Deep Neural Networks
And Where to Find Them

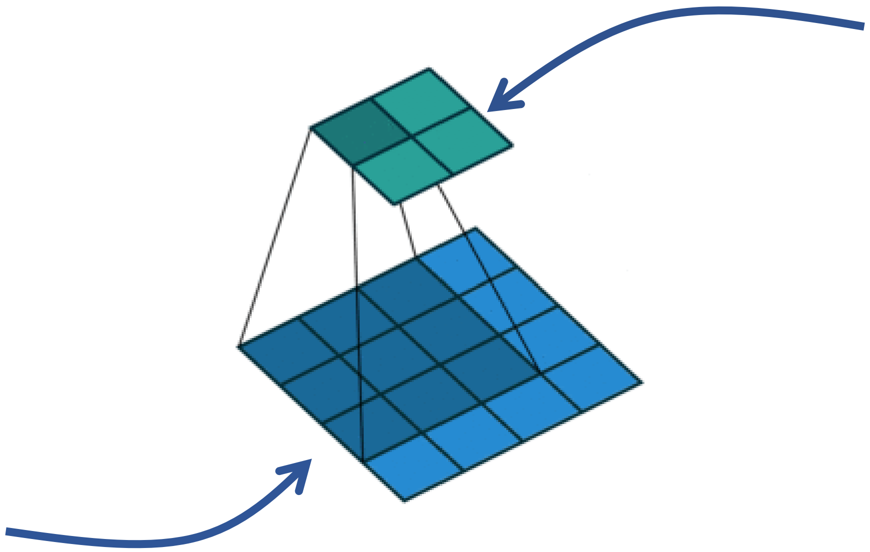
Lecture 5

Artem Korenev, Nikita Gryaznov

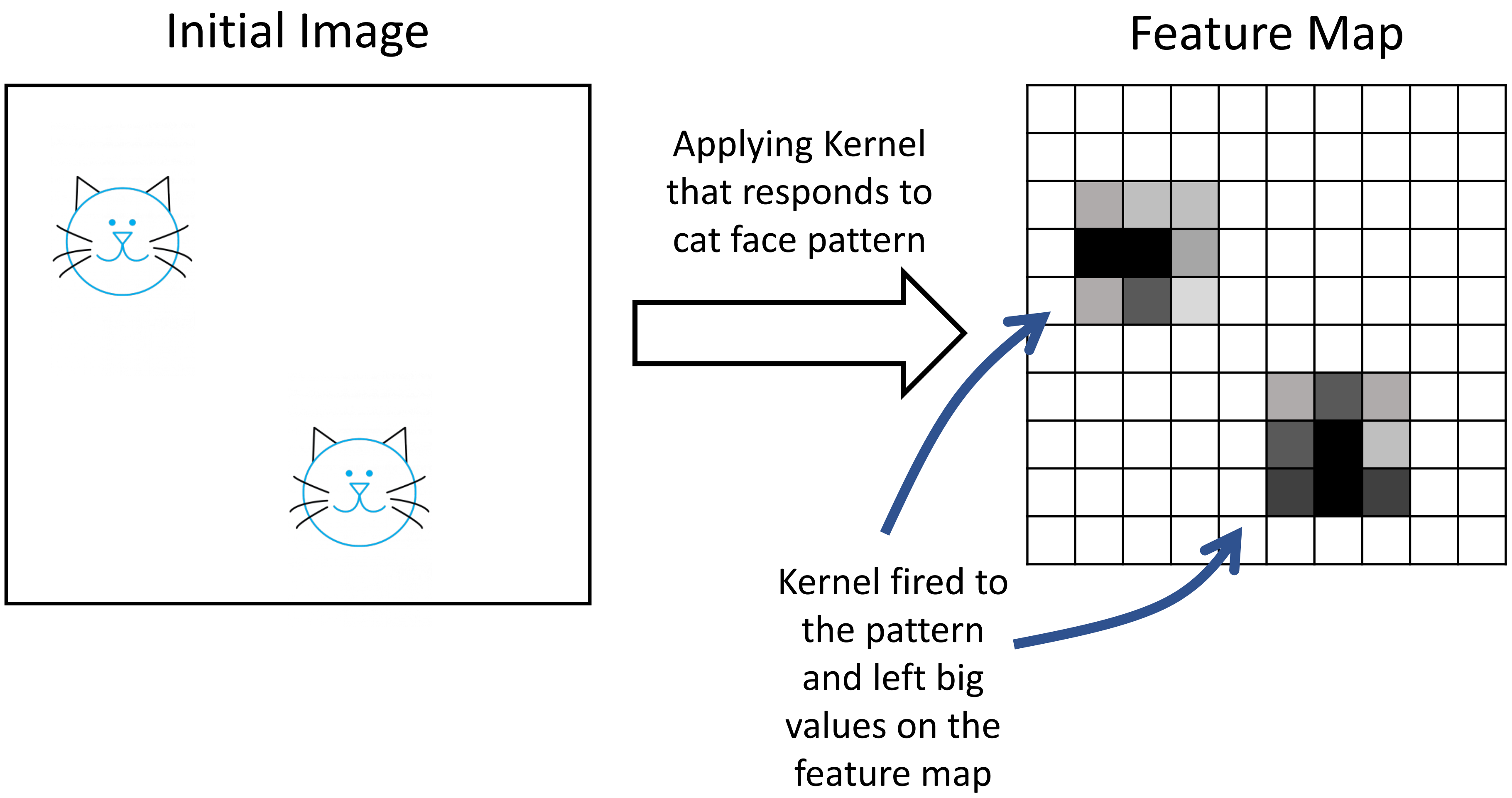
Recap of Convolutional NNs

Recap – Convolutional Neural Network

- Learning small *filters* (*kernels*) to catch useful features
- Using convolution operation to apply filter to every position on the *feature map* (or on the image)
- Adding layers of convolutions – filters learn more complex features



Recap – Kernels Learn Useful Features



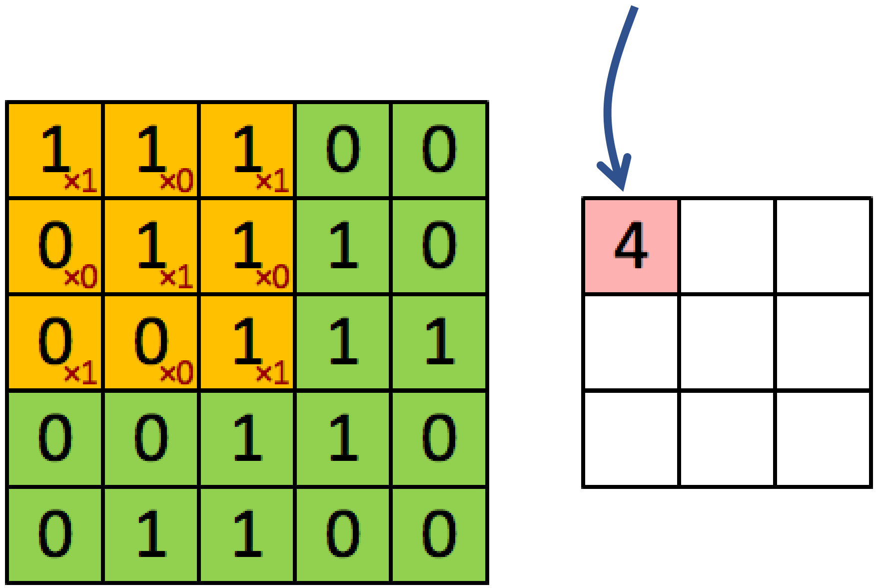
Recap – Convolution Computations

Kernel
(i.e. filter)
has size 3x3

Kernel Values

1	0	1
0	1	0
1	0	1

$$\begin{array}{c} 1*1+1*0+1*1 \\ + \\ 0*0+1*1+1*0 \\ + \\ 0*1+0*0+1*1 \end{array} = 4$$



Image

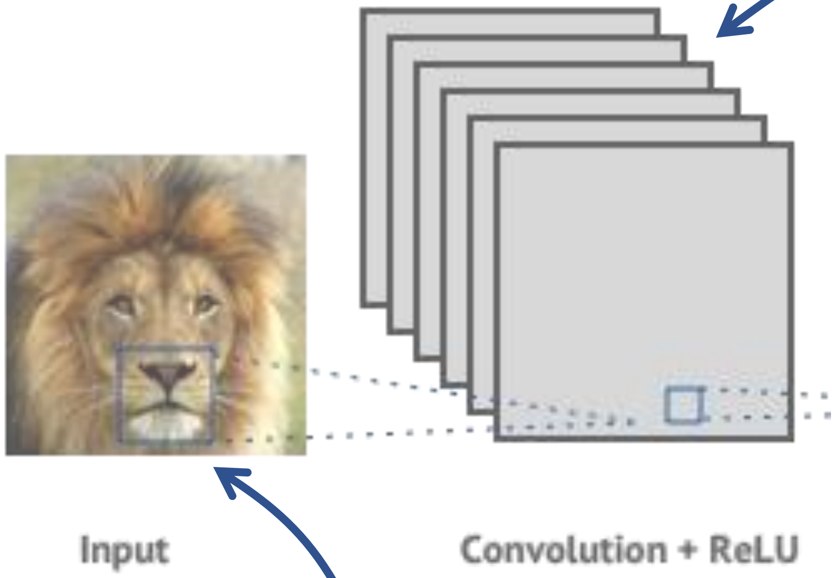
Convolved Feature

Recap – Number of Channels

We applied 6 different kernels, therefore we obtain 6 feature maps

Now we can treat these 6 feature maps as a new image with 6 channels to pass it further so we can apply new HxWx6 kernels on the next layer

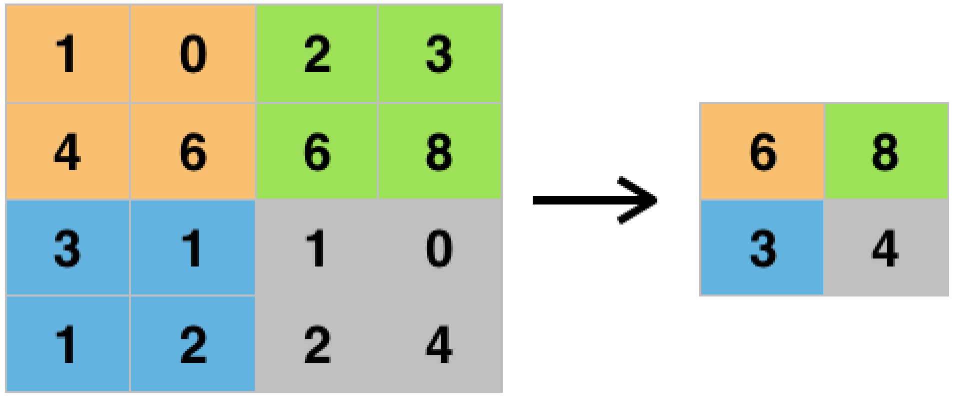
3 color channels for this image (RGB)



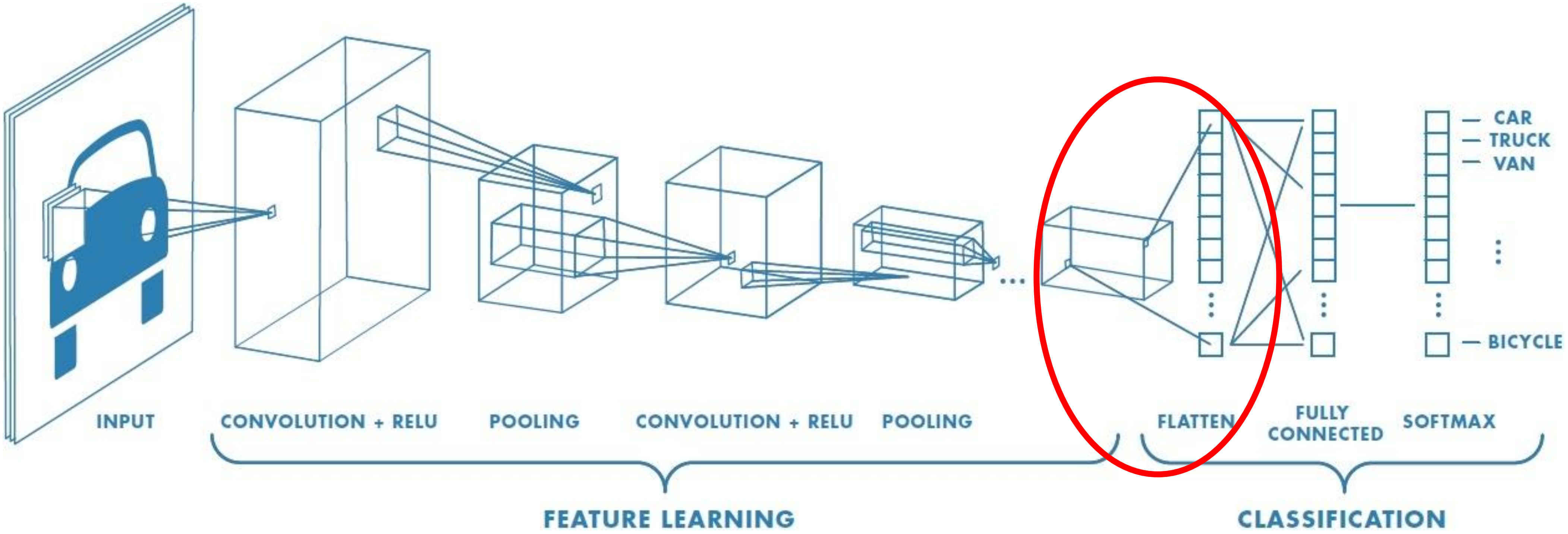
Let’s assume that this window has size of 20x20 pixels
Therefore we apply kernels of size 20x20x3 to the image

