Data Analytics
EEE 4774 & 6777

Module 4 - Classification

Deep Learning, Convolutional Neural Network

Spring 2022

Artificial Neural Network

- Origins in attempts to find mathematical representations of information processing in biological systems

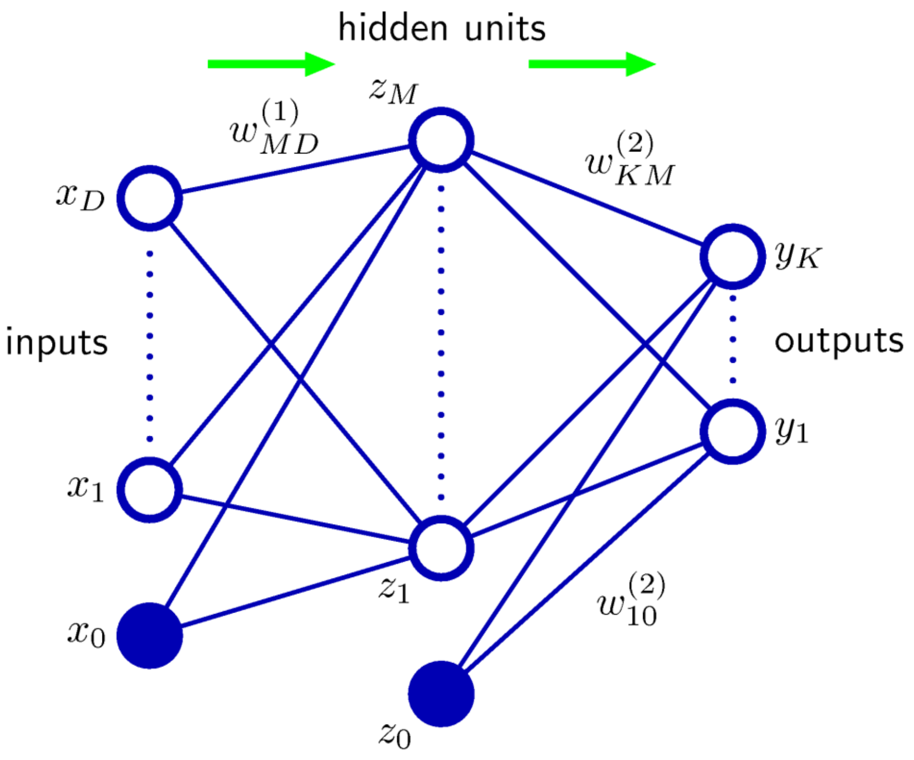
(MLP) : fully connected neural network

- Also known as ***Multilayer Perceptron*** / Can be also regarded as ***Multilayer Logistic Regression***
- Finds basis functions $\boldsymbol{\phi}(\boldsymbol{x})$ adaptive to the training data

$$y(\boldsymbol{x}) = f\big(c\boldsymbol{\phi}(\boldsymbol{x})\big) = \underbrace{\sigma\Big(\underbrace{\boldsymbol{w}_2^T\boldsymbol{\phi}(\boldsymbol{x})}_{\text{linear}}\Big)}_{\text{nonlinear}}$$

$$\boldsymbol{\phi}(\boldsymbol{x}) = h\big(\boldsymbol{w}_1^T\boldsymbol{x}\big)$$

raw data



Mostly black-box approaches

Extensions

Deep Neural Networks (Deep Learning), e.g.,

Supervised Learning

- Convolutional Neural Network (CNN)

image processing

- Recurrent Neural Network (RNN) Module 5

sequential data

- Transformer (self-attention)

- Deep Generative Models module 6

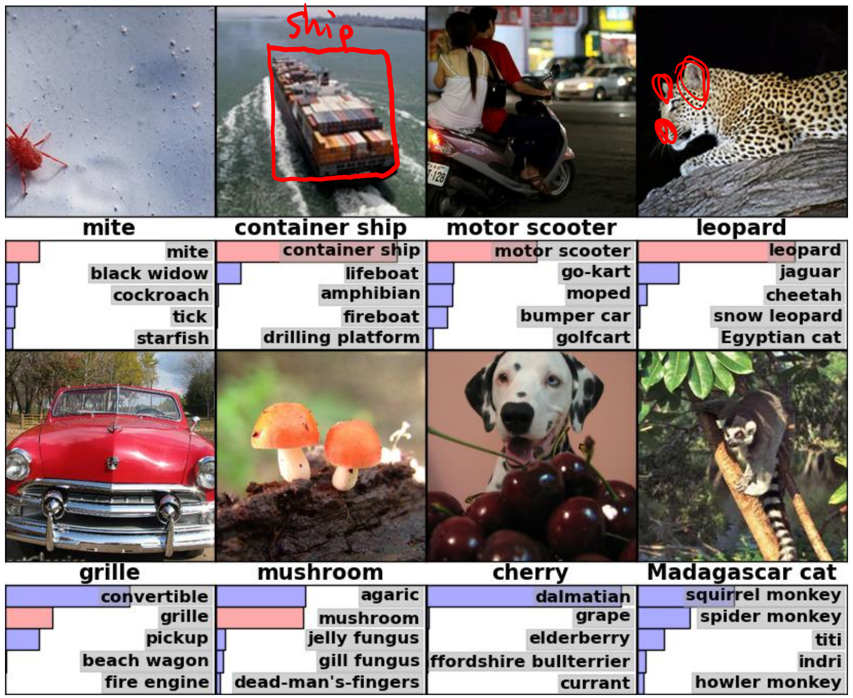
- Deep Reinforcement Learning

RL ANN

e.g., Video Game, Robotics, Chess / Go playing machines

Agent → Actions → Environment

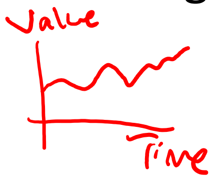
Data, Feedback



Applications

Image Classification

- Object Recognition
- Action Recognition
- Face Recognition
- Speech Recognition
- Natural Language Processing (NLP)
- Video Understanding
- Time-series Prediction



