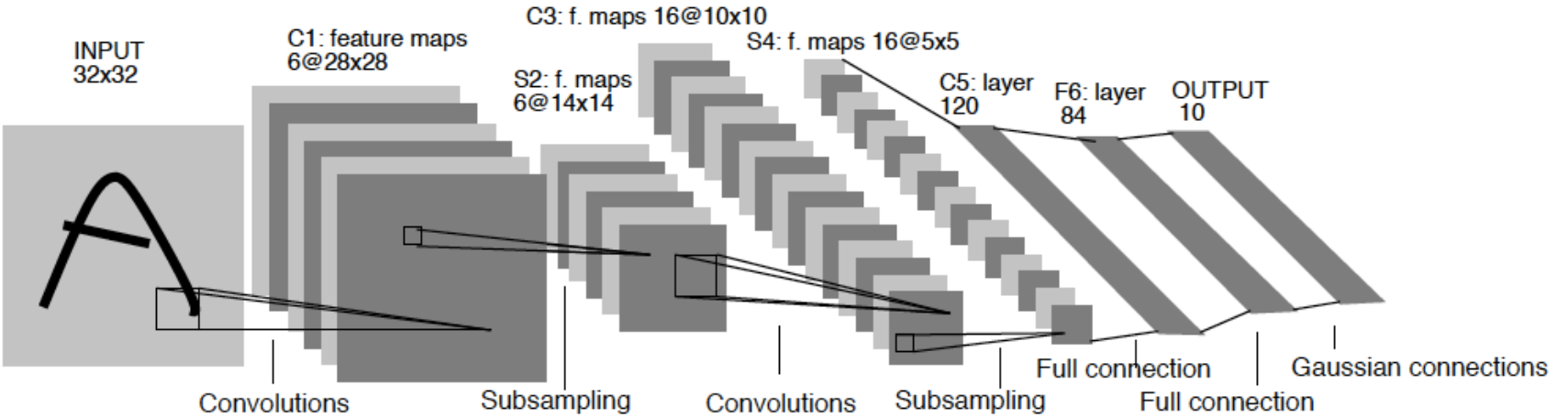
**Dr. Hanhe Lin**

Universität Konstanz, 11.06.2018

# LeNet

- LeCun, et al. 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

# Architecture of LeNet-5




- The early layers were convolutional
- The last two layers were fully-connected
- See a impressive demo of LeNet here

# 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

# Review

- LeNet uses knowledge about the invariance to design:
  - the local connectivity
  - the weight sharing
  - the 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 database, 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 x 256 x 3 color vs 28 x 28 gray)
  - Cluttered scenes requiring segmentation
  - Multiple objects in each image
- Now the question is: will the same type of convolutional neural network work?

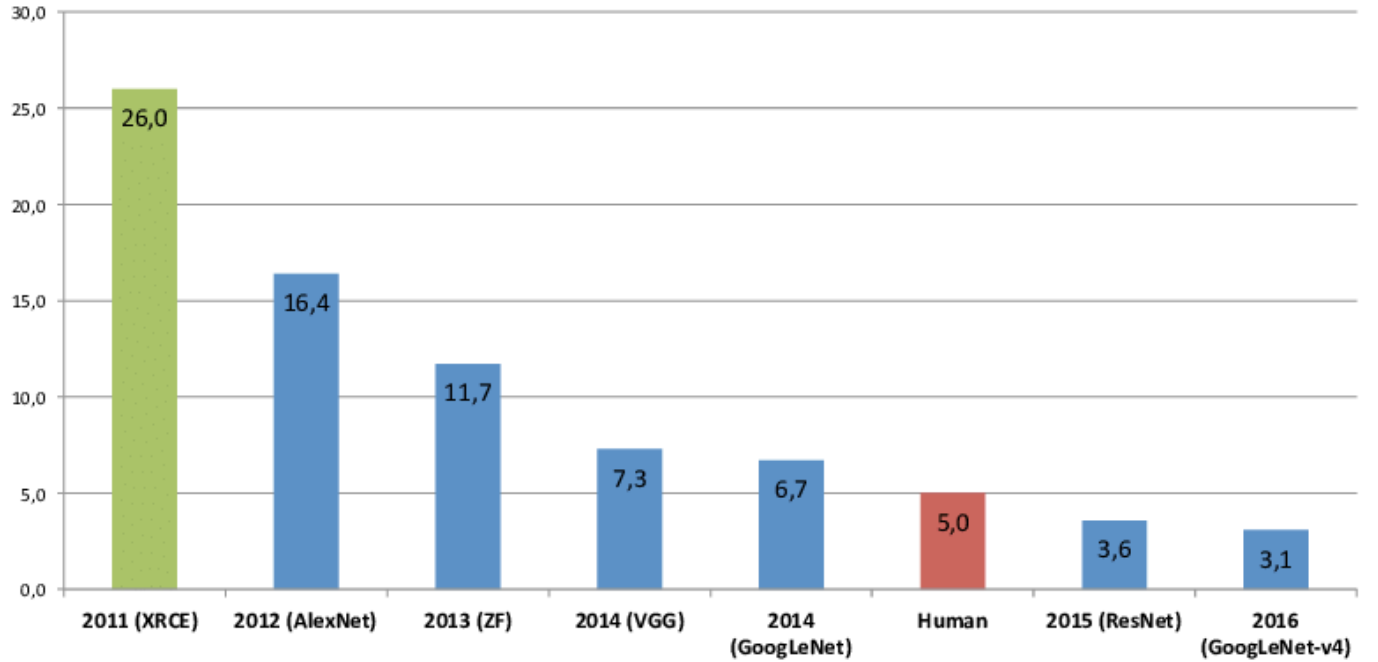
# What is ILSVRC?