Recap – Max Pooling

Max Pooling
window size is
set to 2x2



Recap – Basic Structure of CNN

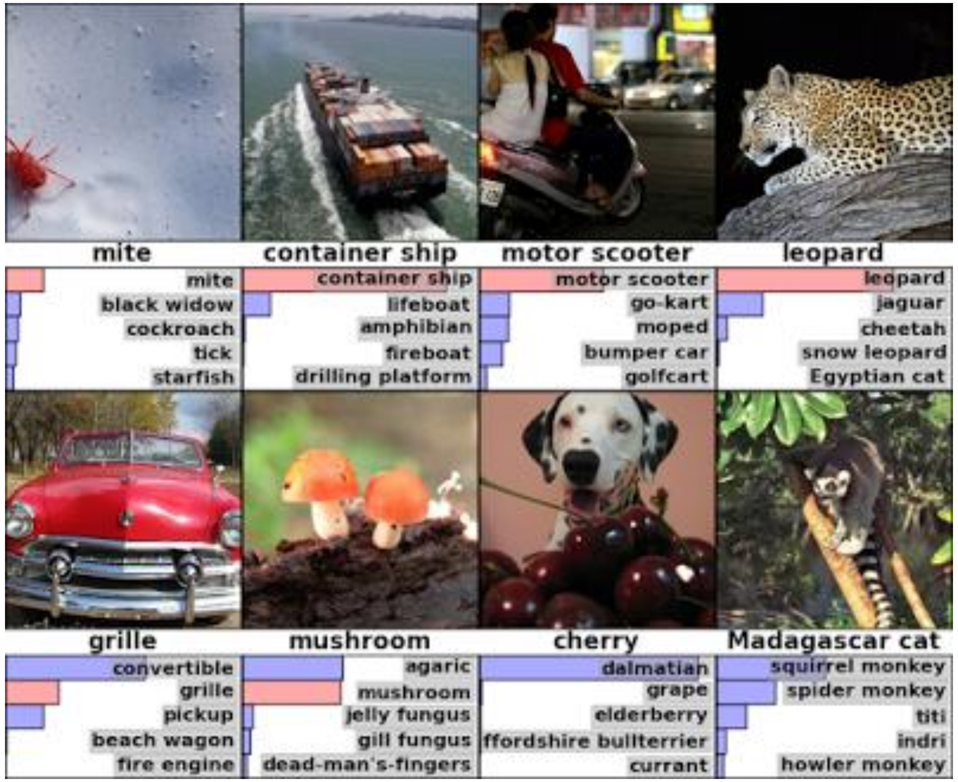


Flattening the feature map – making one shallow vector out of the 'image'

Deep Convolutional NNs

ImageNet Classification Challenge

- 1000 classes
- 1.2kk training images, 150k
- Main metric: top-5 error



LeNet

- Created by Yann LeCun in 1998 (!)
- Classification of handwritten digits dataset (MNIST)

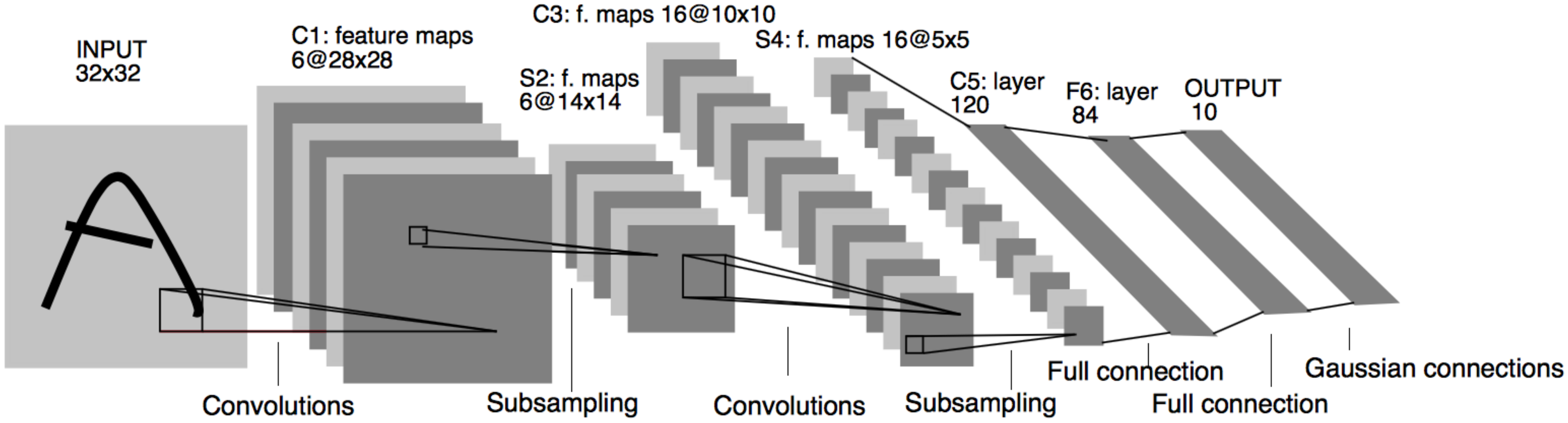
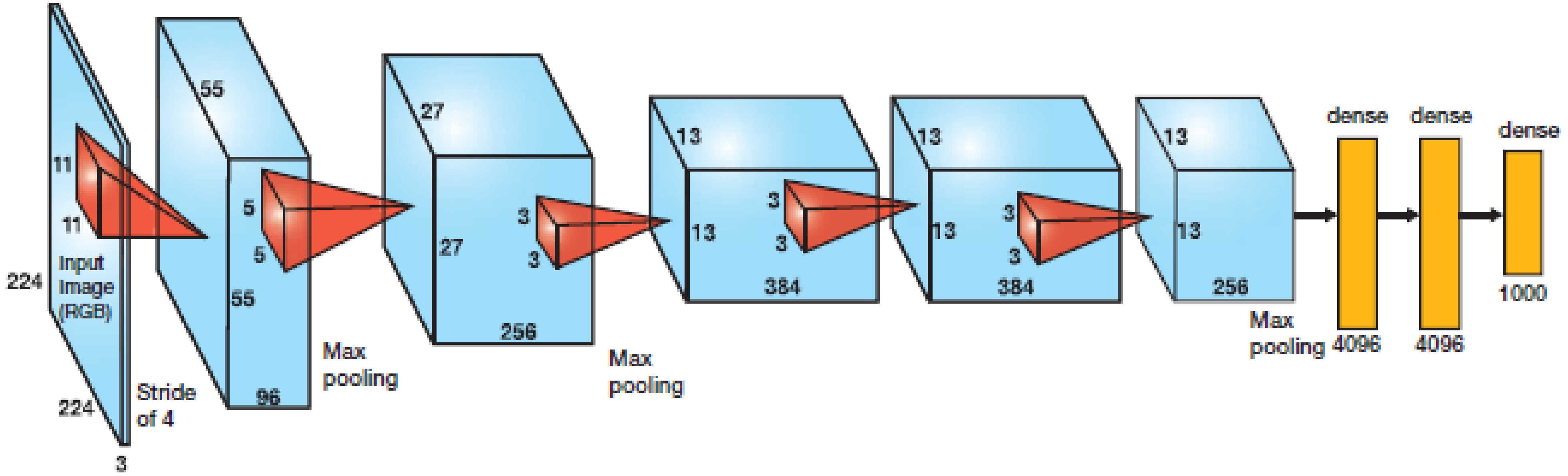


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

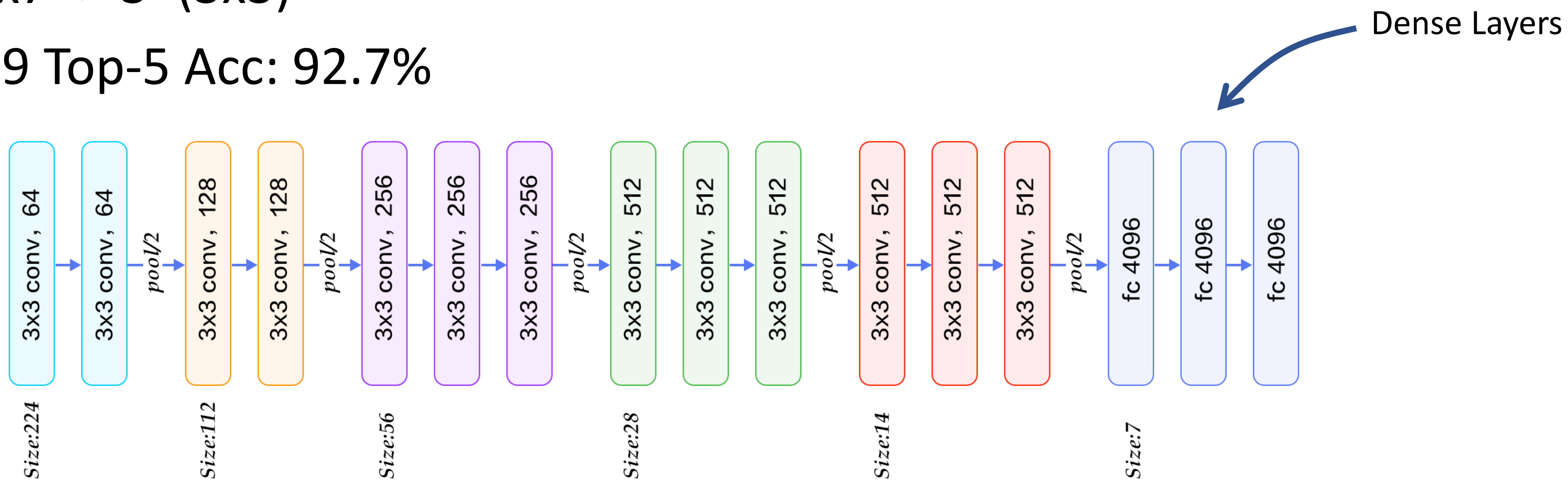
AlexNet - ImageNet 2012 Winner

- First winning deep learning solution
- Convolutions 11x11, 5x5 and 3x3
- 3 Dense layers at the end
- Top-1 Acc: 57%, Top-5 Acc: 80.3%



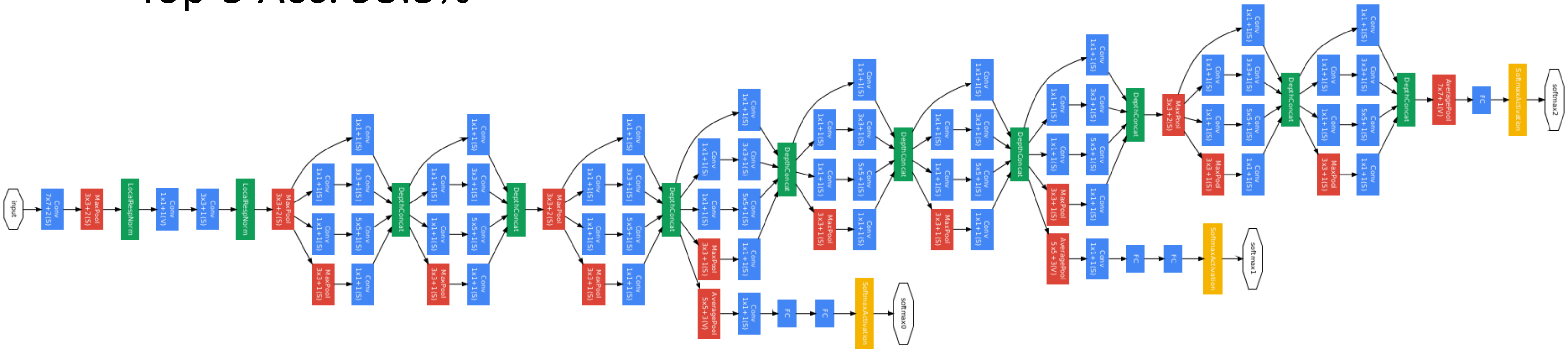
VGG – 2014

- Improved AlexNet
- There are two versions: VGG16 and VGG19 (number of layers)
- Transforming big convolutions to a subsequent 3x3 Convolutions
- E.g. 7x7 -> 3*(3x3)
- VGG19 Top-5 Acc: 92.7%



