UNIVERSITY OF LIVERPOOL

# COMP 532
# Machine Learning and BioInspired Optimization

## Lecture 13: Deep Learning

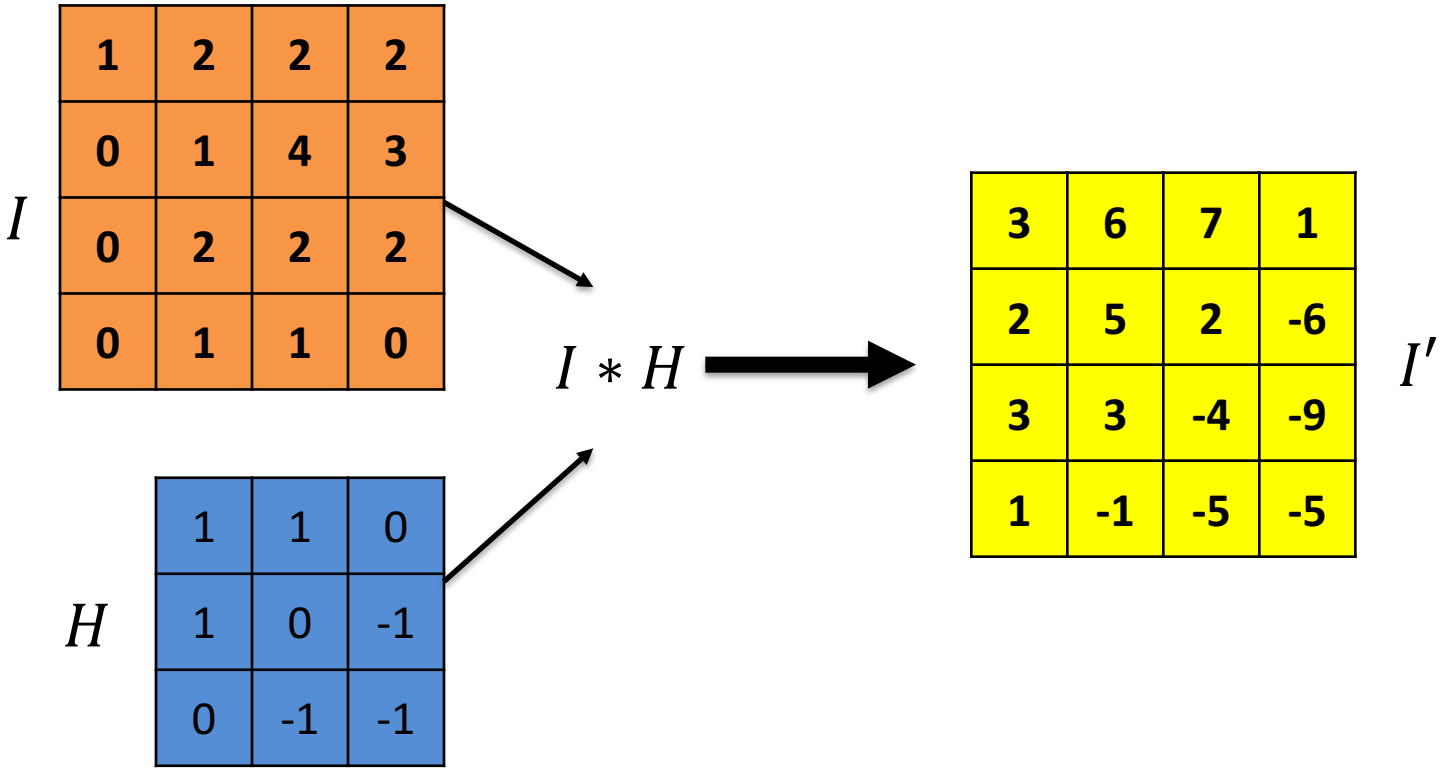
Dr. Shan Luo

Department of Computer Science

shan.luo@liverpool.ac.uk

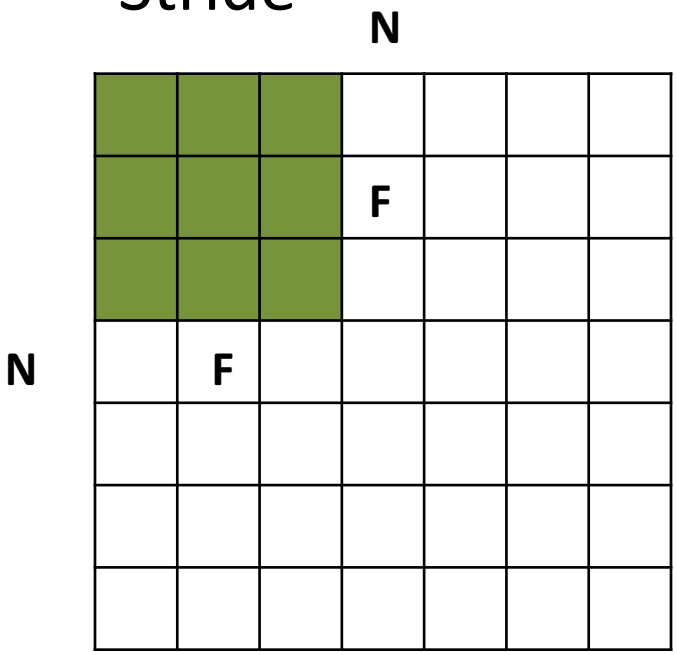
# Recap

- Convolution



# Recap

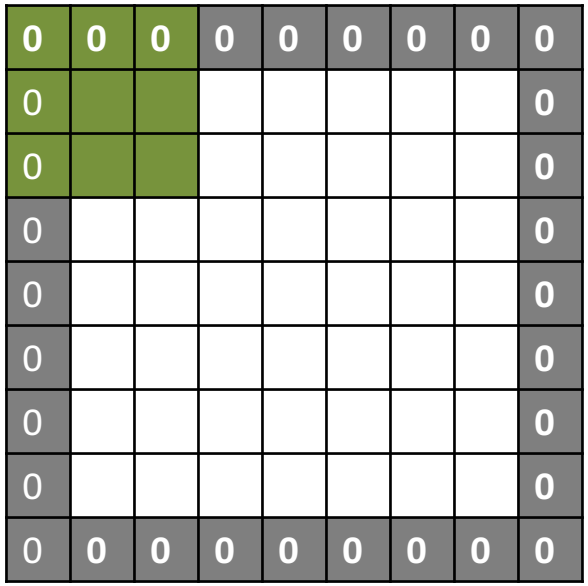
- Convolution
  - Stride




Output size:
(N - F) / stride +1

# Recap

- Convolution
  - Padding




In practice, it is common to zero pad the border

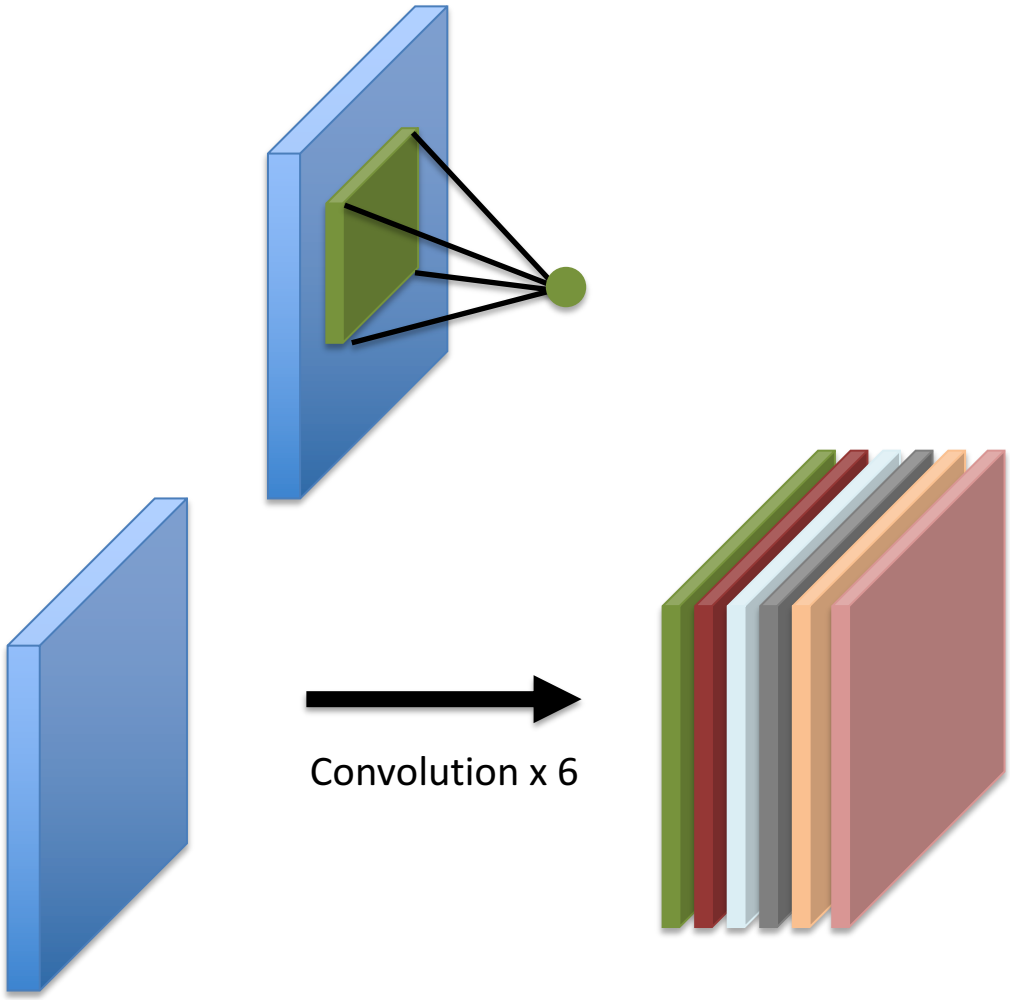
e.g., input 7x7
3x3 filter, applied with stride 1
**pad with 1 pixel border** => what is the output?

**7x7 output!**

In general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g., F=3 => zero pad with 1
F=5 => zero pad with 2