- 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

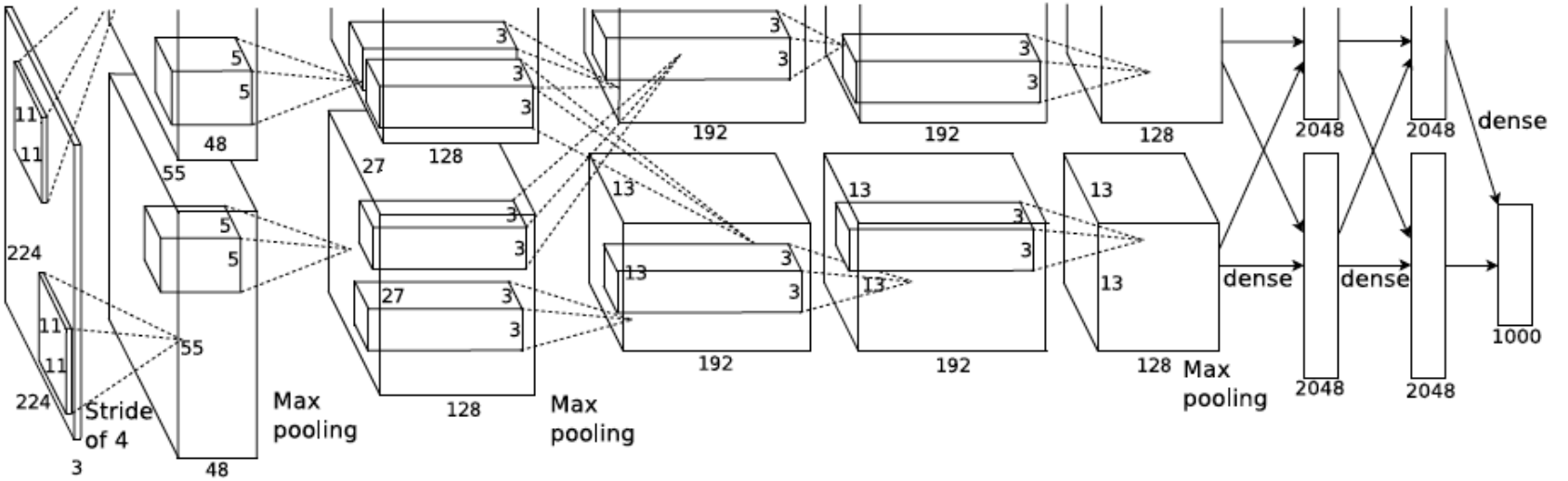
# ILSVRC task

- Image classification (discuss in this week)
  - 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



