COSE474(00): Deep Learning

Lecture 0. Course Overview

Instructor: Jaegul Choo (주재걸)
jchoo@korea.ac.kr

Slides partly made by my former student,
Yunjey Choi

주재걸

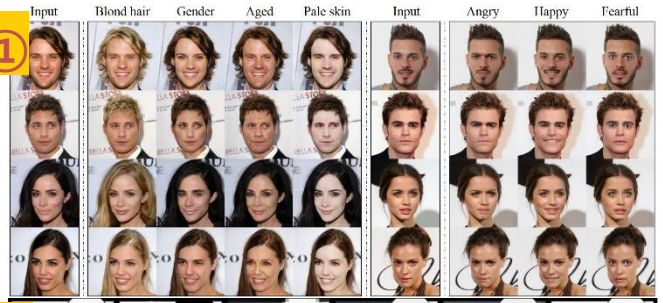
고려대
컴퓨터학과
조교수

[경력]

- 2001 SNU, Electrical Eng., B.S
- 2009 Georgia Tech, Electrical and Computer Eng., M.S
- 2013 Georgia Tech, Computational Science and Eng., Ph.D

[연구 분야]

- **Image Generation and Translation (Ensemble, multi-task, multi-domain)**
  - ①StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation, **CVPR'18**, *Oral presentation* (*2% acceptance rate*)
  - Coloring with Words: Guiding Image Colorization Through Text-based Palette Generation, **ECCV'18**
  - ②Consistent Comic Colorization with Pixel-wise Background Classification, **NIPS'17 Workshop** on Machine Learning for Creativity and Design
- **Natural Language Understanding and Generation**
  - MemoReader: Large-Scale Reading Comprehension through Neural Memory Controller, **EMNLP'18**
  - End-to-End Prediction of Buffer Overruns from Raw Source Code via Neural Memory Networks, **IJCAI'17**
  - Question-Aware Sentence Gating Networks for Question and Answering, ***Under Review***
- **Visual Analytics (Image, text, sequence mining)**
  - ③RetainVis: Visual Analytics with Interpretable and Interactive Recurrent Neural Networks on Electronic Medical Records, **IEEE VIS'18**
  - ReVACNN: Real-Time Visual Analytics for Convolutional Neural Network, **ACM SIGKDD Workshop** on Interactive Data Exploration and Analytics
- **Text and Social Media Mining**
  - ④ExTopicTile: Spatio-Temporal Event Analytics on Social Media, **CHI'18**
  - Short-Text Topic Modeling via Non-negative Matrix Factorization Enriched with Local Word-Context Correlations, **WWW'18**
  - TopicLens: Efficient Multi-Level Visual Topic Exploration of Large-Scale Document Collections, **IEEE VIS'16**



Today's Lecture

- Basic Course Information
- Course Schedule
- Grading Policy

Basic Course Information

- Time and Location
  - Monday 5:00-6:15pm
  - Woojeong Information Building 205 (우정정보통신관 205)
- Instructor: Jaegul Choo (주재걸), Ph.D
  - Office: Room# 502, Woojeong Information Building (우정정보통신관 502)
  - Phone / Email: 02-3290-4602 / jchoo@korea.ac.kr
- Course Website:
  - We will use blackboard.

  (All the class materials including the slides and homework assignments will be available here.)

Goal of This Course

- This course will study the theory and the application of deep neural networks.
- In detail, we will cover deep neural networks techniques including fully connected networks, convolutional neural networks, recurrent neural networks and long short-term memory, attention models, memory networks, generative adversarial networks.
- In addition, we will discuss various techniques necessary to effectively train these models, such as initialization, normalization, gradient descent, pre-training, regularization.

Flipped Class

Our lecture will be flipped classes. You have to watch one or two of the following lectures per week before the class:

- Stanford cs231n (computer vision)
  - http://cs231n.stanford.edu/ (slides + videos)

During our classes, I'll take questions and give in-depth discussions on what you learned from the above lectures.

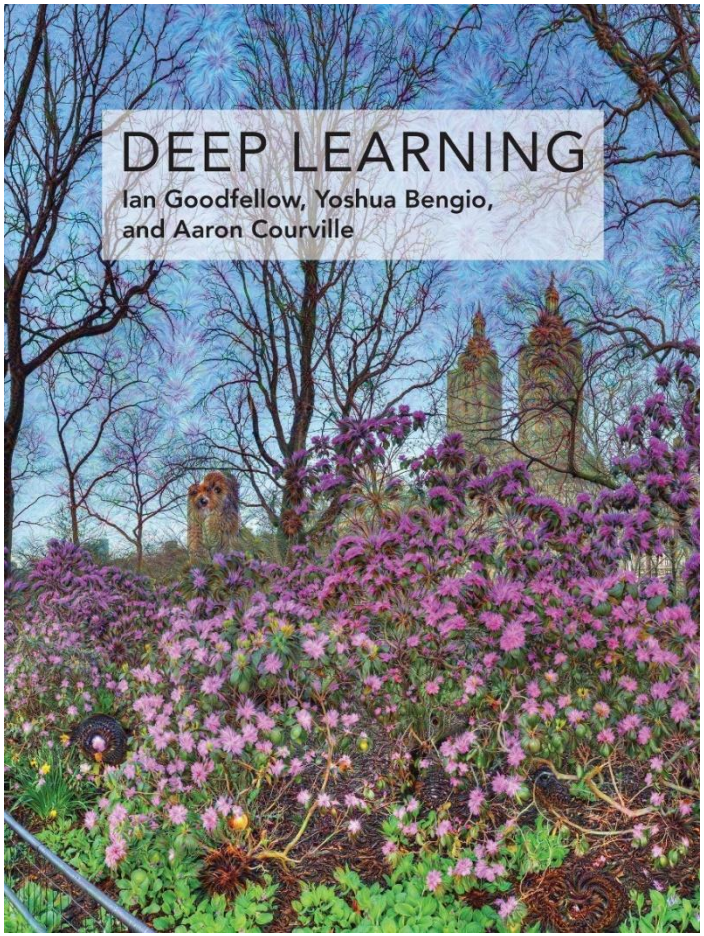
This is a demanding course. You are advised to take the course only if you can devote a substantial amount of your time.

FYI, you can find my advice on how to study machine learning and deep learning at

- http://blog.naver.com/joyfull1/221004891456

Textbook (Optional)

- Deep Learning, MIT Press, Ian Goodfellow, Yoshua Bengio, Aaron Courville
- Freely available:
  - http://www.deeplearningbook.org/



Pre-requisite

- Proficiency in Python
  - All the programming assignments will be in Python (and numpy)
  - I encourage you to also use Python for your term projects.
  - We will mainly use PyTorch as a deep learning library.
- Linear algebra
- Calculus
- Probability and statistics
- Machine Learning
  - I assume that students are familiar with the coursera ML course at https://www.coursera.org/learn/machine-learning
  - I also encourage you to do its homework with Python.

Background Knowledge Check

Poll

- Who took artificial intelligence?
- Who took machine learning?
- Who took data science?
- Who has some background knowledge in deep learning?
- Who currently conducts research related to deep learning?

Things to Know in Machine Learning

- Linear regression and logistic regression
- Ensemble learning
- Bias/variance tradeoff
- Overfitting vs. underfitting
- Regularization
- Gradient descent optimization
- Training/validation/test sets
- Cross-validation
- Model selection and hyper-parameter tuning
- Loss function
- Evaluation metric
  - accuracy, precision, recall, F-1 measure, precision-recall curve, area under the ROC curve, …

Course Schedule (Tentative)

- Week 01 – Basics of machine learning, linear and logistic regression
- Week 02 – Feed-forward neural networks and computational graph
- Week 03 – Convolutional neural networks
- Week 04 – Training neural networks: initialization, batch normalization, dropout, data augmentation, and transfer learning and fine-tuning
- Week 05 – Optimization algorithms: stochastic gradient descent, momentum, AdaGrad, RMSProp, Adam
- Week 06 – CNN architectures: AlexNet, VGG, GoogLeNet, and ResNet
- Week 07 – Word embedding: word2vec and GloVe

Course Schedule (Tentative) – cont'd

- Week 08 – Midterm Exam
- Week 09 – Recurrent neural networks, long short-term memory, gated recurrent units
- Week 10 – Attention model, image captioning, and sequence-to-sequence model
- Week 11 – Self-attention, co-attention, and memory nerworks
- Week 12 – Detection and segmentation
- Week 13 – Visualizing and understanding
- Week 14 – Generative models: generative adversarial networks and variational autoencoders
- Week 15 – Project presentation
- Week 16 – Final Exam

Grading Policy

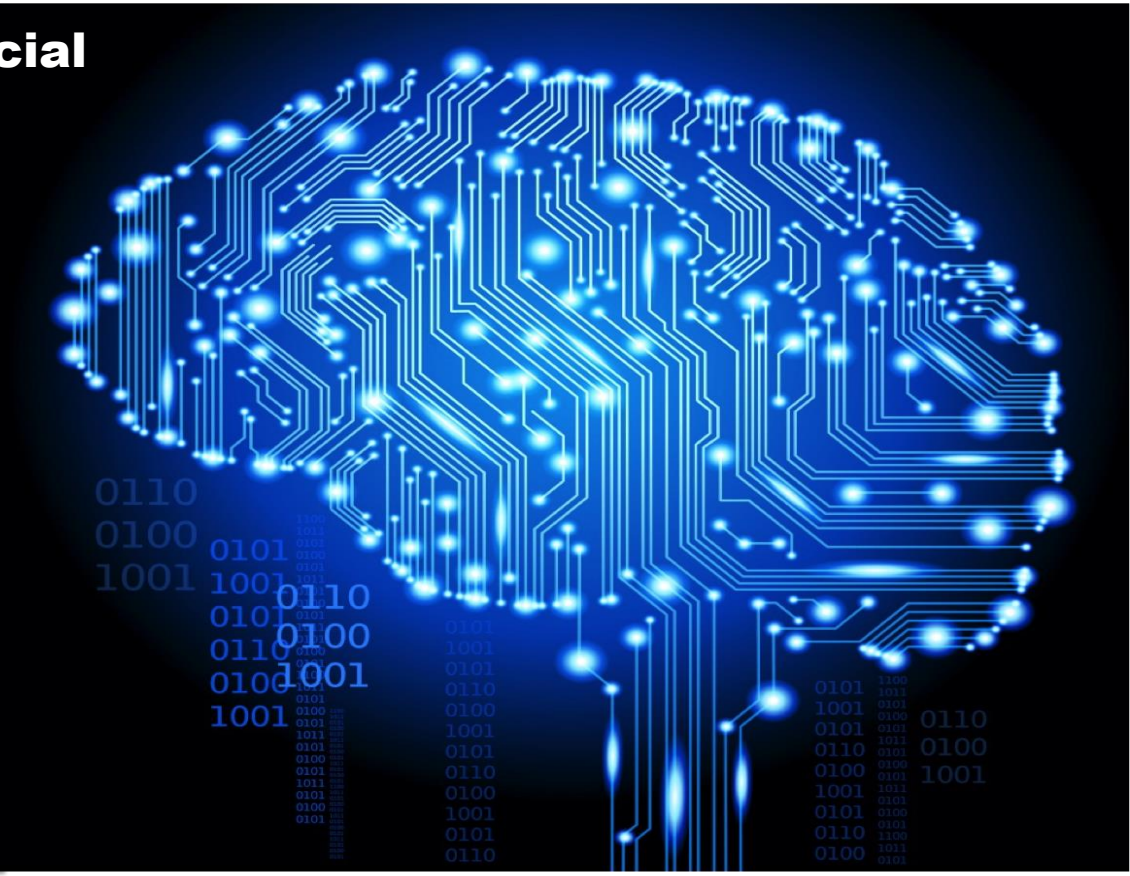
- 20%: Homework assignments
  - Problem sets
  - Programming tasks
- 20%: Midterm
- 25%: Final
- 25%: Term project
- 5%: In-class quiz
- 5%: Class participation
- Attendance: For each missed class, 0.5% point will be deducted.

