Deep Learning

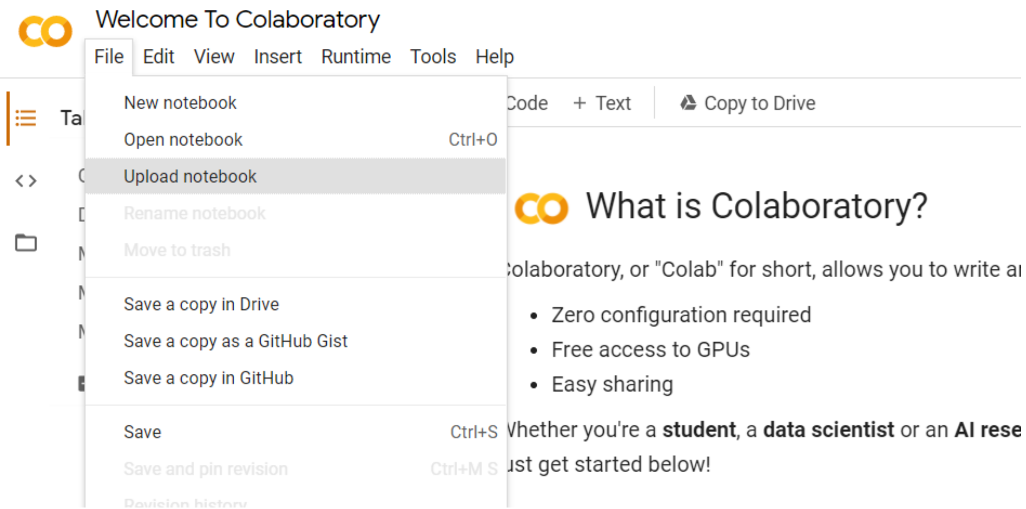
Yan-Fu Kuo
Dept. of Biomechatronics Engineering
National Taiwan University

→

Google Colab

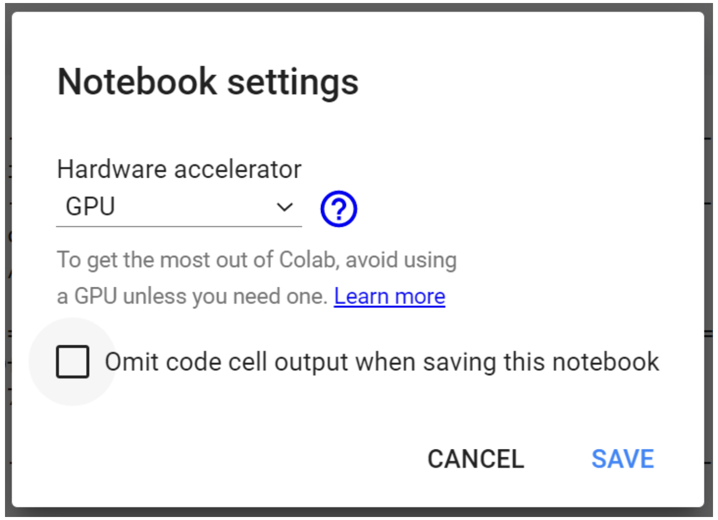
Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs.

1 **Upload 12DeepLearning.ipynb to Google Colab**



2 **Setting Free GPU**

Edit > Notebook settings >Change runtime type and select GPU as Hardware accelerator.



Contents

01 PyTorch and Tensor

PyTorch

PyTorch

A replacement for NumPy to use the power of GPUs

A deep learning research platform that provides maximum flexibility and speed

Pytorch -> Get Started

Note: This project was built and tested using PyTorch version 1.12.0+cu101

GPU Acceleration

Tensors

Tensors are similar to NumPy's ndarrays, with the addition being that Tensors can also be used on a GPU to accelerate computing.

```python
import torch

x = torch.tensor([5.5, 3])
```

```
tensor([5.5000, 3.0000])
```

```python
import numpy as np

y = np.array([5.5, 3])
```

```
[5.5000, 3.0000]
```

Tensors | Operations

Arithmetic

```
x = torch.linspace(1, 10, 10)
y = torch.ones((1, 10))

x + y
```

```
tensor([[ 2., 3., 4., 5., 6., 7., 8., 9., 10., 11.]])
```

Slicing

```
x[:5]
```

```
tensor([1., 2., 3., 4., 5.])
```

Resizing

```
x = torch.randn(4, 4)
y = x.view(16)
z = x.view(-1, 8)
print(x.shape, y.shape, z.shape)
```

```
torch.Size([4, 4]) torch.Size([16]) torch.Size([2, 8])
```

.item()

```
x = torch.randn(1)
print(x.item())
```

```
-0.408441
```

Tensors | Numpy Bridge

NumPy Array → Torch Tensor

```
a = np.ones(5)
b = torch.from_numpy(a)
print(a)
print(b)
```

```
[1. 1. 1. 1. 1.]
tensor([1., 1., 1., 1., 1.],
       dtype=torch.float64)
```

Torch Tensor → NumPy Array

```
a = torch.ones(5)
print(a)
b = a.numpy()
print(b)
```

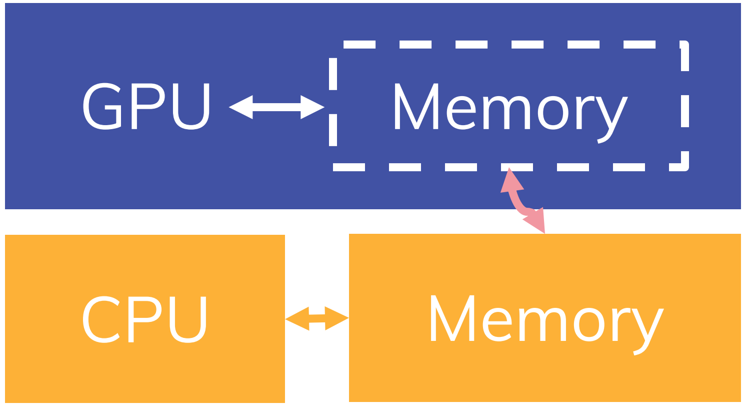
```
tensor([1., 1., 1., 1., 1.])
[1., 1., 1., 1., 1.]
```

Tensors | CUDA

```python
x = torch.randn(1)
if torch.cuda.is_available():
  device = torch.device("cuda")
  y = torch.ones_like(x, device=device)
  x = x.to(device)
  z = x + y
  print(z)
  print(z.to("cpu", torch.double))
```

```
tensor([-0.5981], device='cuda:0')
tensor([-0.5981], dtype=torch.float64)
```

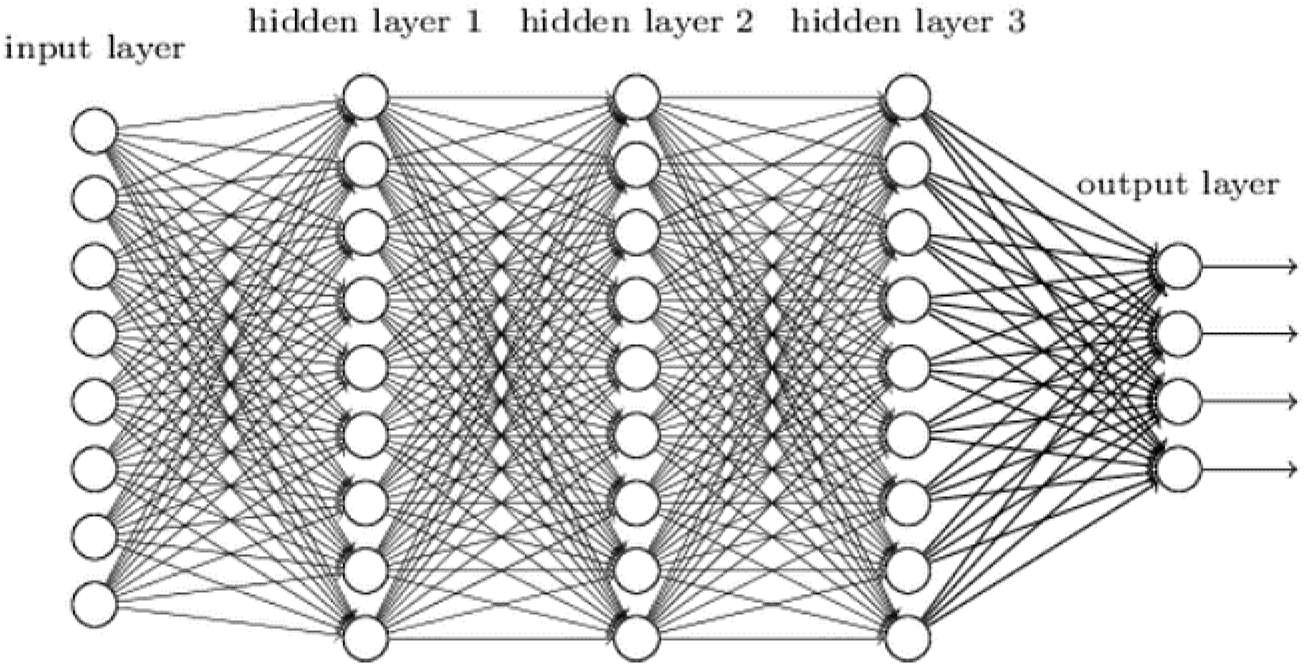
Tensors can be moved onto any device using the `.to` method.