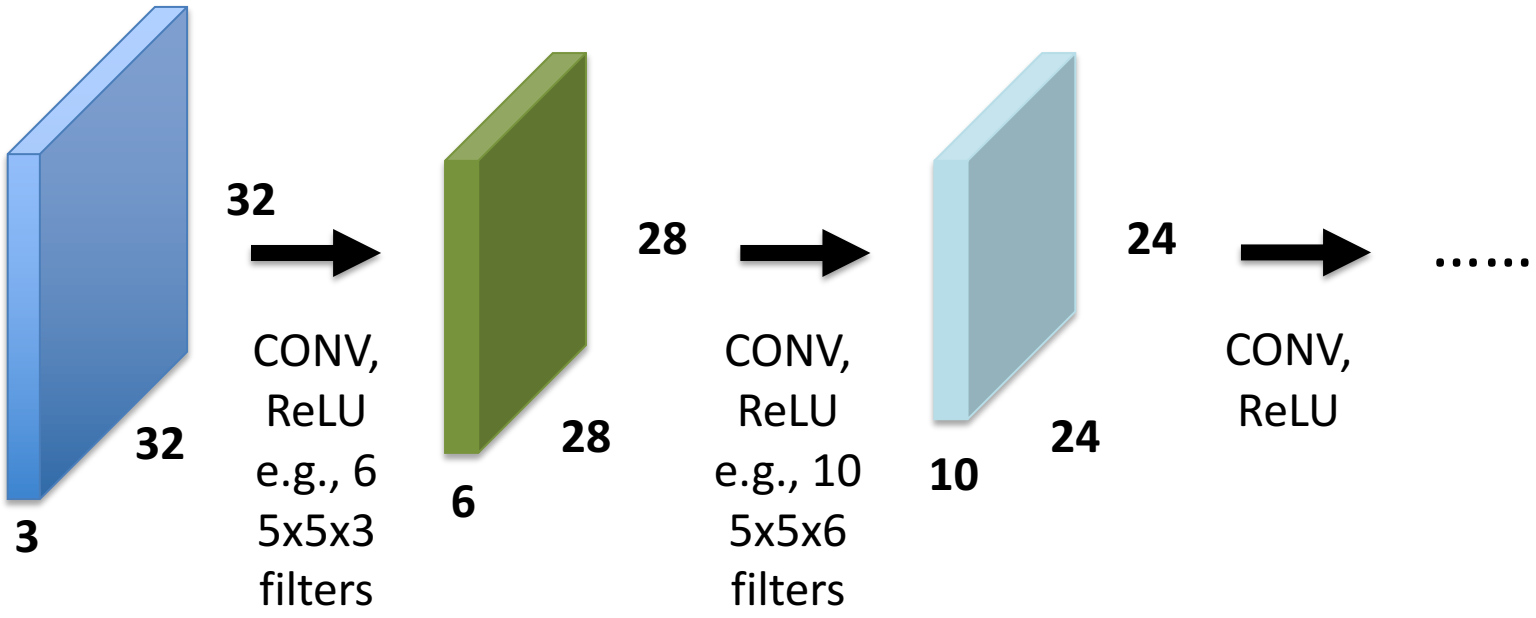
# Recap

- Convolutional layer



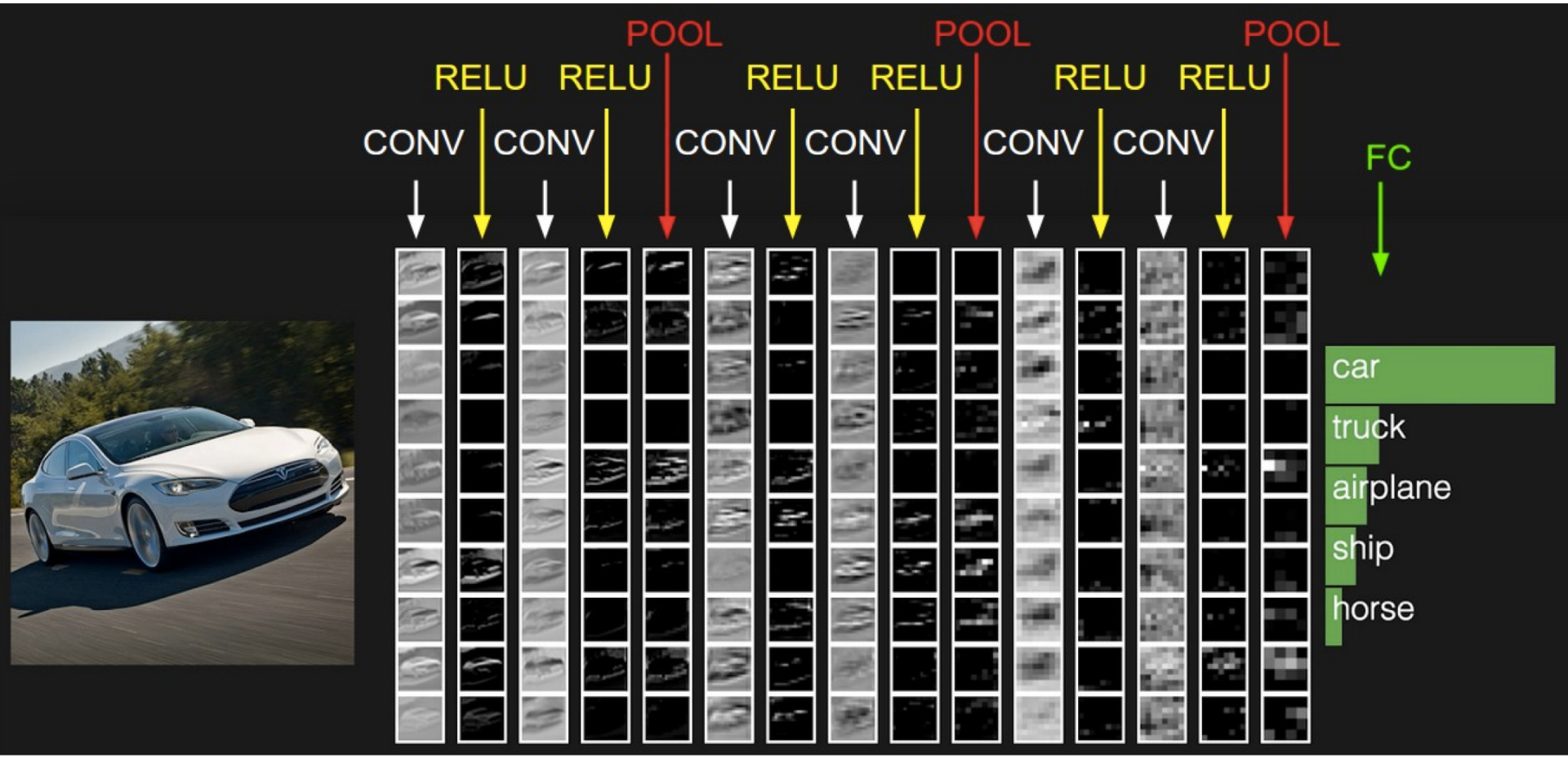
# Recap

- Convolutional layer



# Recap

- ConvNets

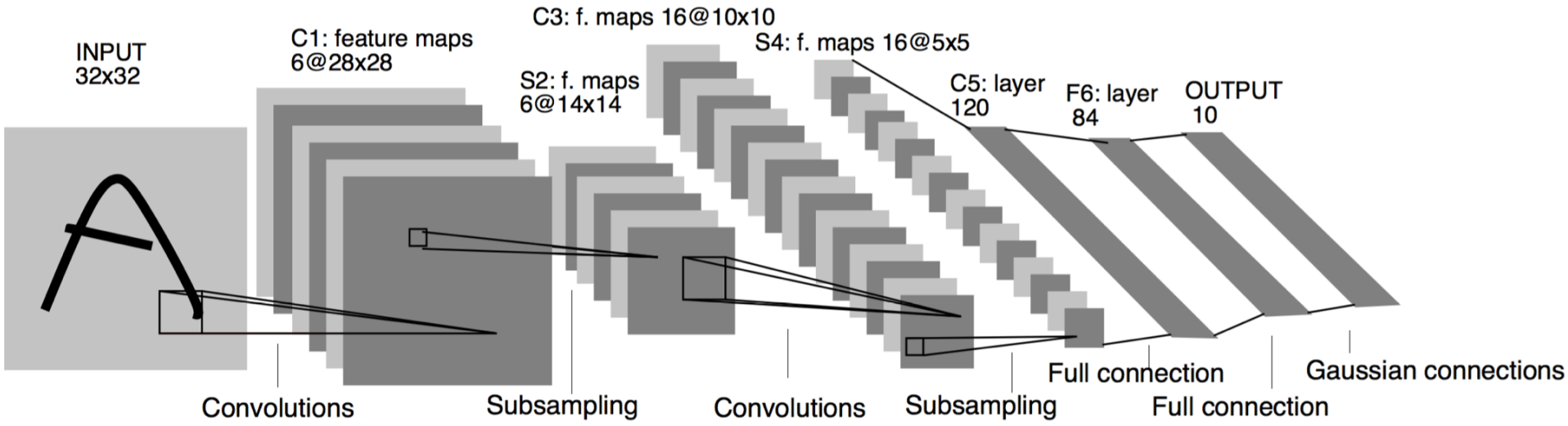


# Overview

- Case studies of ConvNets
  - LeNet 5
  - AlexNet
  - VGG-16
  - GooLeNet
  - ResNet

- Deep Learning Software
  - Model zoos