More Info on Our Class

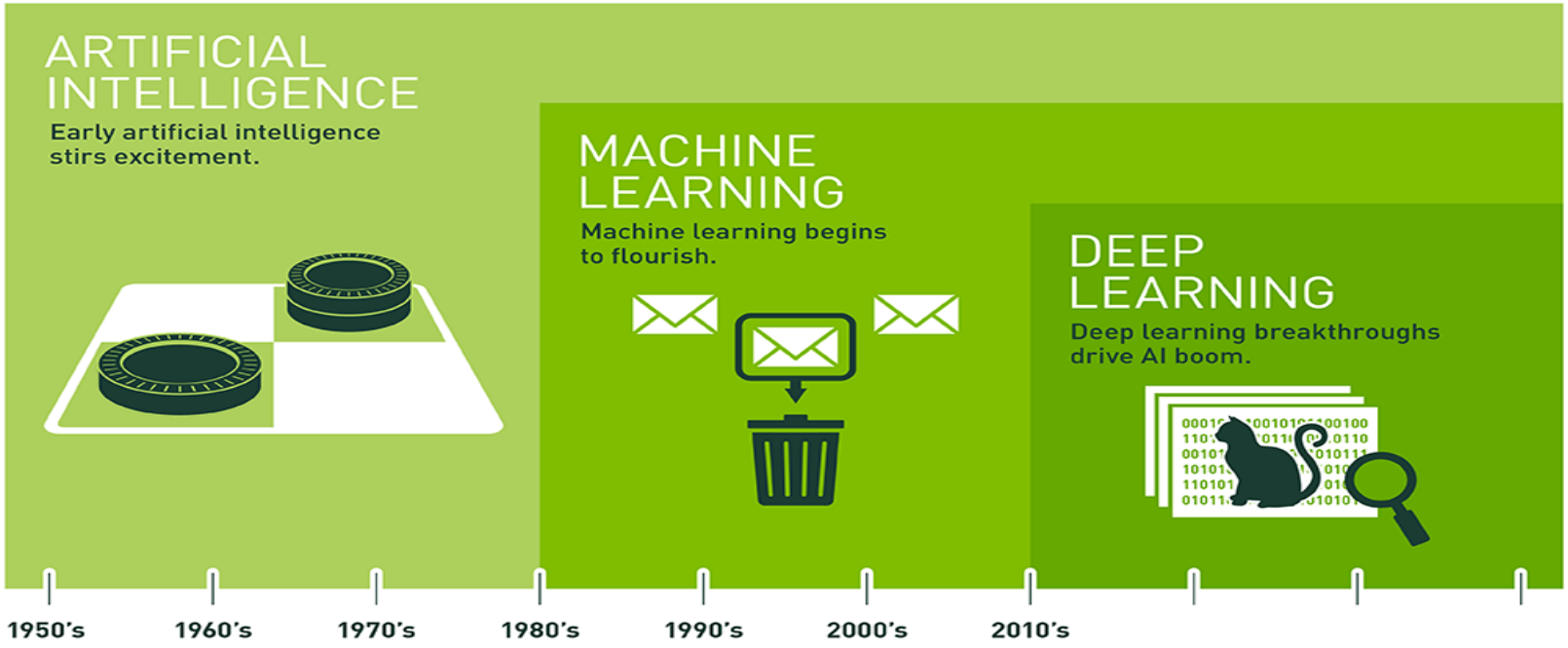
- Every class, we will have a 5-minute quiz in the beginning.
- We will use www.slido.com (access code: kudl), where you can post questions before and during the class.
- Until this Thursday's class, watch 2017 cs231n Lectures:
  - Lecture 1 (optional): https://youtu.be/vT1JzLTH4G4
  - Lecture 2: https://youtu.be/OoUX-nOEjG0
  - Lecture 3: https://youtu.be/h7iBpEHGVNc
- Again, we will start with the quiz from the next class.

Deep Learning

Deep learning refers to artificial neural networks that are composed of many layers.



Deep Learning

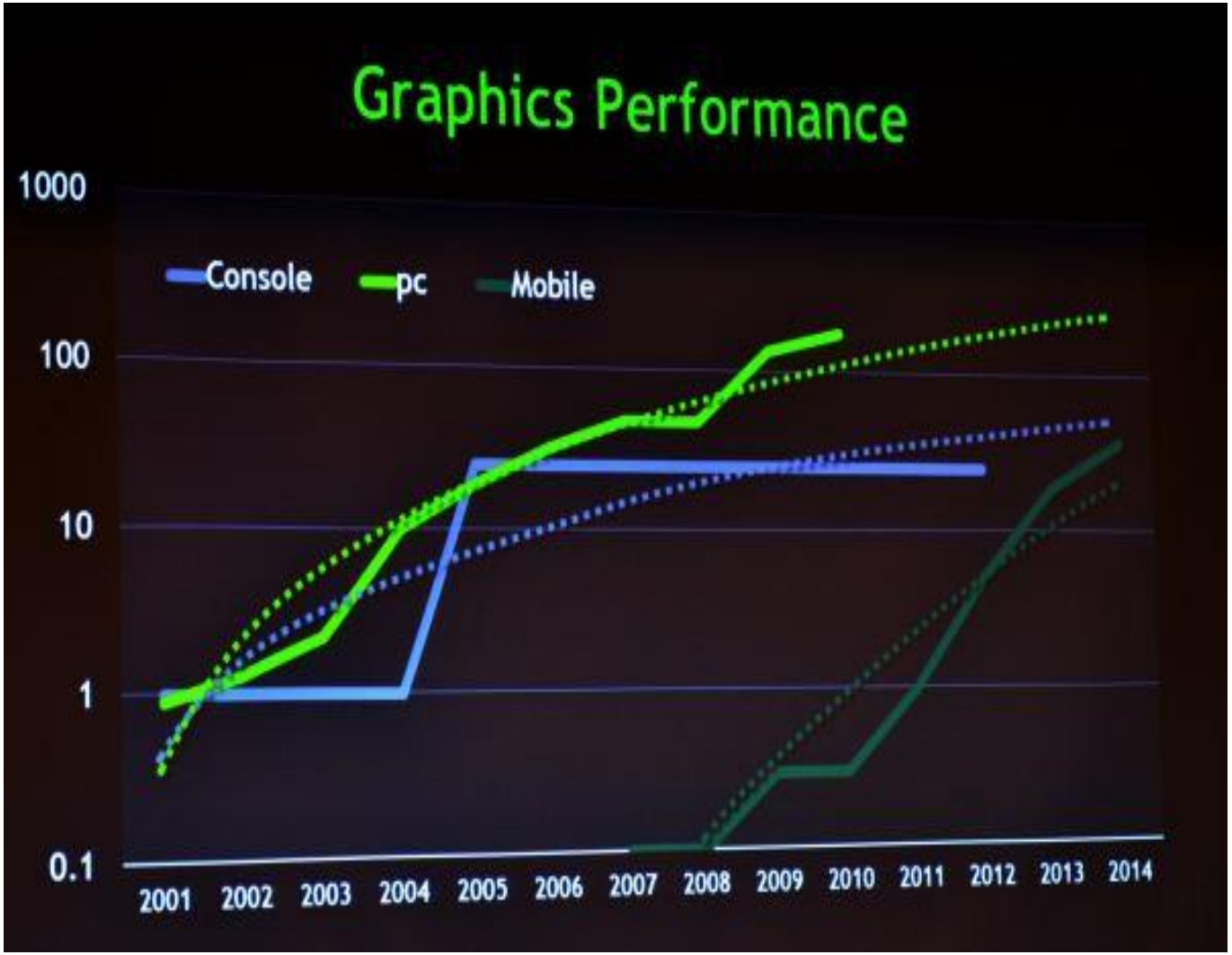


Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Factors for Success of Deep Learning

- Big Data
  - Accumulated data and their labels
- Improved training algorithm
  - Various techniques to properly train deep neural networks
    - ReLU, batch normalization, dropout, …
- Fast hardware
  - GPU Acceleration