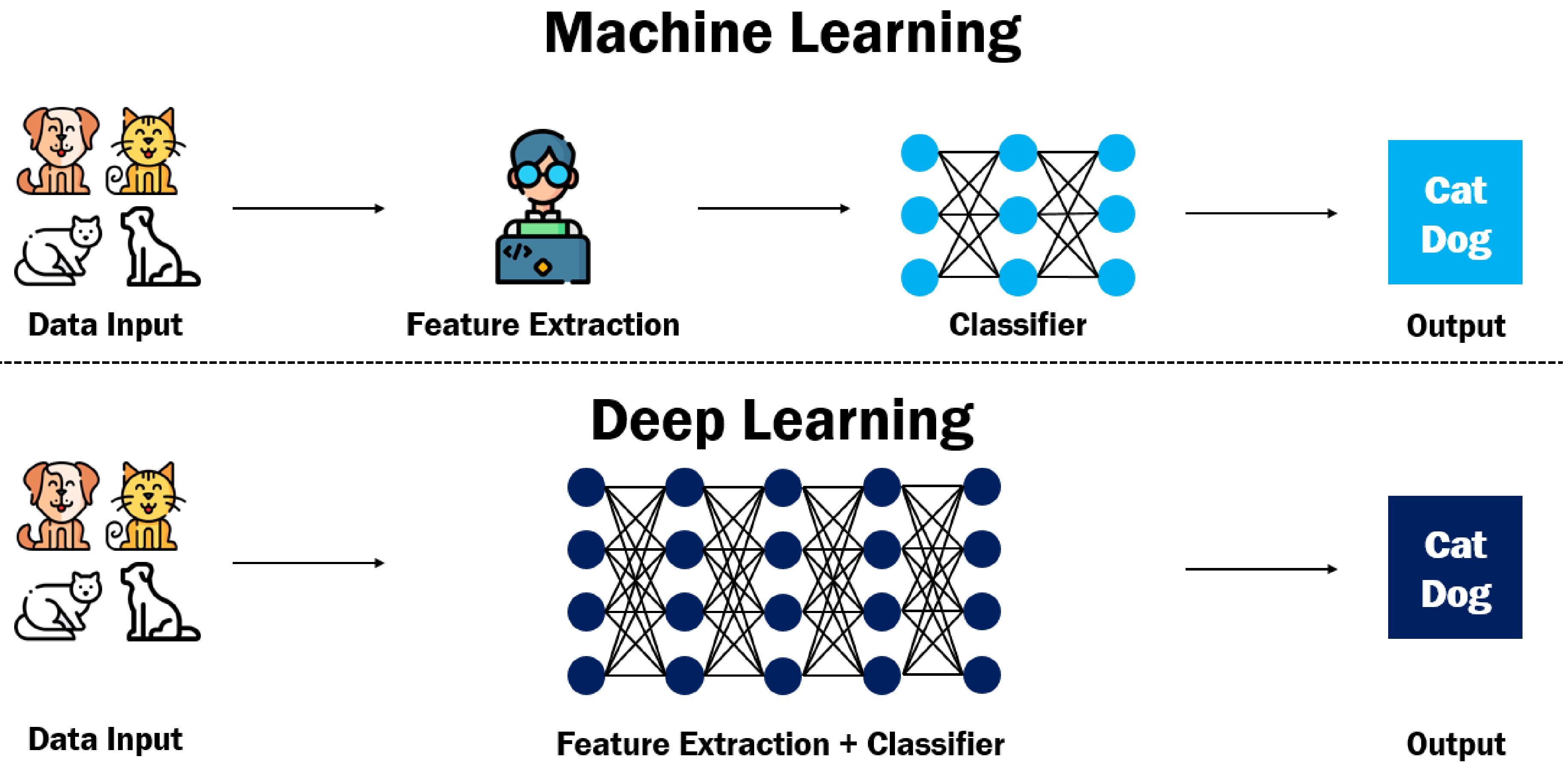
02 Artificial Neural Networks

Deep Learning Models

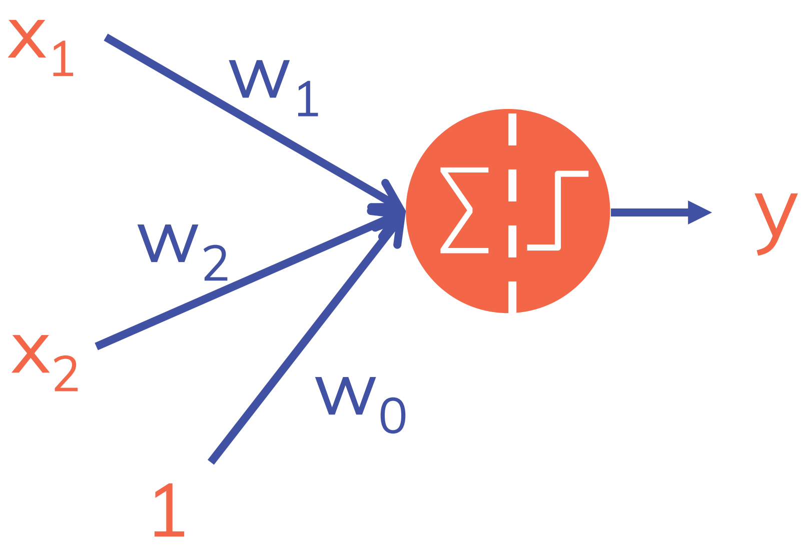
Usually refers to neural networks with large numbers in layers and neurons.



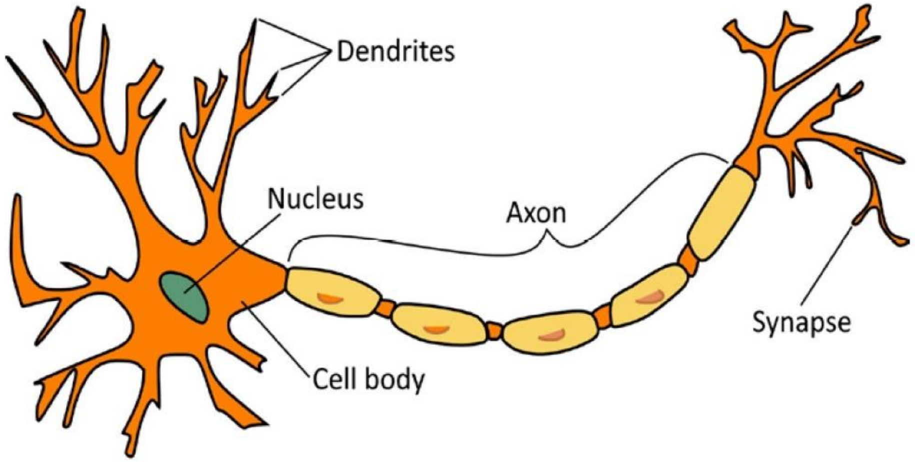
Why Deep Learning?



Perceptron (Neuron)

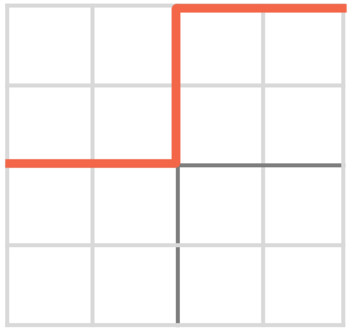


$$y = \begin{cases} 1, & \sum > \text{threshold} \\ 0, & \sum < \text{threshold} \end{cases}$$



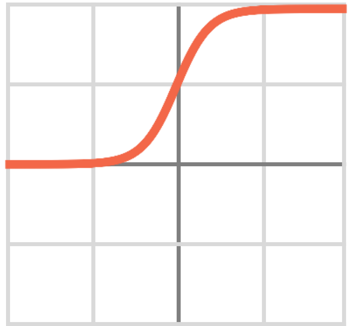
Activation Function

Step Function



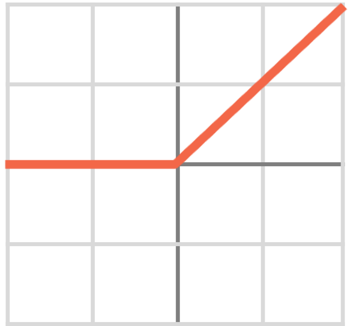
$$f(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

Sigmoid



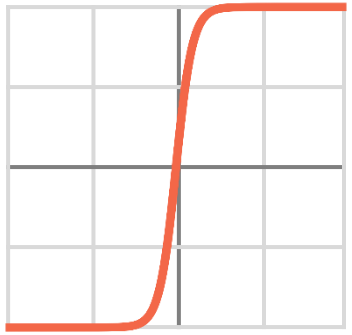
$$f(x) = \frac{1}{1 + e^{-x}}$$

ReLU



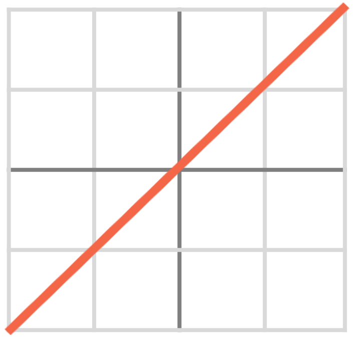
$$f(x) = \max(0, x)$$

TanH



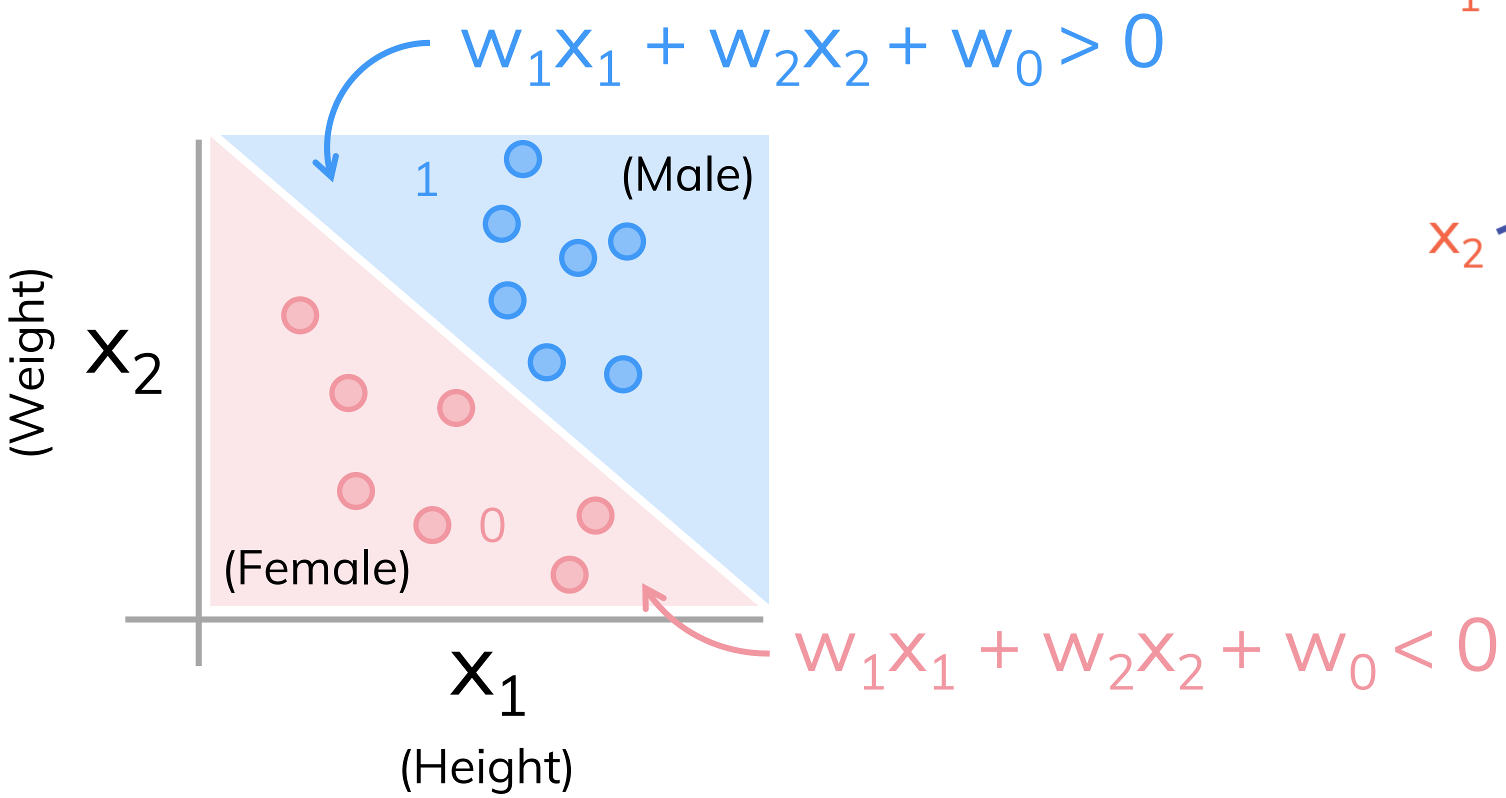
$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