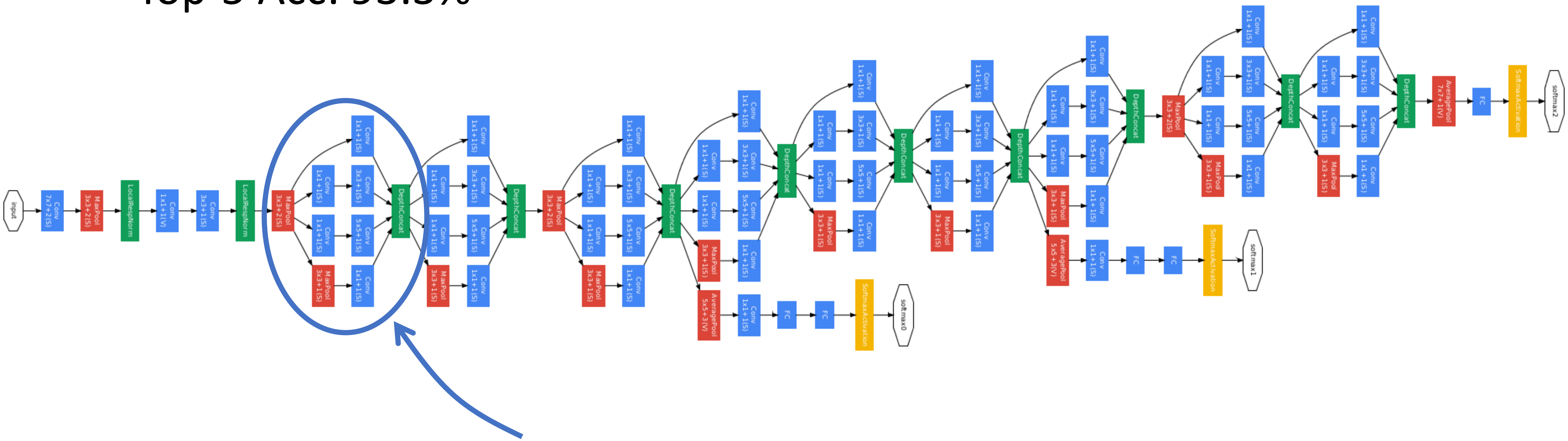
GoogleNet – ImageNet 2014 Winner

- Making NN more tree-structured
- Created Inception Model
- Top-5 Acc: 93.3%



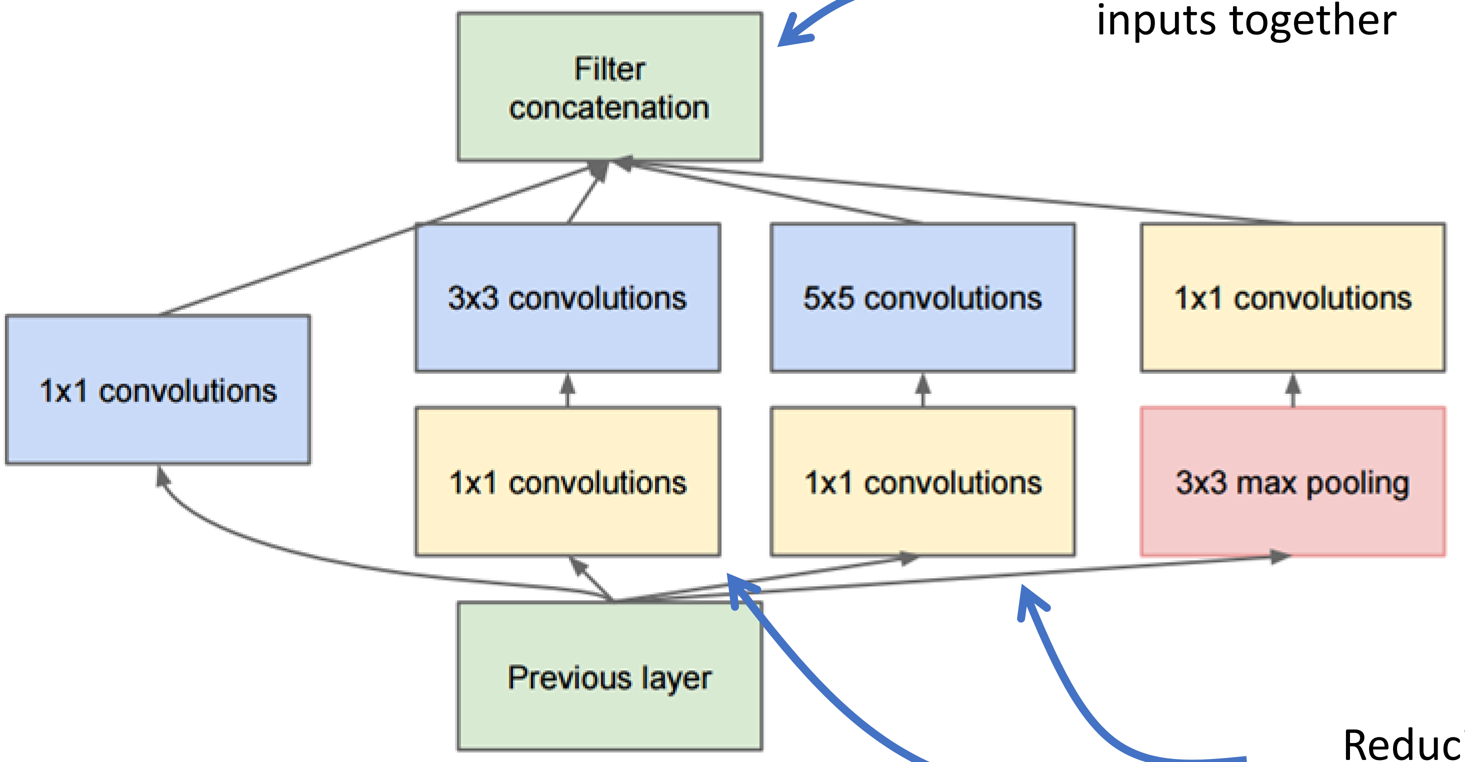
GoogleNet – ImageNet 2014 Winner

- Making NN more tree-structured
- Created Inception Model
- Top-5 Acc: 93.3%



Inception Module

GoogleNet – Inception Module



Stack channels of all inputs together

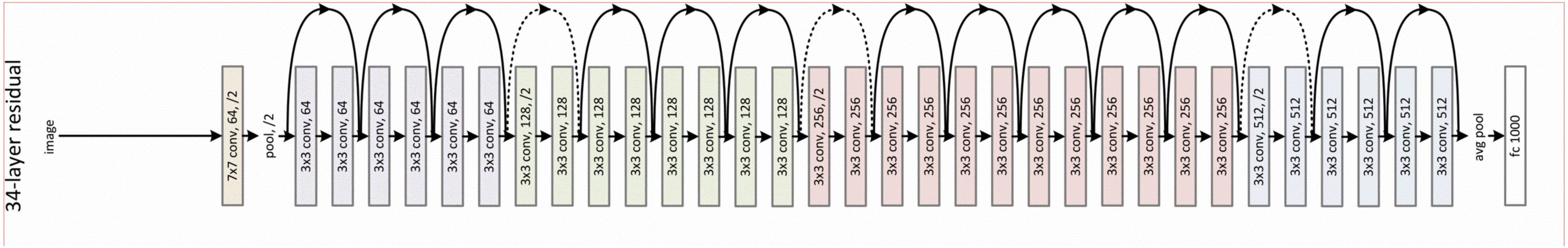
Reducing number of channels to reduce number of computations

Full Inception module

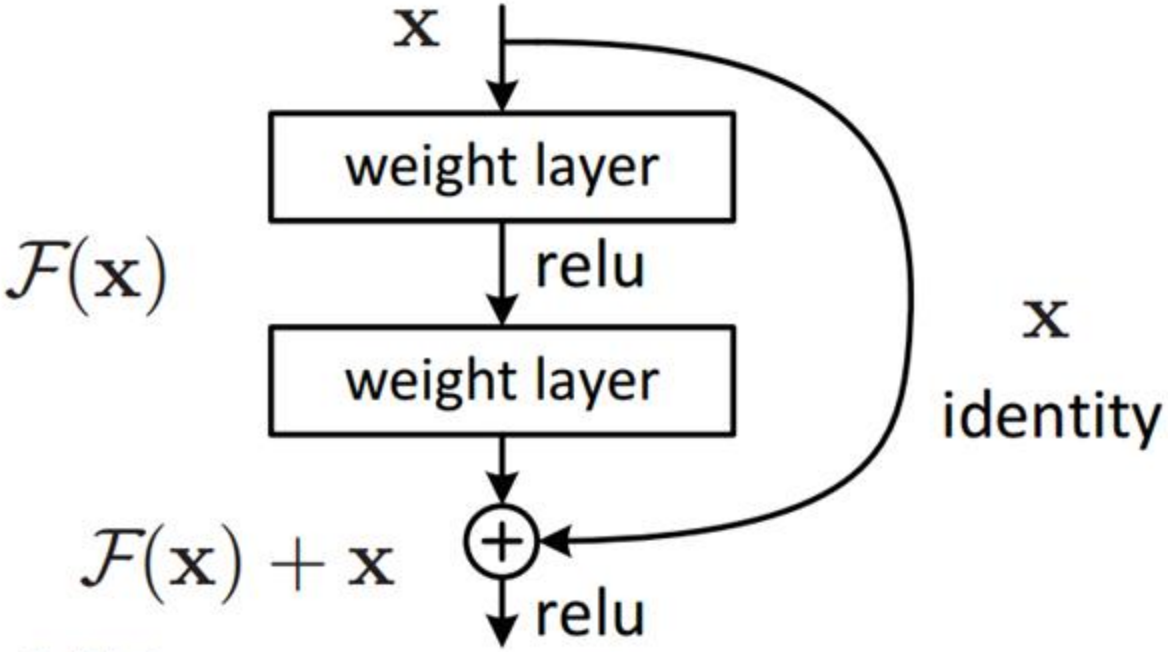
ResNet – ImageNet 2015 Winner

- Released by Microsoft
- They won every other competition as well
- ResNet-50, ResNet-101, ResNet-152
- Using idea of residual connections
- Top-5 Acc: 96.43% (surpasses human performance)

Number of layers!