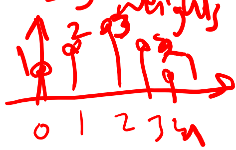
- Anomaly Detection
- Robotics
- Autonomous Driving
- …

Convolutional Neural Networks (CNN)

1D convolution

$x[n]$: input signal

$w[n]$: filter weights



- Uses **convolution** in place of general matrix multiplication in at least one of the layers
- Convolution is a specialized kind of linear operation:

$$y[n] = x[n] * w[n] = \sum_{m=-\infty}^{\infty} w[m]\, x[n-m]$$

convolution operation

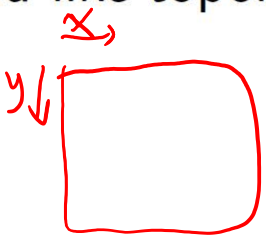
$$s(t) = x(t) * w(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)$$

e.g.,

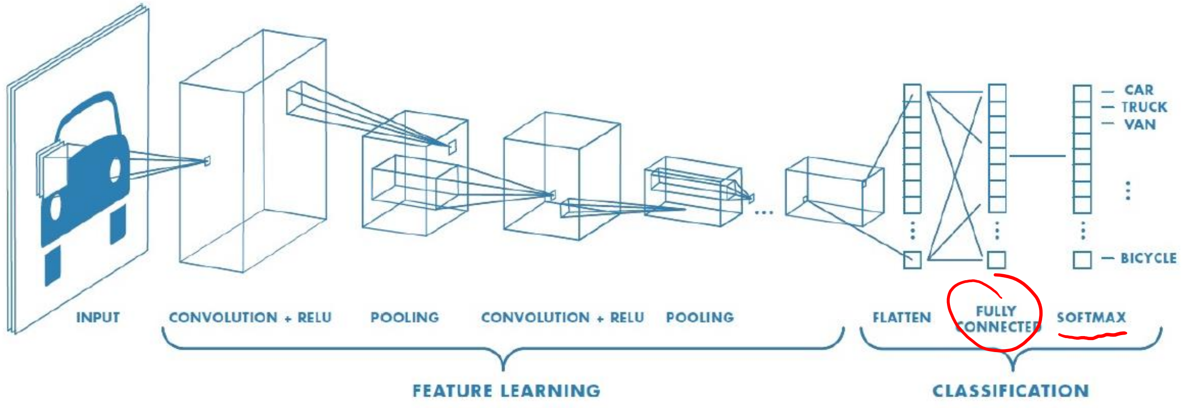
$$\hat{x}[n+1] = y[n] = w[0]\,x[n] + w[1]\,x[n-1] + w[2]\,x[n-2] + w[3]\,x[n-3] + w[4]\,x[n-4]$$

- Used for processing data that has a grid-like topology, e.g., time-series data (1-D grid), image data (2-D grid)
- Very successful in practice