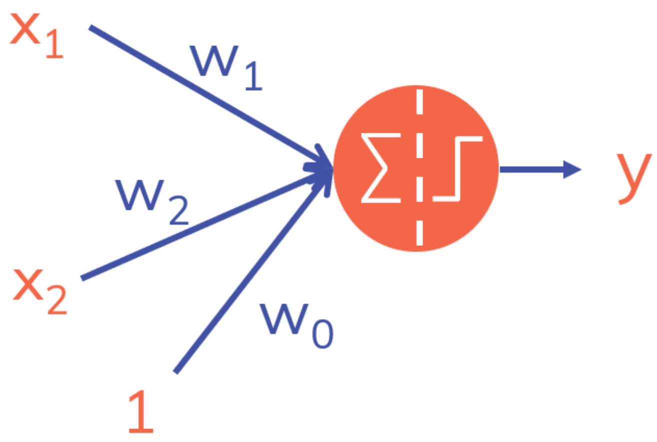
Linear



$$f(x) = x$$

Classification Problem



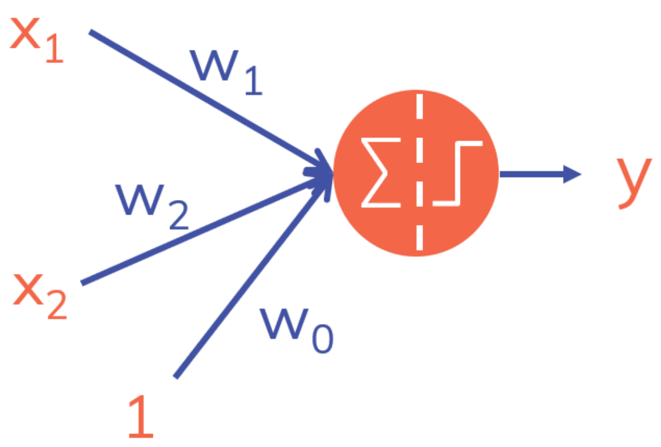


Matrix Multiplication

$$\begin{bmatrix} 1 & x_1 & x_2 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = X^{\mathrm{T}}W$$

(1, 3) (3, 1) (1, 1)

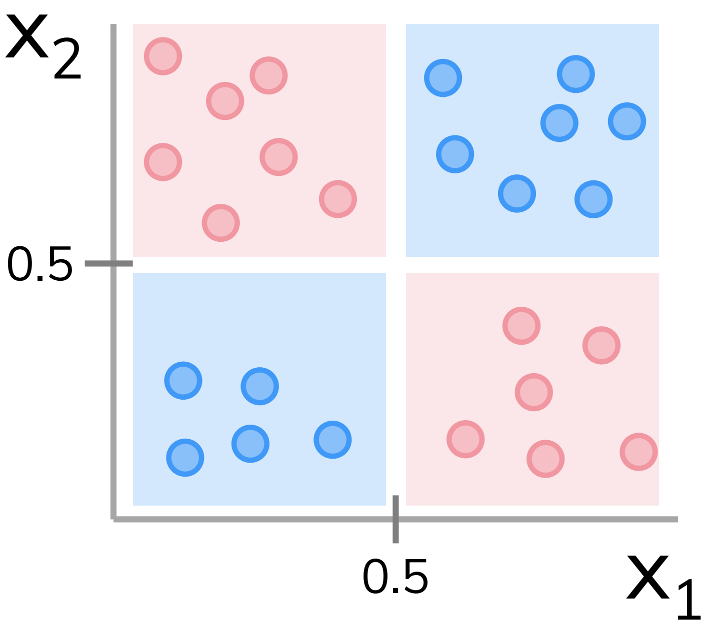
Equal!

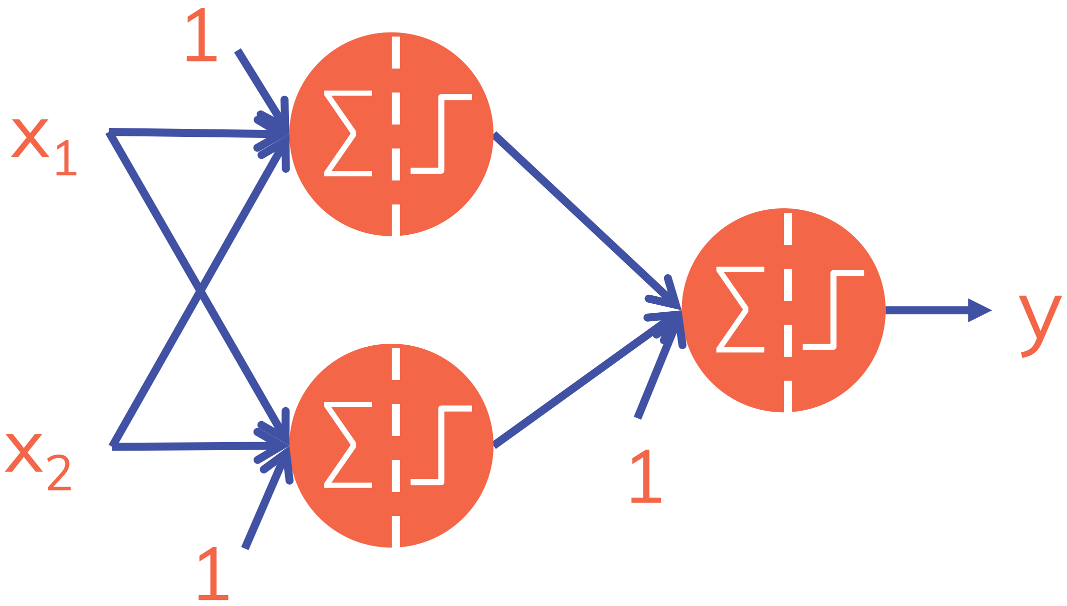


$$\begin{bmatrix} 1 & x_1^1 & x_2^1 \\ 1 & x_1^2 & x_2^2 \\ 1 & x_1^3 & x_2^3 \\ 1 & x_1^4 & x_2^4 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = X^{\mathrm{T}}W$$

(4, 3) (3, 1) (4, 1)

Multilayer Perceptron



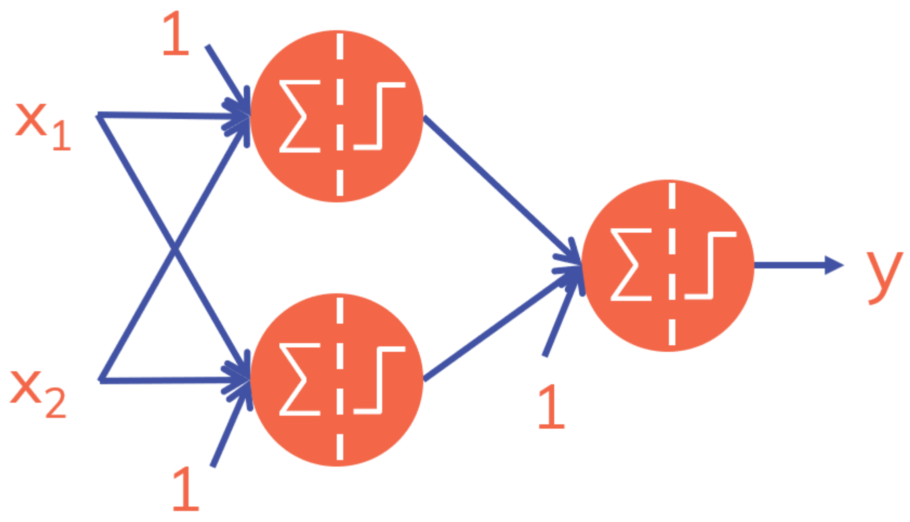


Matrix Multiplication

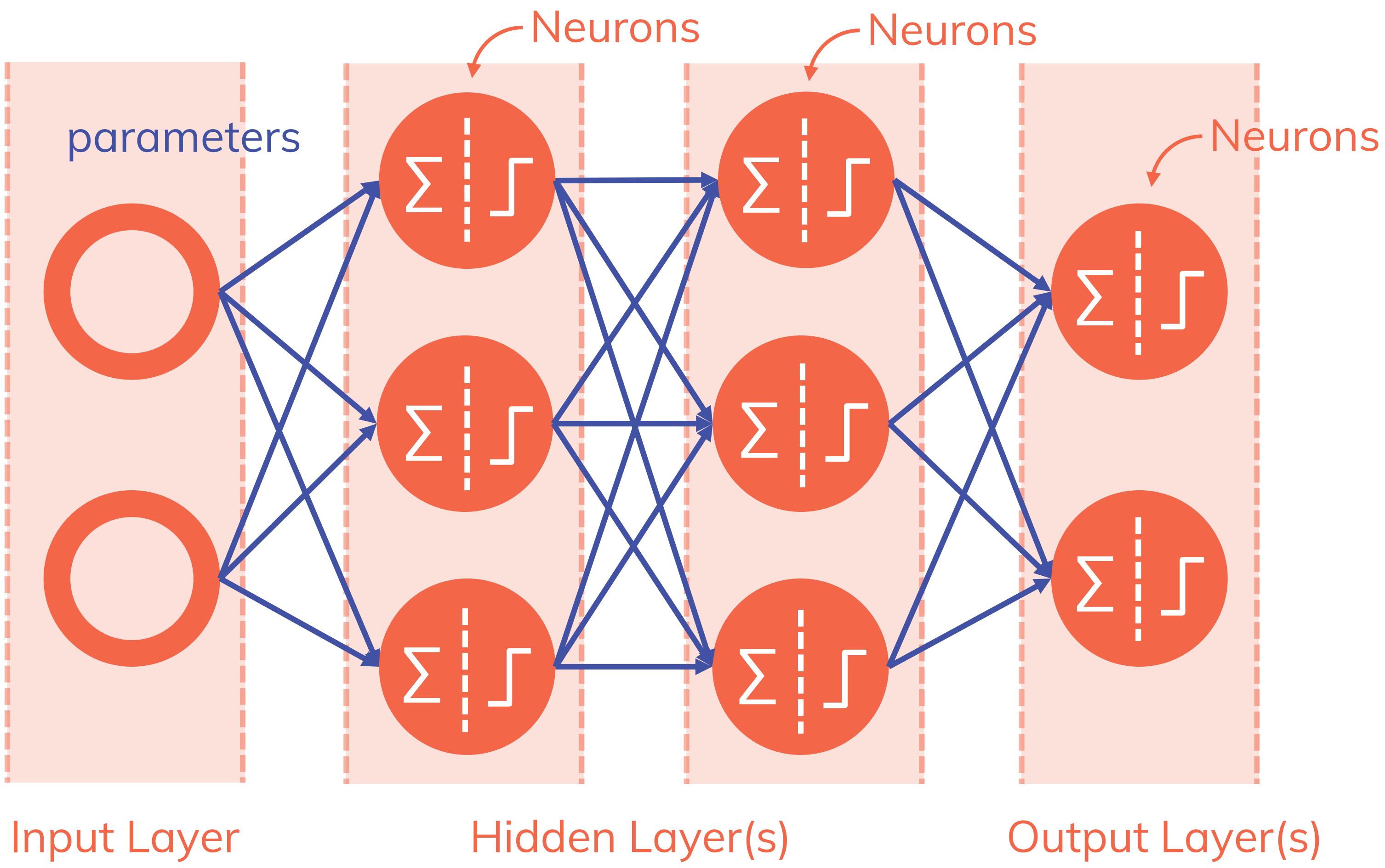
$$\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \end{bmatrix} \begin{bmatrix} w_{11}^{(2)} \\ w_{21}^{(2)} \end{bmatrix} = XW^{(1)}W^{(2)}$$