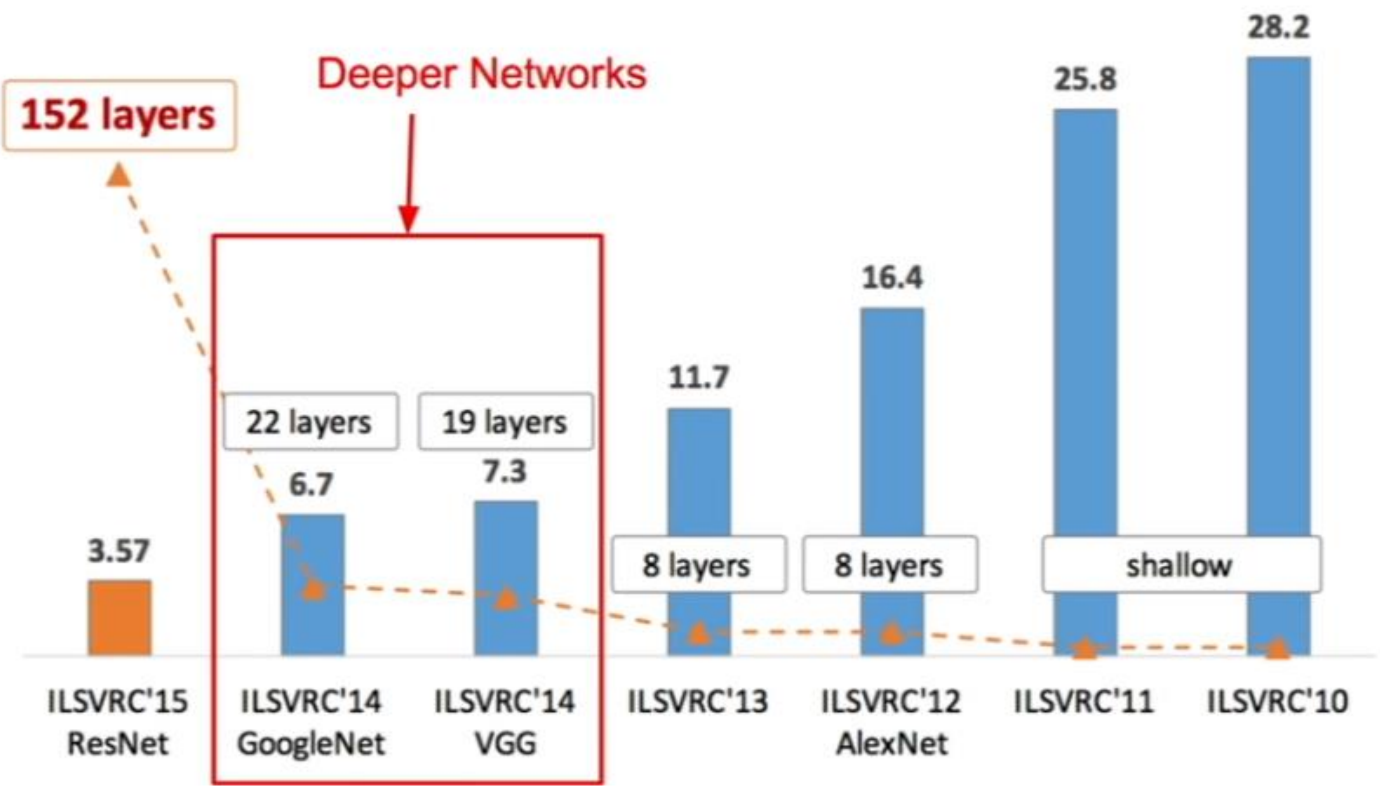
ResNet – Residual Block



A residual block

Layers and Performance

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

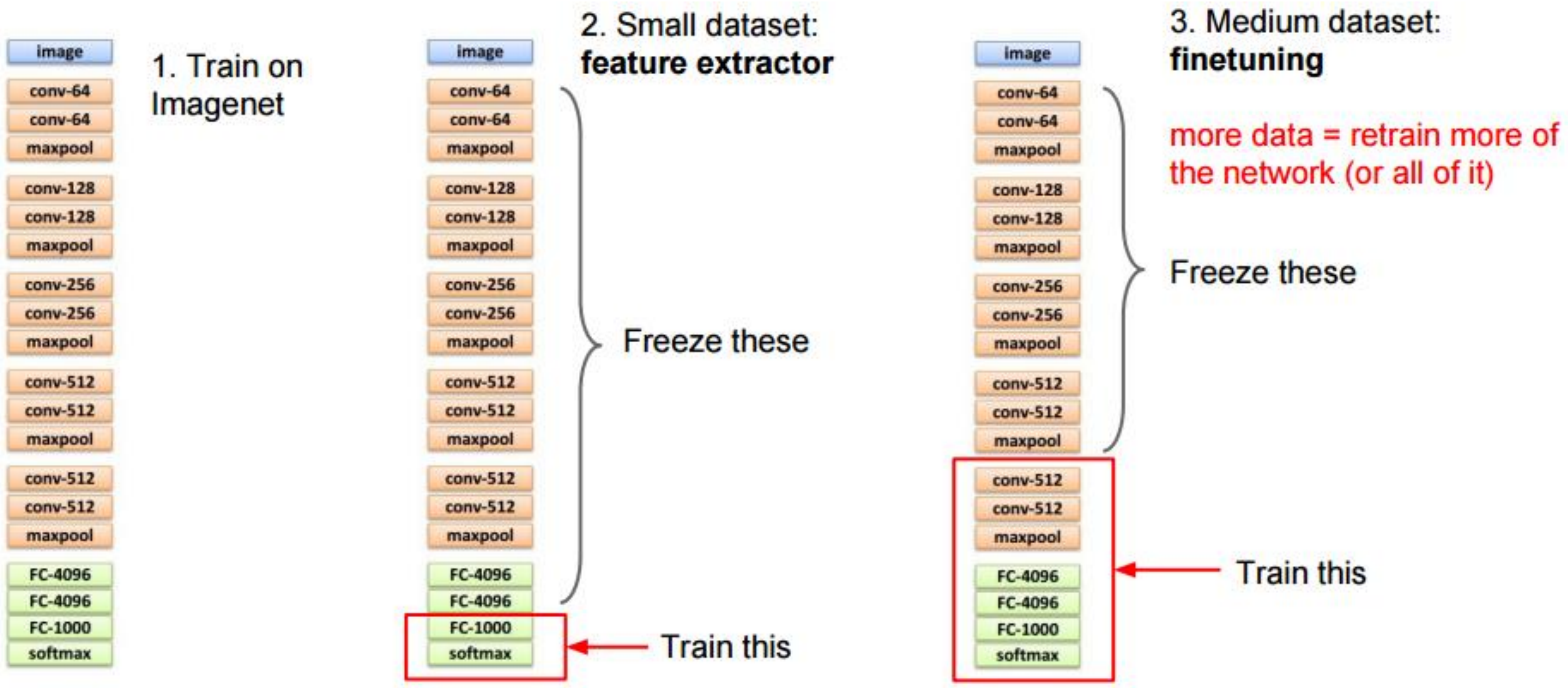


Transfer Learning

- We can reuse these high-performance NNs
- Pretrained NNs are available
- You can create the same NN but with different final layers to adjust it to your problem
- Then you learn the whole network or only small parts of it to train it directly on your model

- These models exists mostly for image classification only

Transfer Learning

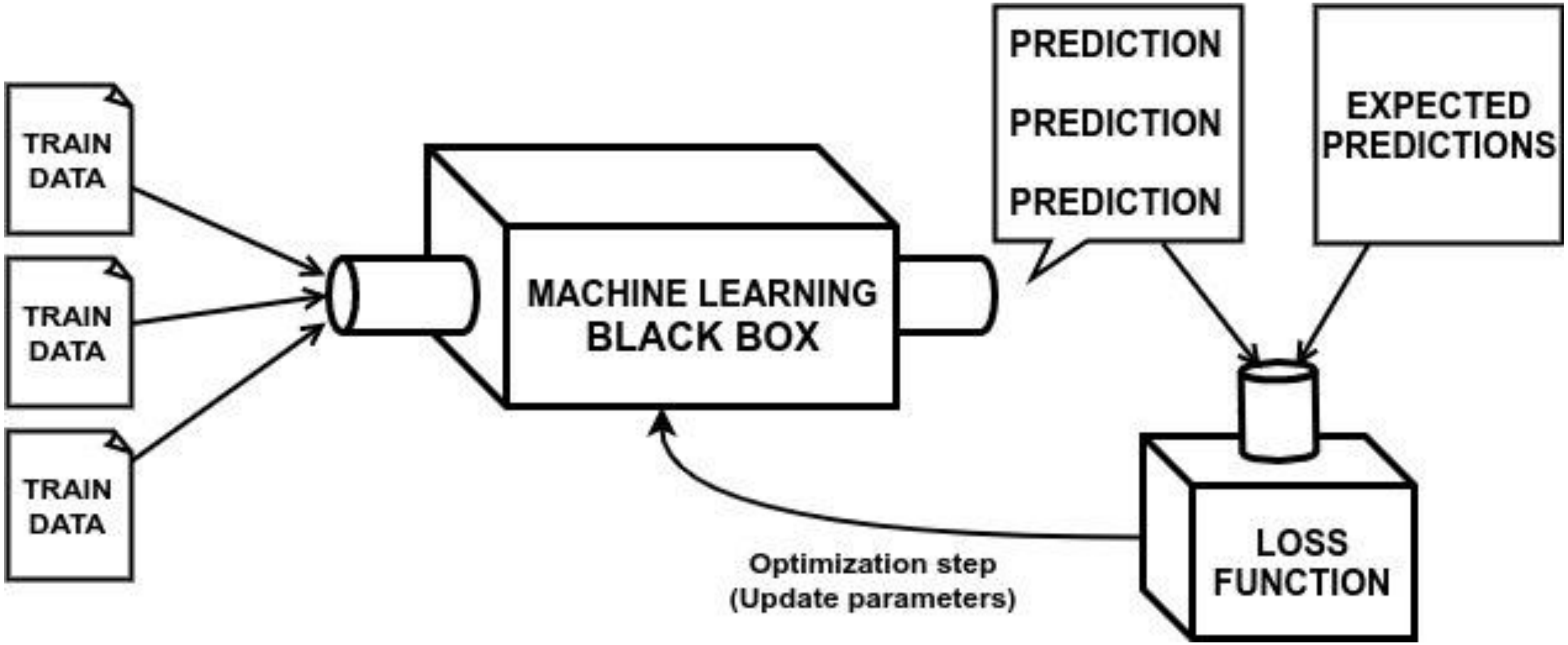


What is Used Nowadays

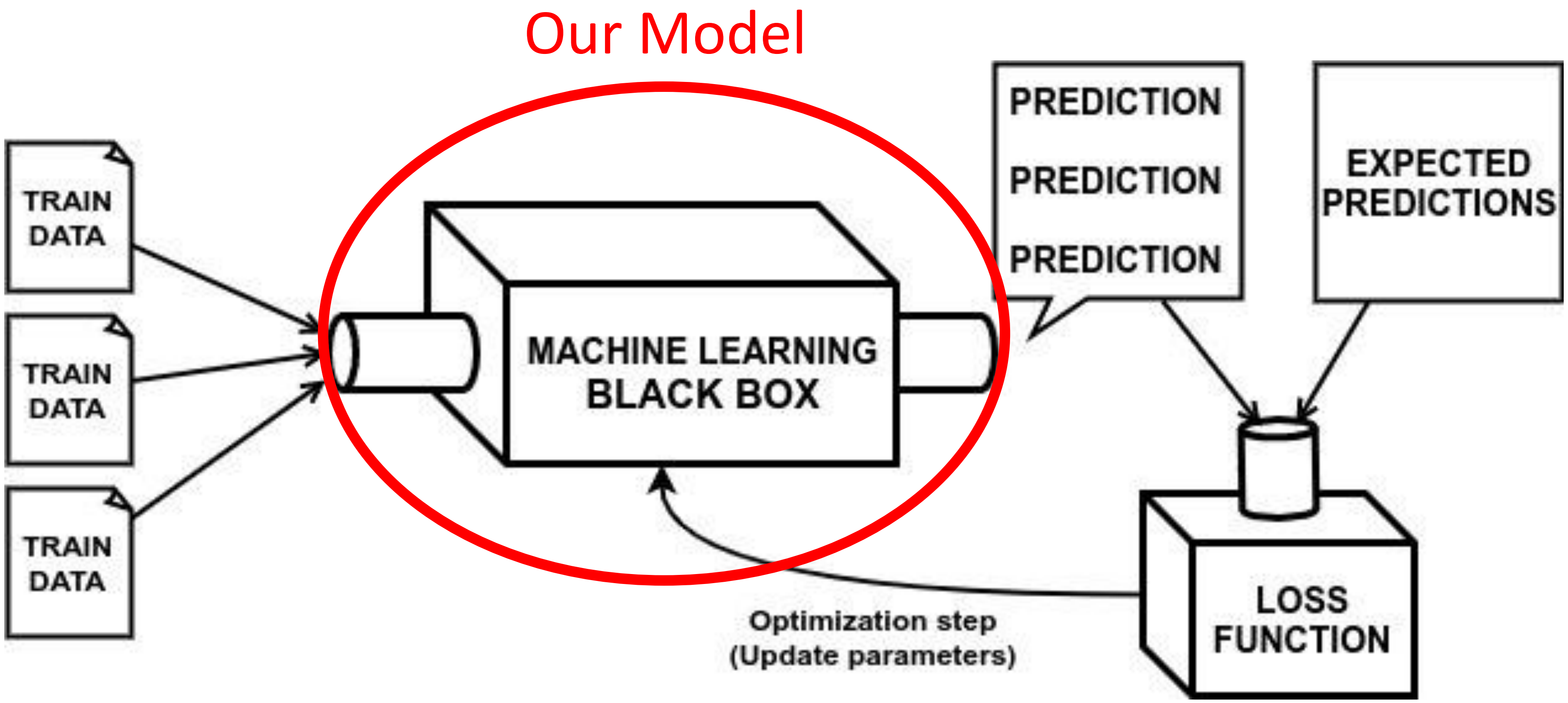
- ResNet, Inception, VGG networks are still used
- ResNet is the most popular choice
- Choose ResNet version for your problem (trade-off for complexity, time, amount of parameters, memory consumption…)
- Never train them from scratch – train it from pretrained version