Fast Hardware



Big Data

ImageNet Challenge

IMAGENET

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

Deep Learning Heroes

Geoffrey Hinton

Yann LeCun

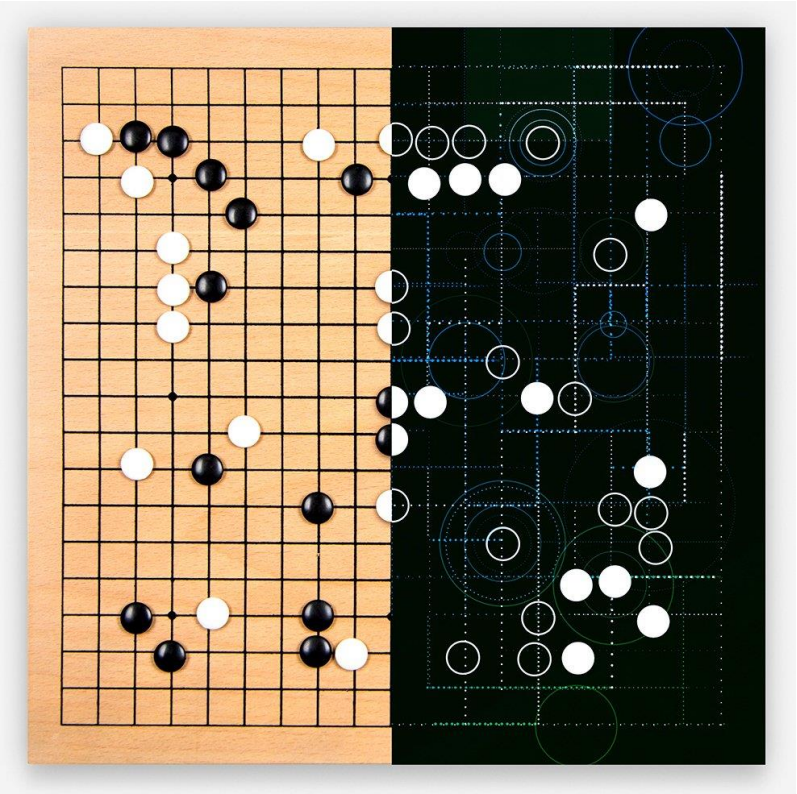
Andrew Ng

Yoshua Bengio

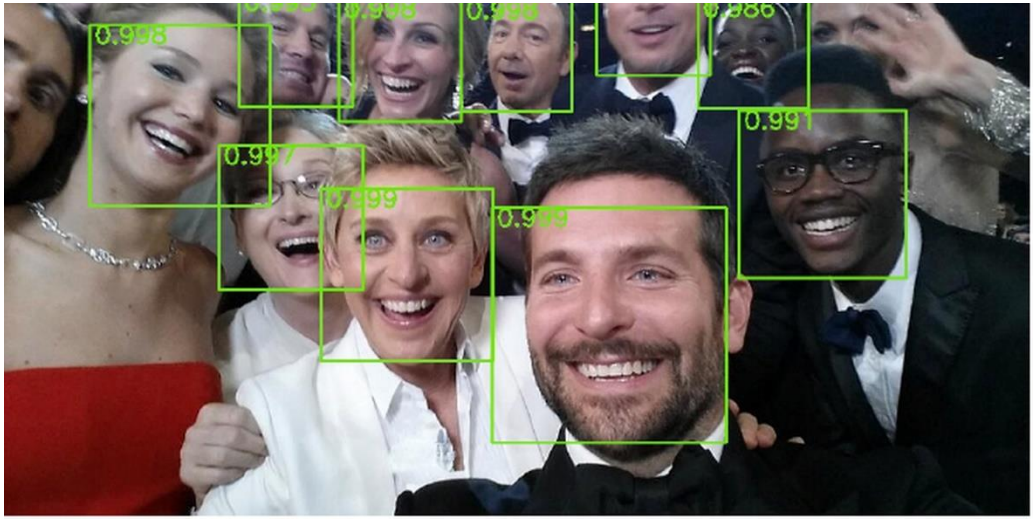
Major Conferences in AI

- Artificial Intelligence / Machine Learning
  - **NIPS**: Neural Information Processing Systems
  - **ICML**: International Conference on Machine Learning
  - **ICLR**: International Conference on Learning Representations
  - **AAAI**: AAAI Conference on Artificial Intelligence
  - **IJCAI**: International Joint Conference on Artificial Intelligence
- Computer Vision
  - **CVPR**: IEEE Conference on Computer Vision and Pattern Recognition
  - **ICCV**: IEEE International Conference on Computer Vision
  - **ECCV**: European Conference on Computer Vision
- Natural Language Processing
  - **ACL**: Meeting of the Association for Computational Linguistics
  - **EMNLP**: Conference on Empirical Methods in Natural Language Processing
  - **NAACL-HLT**: Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies

Game AI



Face Detection & Recognition



Object Detection & Recognition

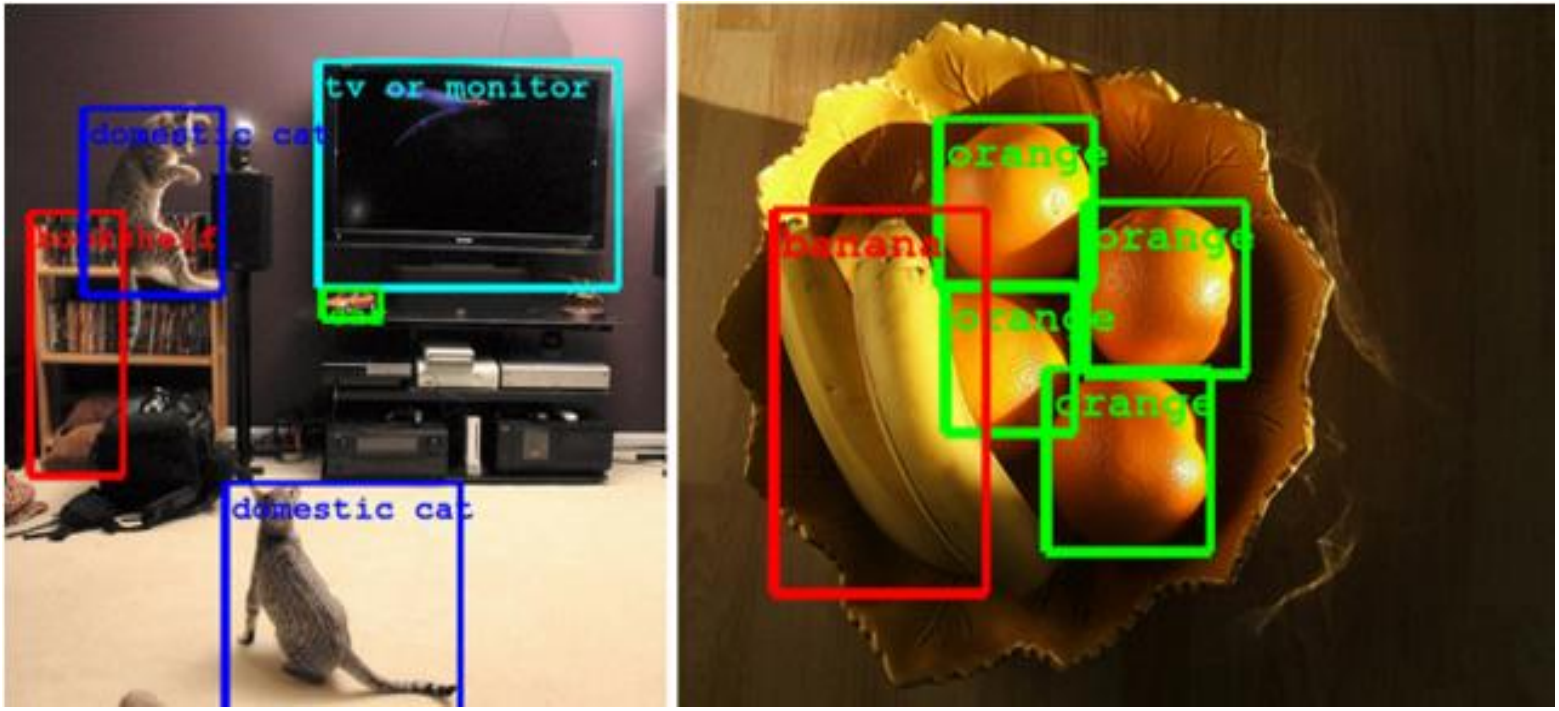
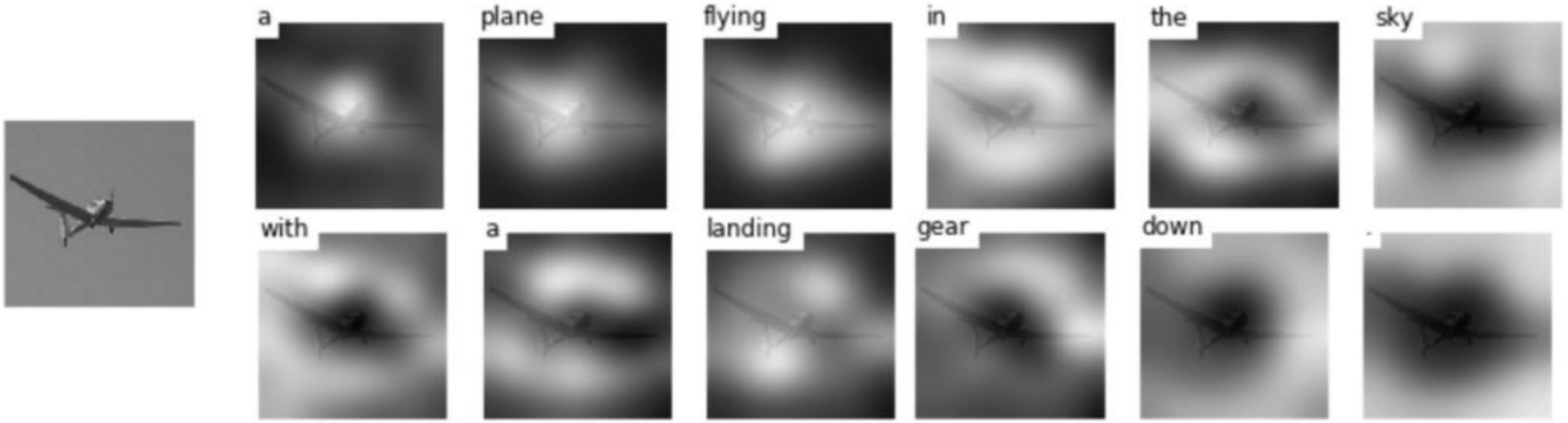
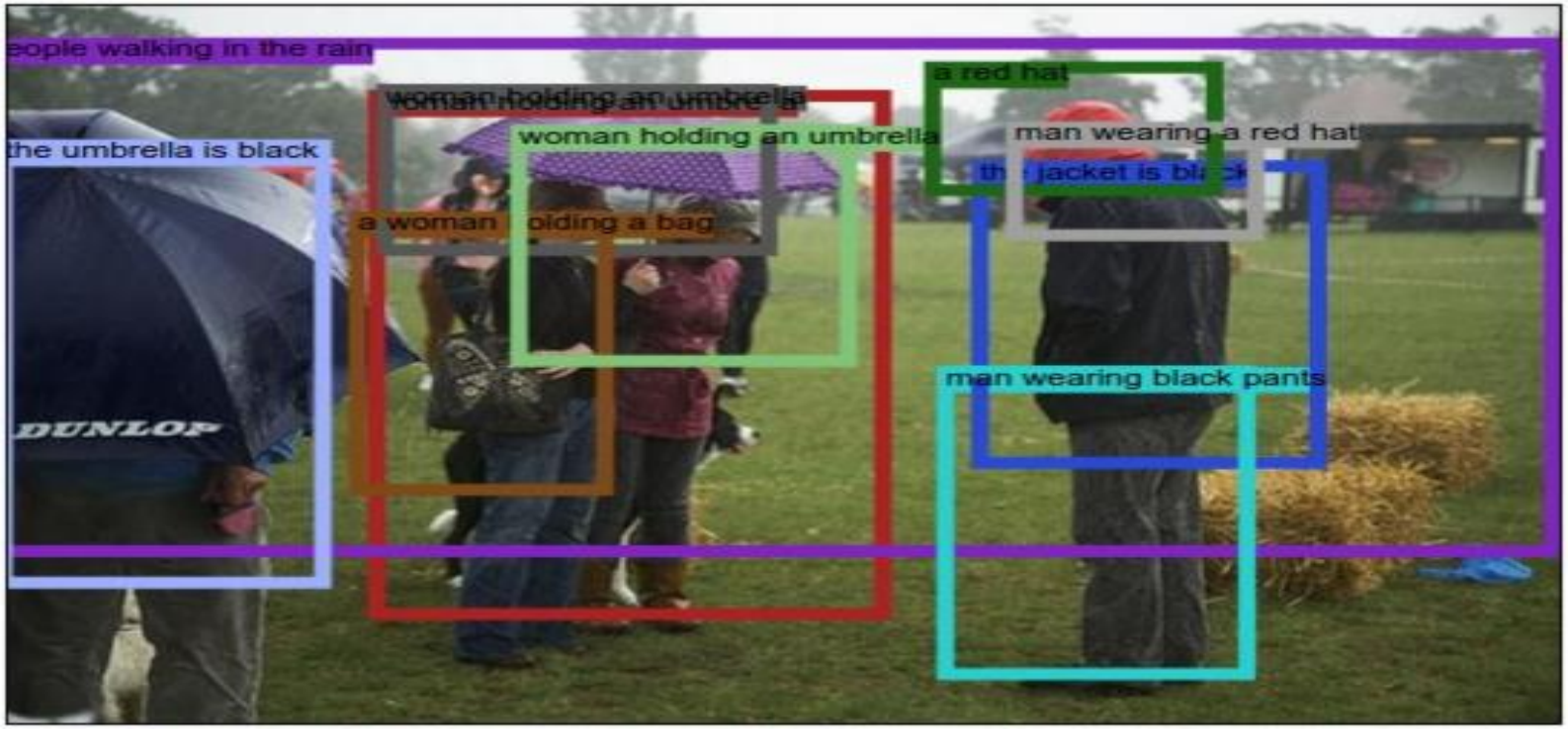