# Architecture of AlexNet



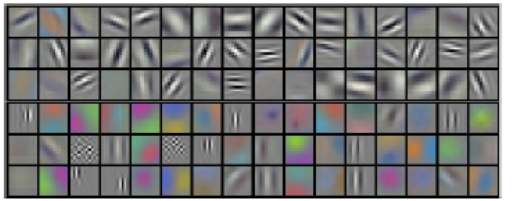
# Architecture of AlexNet

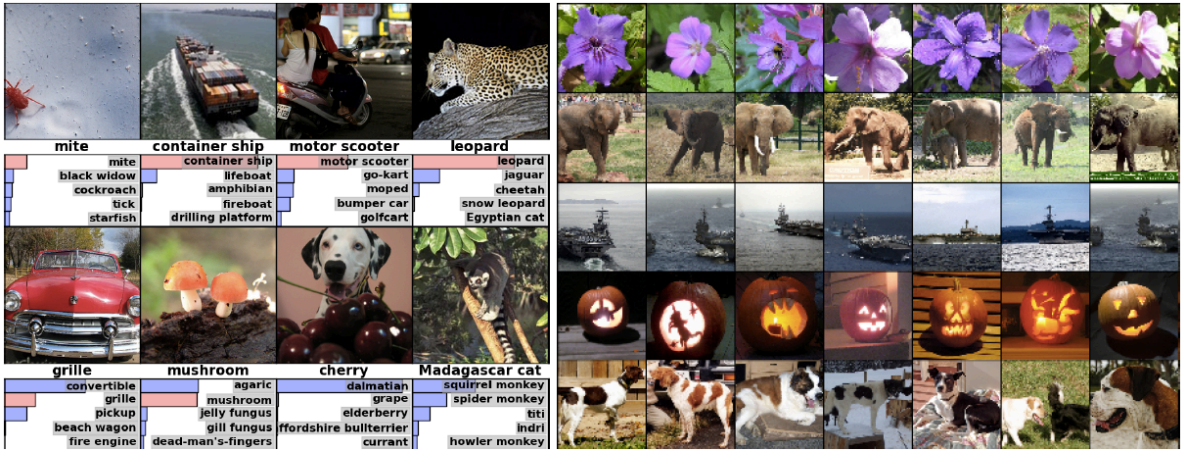
- The net contains eight layers with weights; the first five are convolutional and the remaining three are fully-connected
- Number of neurons: 150,528 $\rightarrow$ 253,440 $\rightarrow$ 186,624 $\rightarrow$ 64,896 $\rightarrow$ 64,896 $\rightarrow$ 43,264 $\rightarrow$ 4096 $\rightarrow$ 4096 $\rightarrow$ 1000
- Difference 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 one 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 overfitting

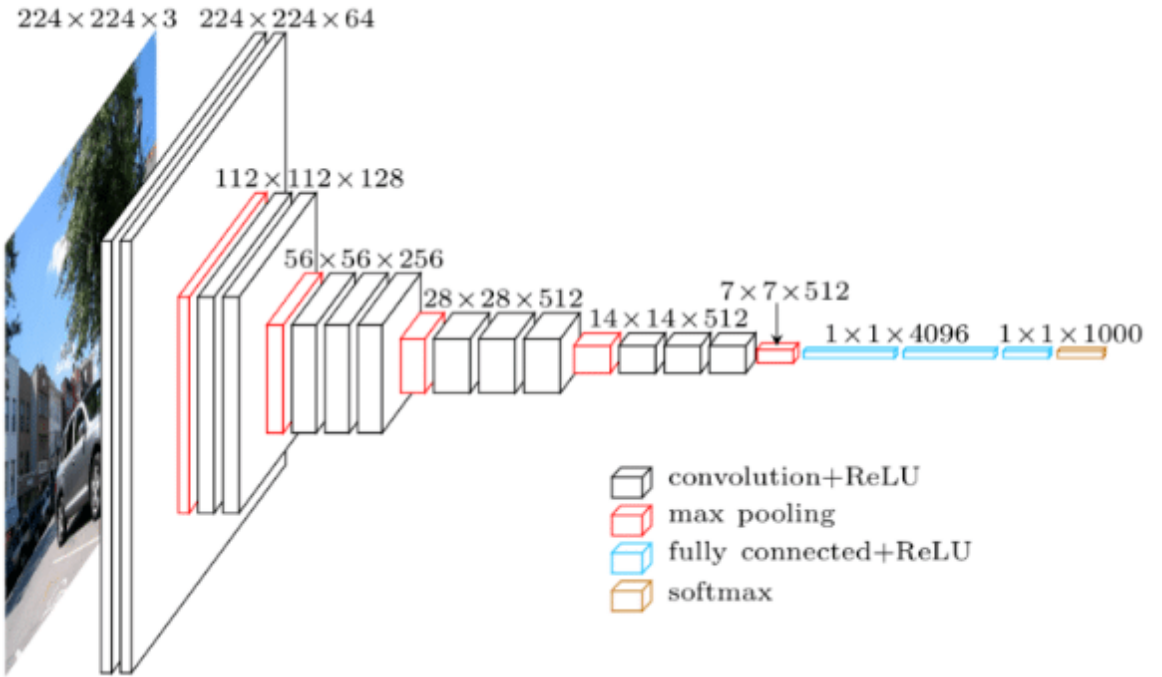
- Data augmentation:
  - Train on random 224x224 patches from the 256x256 images to get more data. Also use left-right reflections of the images
  - At test time, combine the opinions from ten different patches: The four 224x224 corner patches plus the central 224x224 patch plus the reflections of those five patches
- Dropout:
  - Dropout in the first two fully-connected layers