  - Transfer learning

# Case study: LeNet-5




Conv filters were 5x5, applied at stride 1.
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e., architecture is [CONV-POOL-CONV-POOL-CONV-FC]

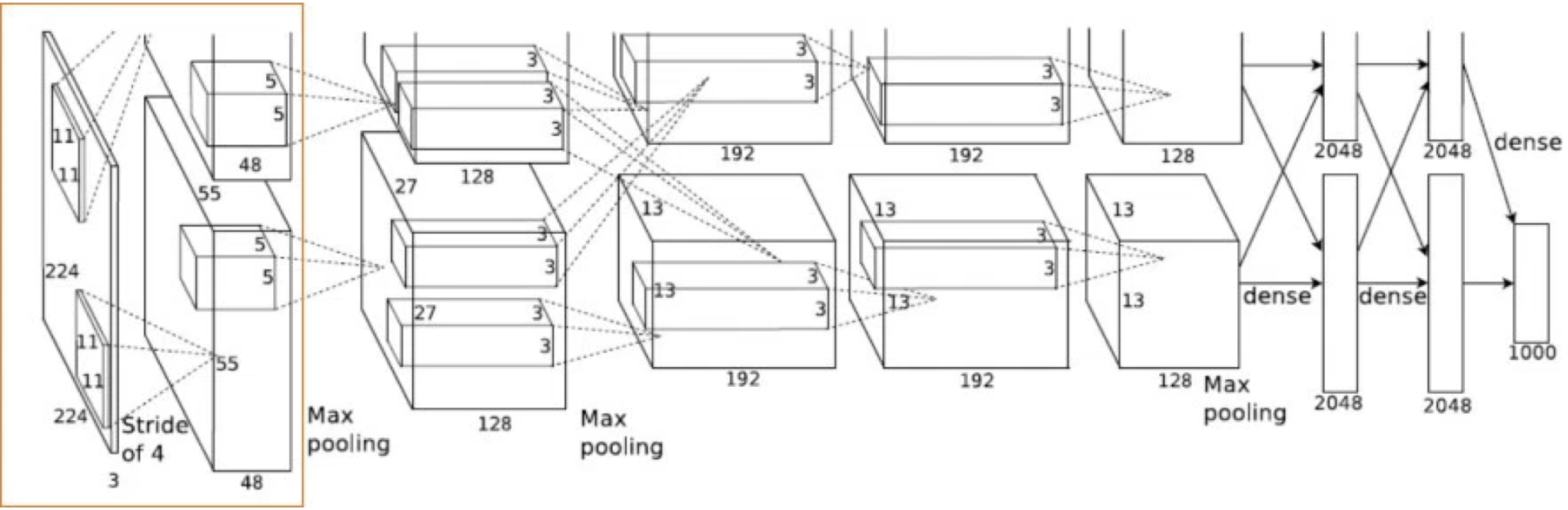
LeCun et al., 1998

# ImageNet Challenge

IMAGENET

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.