(1, 2) (2, 2) (2, 1)

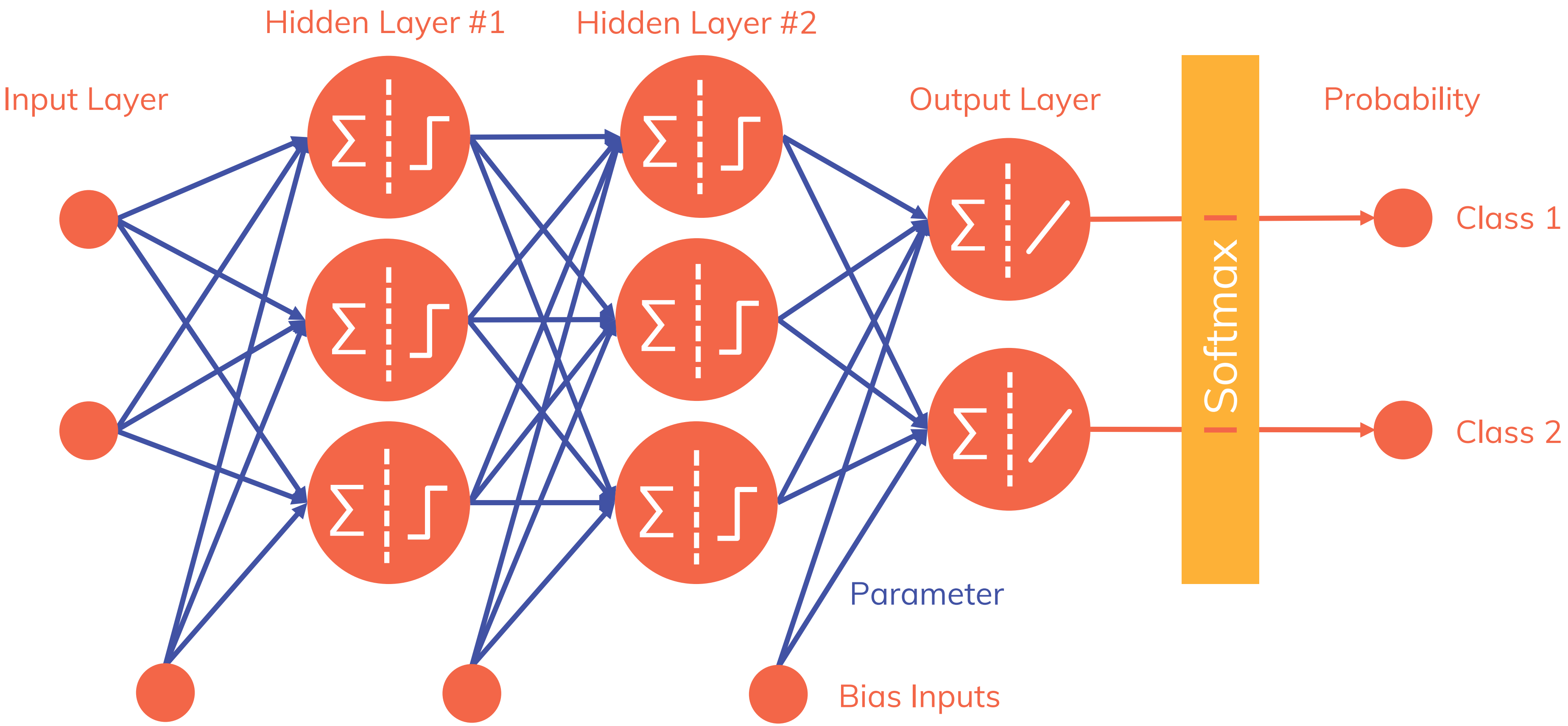
Equal! Equal!



Artificial Neural Network

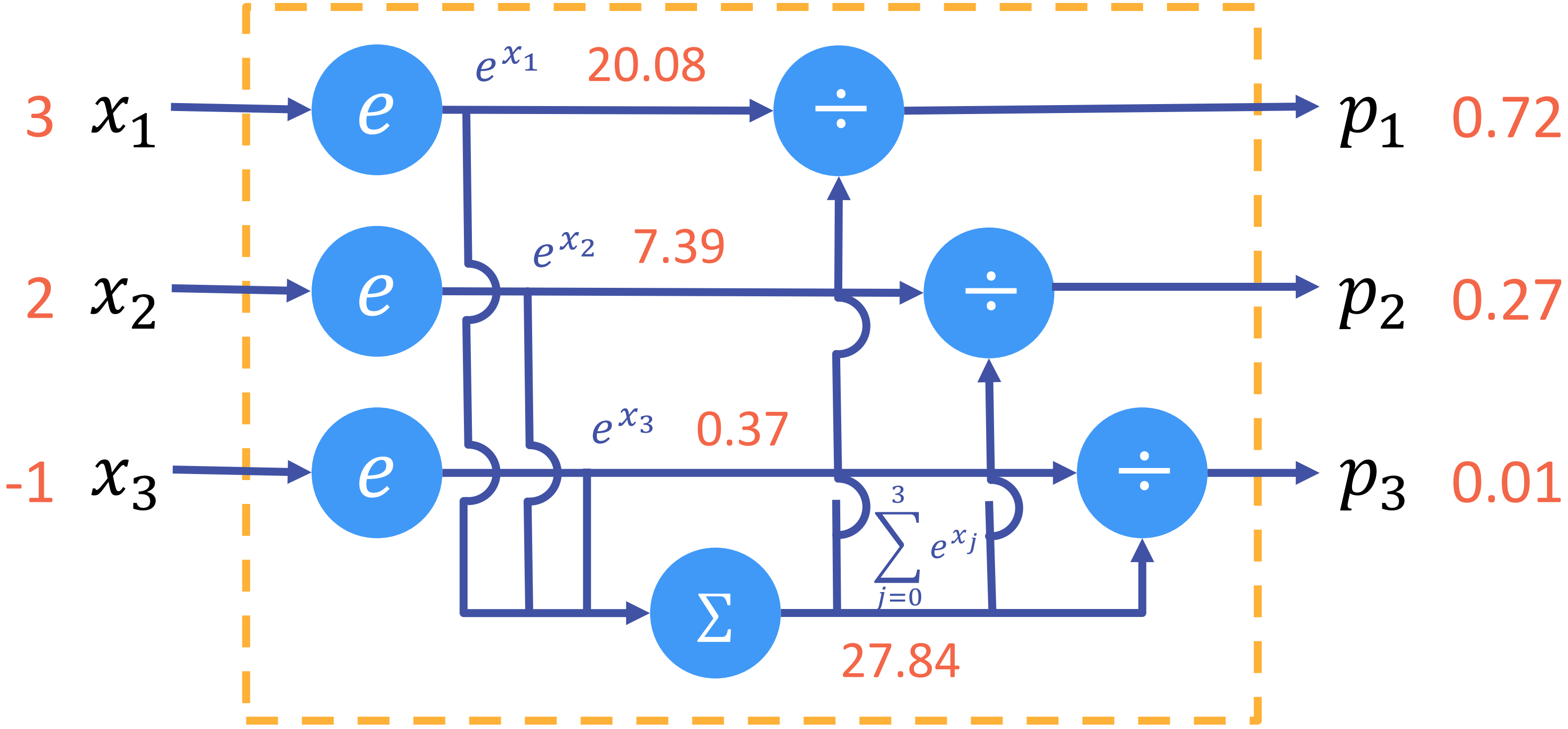


Artificial Neural Network (completed)



Softmax

$$softmax(x_i) = \frac{exp(x_i)}{\sum_{j=0}^{n} exp(x_j)}$$



03 How to Train A Model

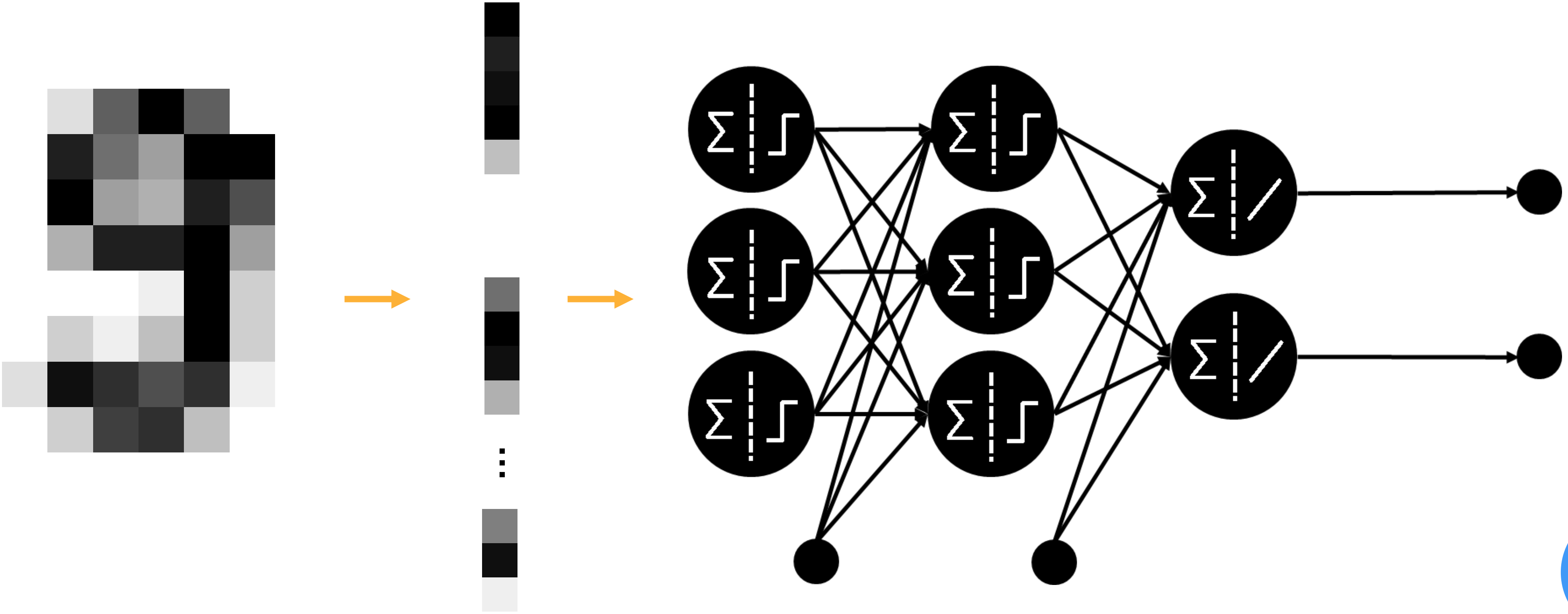
Procedure of Training a DL Model

- Step 1 | Arrange data into in “tensor” format
- Step 2 | Define the network
- Step 3 | Define a loss function and an optimizer
- Step 4 | Train the model
- Step 5 | Test the model

MNIST Dataset

Handwritten digits

Define a Model in PyTorch



Error Rate (1-Accuracy)

Model 1 **Error rate = 0.25**

		Dog	Cat	Snake
Image 1	**Cat**	0.3	0.4	0.3
Image 2	**Dog**	0.5	0.3	0.2
Image 3	**Cat**	0.6	0.1	0.3
Image 4	**Snake**	0.3	0.2	0.5