Final Recap

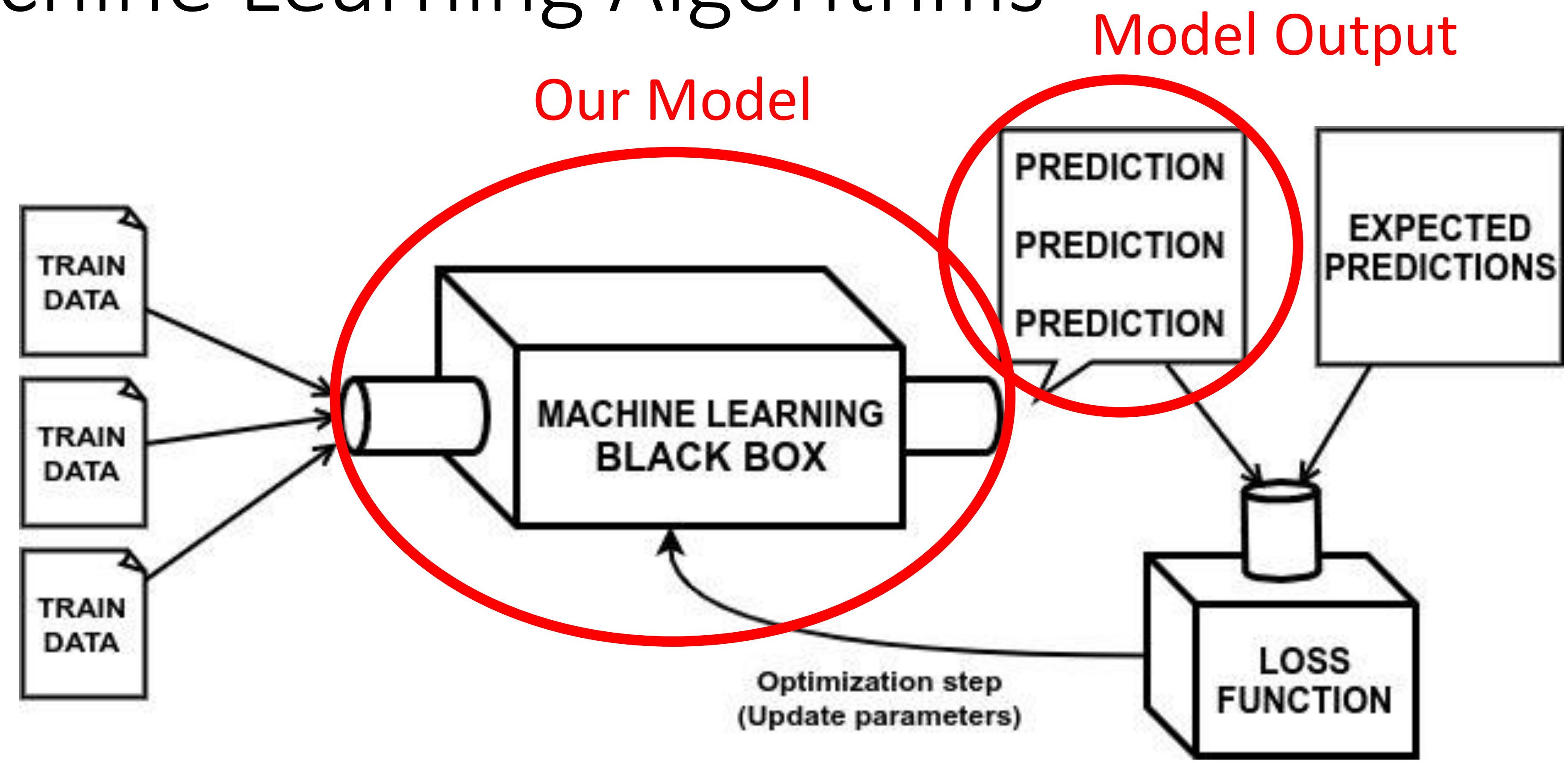
Machine Learning Algorithms



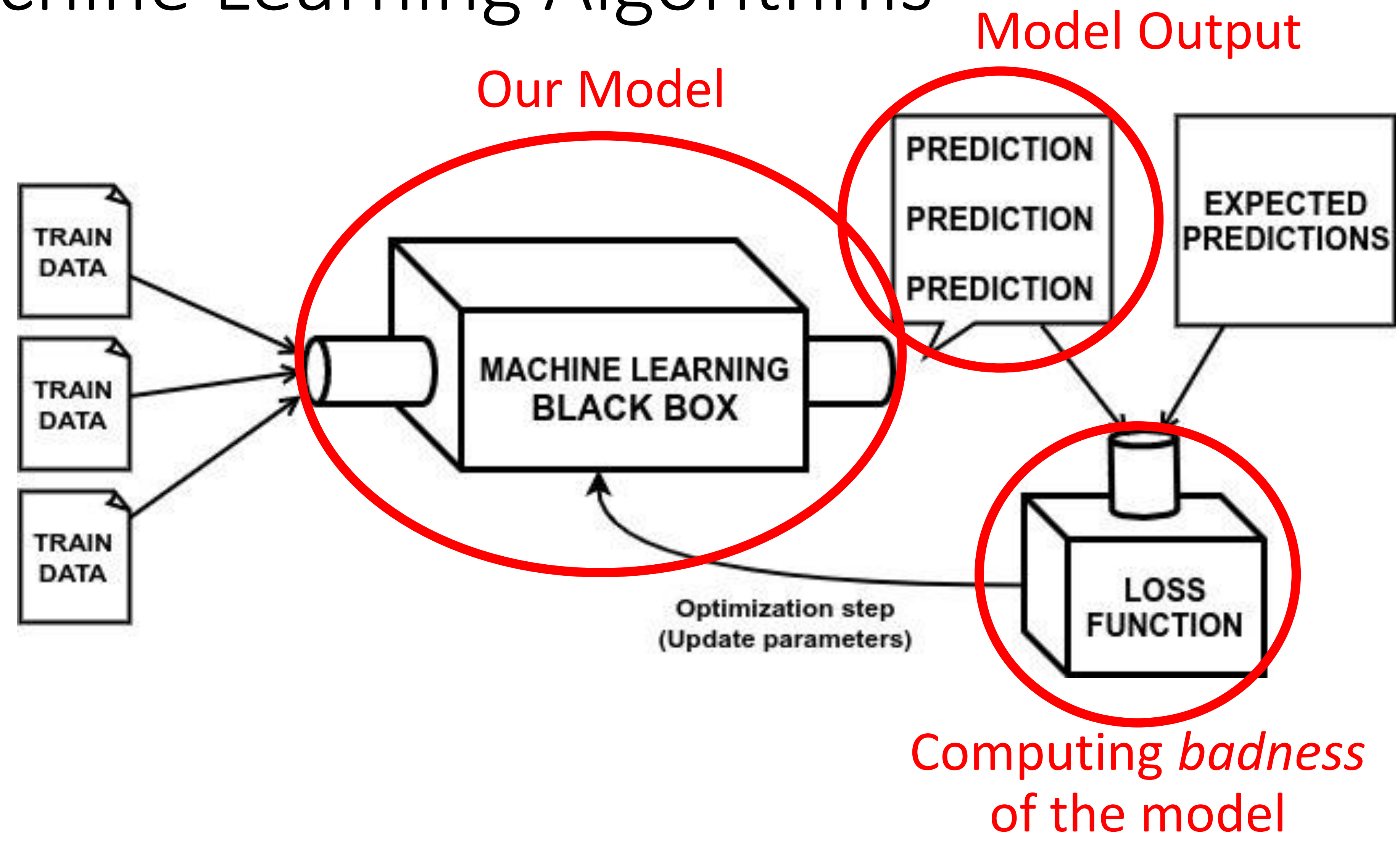
Machine Learning Algorithms



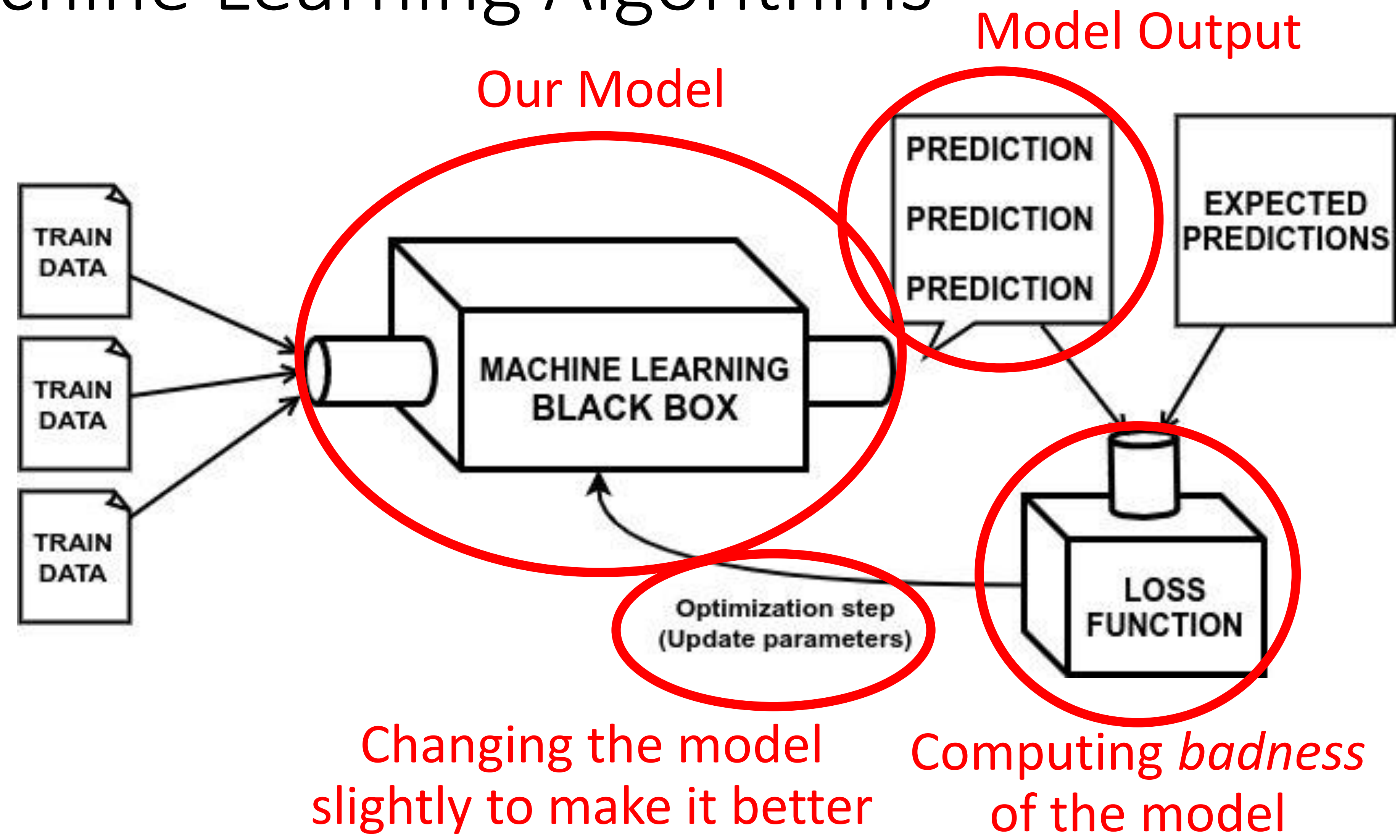
Machine Learning Algorithms



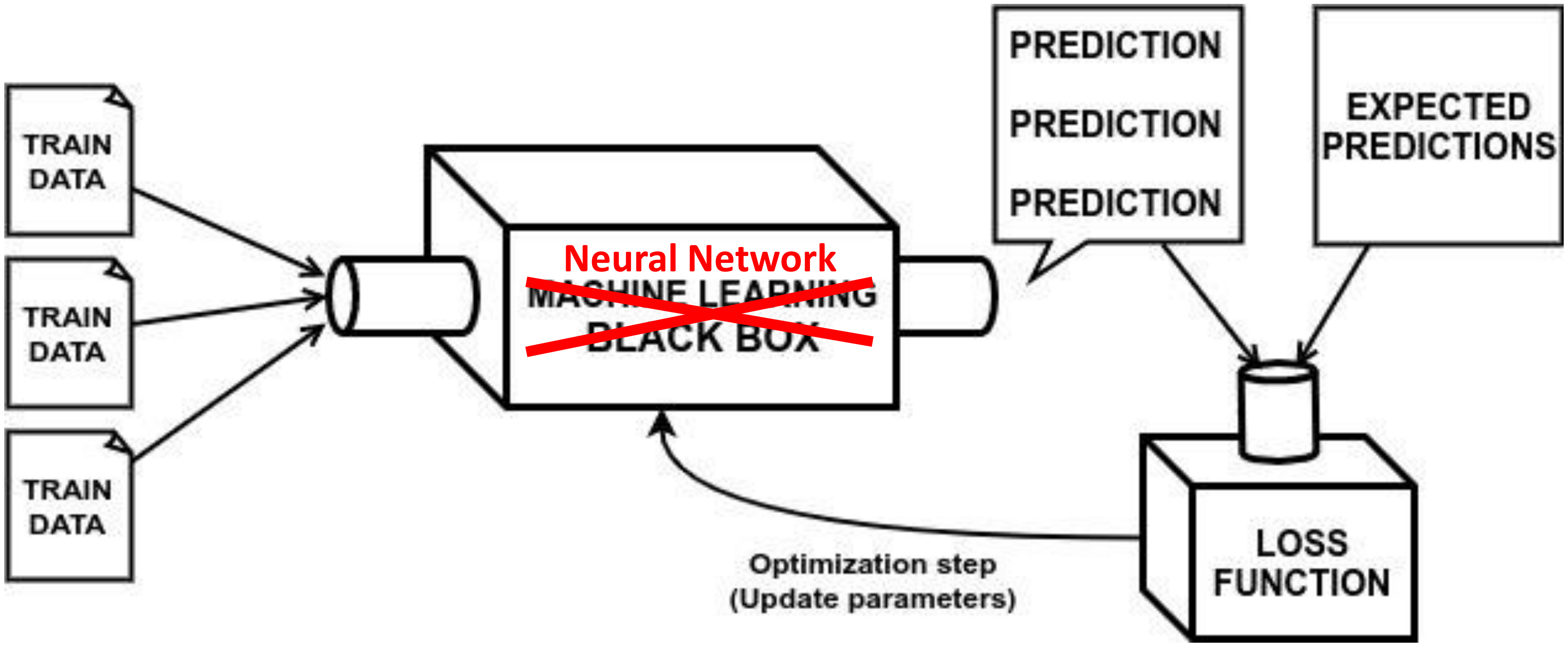
Machine Learning Algorithms



Machine Learning Algorithms



Neural Network as a ML Black Box



Types of Problems and Losses

Most frequent problems:

- Classification (predicting label(s) across the predefined set)

  Losses:

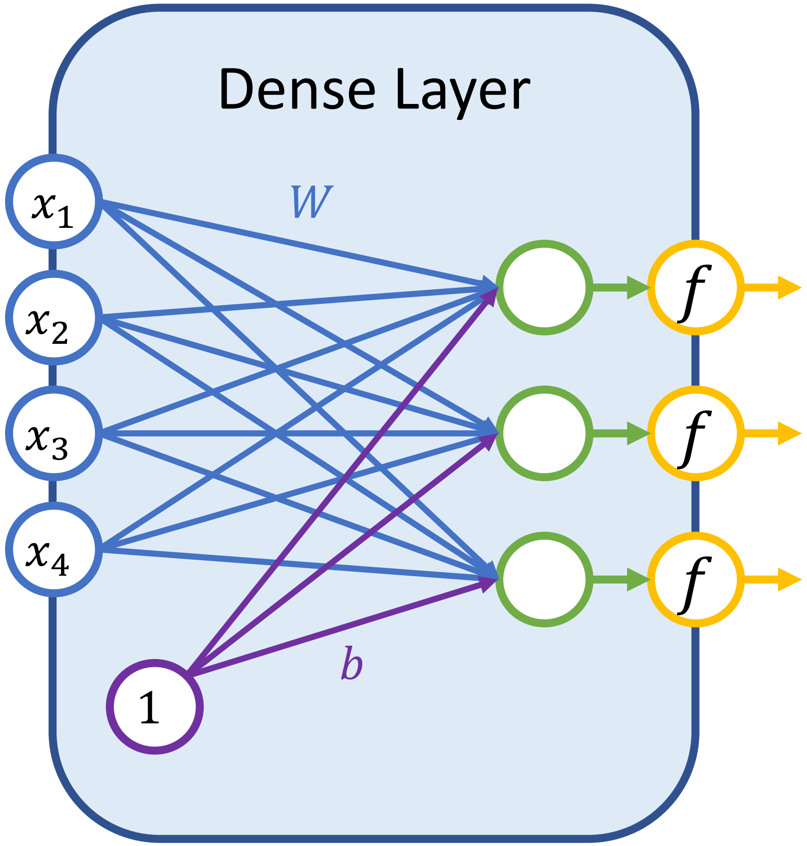
  - Binary Cross Entropy (2 classes) or Categorical Cross Entropy (>2 classes)
  - (rarely) Hinge Loss