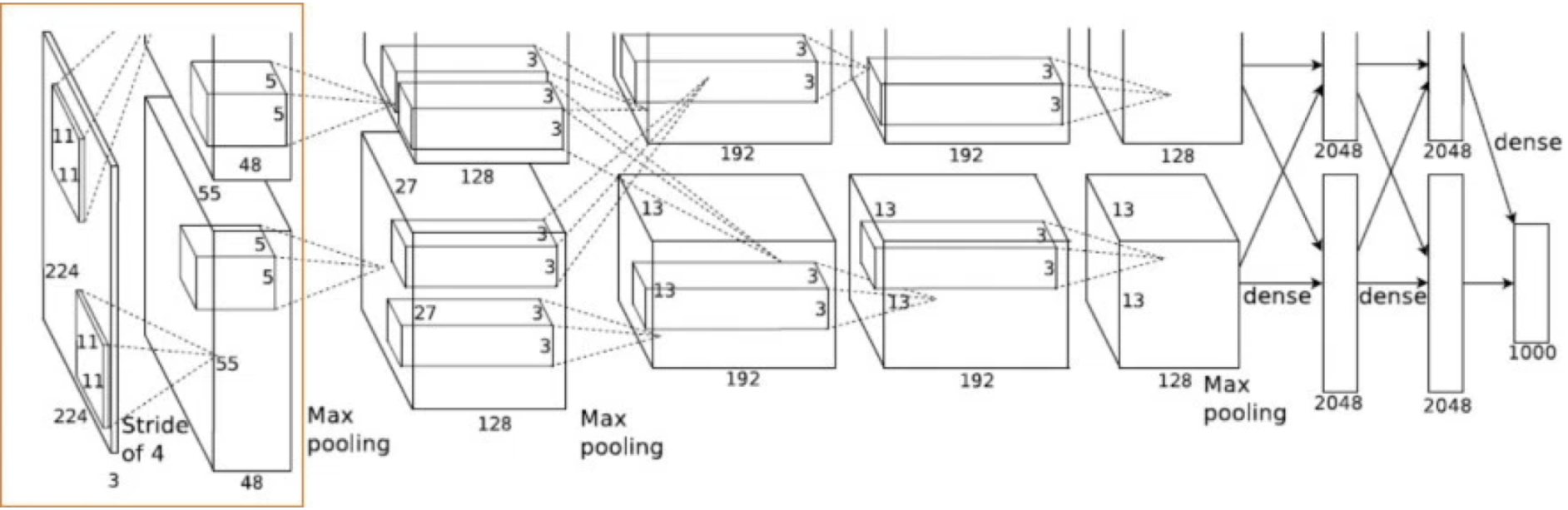
# Case study: AlexNet




Architecture:
[CONV1-MaxPOOL1-CONV2-MaxPOOL2-CONV3-CONV4-CONV5-MaxPOOL3-FC6-FC7-FC8]

Krizhevsky et al., 2012

# Case study: AlexNet

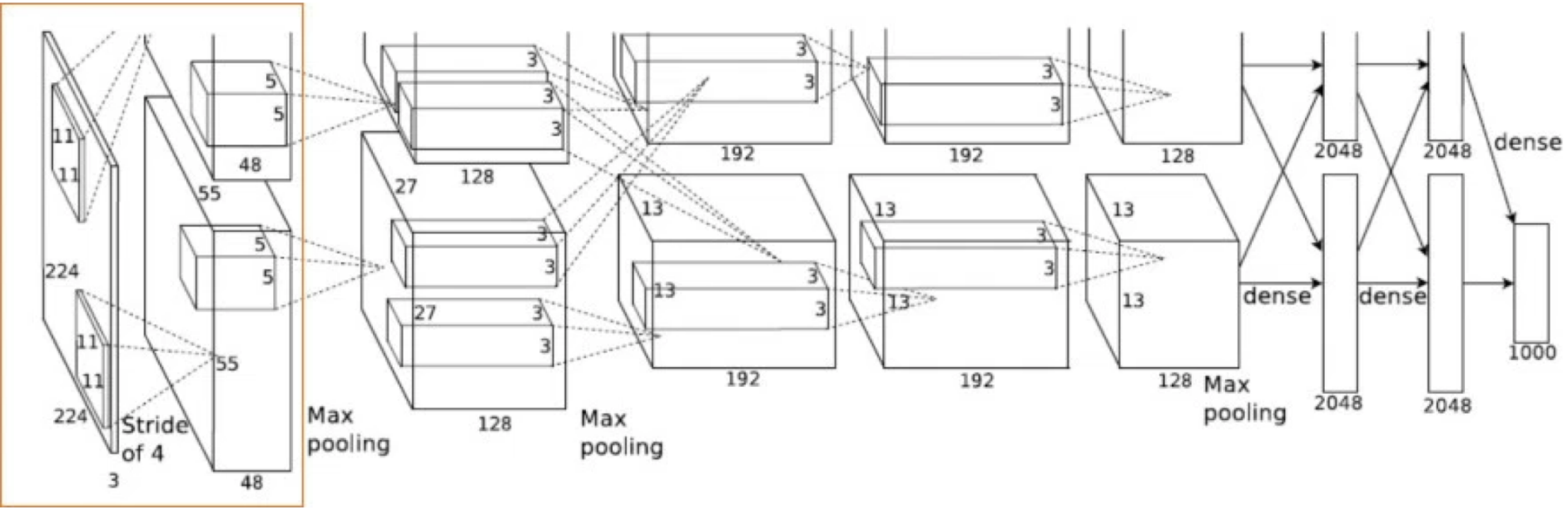



Architecture:
[CONV1-MaxPOOL1-CONV2-MaxPOOL2-CONV3-CONV4-CONV5-MaxPOOL3-FC6-FC7-FC8]

Input: 227x227x3 images
First layer (CONV1) 96 11x11 filter applied at stride 4
Output volume [55x55x96]