Image Captioning



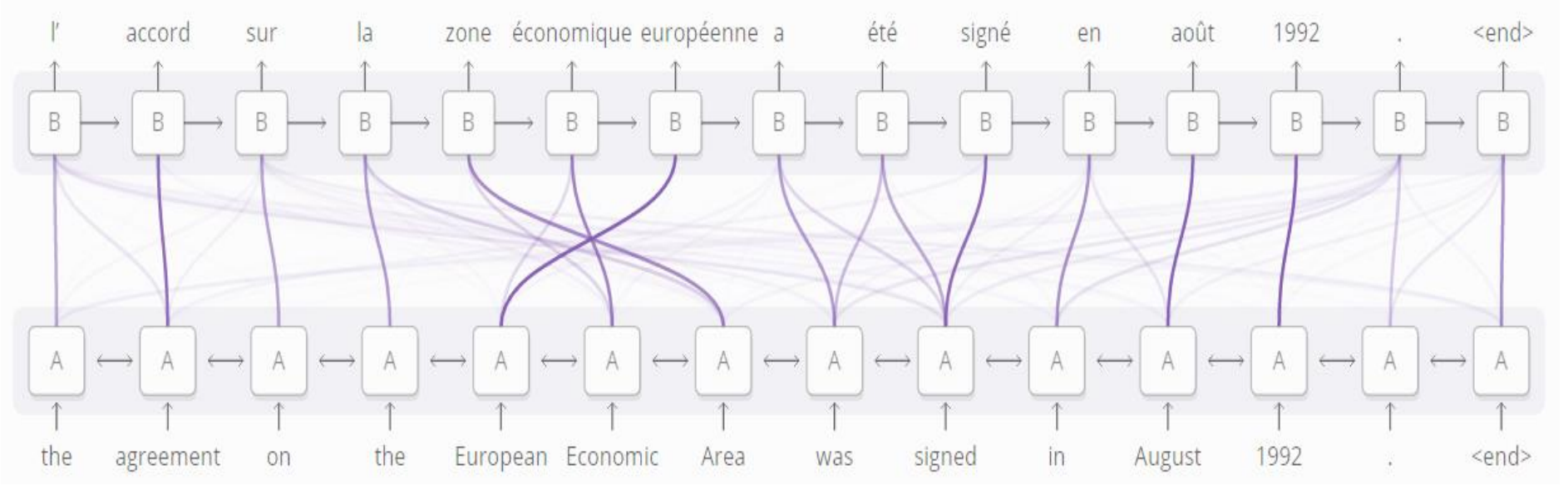
https://github.com/yunjey/show-attend-and-tell

Image Captioning



DenseCap: Fully Convolutional Localization Networks for Dense Captioning

Machine Translation



Speech Synthesis

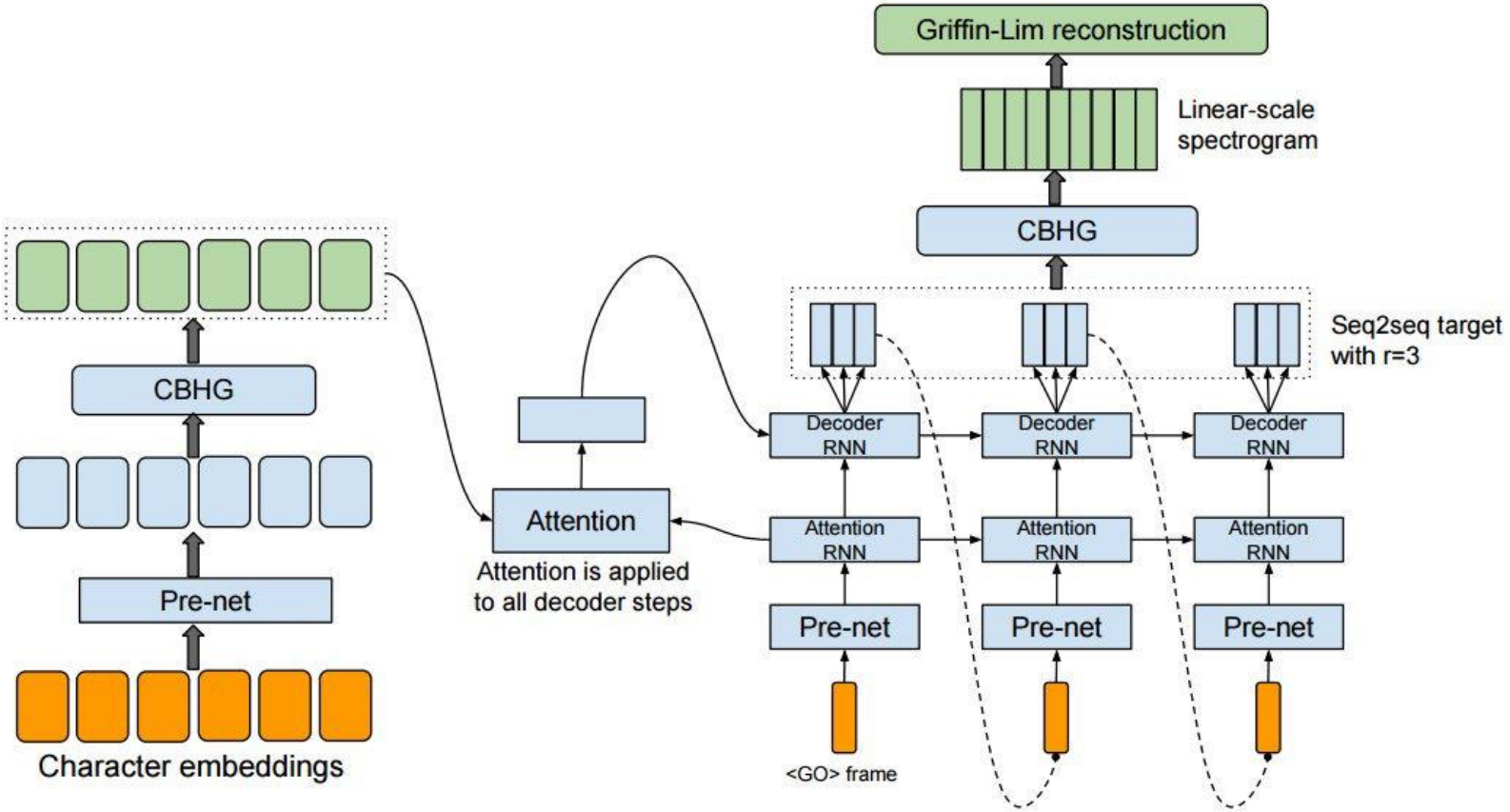


Figure 1: *Model architecture. The model takes characters as input and outputs the corresponding raw spectrogram, which is then fed to the Griffin-Lim reconstruction algorithm to synthesize speech.*

Question Answering

Task 1: Single Supporting Fact

Mary went to the bathroom.
John moved to the hallway.
Mary travelled to the office.
Where is Mary? A:office

Task 2: Two Supporting Facts

John is in the playground.
John picked up the football.
Bob went to the kitchen.
Where is the football? A:playground

Task 3: Three Supporting Facts

John picked up the apple.
John went to the office.
John went to the kitchen.
John dropped the apple.
Where was the apple before the kitchen? A:office

Task 4: Two Argument Relations

The office is north of the bedroom.
The bedroom is north of the bathroom.
The kitchen is west of the garden.
What is north of the bedroom? A: office
What is the bedroom north of? A: bathroom

Task 5: Three Argument Relations

Mary gave the cake to Fred.
Fred gave the cake to Bill.
Jeff was given the milk by Bill.
Who gave the cake to Fred? A: Mary
Who did Fred give the cake to? A: Bill

Task 6: Yes/No Questions

John moved to the playground.
Daniel went to the bathroom.
John went back to the hallway.
Is John in the playground? A:no
Is Daniel in the bathroom? A:yes

Task 7: Counting

Daniel picked up the football.
Daniel dropped the football.
Daniel got the milk.
Daniel took the apple.
How many objects is Daniel holding? A: two

Task 8: Lists/Sets

Daniel picks up the football.
Daniel drops the newspaper.
Daniel picks up the milk.
John took the apple.
What is Daniel holding? milk, football