- Regression (predicting real value without predefined set of outcomes)

  Losses:

  - Mean Squared Error (L2 Loss)
  - Mean Absolute Error (L1 Loss)

Dense Layer



x_1 Layer input

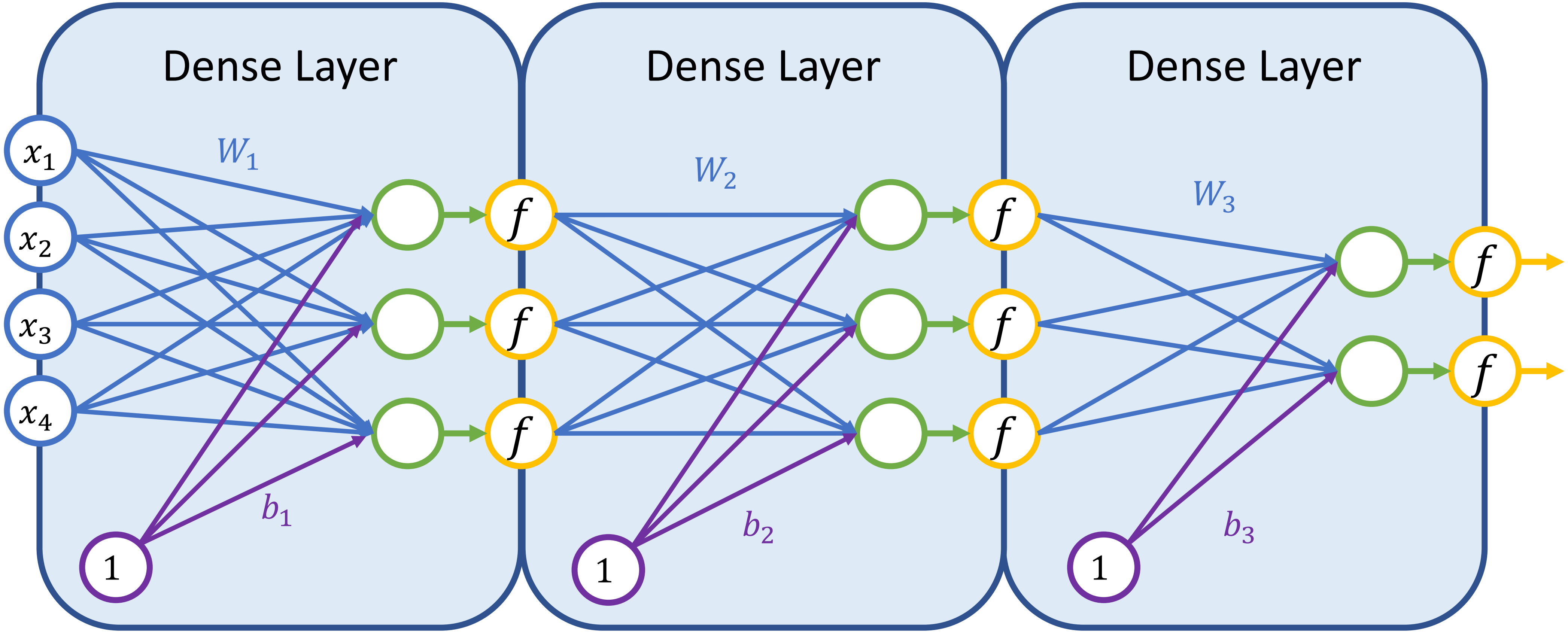
Neuron Output
$(W_{i1}x_1 + \cdots + W_{i2}x_2 + b_i)$

f Activation function
(Non-linearity)
(Neuron activation)

Layer weight
(trainable parameter)

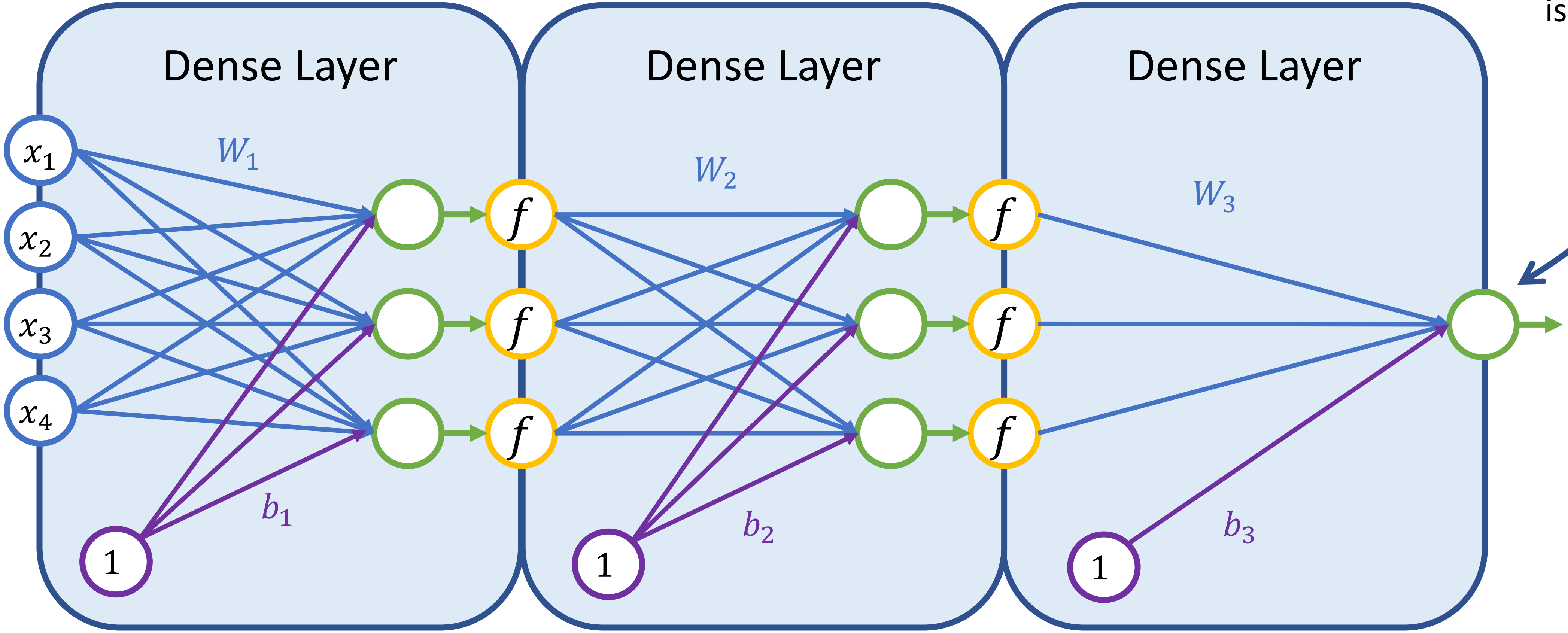
Layer bias
(trainable parameter)

Multi Layer Neural Network

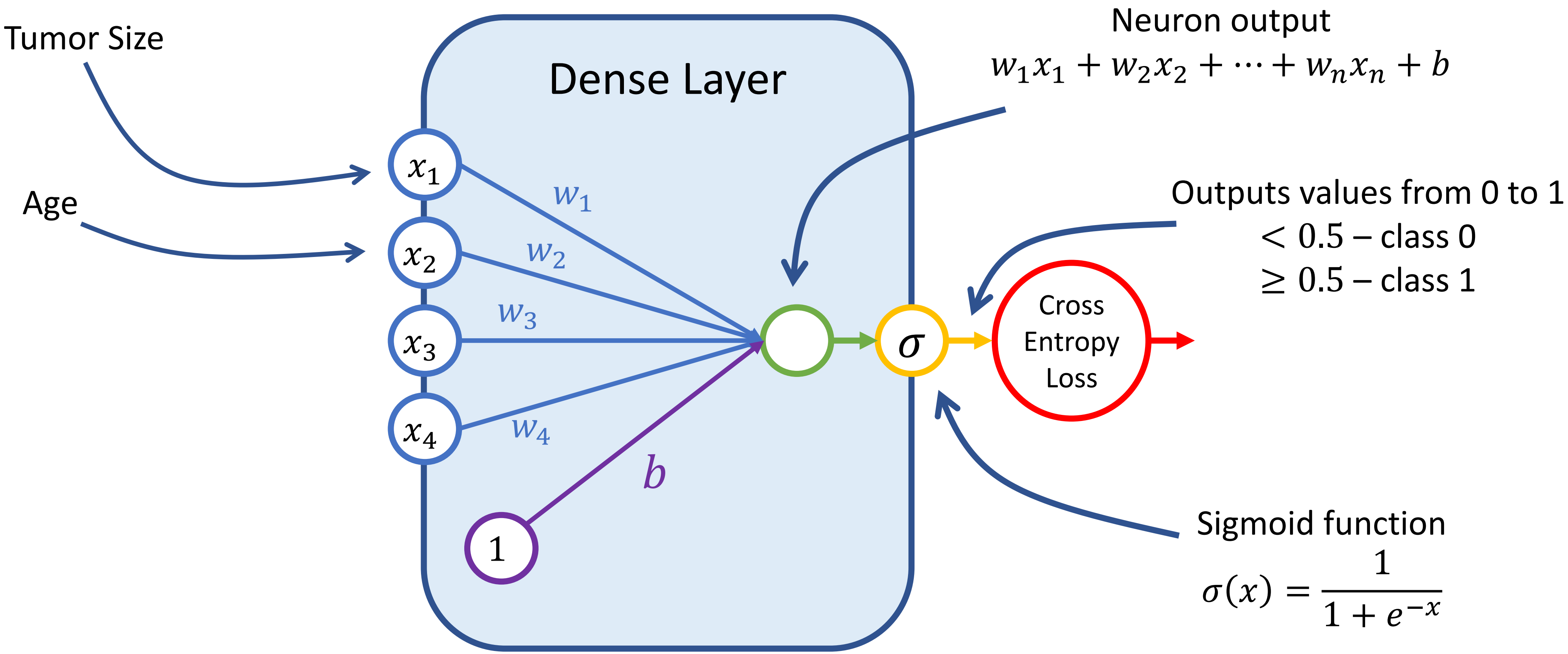


Regression Problem

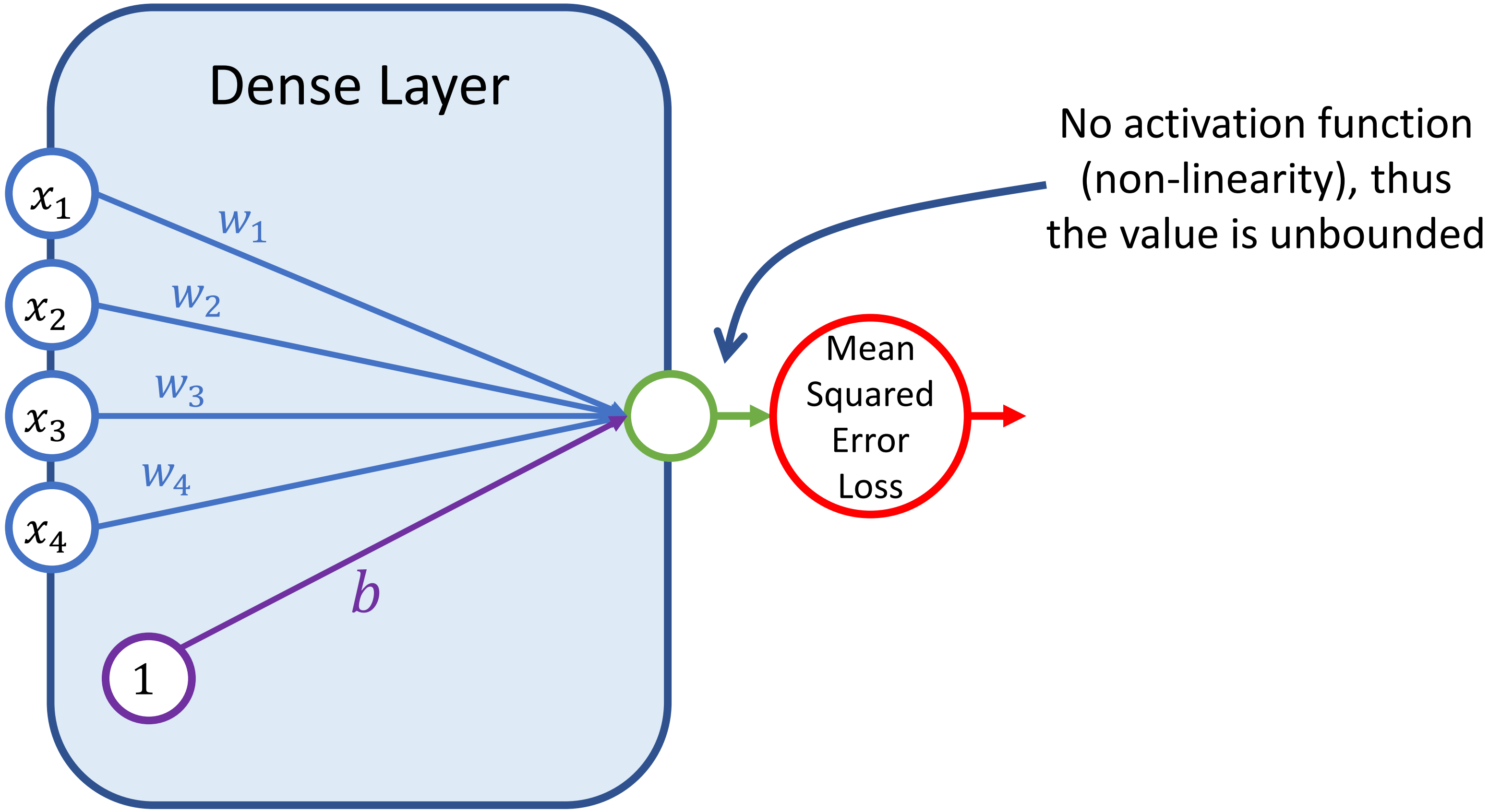
Often no activation function is used!