Krizhevsky et al., 2012

# Case study: AlexNet

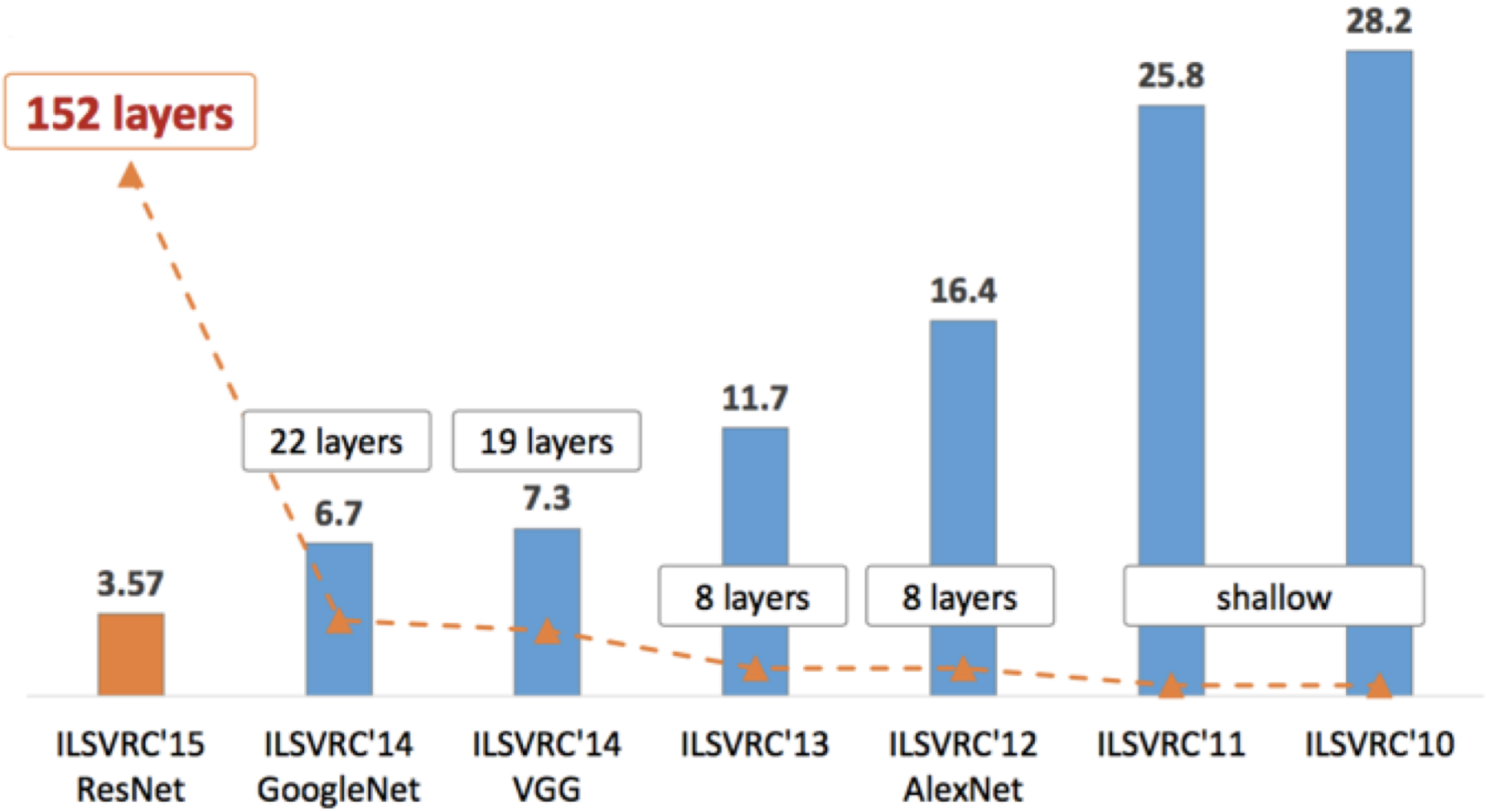



Architecture:
[CONV1-MaxPOOL1-CONV2-MaxPOOL2-CONV3-CONV4-CONV5-MaxPOOL3-FC6-FC7-FC8]

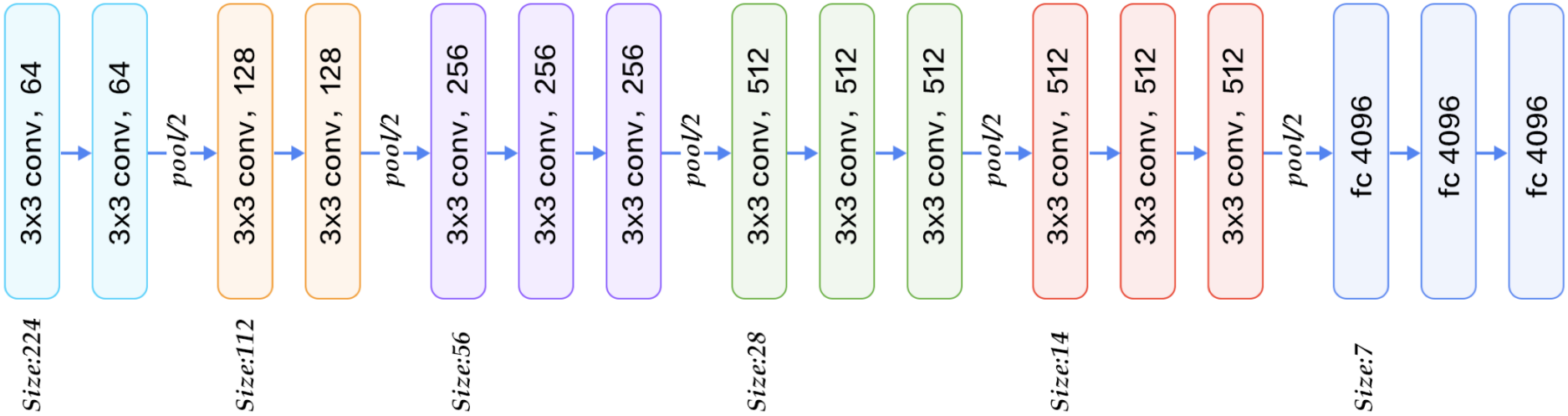
Input: 227x227x3 images After CONV1: 55x55x96
Second layer (POOL1): 3x3 filters applied at stride 2
Output volume (55-3)/2+1=27 [27x27x96]

Krizhevsky et al., 2012

# ImageNet Large Scale Visual Recognition Challenge (ILSRC) winners



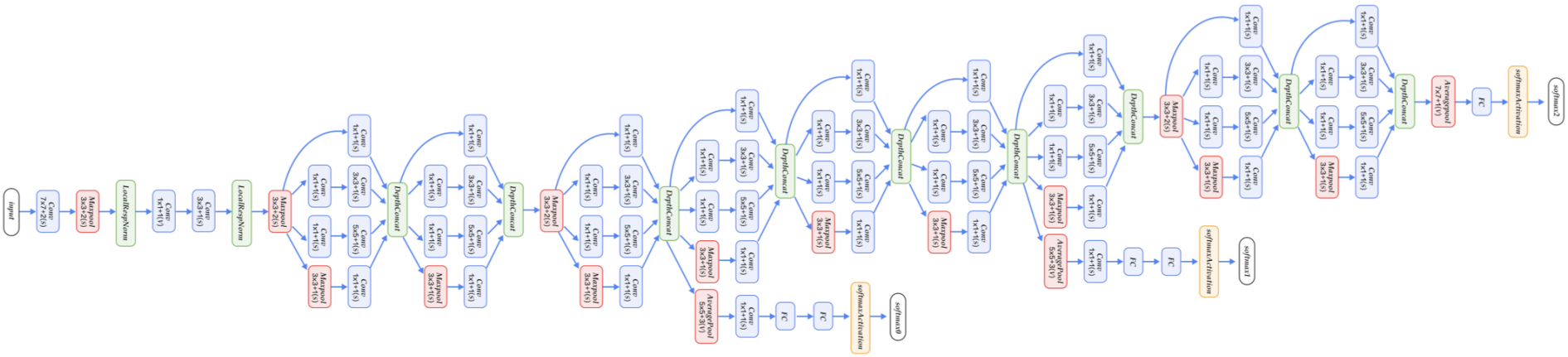
# Case study: VGG-16




In total, there are 16 convolutional/fc layers
2-3 convolutional layers between the pooling layers
Smaller filters, Deeper networks

Simonyan and Zsserman., 2014

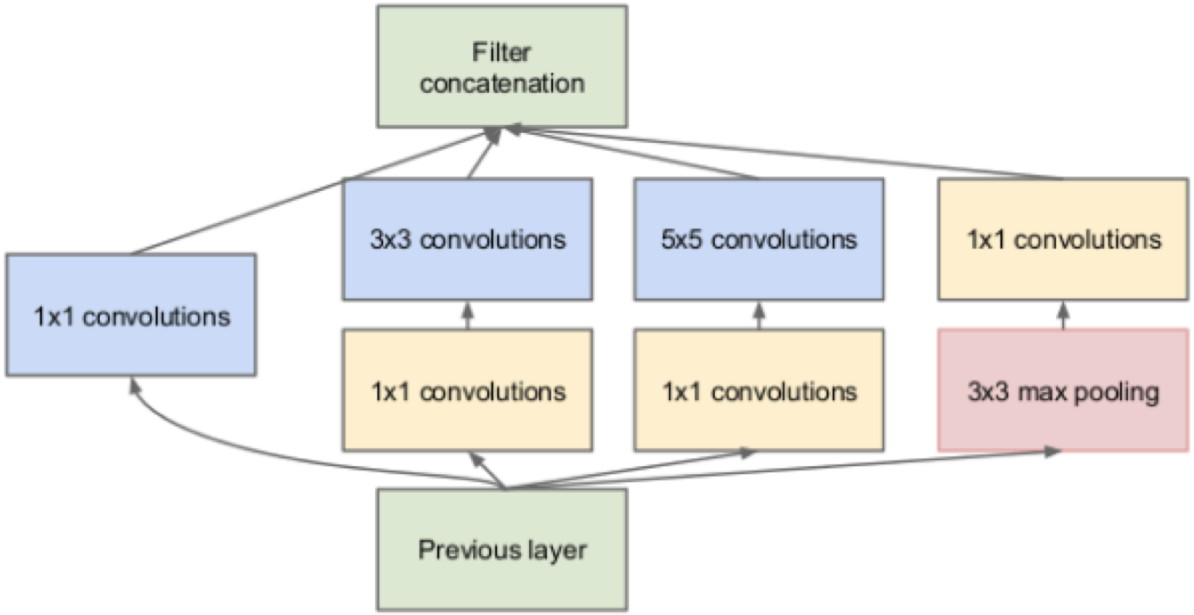
# Case study: GoogLeNet




Deeper networks, with computational efficiency
22 layers, efficient “Inception” module