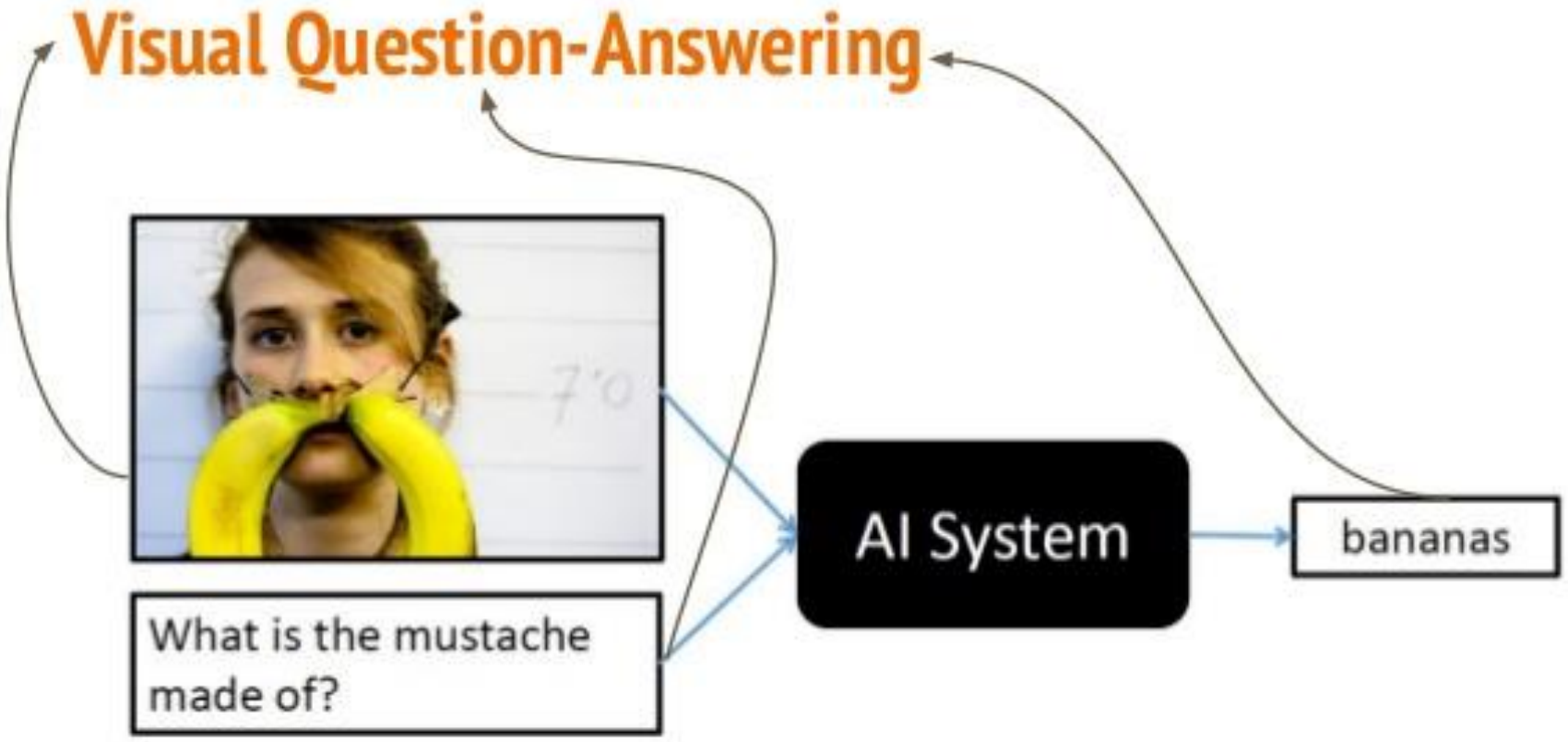
Task 9: Simple Negation

Sandra travelled to the office.
Fred is no longer in the office.
Is Fred in the office? A:no
Is Sandra in the office? A:yes

Task 10: Indefinite Knowledge

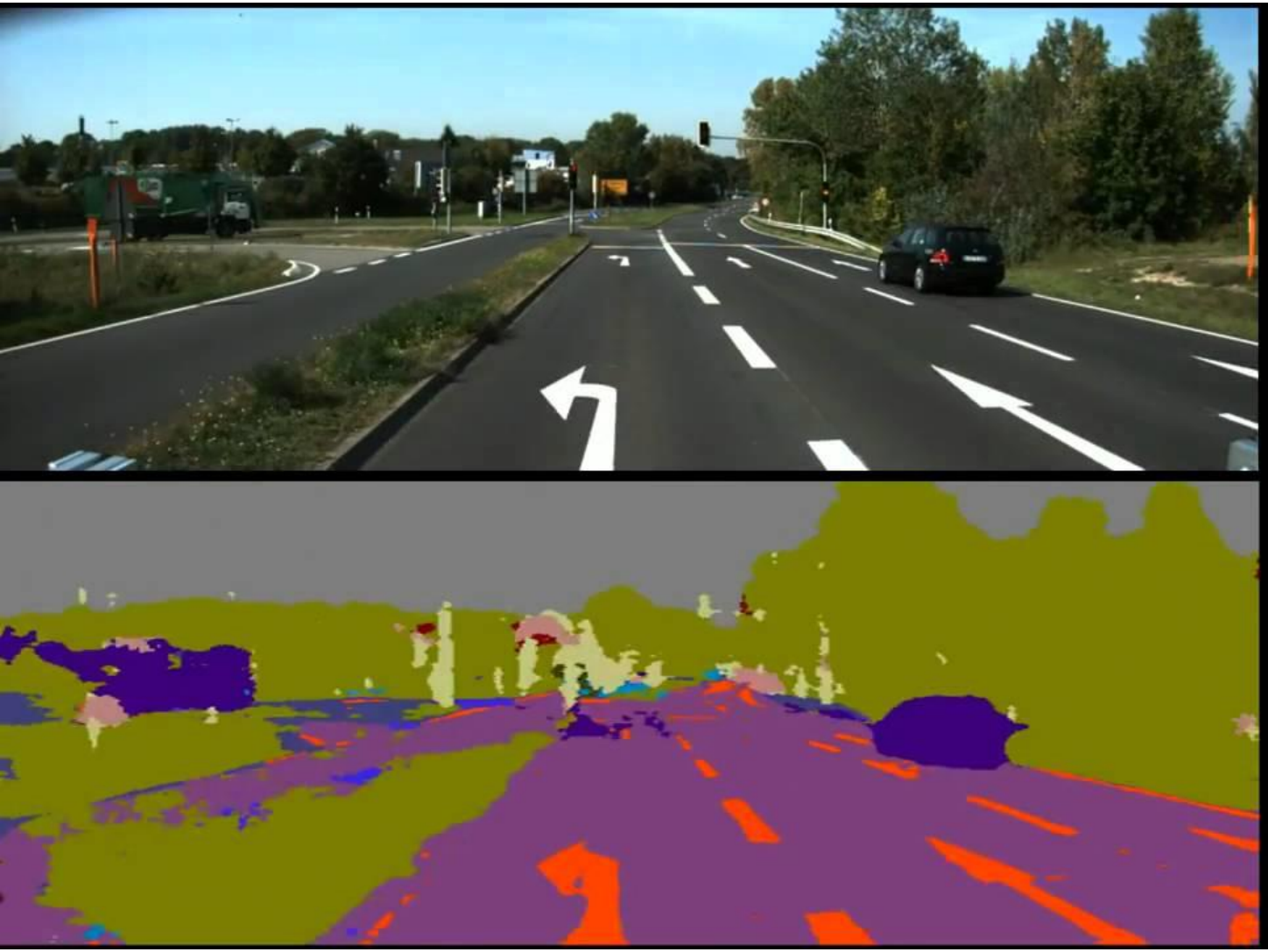
John is either in the classroom or the playground.
Sandra is in the garden.
Is John in the classroom? A:maybe
Is John in the office? A:no

Visual Question Answering



Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Lawrence Zitnick, C., & Parikh, D. (2015). Vqa: Visual question answering. In Proceedings of the IEEE International Conference on Computer Vision (pp. 2425-2433).

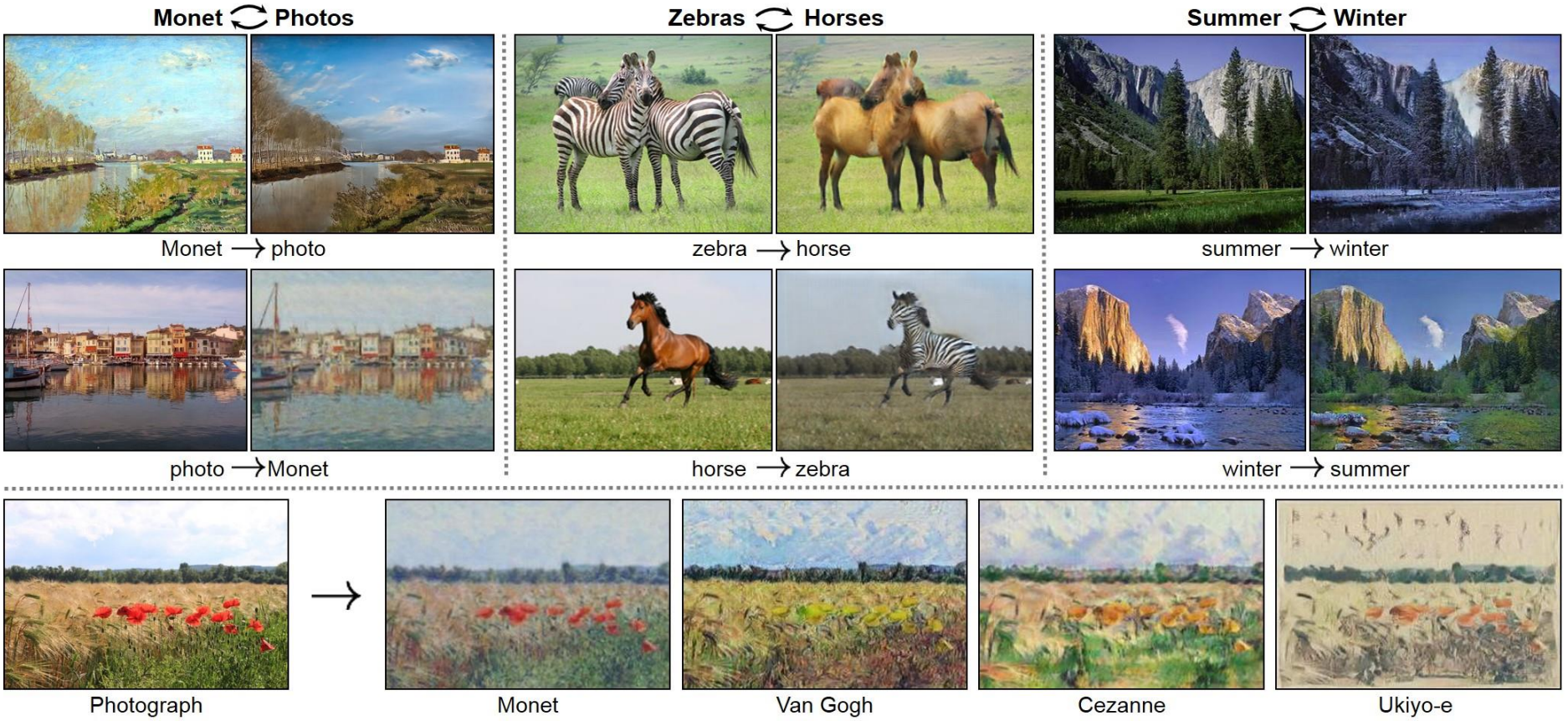
Self-Driving Cars



Style Transfer

https://medium.com/element-ai-research-lab/stabilizing-neural-style-transfer-for-video-62675e203e42