image



$$w = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_4 \end{bmatrix}$$



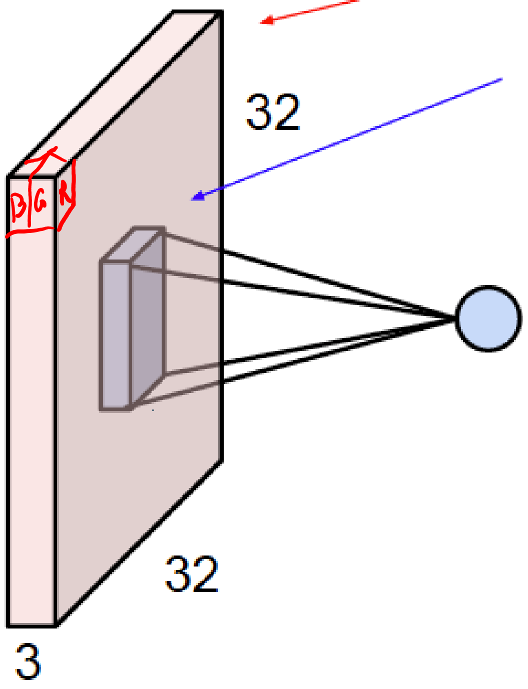
Convolution

RGB

32x32x3 image

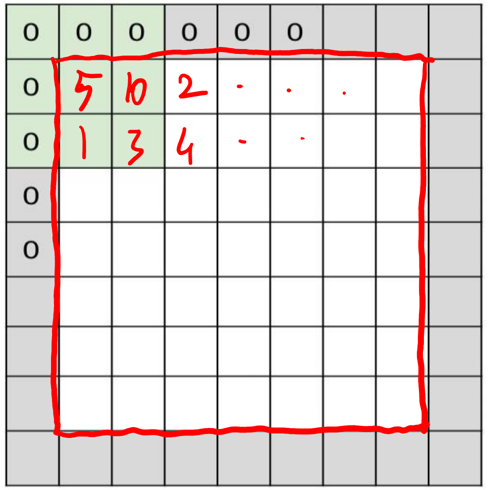
3x3x3 filter w

color channels



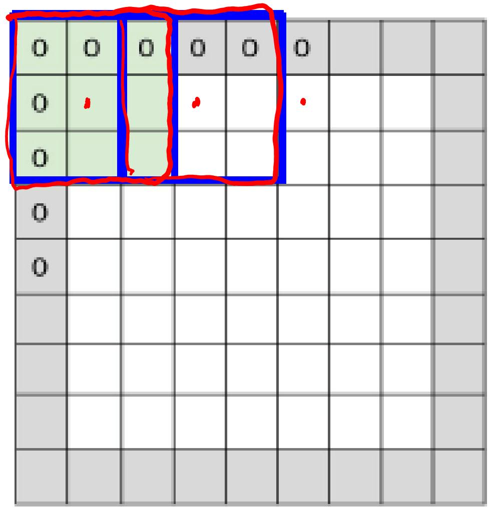
Ex:

$$\begin{bmatrix} 2 & 1 & -1 \\ 0 & 3 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$



Padding

16x16x3



Stride $= 2$

$$3 \times 5 + (-2)10 + (-1)3 = -8$$

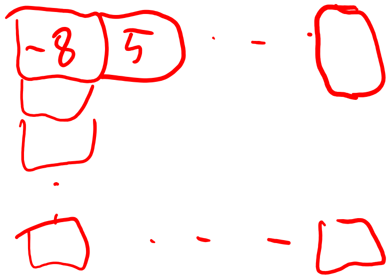
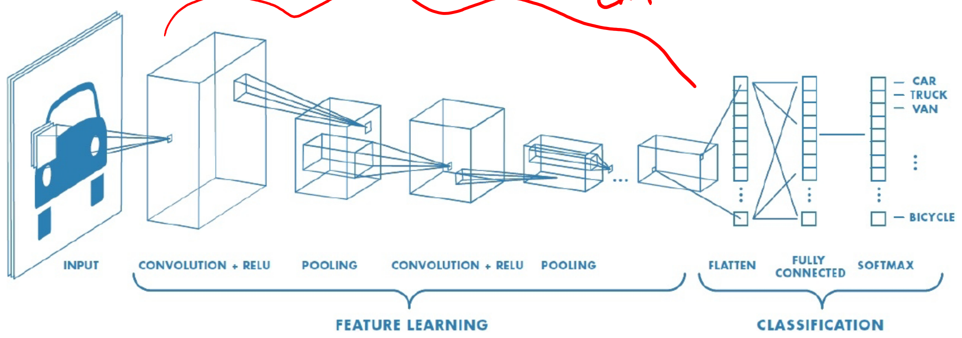


Image courtesy of Fei-Fei Li, Ranjay Krishna, Danfei Xu

Convolutional Neural Networks

- 3 stages:
  - convolutional stage: linear activations
  - detector stage: nonlinear activation function such as Rectified Linear Unit (ReLU)
  - pooling stage: modify the output with a summary statistic of nearby outputs, e.g.,
- **Max Pooling:** reports the maximum output within a rectangular neighborhood
- **Average Pooling:** average of a rectangular neighborhood
- Pooling helps to make the representation approximately invariant (i.e., robust) to small translations of the input