Szegedy et al., 2014

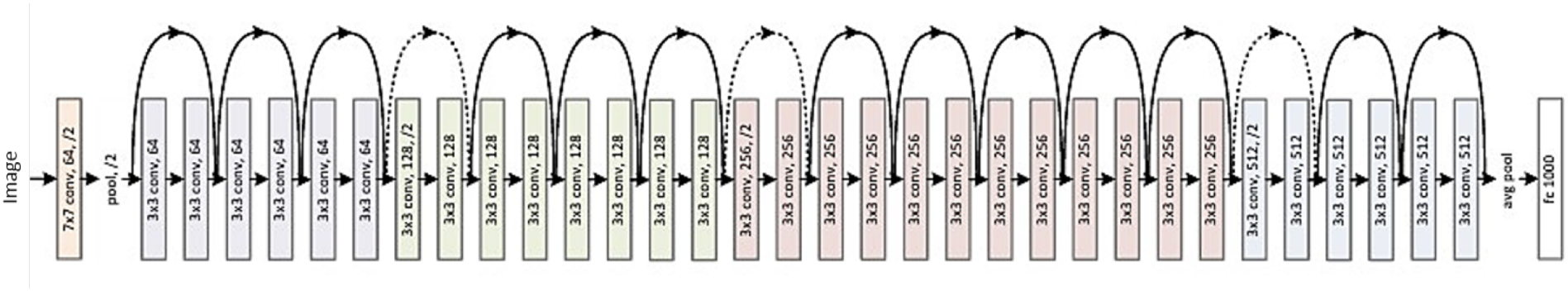
# Case study: GoogLeNet




"Inception module": design a good local network topology (network within a network) and then stack these modules on top of each other

Szegedy et al., 2014

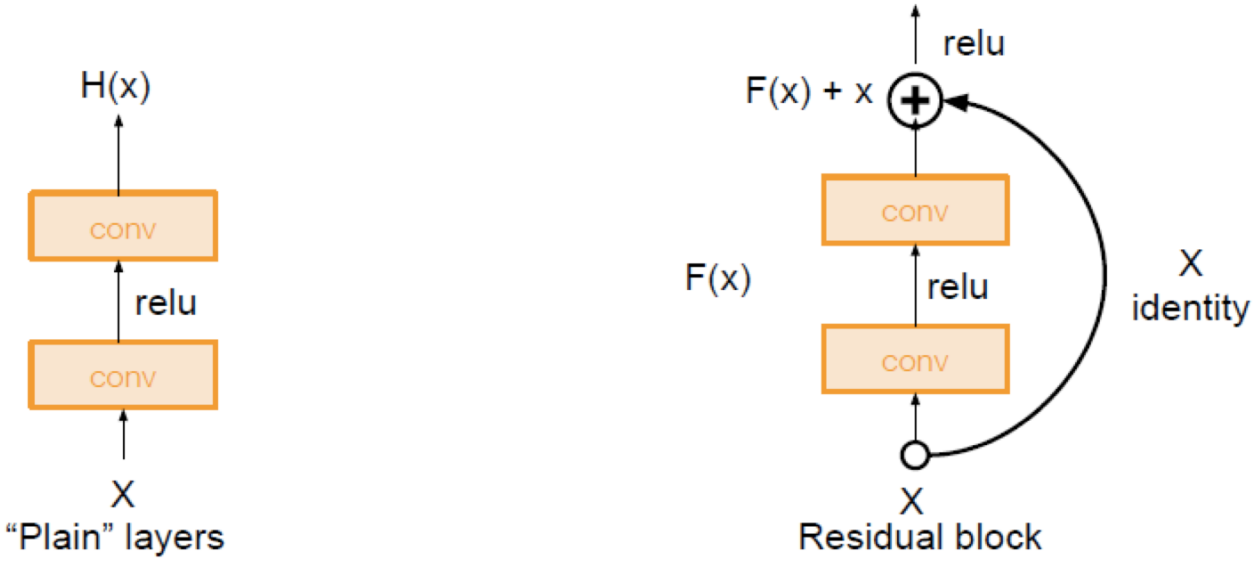
# Case study: ResNet




Very deep networks using residual connections
152 layers for ImageNet
Residual connections

He et al., 2015

# Case study: ResNet

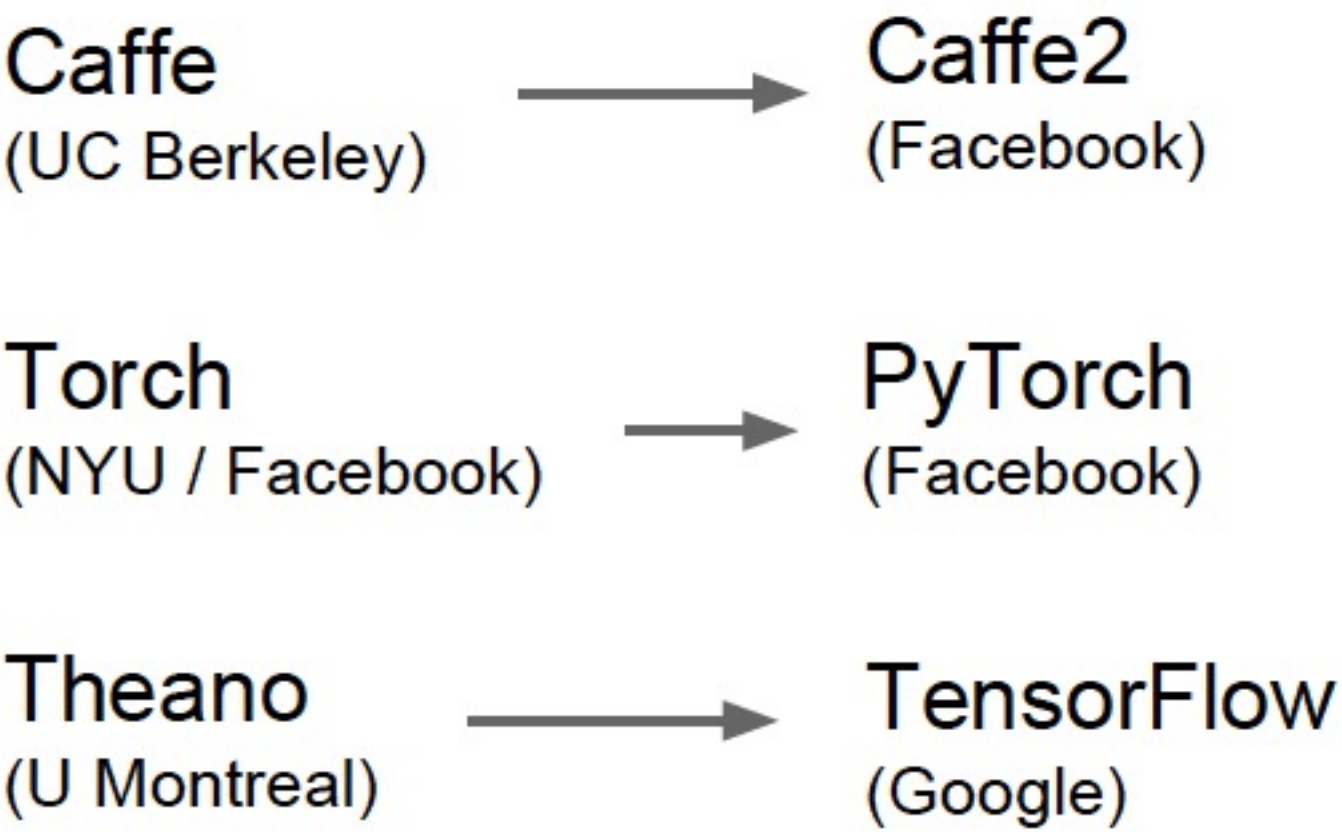



Use network layers to fit a residual mapping instead of directly trying to fit the desired underlying mapping