Logistic Regression Problem as a NN

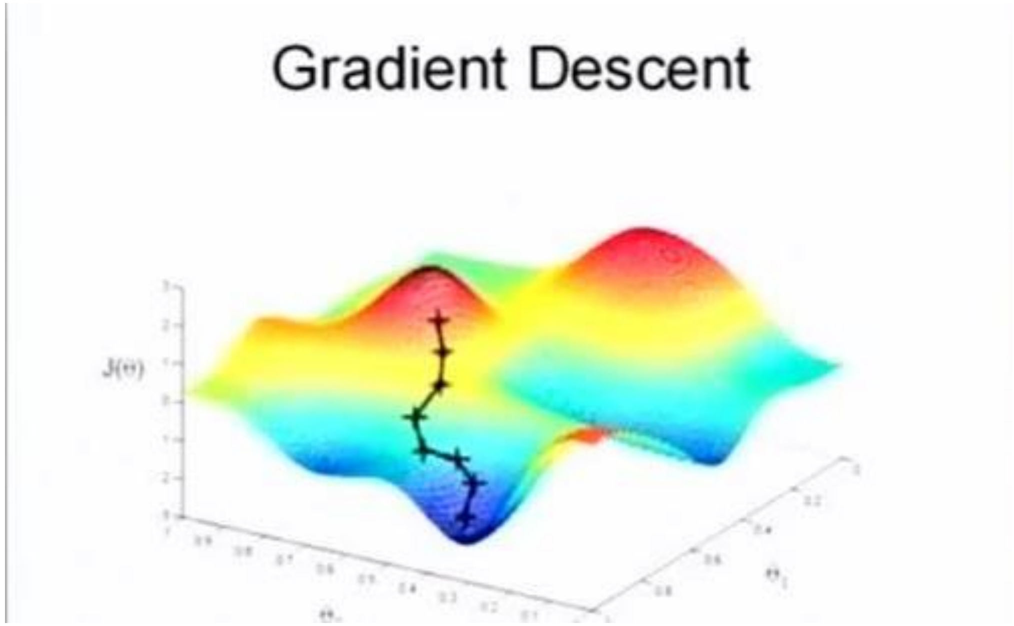


Least Squares Regression Problem as a NN



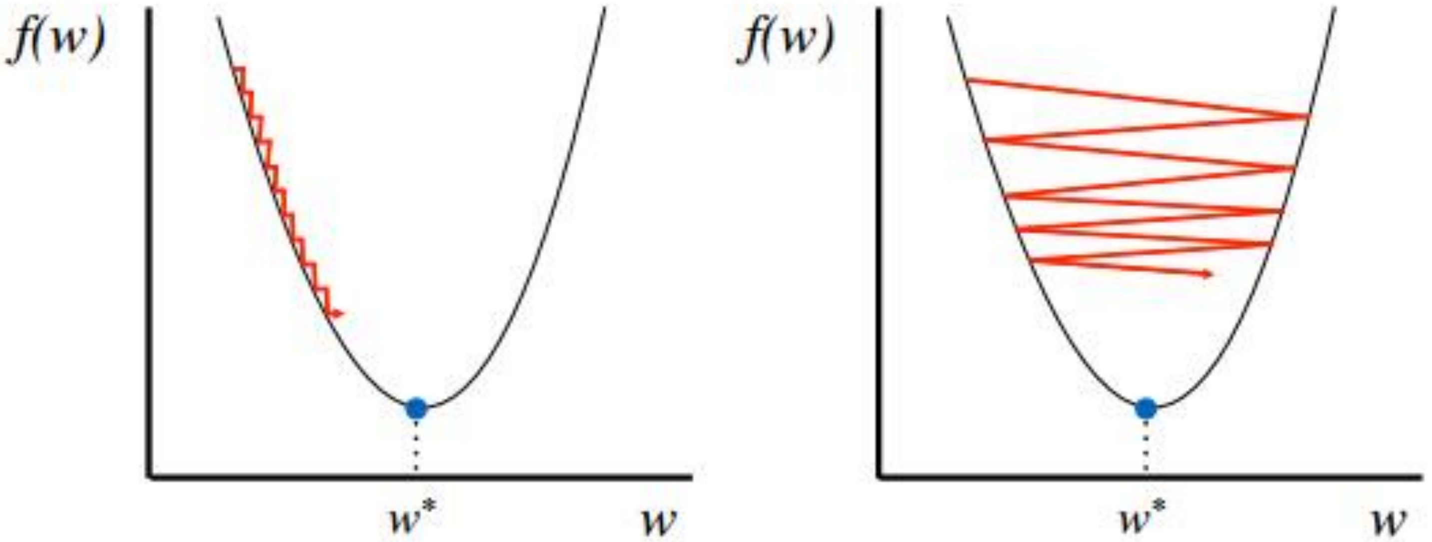
Optimization

- *Backpropagation* using *chain rule* allows us to compute gradients for all parameters of deep networks
- We use *mini-batch gradient descent* to optimize the network
- Stochastic means we use small batches to update the model instead of the whole dataset



Learning Rate

- *Learning rate* is a parameter you need to adjust wisely

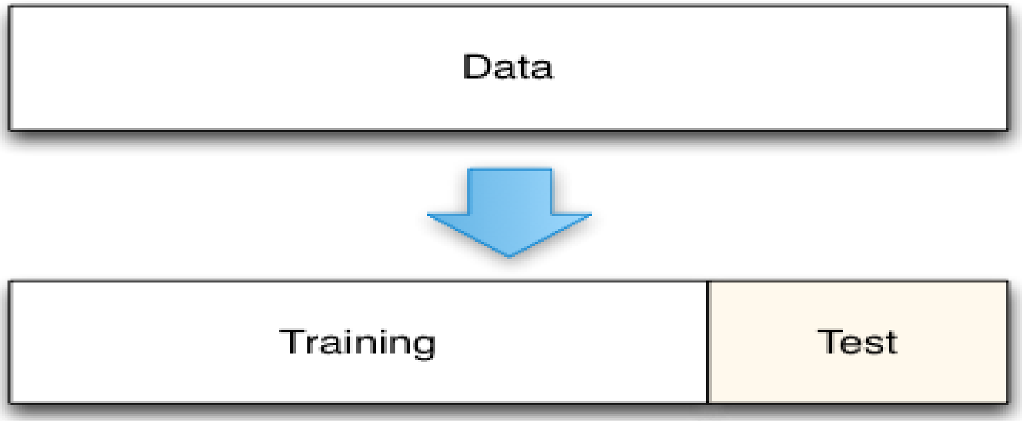


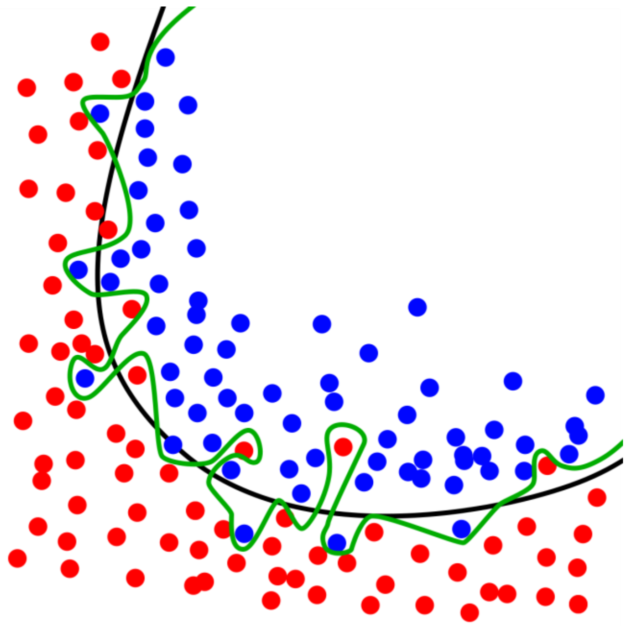
Too small: converge very slowly

Too big: overshoot and even diverge

Training Neural Networks

- Split all data to *training* and *testing data*
- Train on *training data*
- Evaluate performance on *testing data*
- Make sure you don't overfit the data



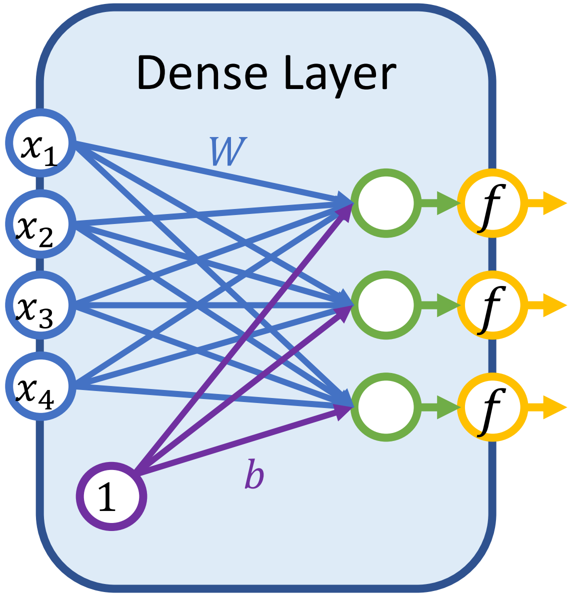


Overfitting Data
(green line)

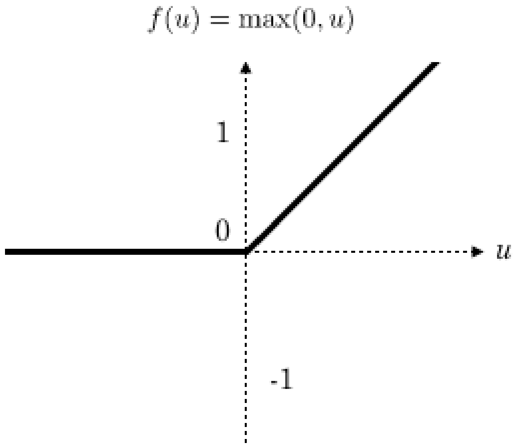
Activation Functions

- A non-linearity function between layers of Neural Networks
- The main reason why NNs are so powerful

Popular Activation Functions: Sigmoid, Tanh, **ReLU** (best in practice)



Activation function
(Non-linearity)
(Neuron activation)



ReLU Function

Metrics

- You should use *metrics* to track your performance
- Popular metrics for classification:
  - Accuracy
  - Precision
  - Recall
  - PR AUC, ROC AUC

Regularization

- Helps to prevent overfitting
- Restricting your model to continue learning the same stuff it has already learnt
- L2 Weight Regularization (sometimes used)

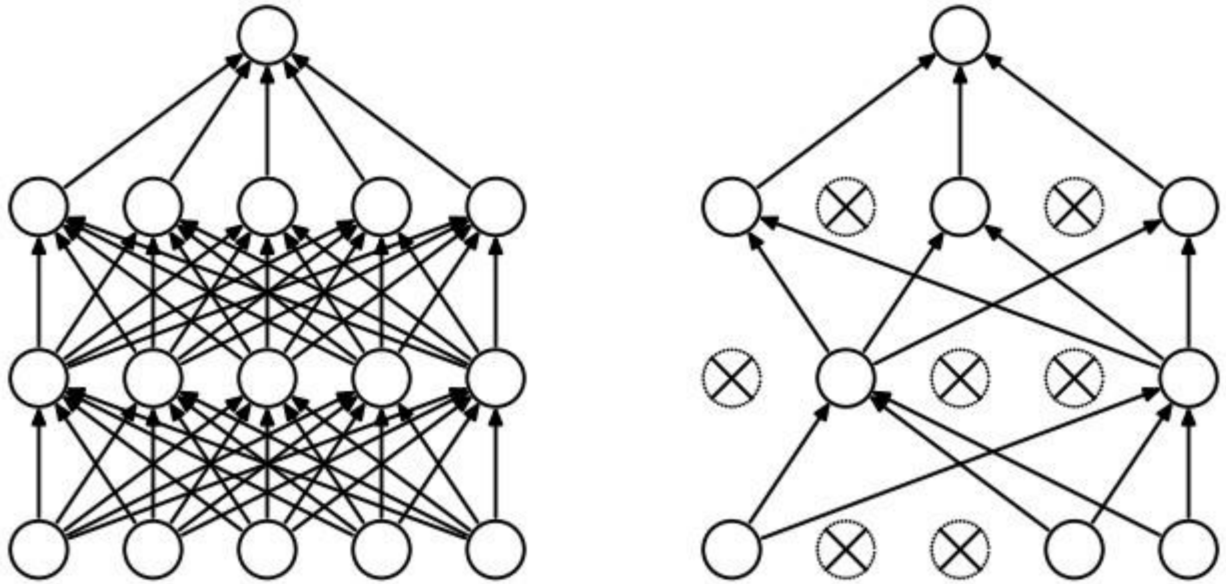
$$L(y, f(x)) + \frac{1}{2}\lambda\|w\|_2^2$$

- L1 Weight Regularization (very rarely used)

$$L(y, f(x)) + \lambda\|w\|_1$$

Dropout

- Switching off random neurons of the layer with the given probability
- Harder to train but harder to overfit as well



(a) Standard Neural Net (b) After applying dropout.