feature extraction



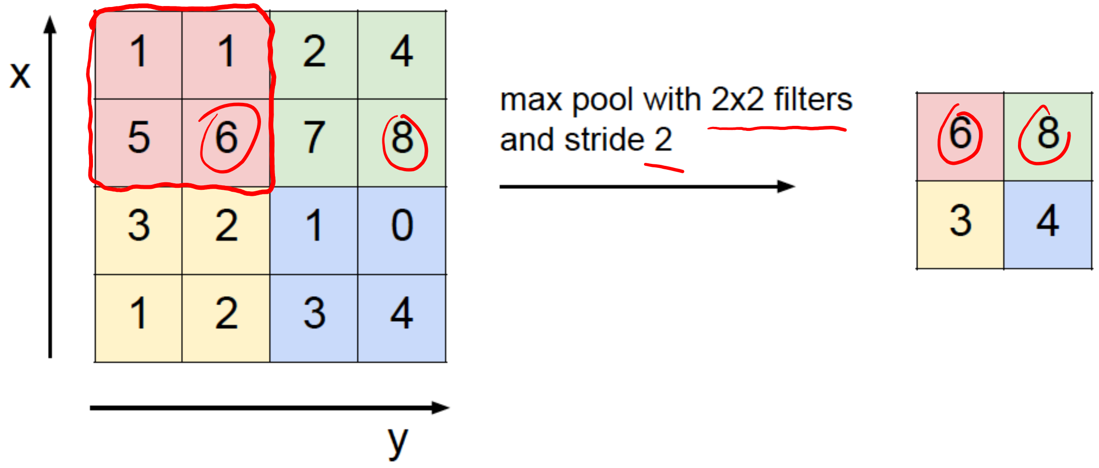


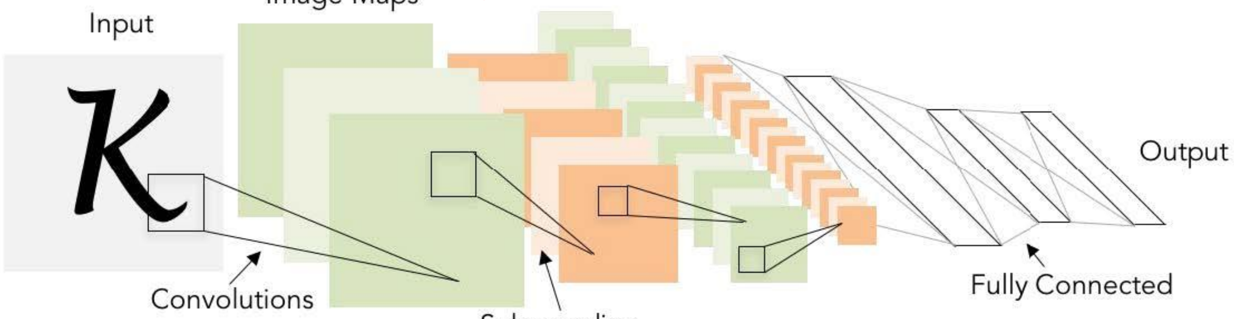
Image courtesy of Fei-Fei Li, Ranjay Krishna, Danfei Xu

CNN Architectures

- LeNet-5 [LeCunn et al. 1998]
- AlexNet [Krizhevsky et al. 2012]
- VGG [Simonyan et al. 2014]
- GoogLeNet [Szegedy et al. 2015]
- ResNet [He et al. 2016]
- …
- …

size = size w/ stride 1

size decreases

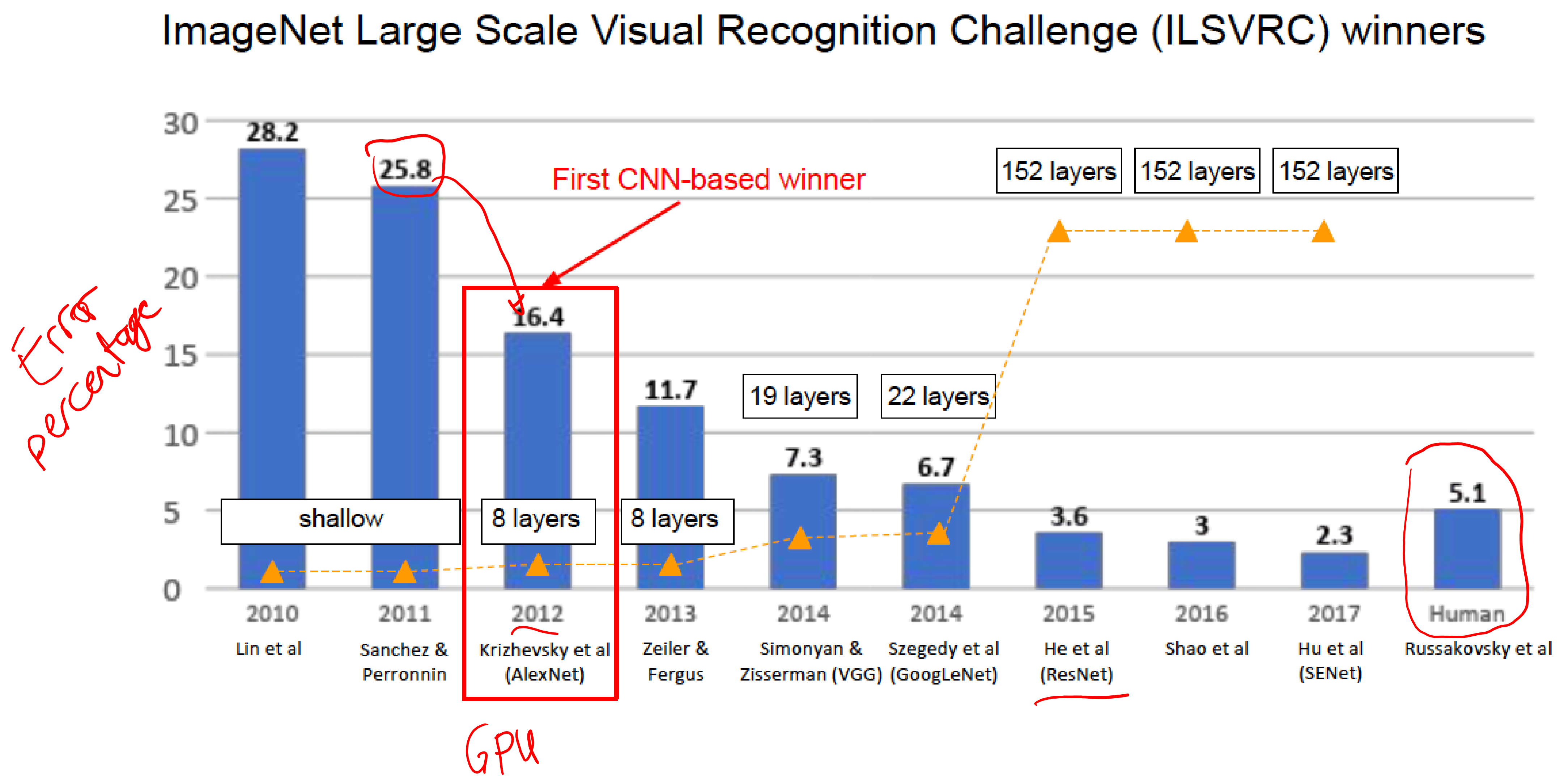


Pooling

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-FC-FC]