He et al., 2015

- Deep Learning Software
  - Caffe / Caffe2
  - Theano/Tensorflow - Keras
  - Torch/PyTorch

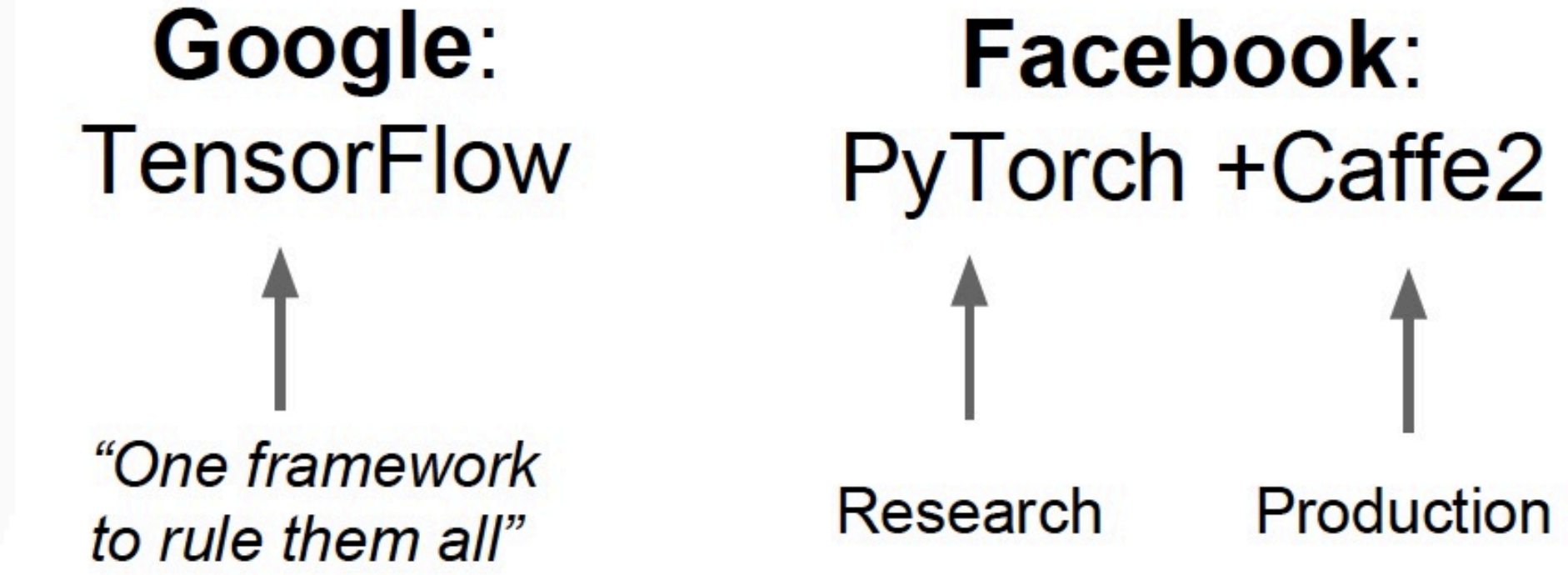
# Deep Learning software



# Deep Learning software

| | Languages | Tutorials and training materials | CNN modeling capability | RNN modeling capability | Architecture: easy-to-use and modular front end | Speed | Multiple GPU support | Keras compatible |
|---|---|---|---|---|---|---|---|---|
| Theano | Python, C++ | ++ | ++ | ++ | + | ++ | + | + |
| Tensor-Flow | Python | +++ | +++ | ++ | +++ | ++ | ++ | + |
| Torch | Lua, Python (new) | + | +++ | ++ | ++ | +++ | ++ | |
| Caffe | C++ | + | ++ | | + | + | + | |
| MXNet | R, Python, Julia, Scala | ++ | ++ | + | ++ | ++ | +++ | |
| Neon | Python | + | ++ | + | + | ++ | + | |
| CNTK | C++ | + | + | +++ | + | ++ | + | |

# Deep Learning software



# Deep Learning software

Keras is a layer on top of TensorFlow/Theano, makes common things easy to do.

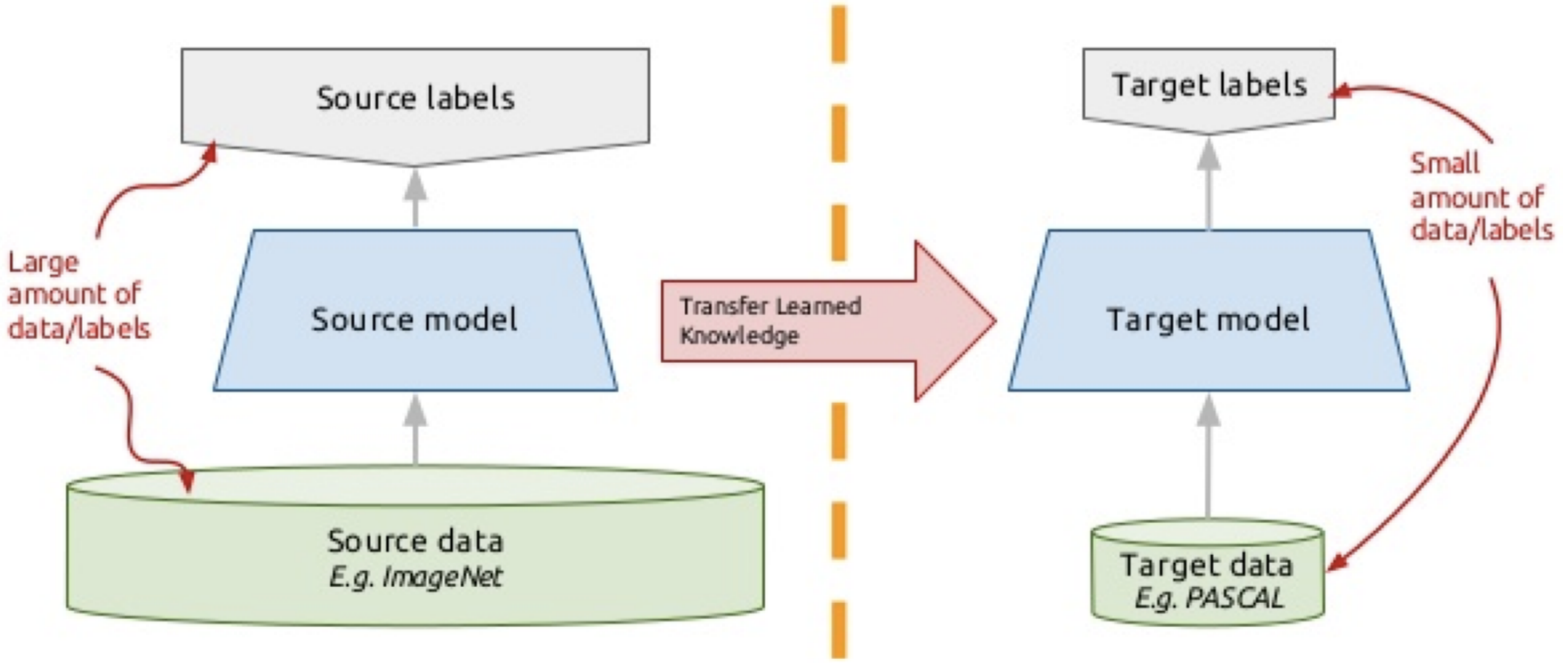
```
from keras.models import Sequential
from keras.layers.core import Dense, Activation
from keras.optimizers import SGD

N, D, H = 64, 1000, 100

model = Sequential()
model.add(Dense(input_dim=D, output_dim=H))
model.add(Activation('relu'))
model.add(Dense(input_dim=H, output_dim=D))

optimizer = SGD(lr=1e0)
model.compile(loss='mean_squared_error',
              optimizer=optimizer)

x = np.random.randn(N, D)
y = np.random.randn(N, D)
history = model.fit(x, y, nb_epoch=50,
                    batch_size=N, verbose=0)
```