# Results



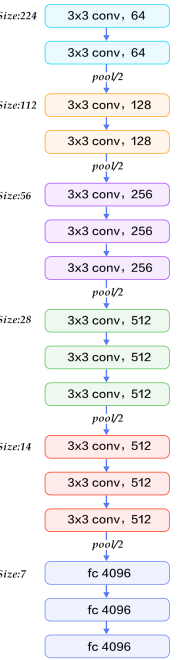

mite
container ship
motor scooter
leopard
mite
black widow
cockroach
tick
starfish
container ship
lifeboat
amphibian
fireboat
drilling platform
motor scooter
go-kart
moped
bumper car
golfcart
leopard
jaguar
cheetah
snow leopard
Egyptian cat
grille
mushroom
cherry
Madagascar cat
convertible
grille
pickup
beach wagon
fire engine
agaric
mushroom
jelly fungus
gill fungus
dead-man's-fingers
dalmatian
grape
elderberry
ffordshire bullterrier
currant
squirrel monkey
spider monkey
titi
indri
howler monkey

# Architecture of VGG16



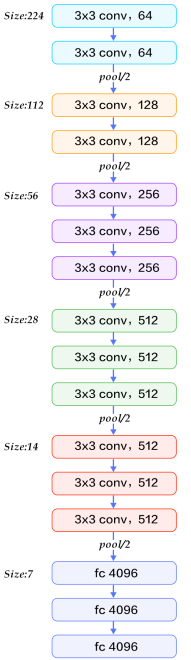
# VGGNet

- VGG short for Visual Geometry Group from University of Oxford
- 16 - 19 layers
- Only 3x3 kernel stride 1, zero-padding 1 and 2x2 max pooling stride 2



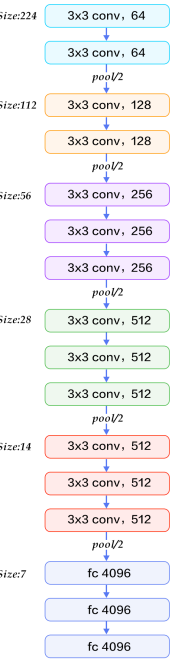
# Smaller kernels, deeper network

- 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 $3(3^2C^2) = 27C^2$



# Review

- ILSVRC'14 2nd in classification, 1st in localization
- Similar training procedure as AlexNet
- VGG19 only slightly better than VGG16, but requires more memory
- FC7 features generalize well to other tasks

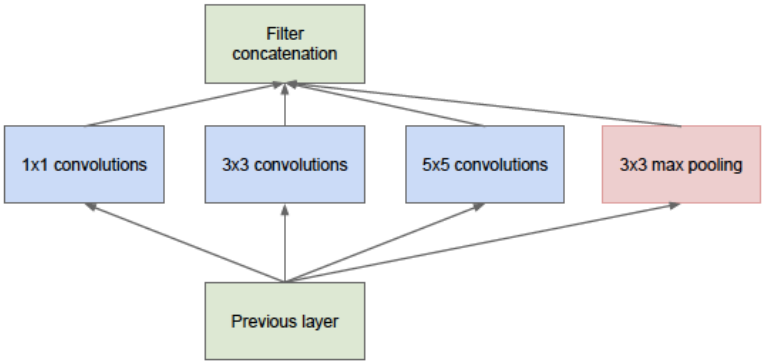


# Motivation

- 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:
  - means a larger number of parameters $\rightarrow$ prove to overfitting
  - the dramatically increased use of computational resources
- Increase the depth and width of the network while keeping the computational budget constant

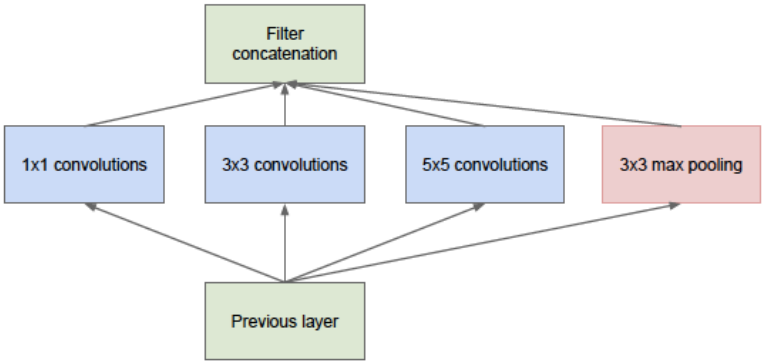
# Inception module - naive

- Apply parallel operations on the input from from previous layer:
  - Multiple kernel size for convolution (1x1, 3x3, 5x5)
  - Pooling operation (3x3)
- Concatenate all filter outputs together depth-wise
- What is the problem with this structure?