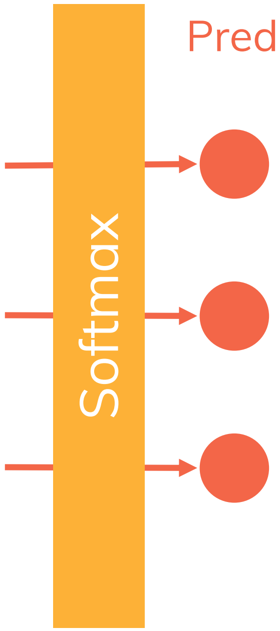
Model 2 **Error rate = 0.25**

		Dog	Cat	Snake
Image 1	**Cat**	0.1	0.8	0.1
Image 2	**Dog**	0.9	0.1	0.0
Image 3	**Cat**	0.4	0.3	0.3
Image 4	**Snake**	0.2	0.0	0.8

Mean Square Error

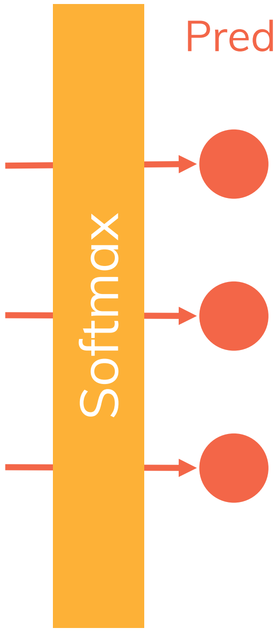
$$MSE = \frac{1}{N}\sum_{i}(y_i - \hat{y}_i)^2$$

Model 1 **MSE = 0.18**



	Predicted Probability	True Label
Dog	0.3	0
Cat	0.4	1
Snake	0.3	0

Model 2 **MSE = 0.02**



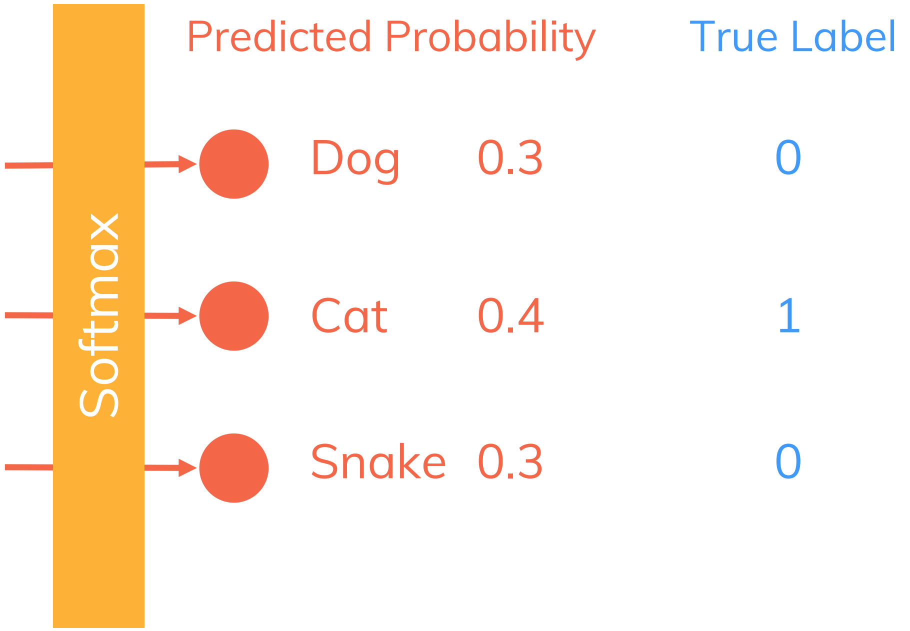
	Predicted Probability	True Label
Dog	0.1	0
Cat	0.8	1
Snake	0.1	0

One hot encoding

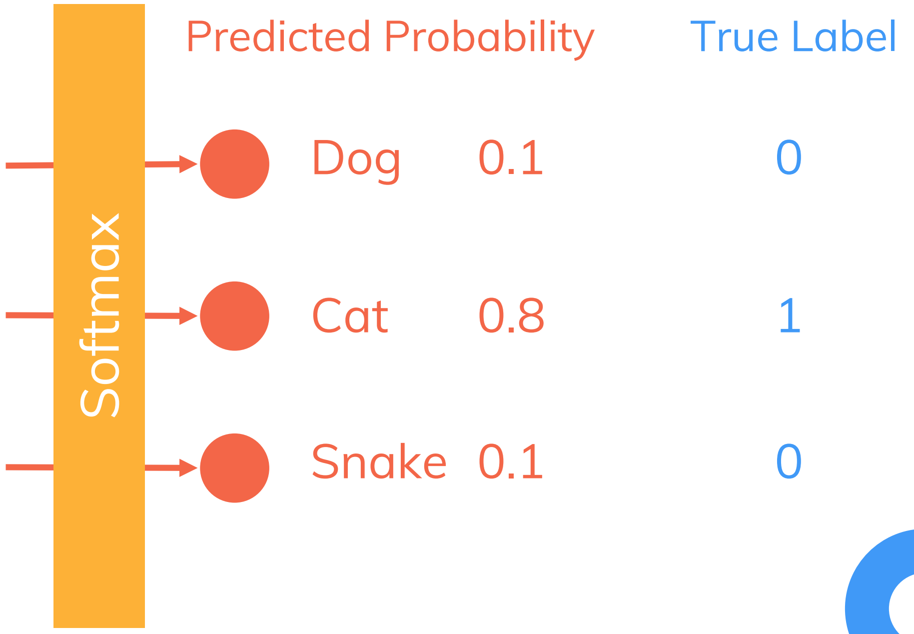
Cross-Entropy

$$CE = \sum_{c=1}^{C}\sum_{i=1}^{n} -y_{c,i} log_2(p_{c,i})$$

Model 1 **CE = 0.22**



Model 2 **CE = 0.09**



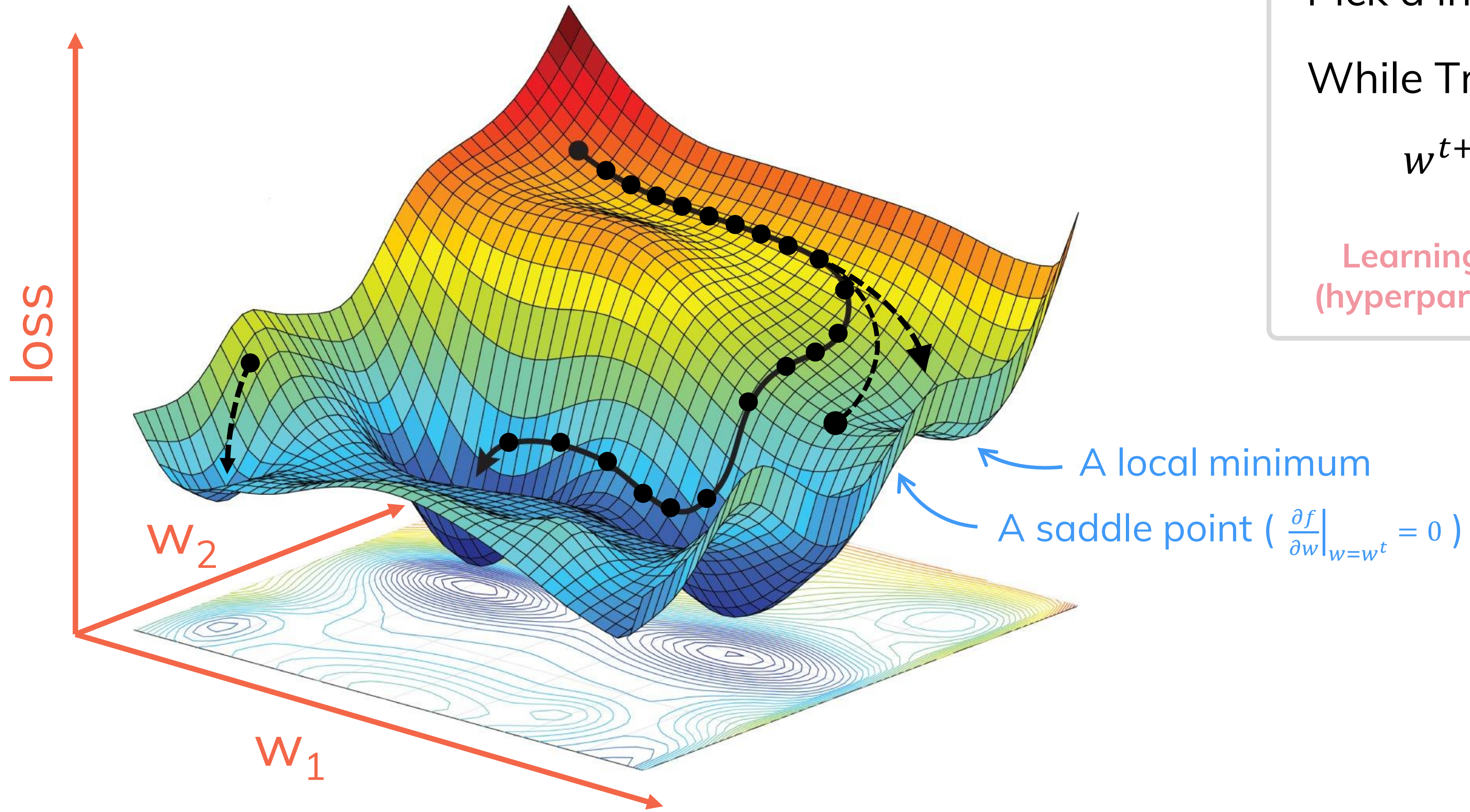
Solving the Loss Function

Typically we estimate a parameter by minimizing the loss function, and using as the estimator the parameter which minimizes the loss.

$$\min_{w} loss(w)$$

Usually (but not always) the way to solve the loss function is to differentiate it and equate it to zero.