# Deep Learning software

Model zoos:

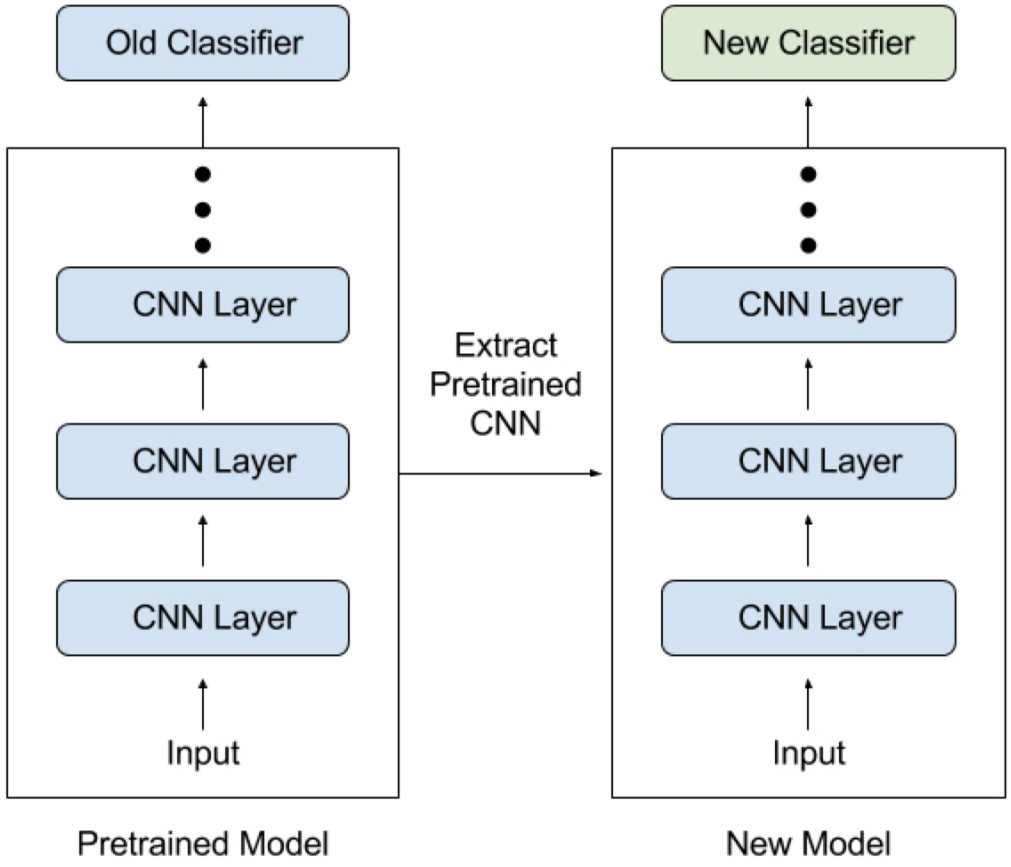
Caffe: https://github.com/BVLC/caffe/wiki/Model-Zoo
Keras: https://github.com/albertomontesg/keras-model-zoo
Torch: https://github.com/torch/torch7/wiki/ModelZoo
Theano: https://github.com/Theano/Theano/wiki/Related-projects
Tensorflow: https://github.com/tensorflow/models

# Transfer learning



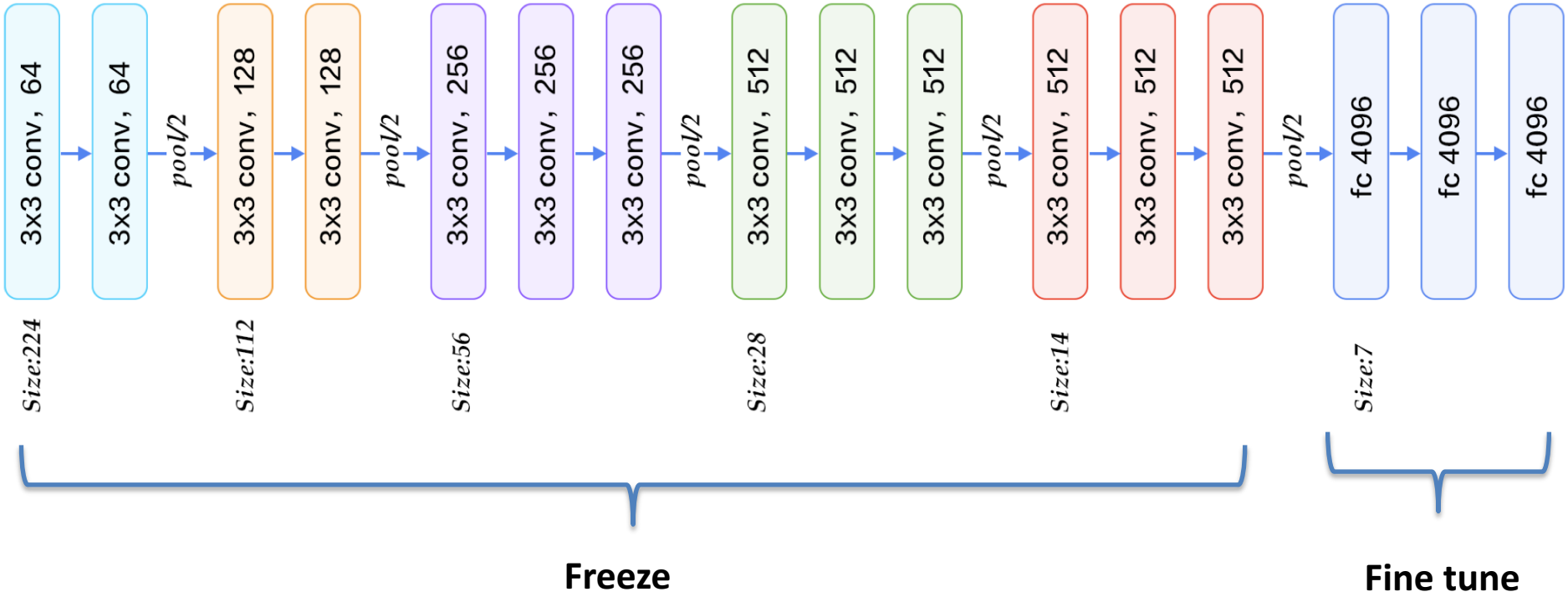
# Transfer learning

**Pretrained models:** https://github.com/fchollet/deep-learning-models



# Transfer learning

**Pretrained models:** https://github.com/fchollet/deep-learning-models



# Summary

- Case studies of ConvNets
  - LeNet 5
  - AlexNet
  - VGG-16
  - GooLeNet
  - ResNet

- Deep Learning Software
  - Model zoos
  - Transfer learning