4 layers

Rise of Deep Learning



AlexNet

Img size = 224x224x3

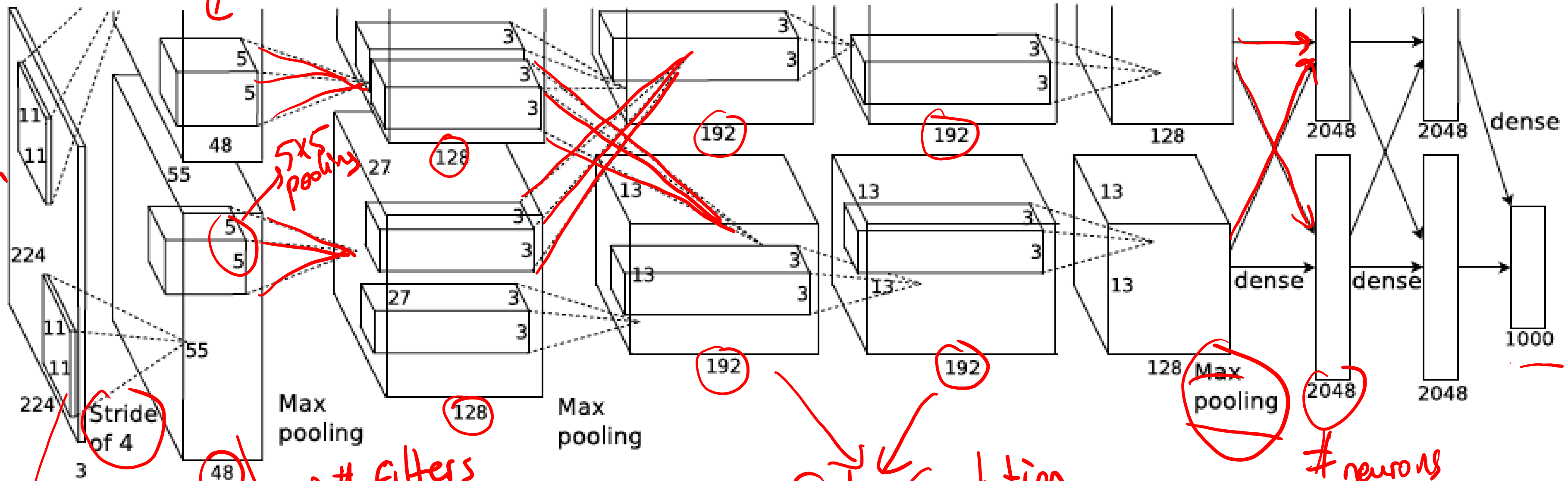
GPU 2 (48 filters/neurons)

Architecture:

- CONV1
- MAX POOL1
- NORM1 → Batch/Layer Normalization
- CONV2
- MAX POOL2
- NORM2
- CONV3
- CONV4
- CONV5
- Max POOL3
- FC6
- FC7
- FC8