Gradient Descent



Pick a initial value w^0 (randomly)

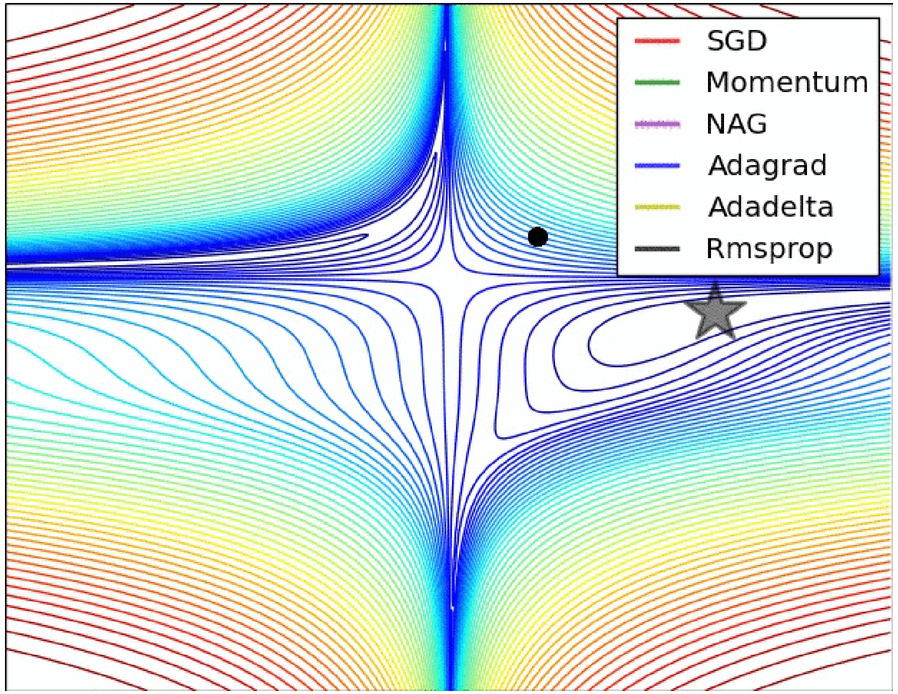
While True:

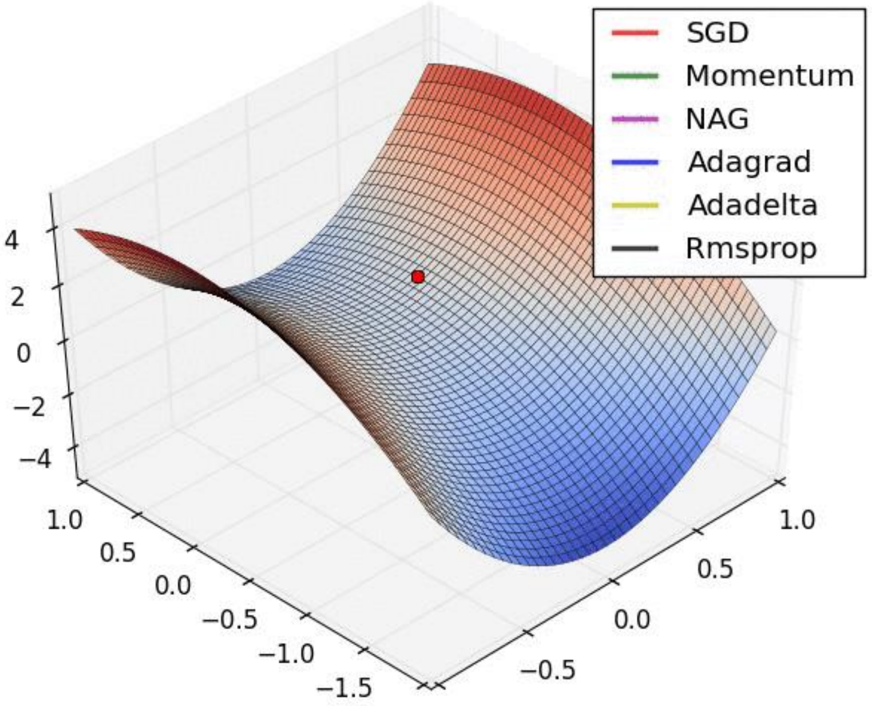
$$w^{t+1} \leftarrow w^t - \eta \left.\frac{\partial f}{\partial w}\right|_{w=w^t}$$

Learning Rate
(hyperparameter)

Gradient

Optimizers



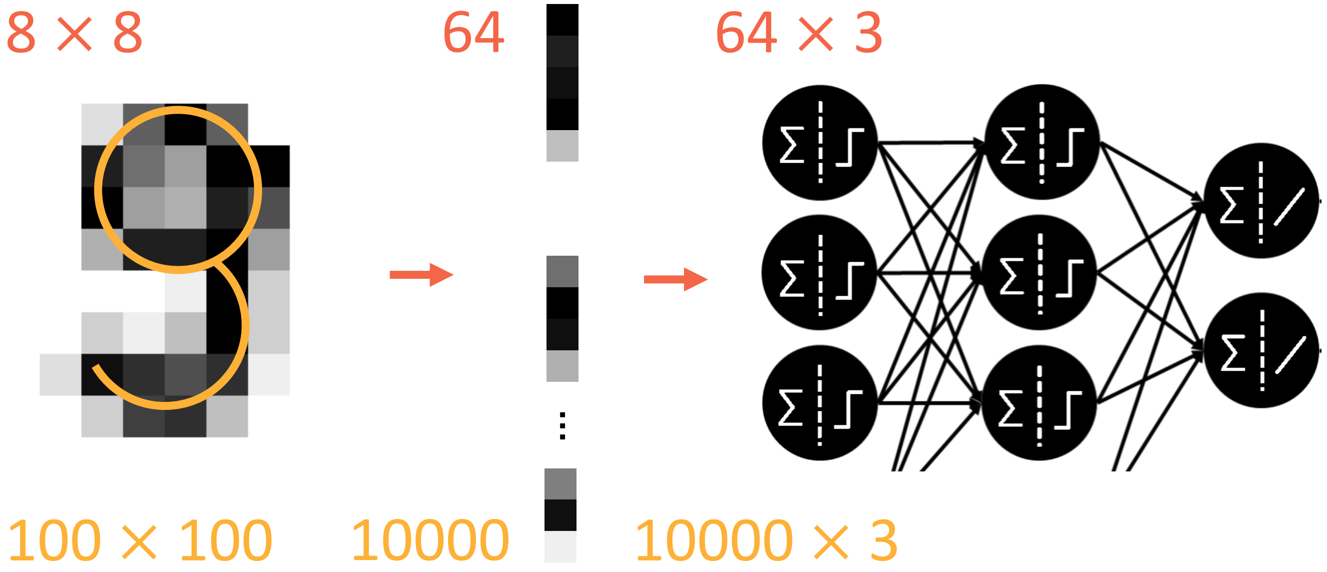


http://louistiao.me/notes/visualizing-and-animating-optimization-algorithms-with-matplotlib/

04 Convolutional Neural Networks

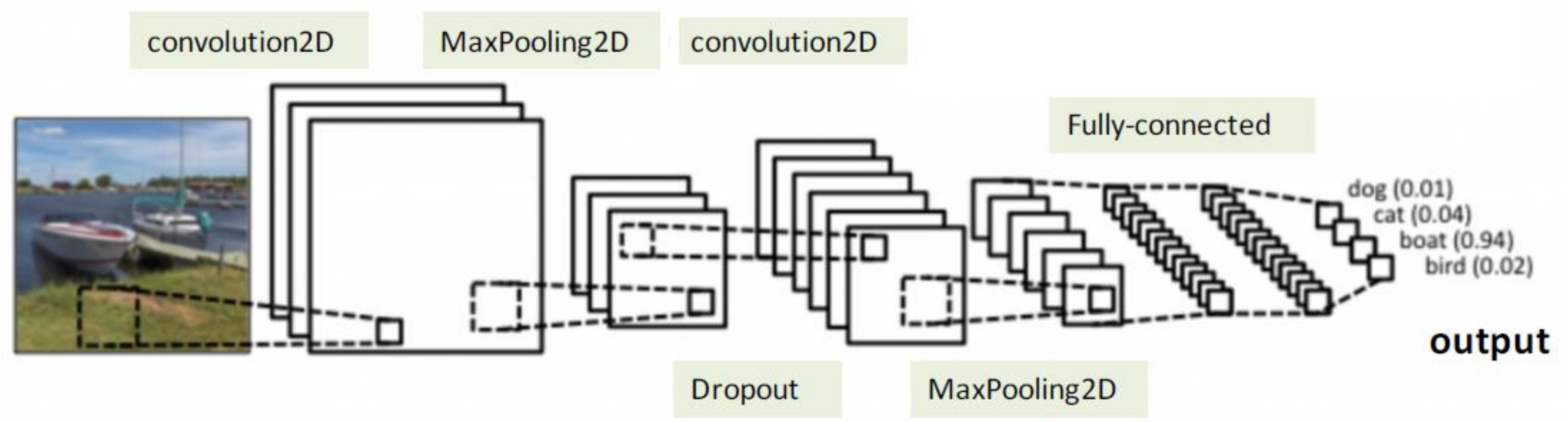
Challenges with ANN

- ANN loses the spatial features of an image.
- The number of trainable parameters increases drastically with an increase in the size of the image.

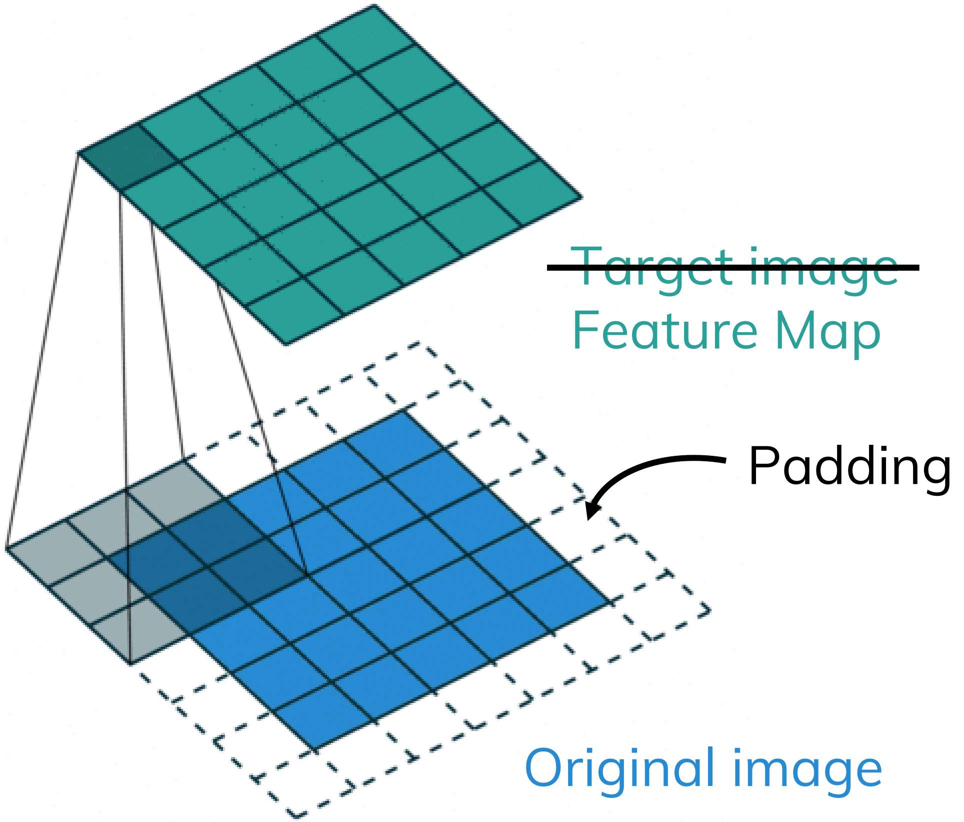


Convolutional Neural Networks

Convolution + Artificial Neural Network



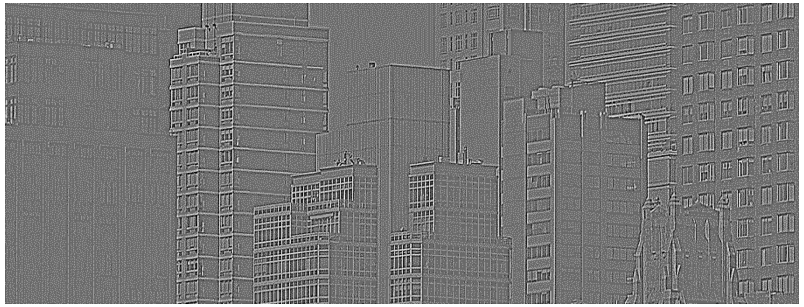
Convolution (a review)