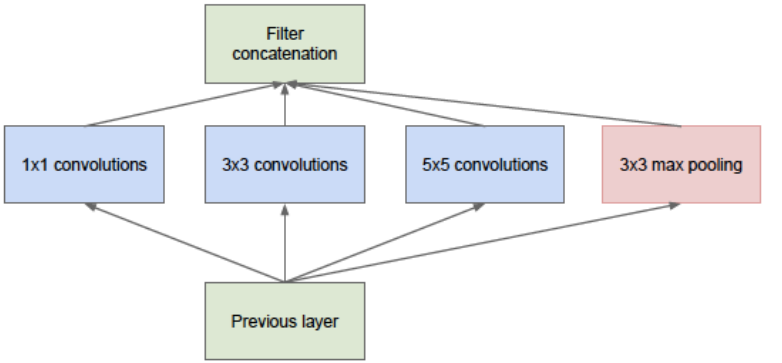
# Inception module - naive

- Let assume the setup of inception module is:
  - Input size: 28x28x256
  - 1x1 convolutional kernels: 128 with stride 1
  - 3x3 convolutional kernels: 192 with stride 1, zero-padding 1
  - 5x5 convolutional kernels: 96 with stride 1, zero-padding 2
  - 3x3 max pooling: stride 1, zero-padding 1
- The output is 28x28x(128+192+96+256) = 28x28x672



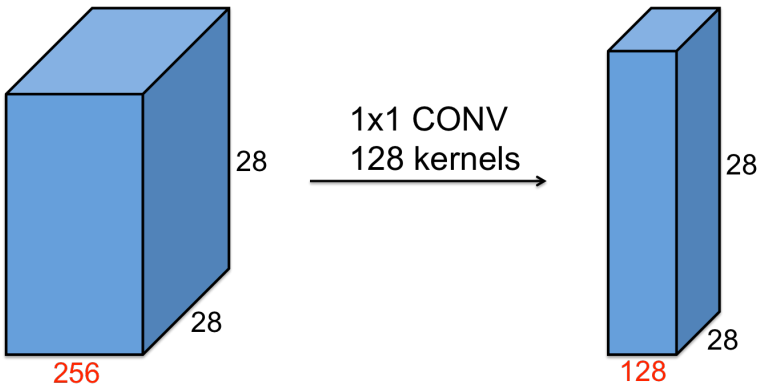
# Inception module - naive

- - **Conv Ops**:
  - 1x1 conv, 128: 28x28x128x1x1x256
  - 3x3 conv, 192: 28x28x192x3x3x256
  - 5x5 conv, 96: 28x28x96x5x5x256
  - **Total: 854M ops**
- Very expensive to compute
- Pooling layer also preserves feature depth, which means total depth always grow dramatically after concatenation
- Solution: “bottleneck” layers that use 1x1 convolutions to reduce feature depth

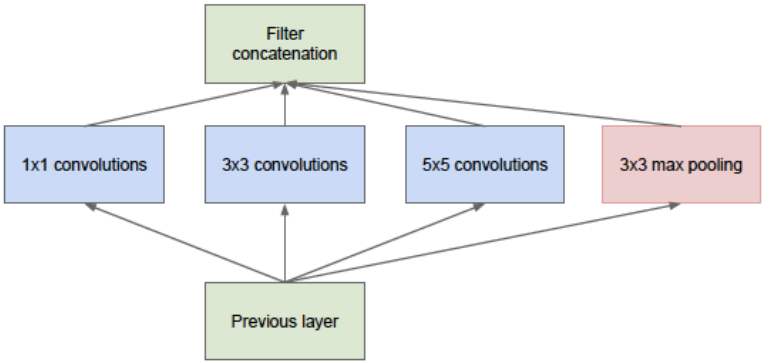


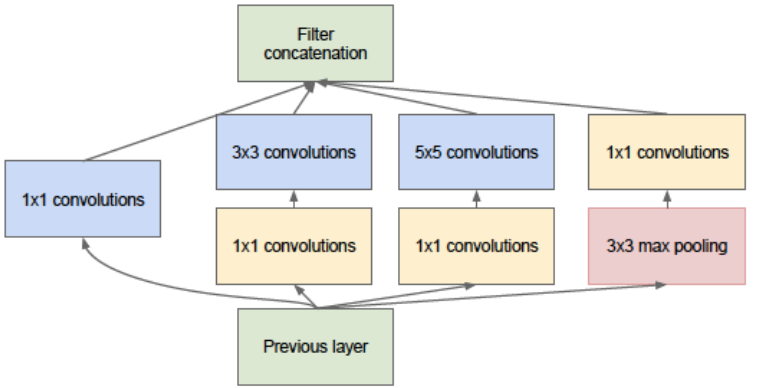
# 1x1 convolutions

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



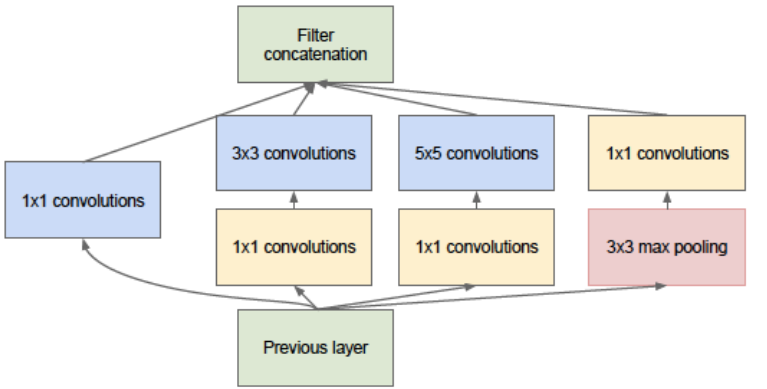
# Inception module with dimensionality reduction