5 Conv.

3 FC



5x5 pooling

Filter size = 11x11

GPU1

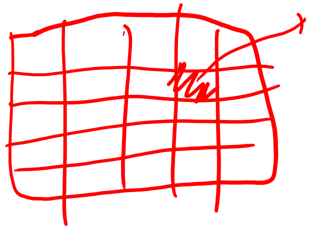
filters
neurons

Only Convolution

neurons

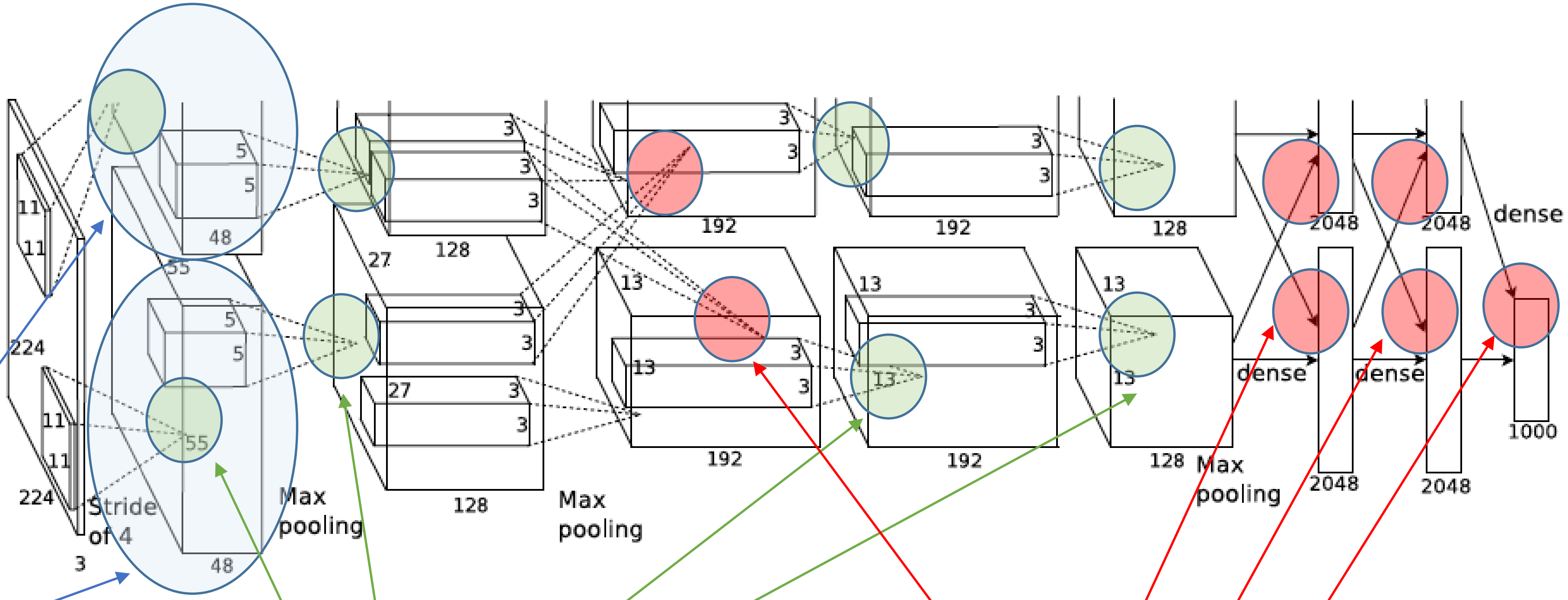
2 GPU

Max Pooling



max value

AlexNet

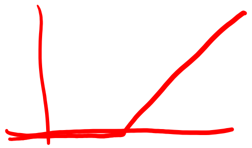


Trained on 2 GPUs. Half the neurons on each GPU.

Connections only within the same GPU

Connections across GPUs

AlexNet



Nonlin. activation func.

- First use of ReLU
- Used Norm layers

increase diversity in training dataset

- **Data augmentation (overfitting)**
- **Dropout (overfitting)**
  - Randomly drop neurons for each training instance in feedforward and backpropagation with probability 0.5
  - “Every time an input is presented, the neural network samples a different architecture, but all these architectures share weights”
  - “At test time, use all the neurons but multiply their outputs by 0.5”
- SGD for weight update in training
  - Gradient batch size 128
  - Momentum 0.9

$0.01 \quad w_{n+1} = 0.9 w_n + 0.01 \nabla_w E$

  - Learning rate 1e-2, reduced by 10 manually when validation accuracy plateaus
- 7 CNN ensemble: 18.2% -> 15.4%

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

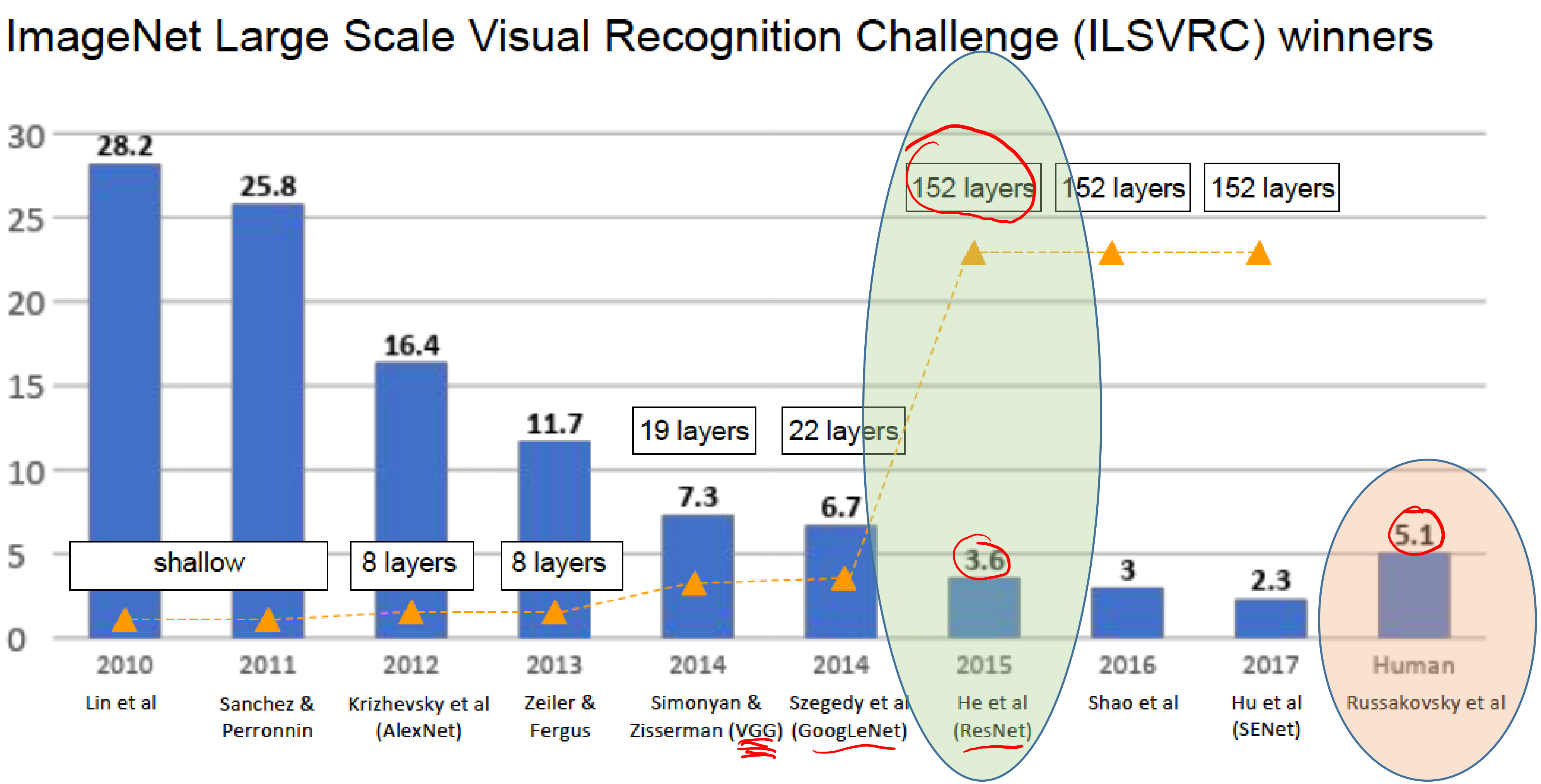
Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called “dropout” that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