Image-to-Image Translation



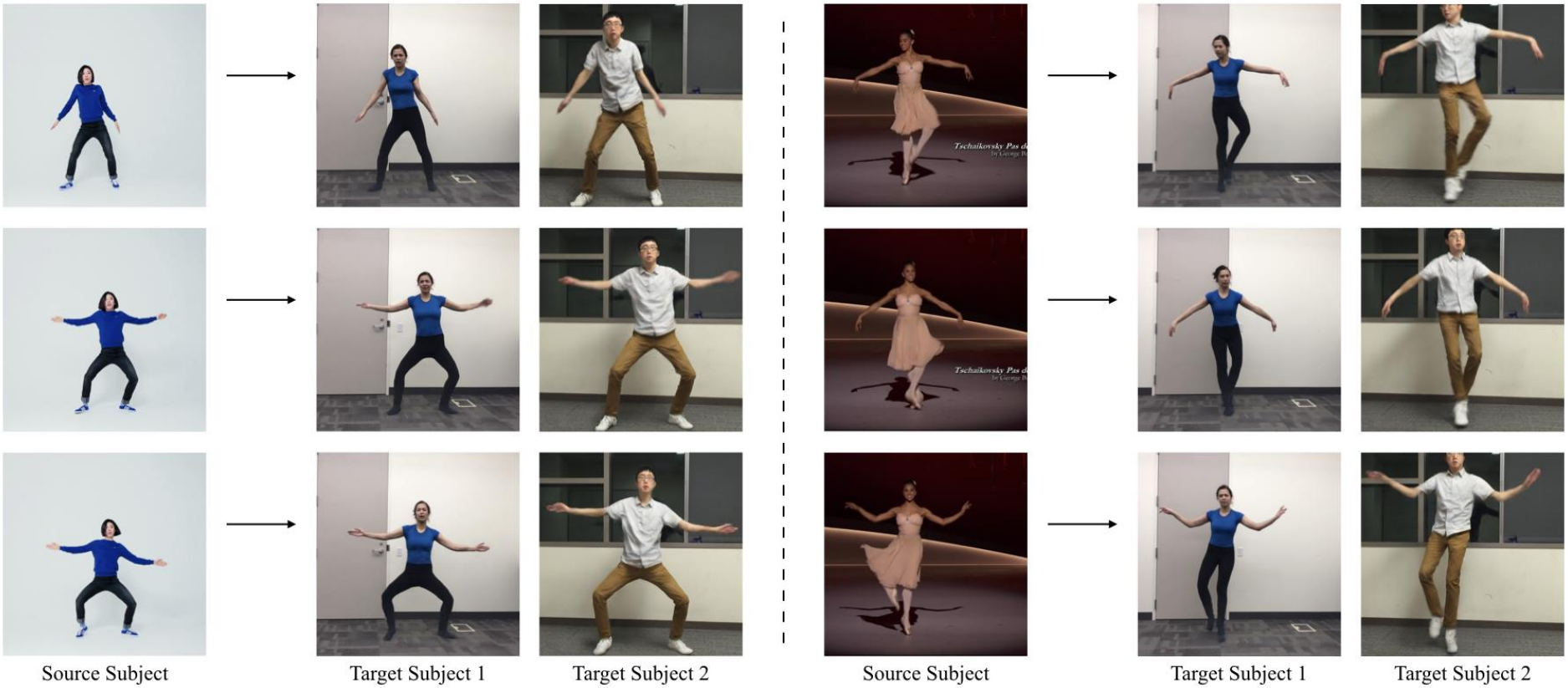
CycleGAN: Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, Zhu et al., ICCV'17

https://github.com/junyanz/CycleGAN

Video-to-Video Translation

Video-to-Video Synthesis, Wang et al., arXiv, 2018
https://tcwang0509.github.io/vid2vid/

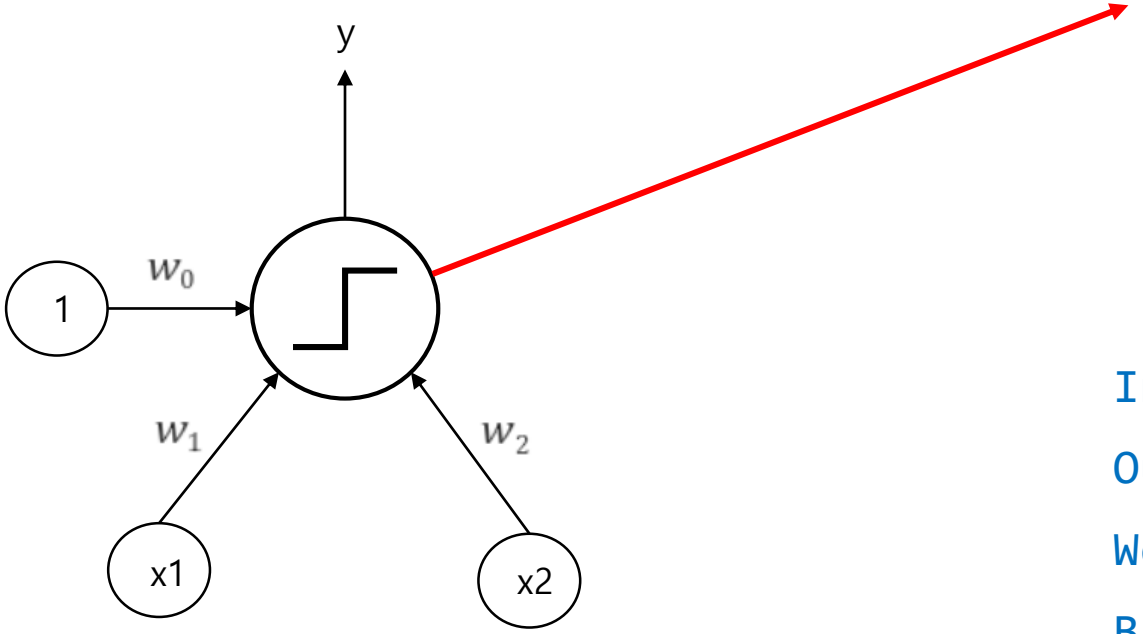
Video-to-Video Translation



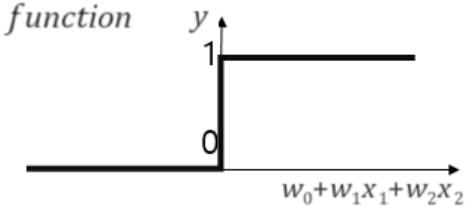
Everybody Dance Now, Chan et al., arXiv, 2018
Video link: https://youtu.be/PCBTZh41Ris

Perceptron

Single Layer Perceptron



hard thresholding function



$$y = \begin{cases} 1 & w_0+w_1x_1+w_2x_2 \geq 0 \\ 0 & otherwise \end{cases}$$

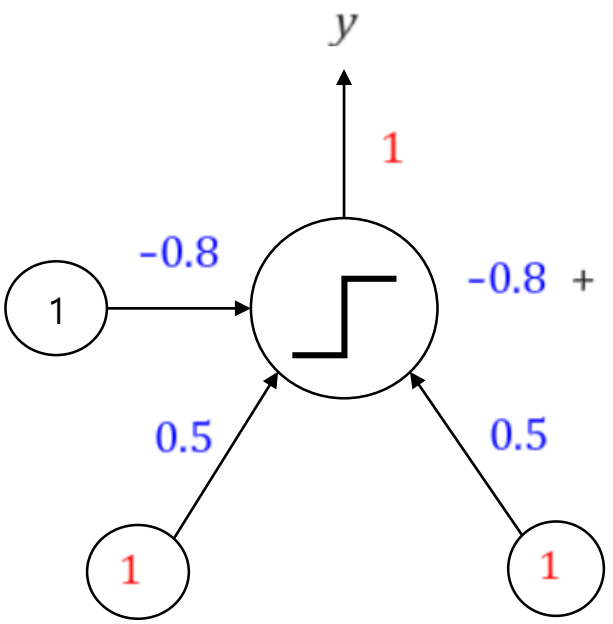
Inputs: x1, x2

Output: y

Weights: w0, w1, w2

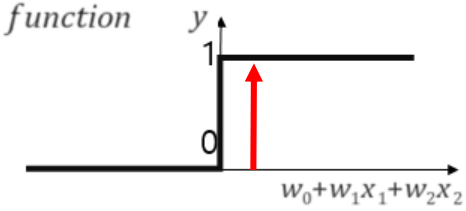
Bias: 1

Single Layer Perceptron



$-0.8 + 0.5\times1 + 0.5\times1 = 0.2$

hard thresholding function

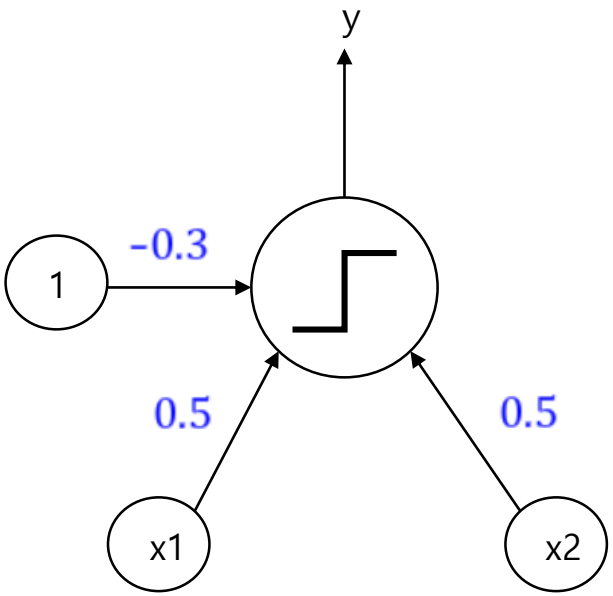


$$y = \begin{cases} 1 & w_0+w_1x_1+w_2x_2 \geq 0 \\ 0 & otherwise \end{cases}$$

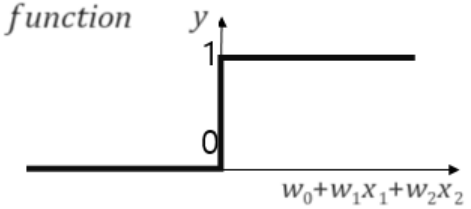
AND Gate

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Single Layer Perceptron



hard thresholding function



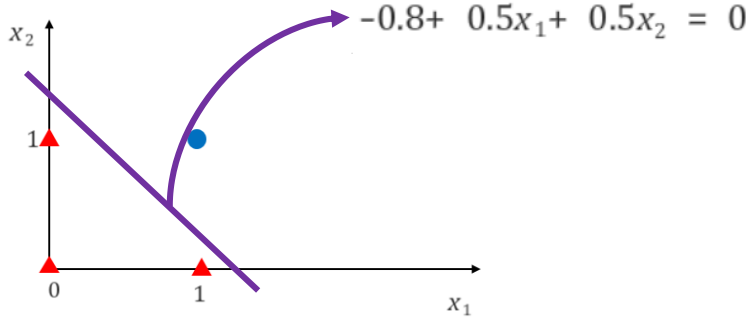
$$y = \begin{cases} 1 & w_0+w_1x_1+w_2x_2 \geq 0 \\ 0 & otherwise \end{cases}$$