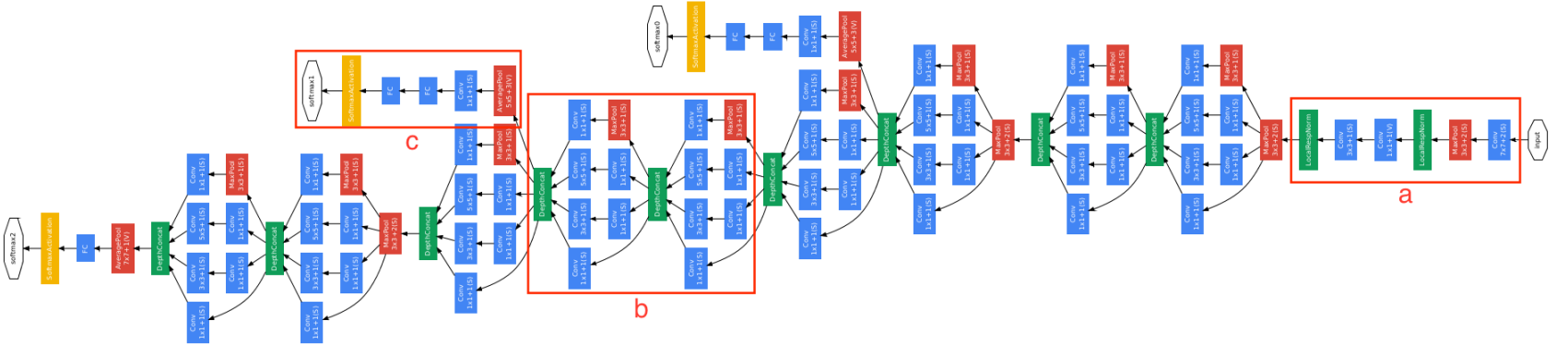



# Inception module with dimensionality reduction

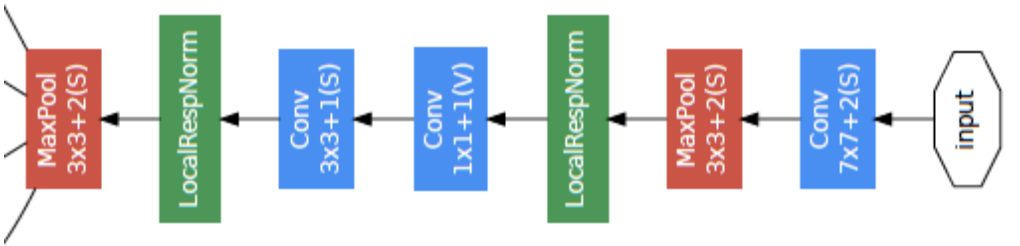
- If we adding three layers with“1x1 conv, 64 kernels”, then
  - **Conv Ops**:
  - 1x1 conv, 128: 28x28x128x1x1x256
  - 1x1 conv, 64: 28x28x64x1x1x256
  - 3x3 conv, 192: 28x28x192x3x3x64
  - 1x1 conv, 64: 28x28x64x1x1x256
  - 5x5 conv, 96: 28x28x96x5x5x64
  - 1x1 conv, 64: 28x28x64x1x1x256
  - **Total: 358M ops**