ResNet

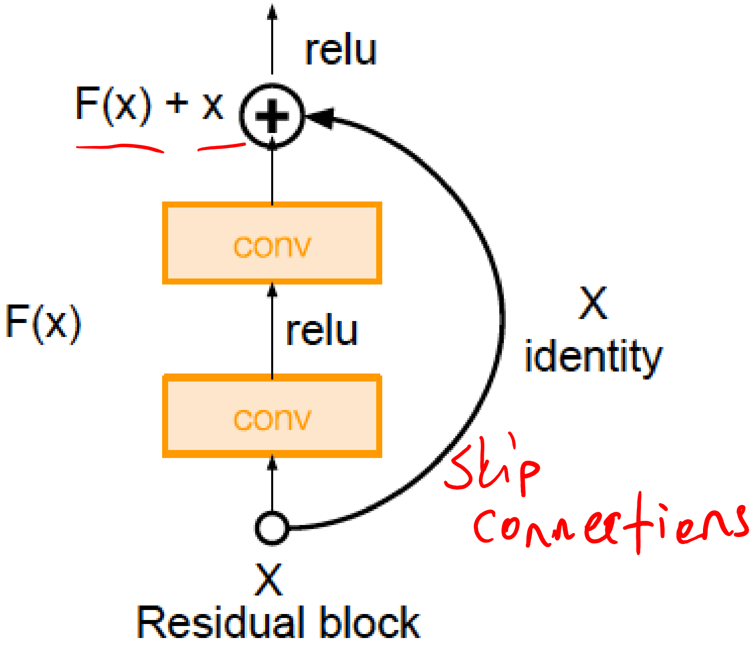


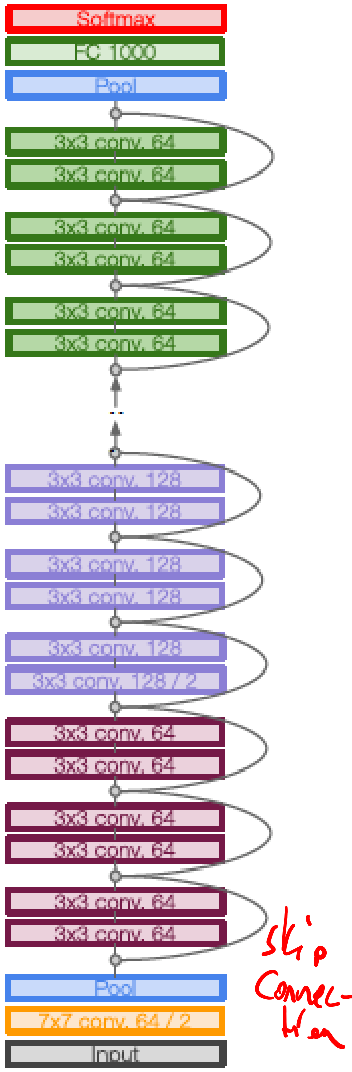
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners
30
25
20
15
10
5
0
28.2
25.8
16.4
11.7
7.3
6.7
3.6
3
2.3
5.1
shallow
8 layers
8 layers
19 layers
22 layers
152 layers
152 layers
152 layers
2010
Lin et al
2011
Sanchez & Perronnin
2012
Krizhevsky et al (AlexNet)
2013
Zeiler & Fergus
2014
Simonyan & Zisserman (VGG)
2014
Szegedy et al (GoogLeNet)
2015
He et al (ResNet)
2016
Shao et al
2017
Hu et al (SENet)
Human
Russakovsky et al

ResNet

Residual

- Very deep networks using residual connections
- 152-layer model for ImageNet
  - Outperformed the human-level performance
- Now focus shifted to Efficient Networks:
  - Lots of tiny networks aimed at mobile devices: MobileNet, ShuffleNet, etc.