Laplacian Filter

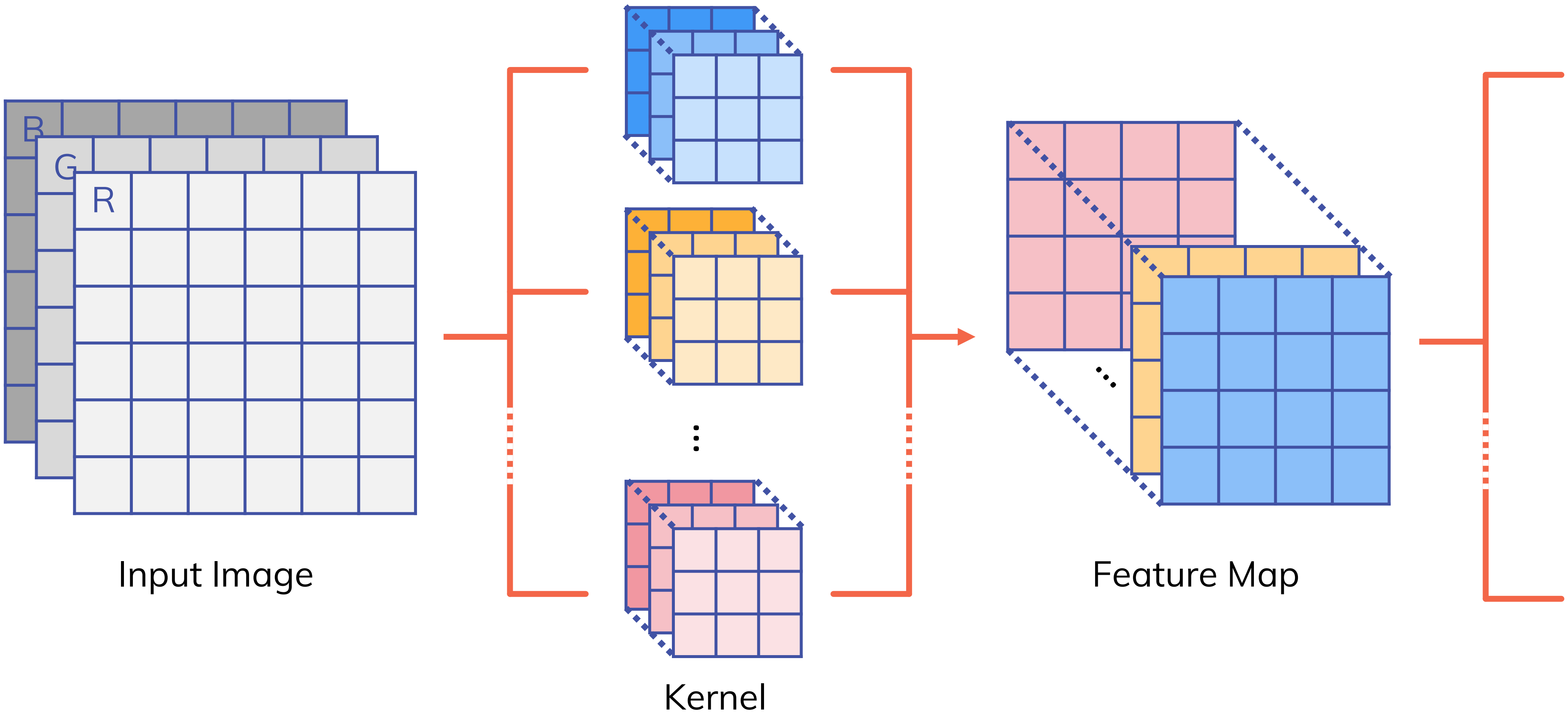
0	1	0
1	-4	1
0	1	0

1	1	1
1	-8	1
1	1	1



CNN learns the filters by itself

Convolution Layer



Max Pooling

Max pooling is done by applying a max filter to (usually) non-overlapping subregions of the initial representation.

146	155	144	130	145	151
142	153	150	128	131	151
131	141	142	130	128	148
122	123	125	127	130	135
130	123	107	118	150	154
127	120	125	143	153	161

→

155	150	151
141	142	148
130	143	161

It reduces the computational cost by reducing the number of parameters to learn and provides basic translation invariance to the internal representation.

This layer requires no training!

Convolutional Layer in PyTorch

```
import torch.nn as nn
import torch.nn.functional as F

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)
```

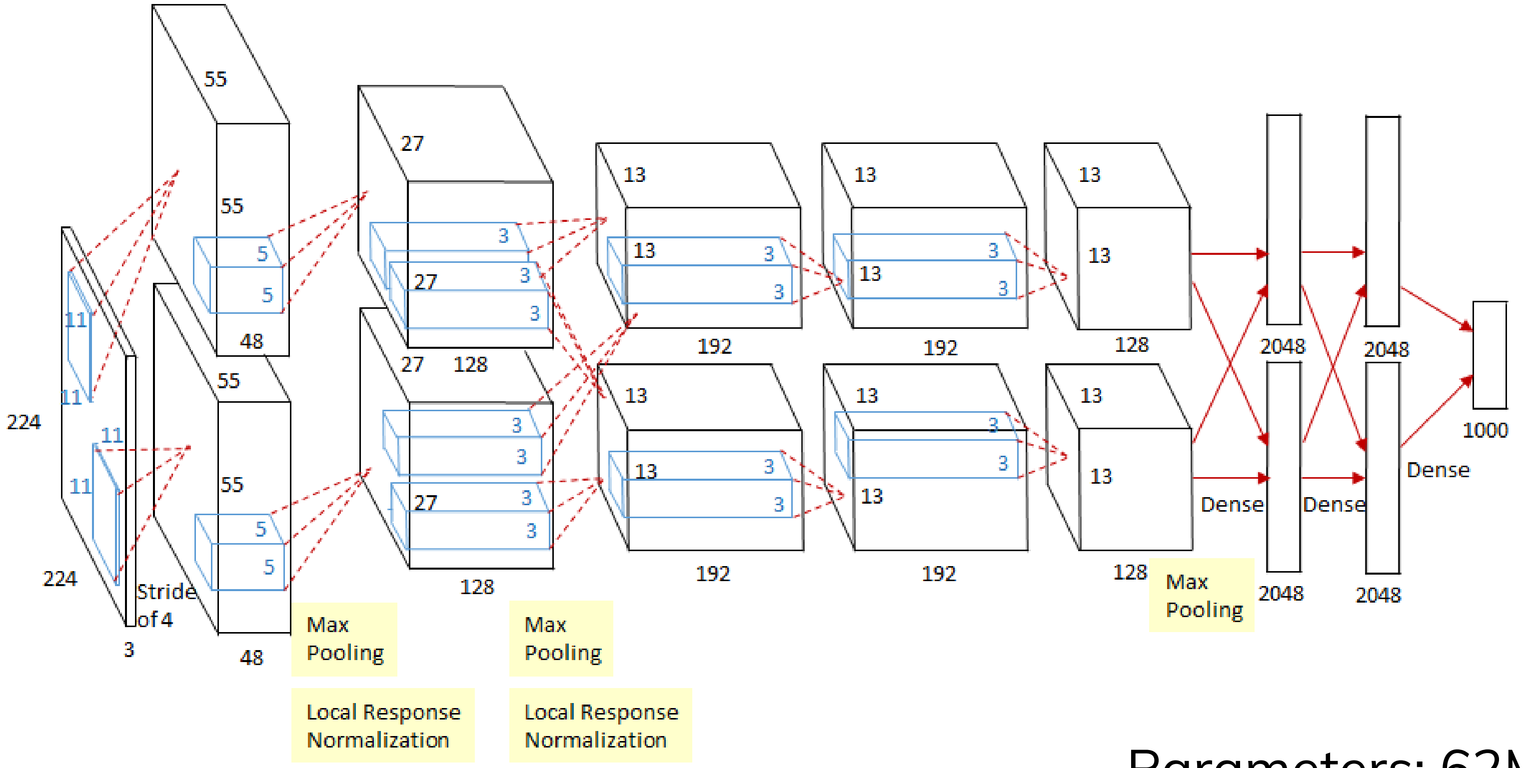
Convolutional Layer in PyTorch

```python
def forward(self, x):
    x = self.pool(F.relu(self.conv1(x)))
    x = self.pool(F.relu(self.conv2(x)))
    x = x.view(-1, 16 * 5 * 5)
    x = F.relu(self.fc1(x))
    x = F.relu(self.fc2(x))
    x = self.fc3(x)
    return x
```

torch.nn

Deep neural network (DNN) models can be composed just like building LEGO buildings