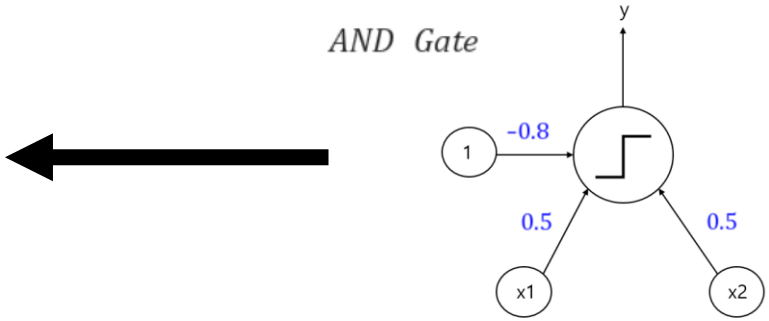
OR Gate

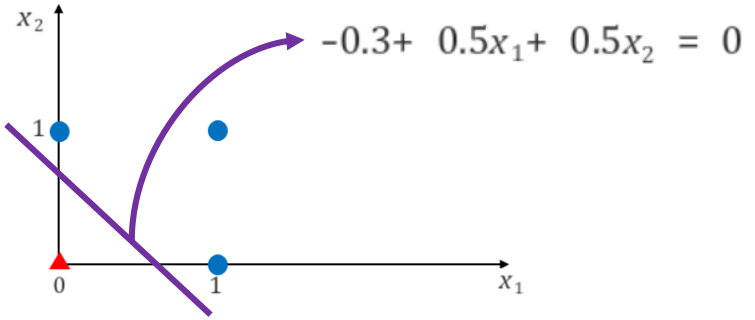
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

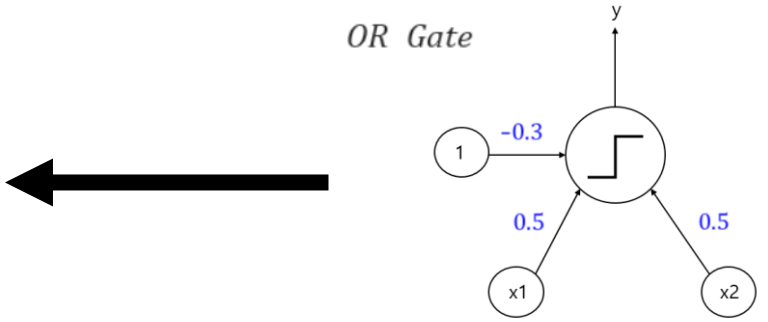
Single Layer Perceptron

● : 1 ▲ : 0





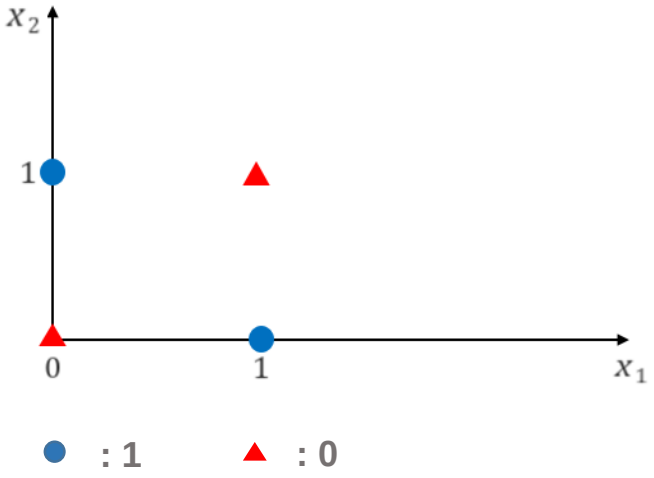




XOR Gate

Is it possible to solve XOR problem using a single layer perceptron?

No. Single layer perceptron can only solve linear problem. XOR problem is non-linear.

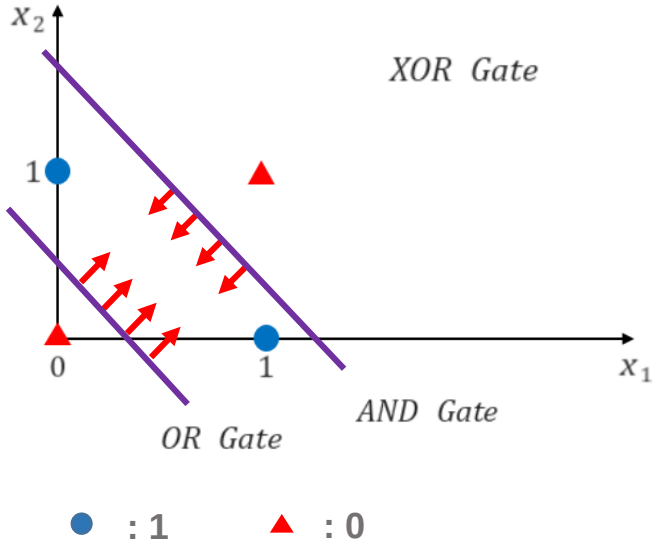


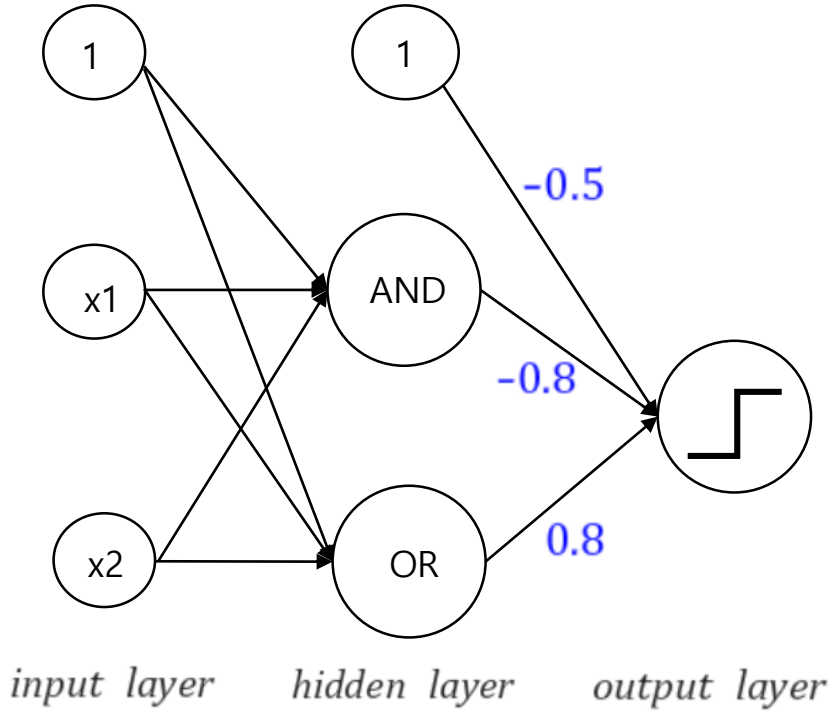
XOR Gate

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

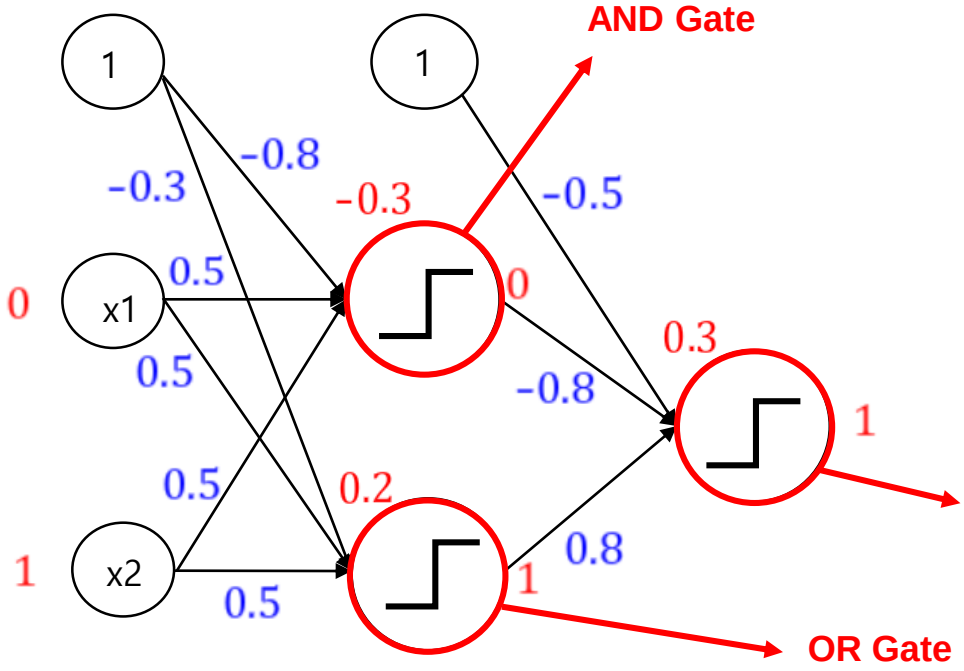
Multi Layer Perceptron

But if we use 2 single layer perceptron, we can solve XOR problem. This model is called multi layer perceptron.