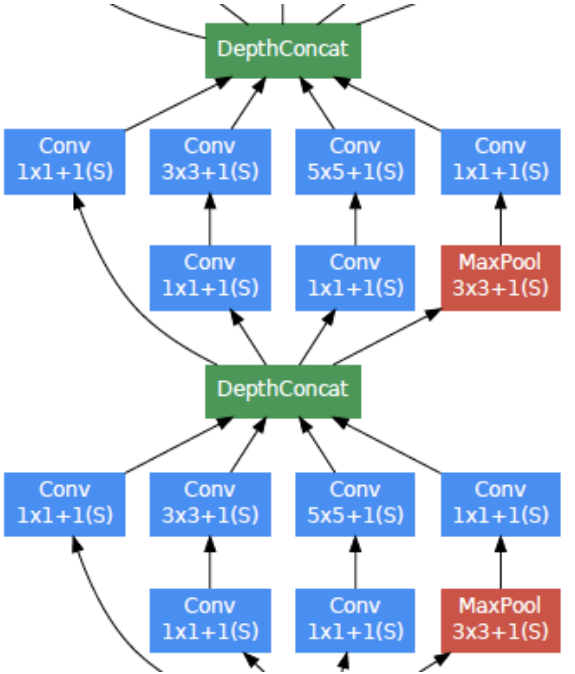
# Architecture of GoogLeNet



# Part a: stem network

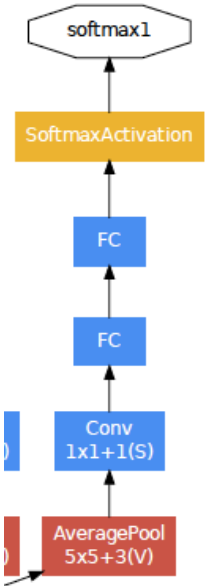


# Part b: stacked inception modules



# Part c: auxiliary classifiers

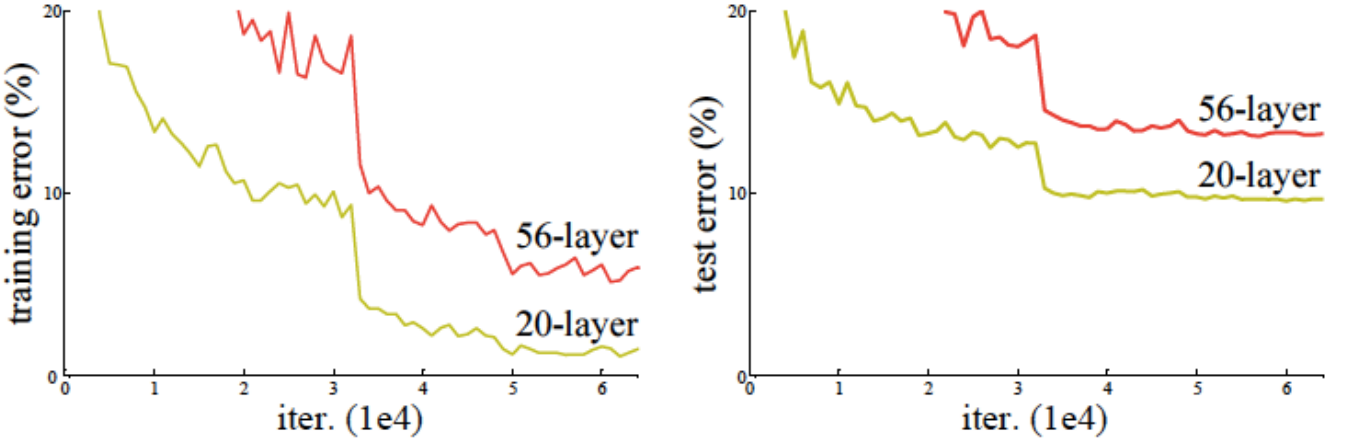
- Features produced by the layers in the middle of the network should be very discriminative
- Auxiliary classifiers connected to these intermediate layers, discrimination in the lower stages in the classifier was expected
- During training, their loss gets added to the total loss of the network with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



# 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)

# Motivation

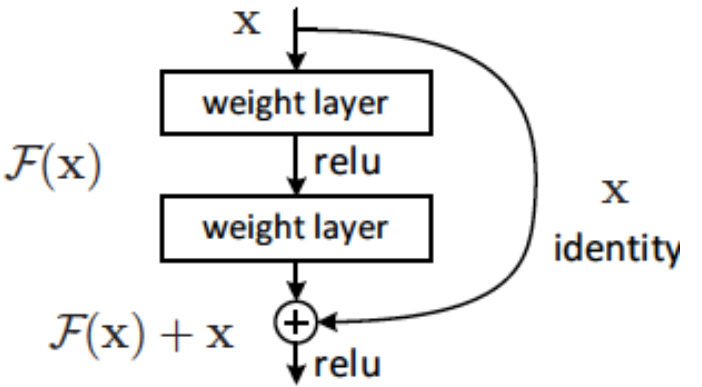



- Stacking more layers does not mean better performance
- With the network depth increasing, accuracy gets saturated and then degrades rapidly
- Such degradation is not caused by overfitting optimize

# 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
- Solution: Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping
-