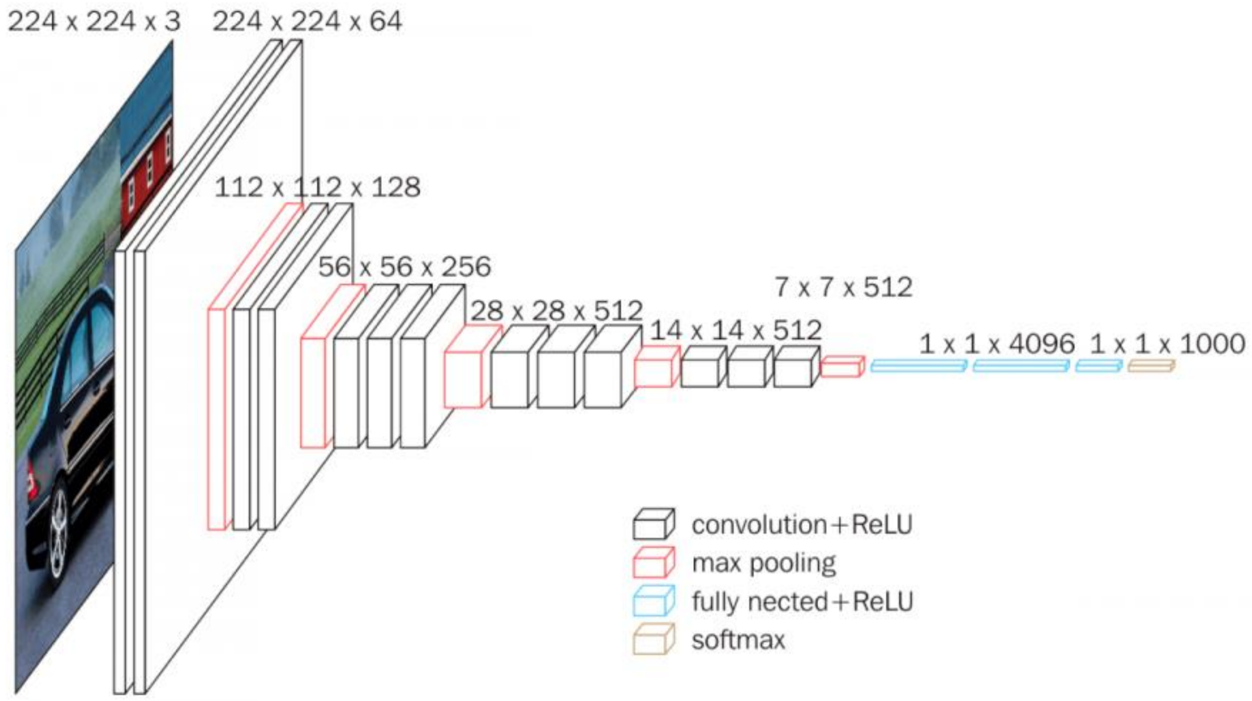
CNN Architectures | AlexNet



Parameters: 62M

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

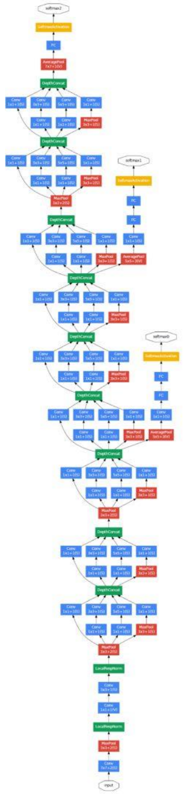
CNN Architectures | VGG



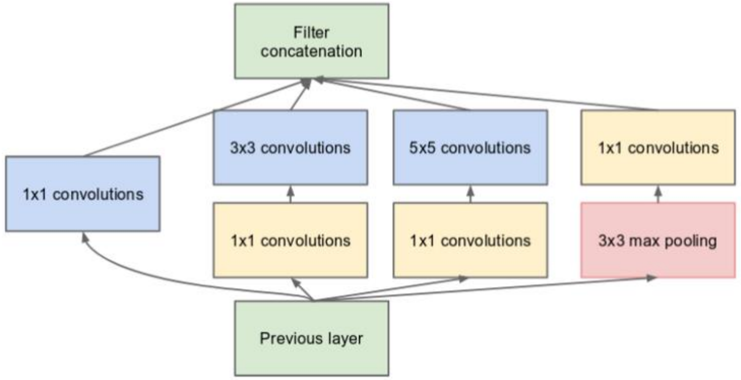
Parameters: 138M

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556.*

CNN Architectures | GoogLeNet



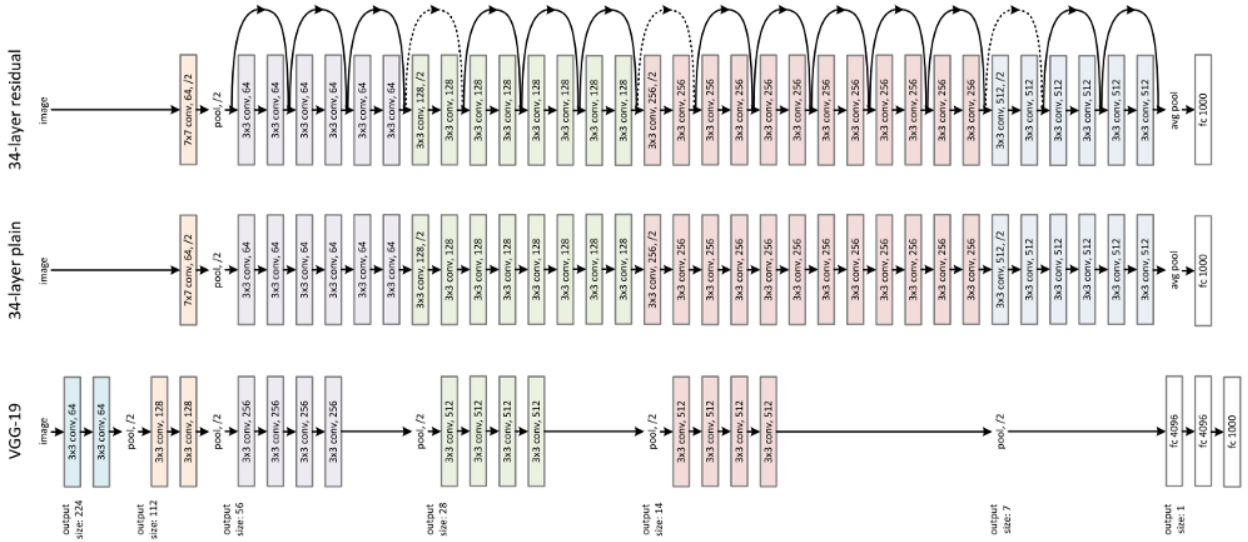
Inception module



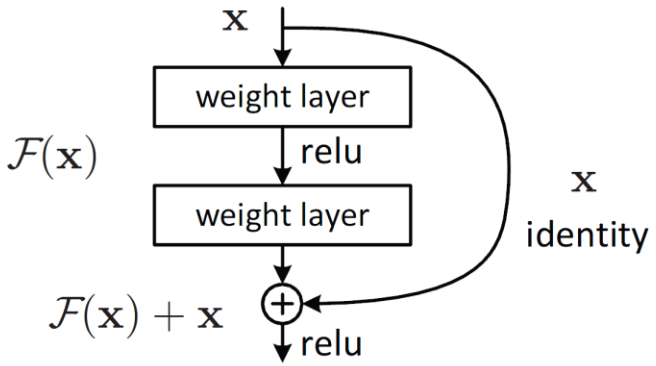
Parameters: 4M

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

CNN Architectures | ResNet



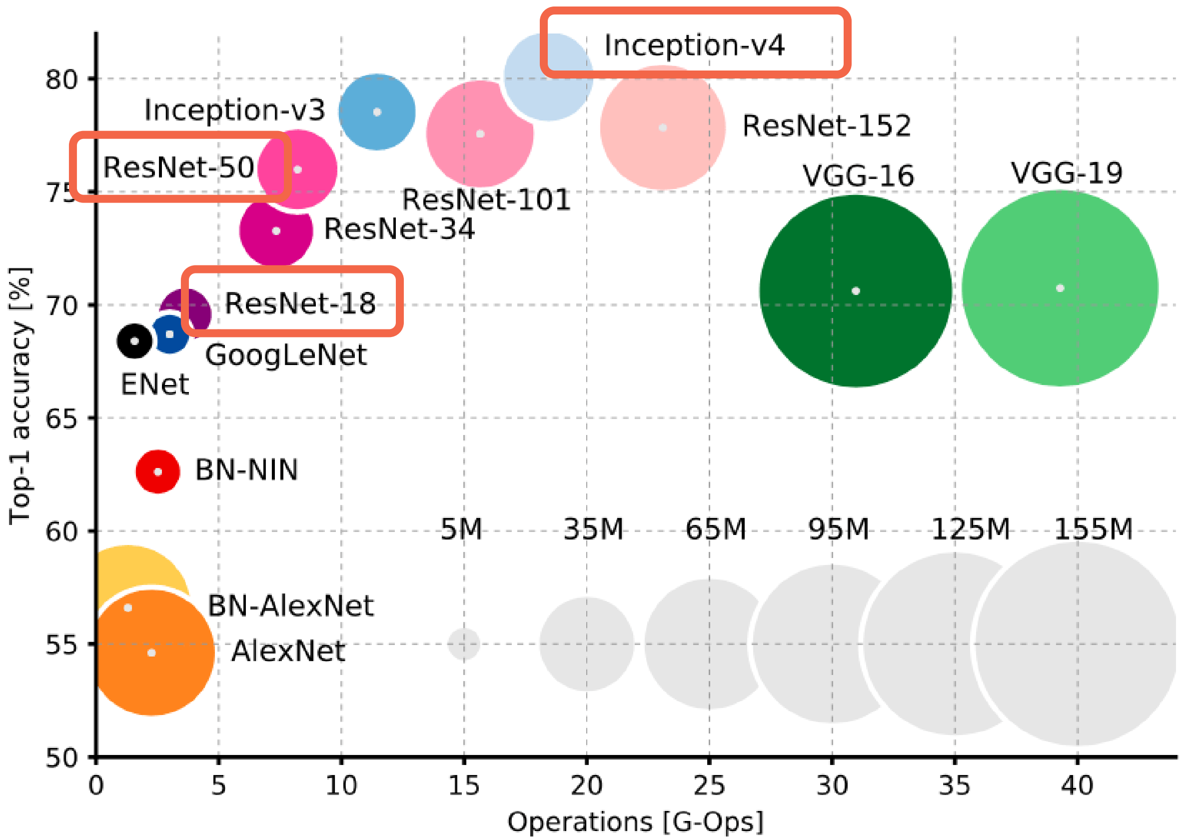
residual block



Parameters: 11-58M

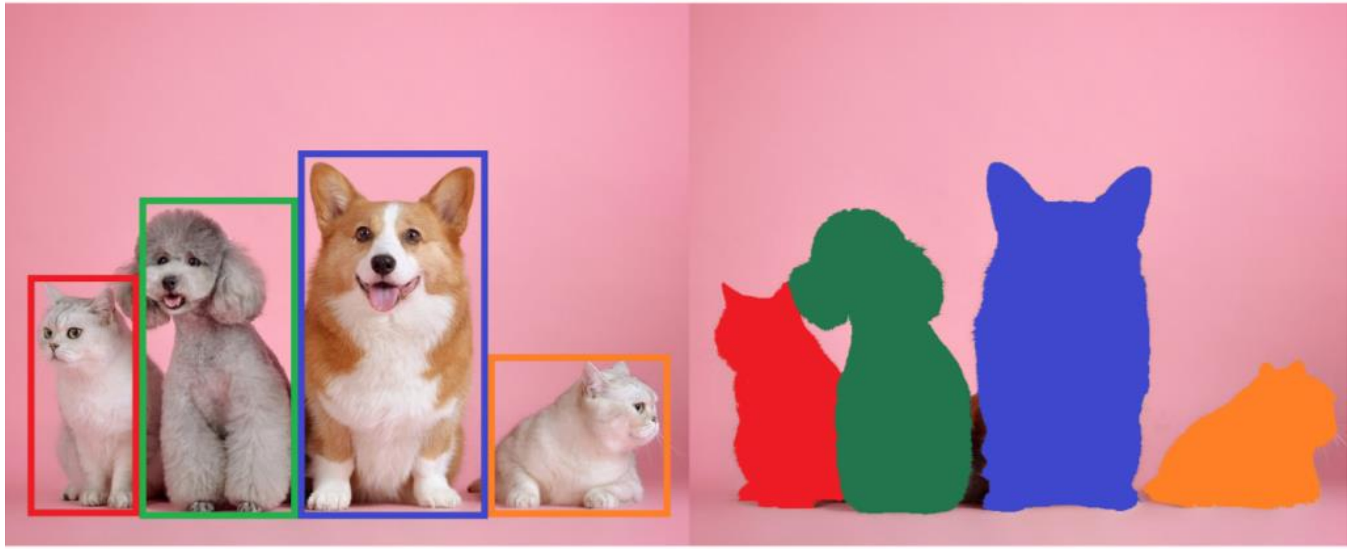
Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

CNN Architectures | Summery



Computer Vision Tasks

Semantic Segmentation | Classification + Localization | Object Detection | Instance Segmentation



TRUNK, CAT, LEAF | CAT | CAT, DOG, DOG, CAT | CAT, DOG, DOG, CAT

Only pixels, No object | Single Object | Multiple Objects

Deep Learning Frameworks

Microsoft CNTK

Caffe

Caffe2

PYTORCH

Chainer

Keras

TensorFlow

theano

dy/net

mxnet

GLUON

thank you!