Batch Normalization

m - Batch Size

x - One output from the layer

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Output of Batch Normalization

Trained parameters via backpropagation

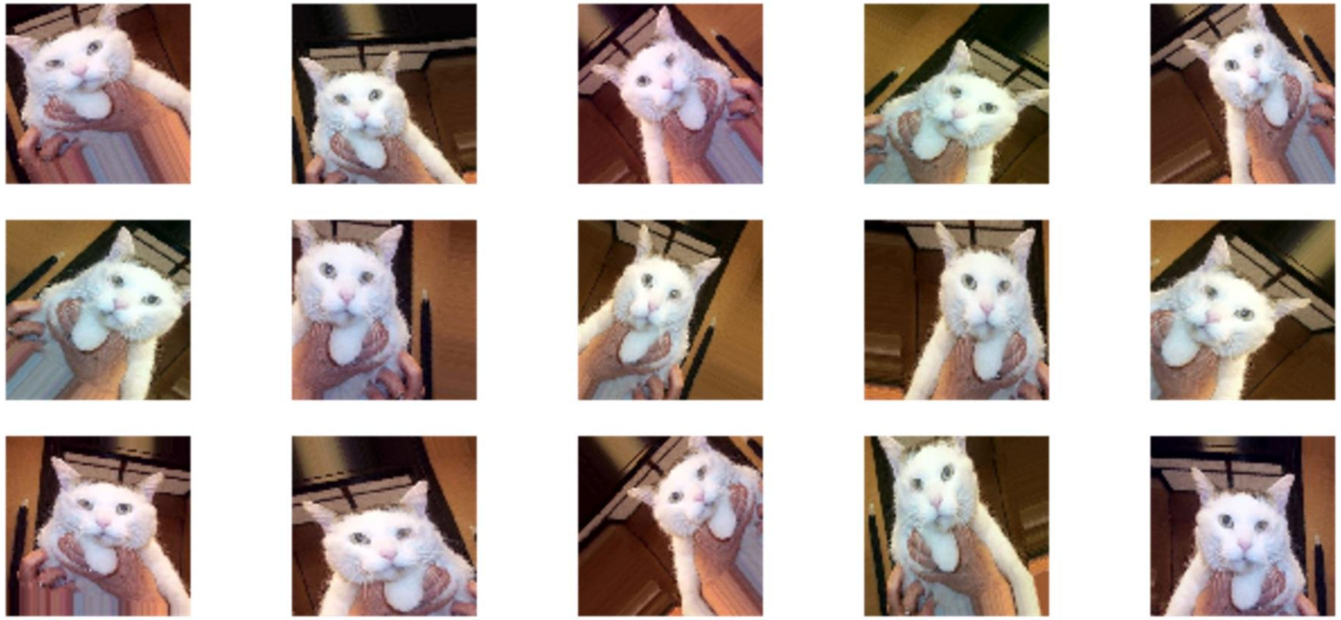
Batch Normalization

- Extremely powerful technique
- Decreases training time
- A rule of thumb: Dense -> Batch Normalization -> Activation

```
x = Dense(128)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
```

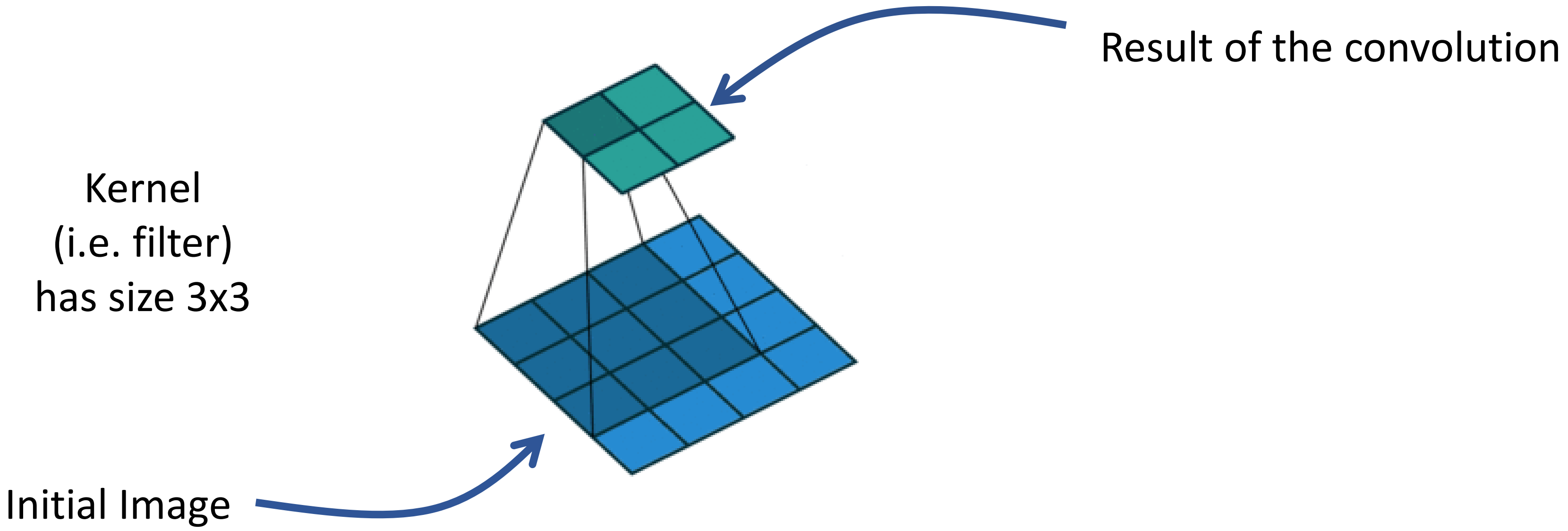
Data Augmentation

- Artificially adding more data
- Very specific to your task



Convolutional Neural Network

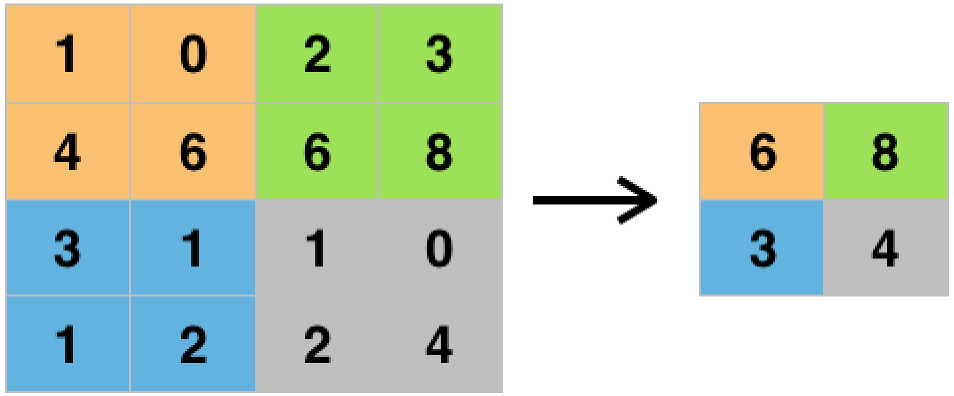
- Learning small *filters* (*kernels*) to catch useful features
- Using convolution operation to apply filter to every position on the *feature map* (or on the image)
- Adding layers of convolutions – filters learn more complex features



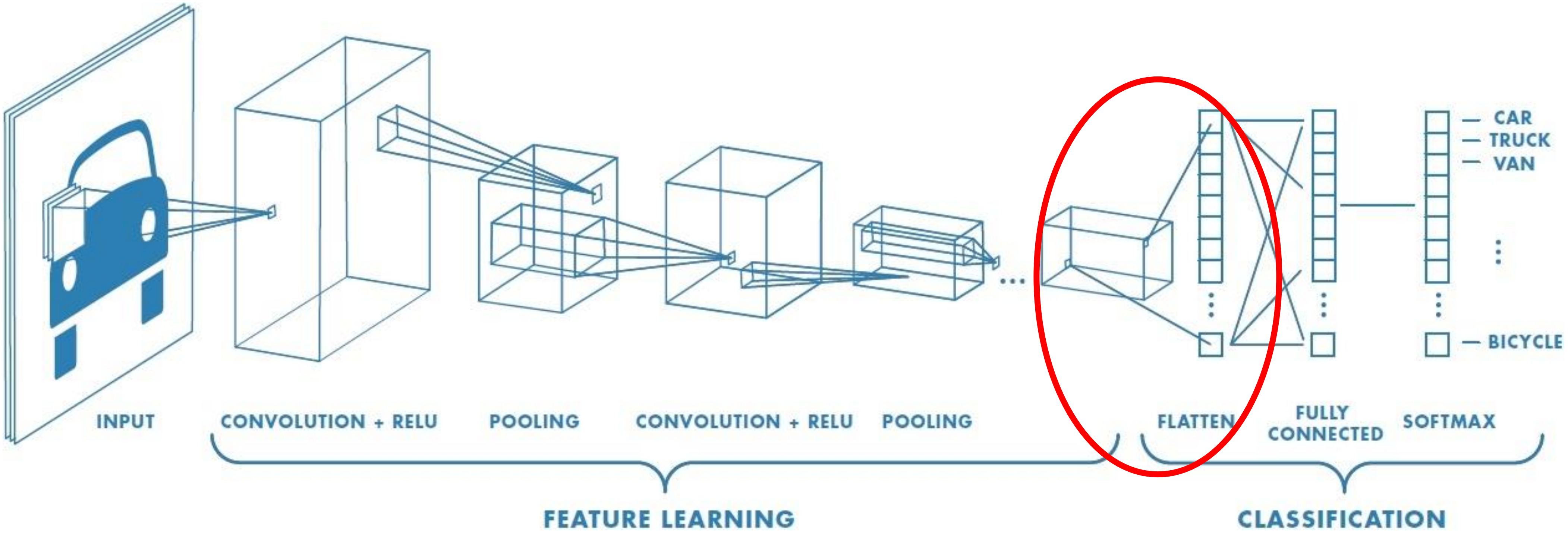
Max Pooling

- Reducing spatial size (thus amount of computations)
- Adding more non-linearity

Max Pooling window size is set to 2x2



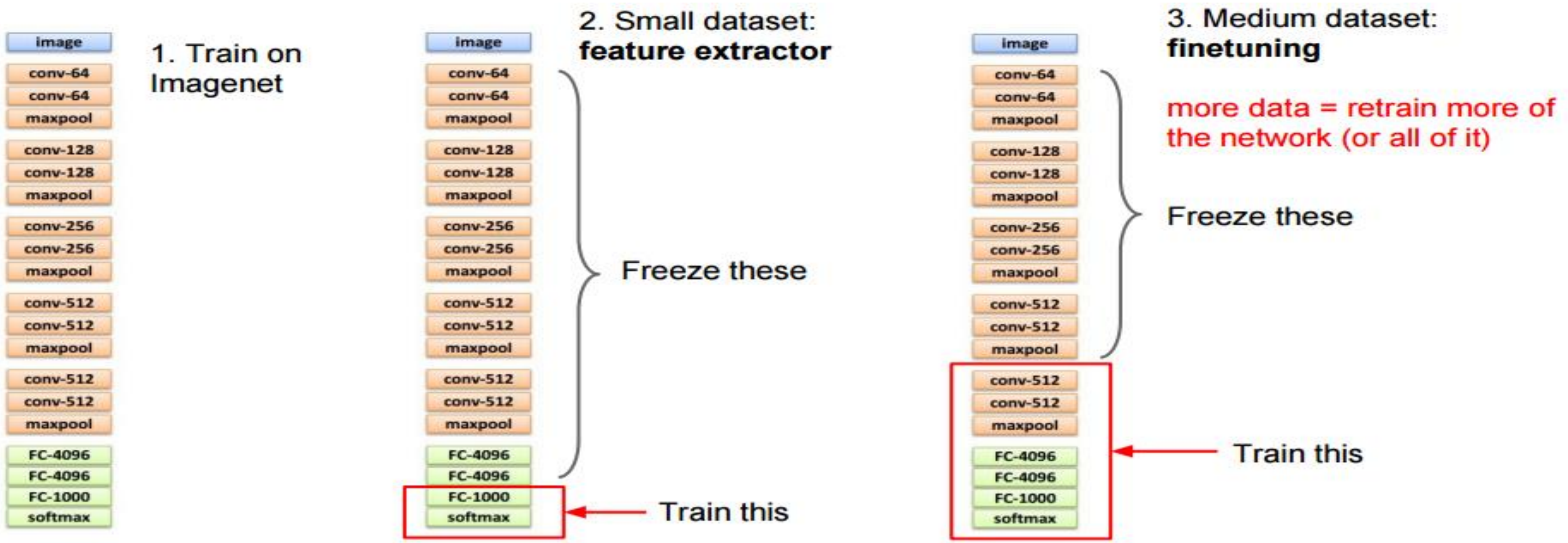
Basic Structure of CNN



Flattening the feature map –
making one shallow vector
out of the 'image'

Using Pretrained CNNs – Transfer Learning

- There are plenty of high-performance pretrained NNs
- Use pretrained weights and train from them instead of from scratch



This is it guys

To learn more we would recommend

- Attend Deep Learning course by V. Lempitsky in Term 4
- cs231n course from Stanford Univerity (http://cs231n.stanford.edu/)
- Book – “Deep Learning” by Ian Goodfellow
- Try to solve Kaggle Competitions

This is It For the Final Lecture