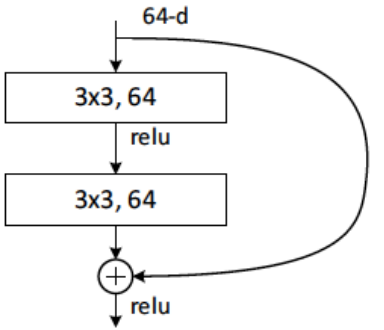
$$y = F(x, W_i) + x$$

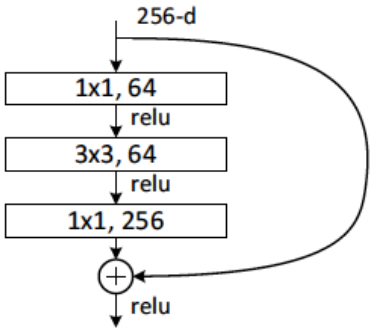



$$F(x) = W_2\sigma(W_1x)$$

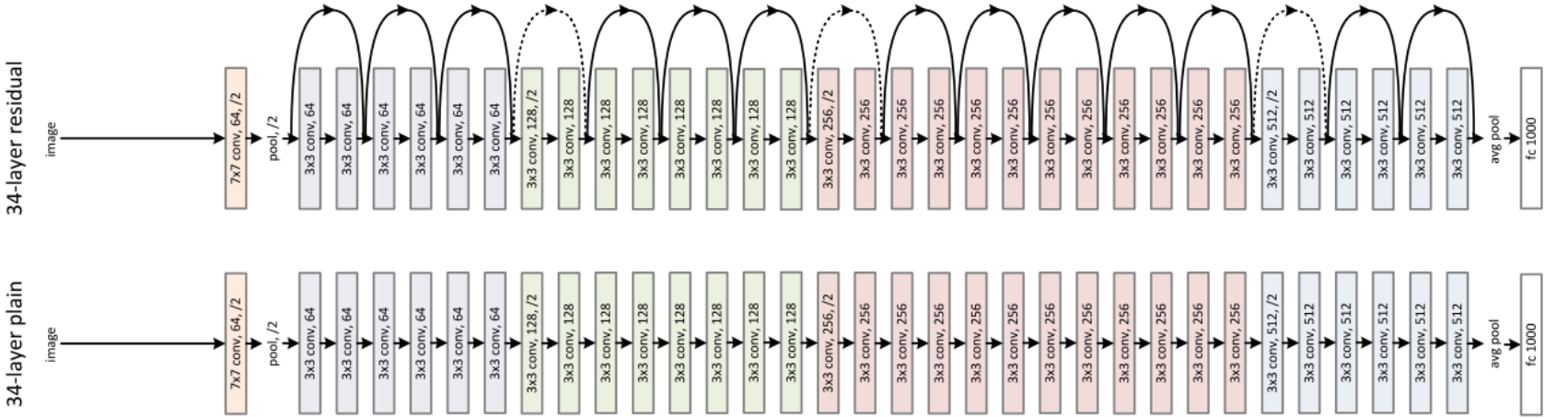
# Residual block (cont.)

- Similar to GoogLeNet, use “bottleneck” layer to improve efficiency for deeper networks

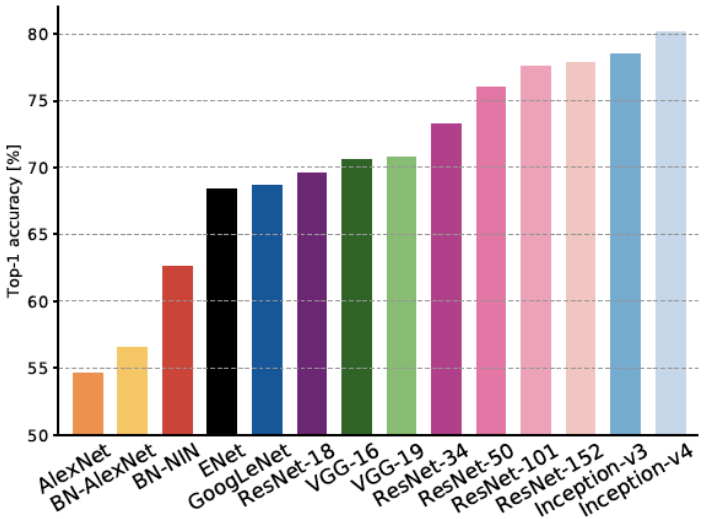


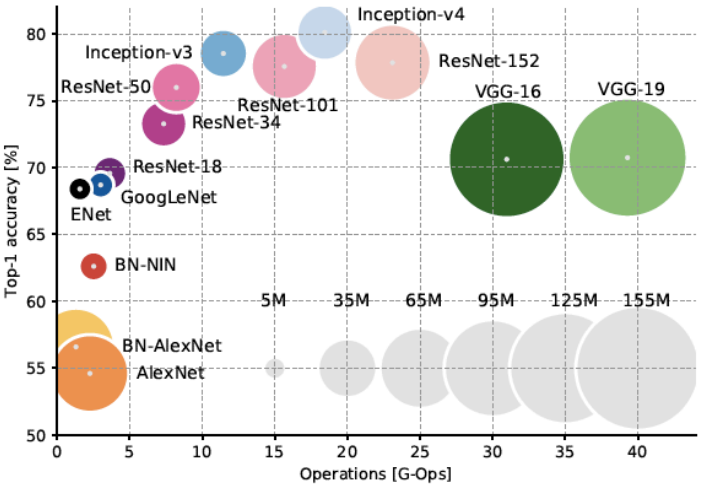



# Architecture of ResNet



# Performance comparison



# Summary

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