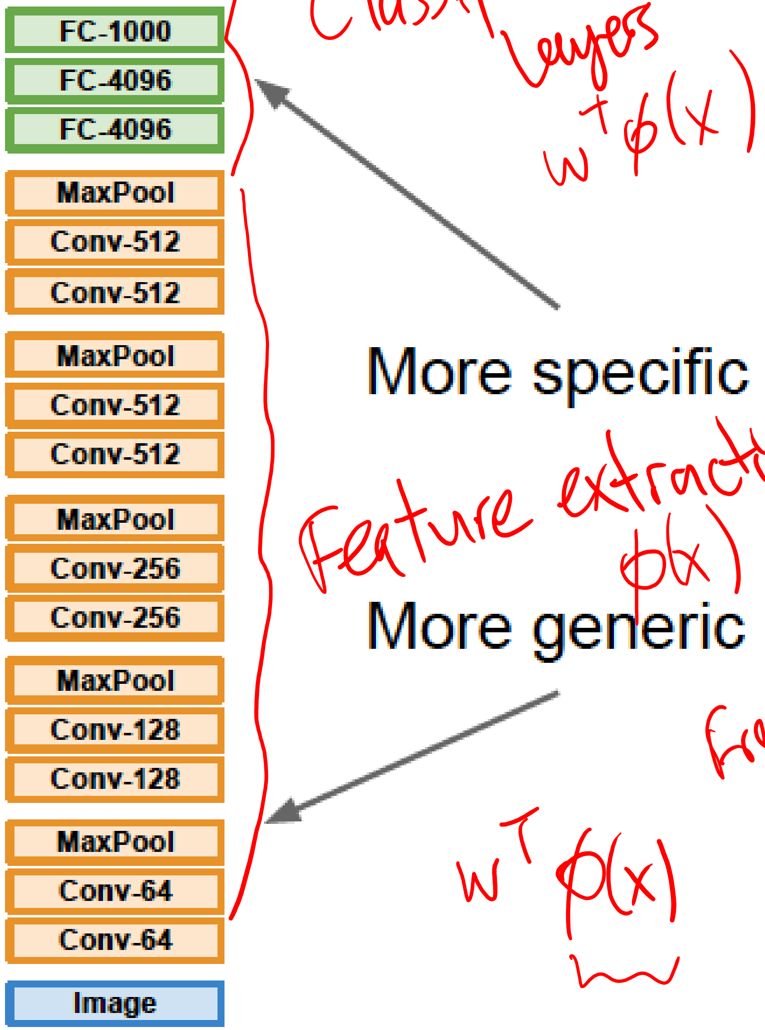


Transfer Learning

Trained on new data

Classification layers

$w^T \phi(x)$



Feature extraction

$\phi(x)$

Freeze

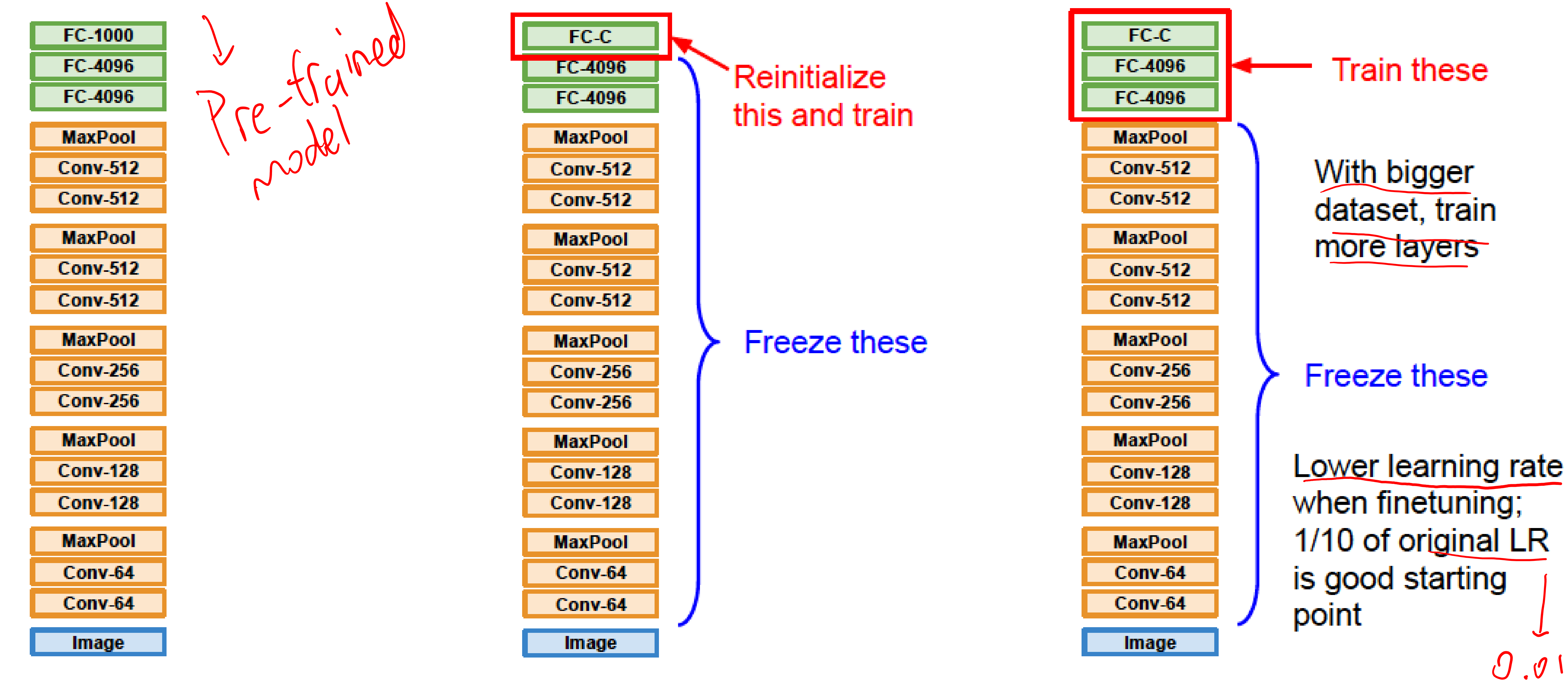
$w^T \phi(x)$

	very similar dataset	very different dataset
very little data	Finetune linear classifier on top layer	You're in trouble… Try data augmentation / collect more data
quite a lot of data	Finetune a few layers	Finetune a larger number of layers

Transfer Learning with CNNs

Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014
Razavian et al, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", CVPR Workshops 2014

1. Train on Imagenet
2. Small Dataset (C classes)
3. Bigger dataset



Transfer Learning

In practice:

- Take a pretrained model
  - Trained on a very large dataset such as ImageNet
  - "Model Garden" of pretrained models:
    https://github.com/tensorflow/models
    https://github.com/pytorch/vision

- Train only a few last layers on your dataset