Multi Layer Perceptron



OR Gate

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

AND Gate

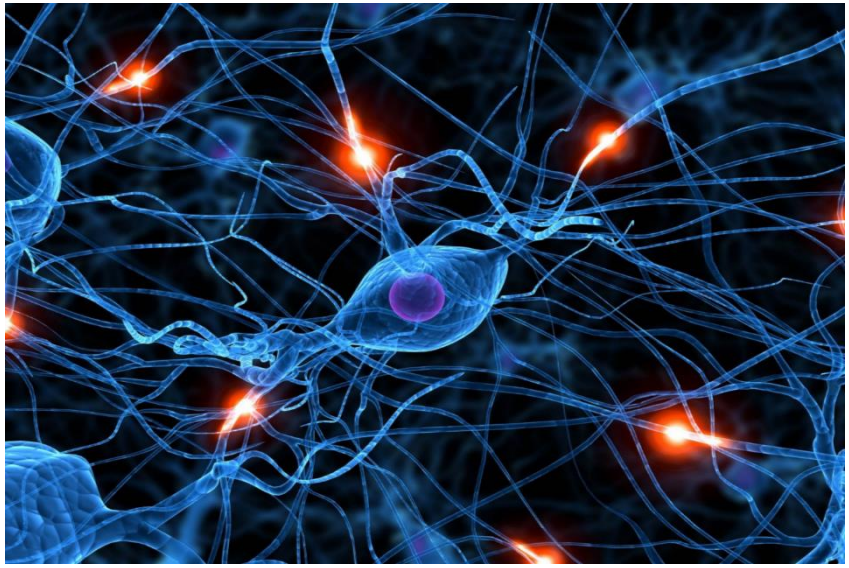
x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

XOR Gate

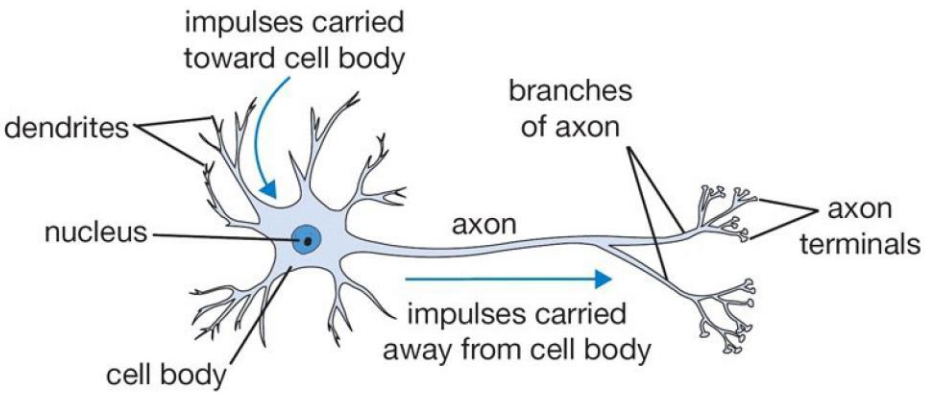
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Neural Networks

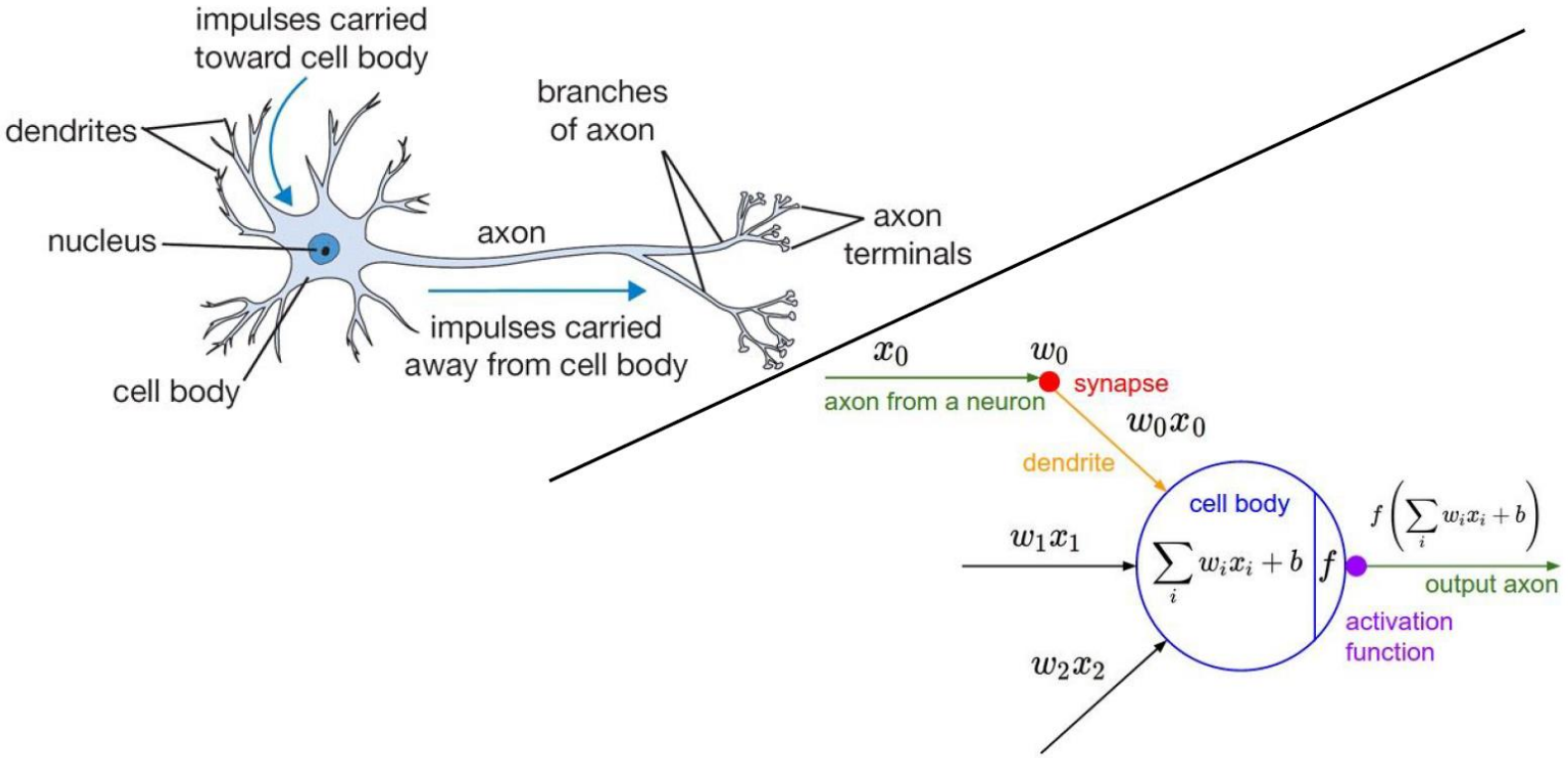
Human Brain and Neural Networks



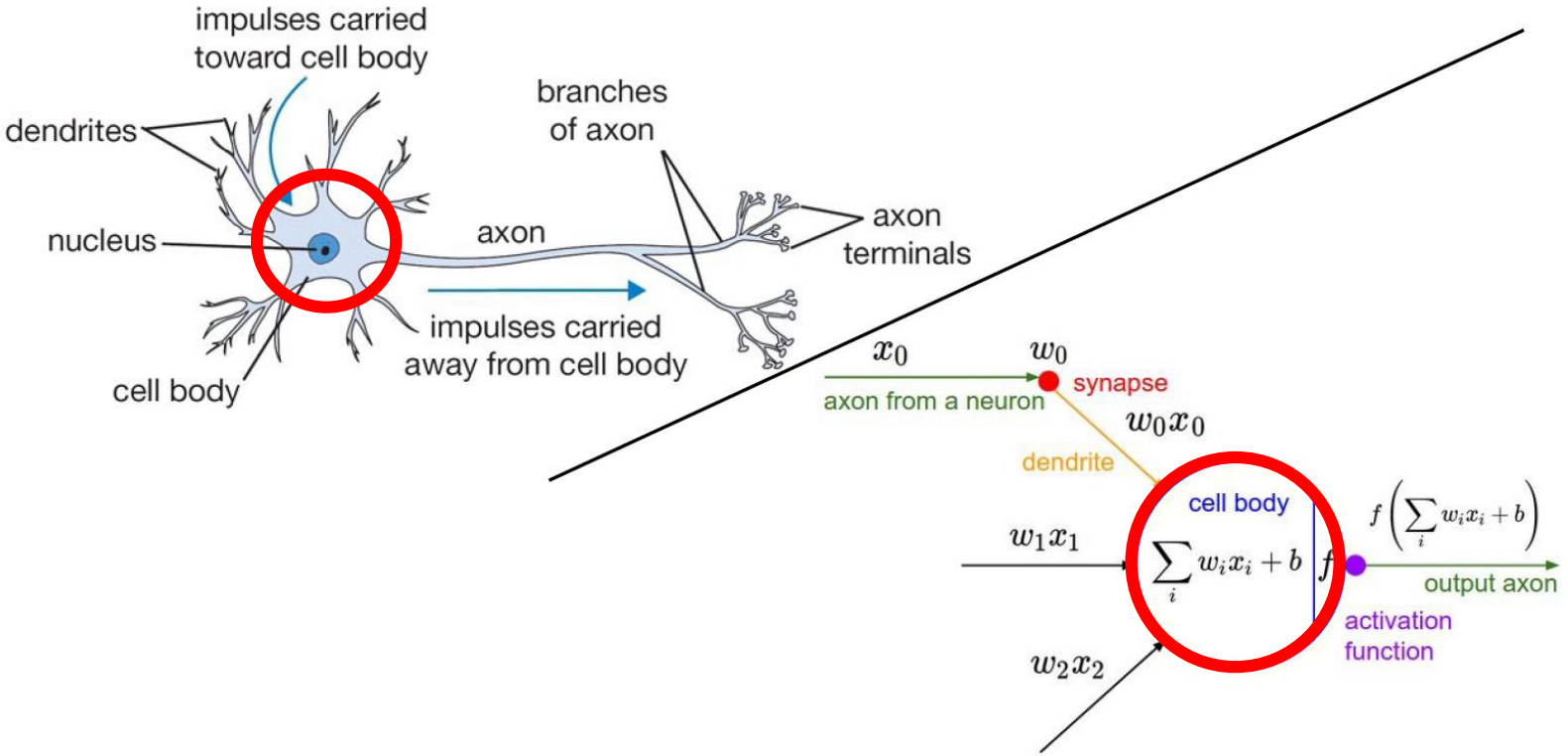
Human Brain and Neural Networks



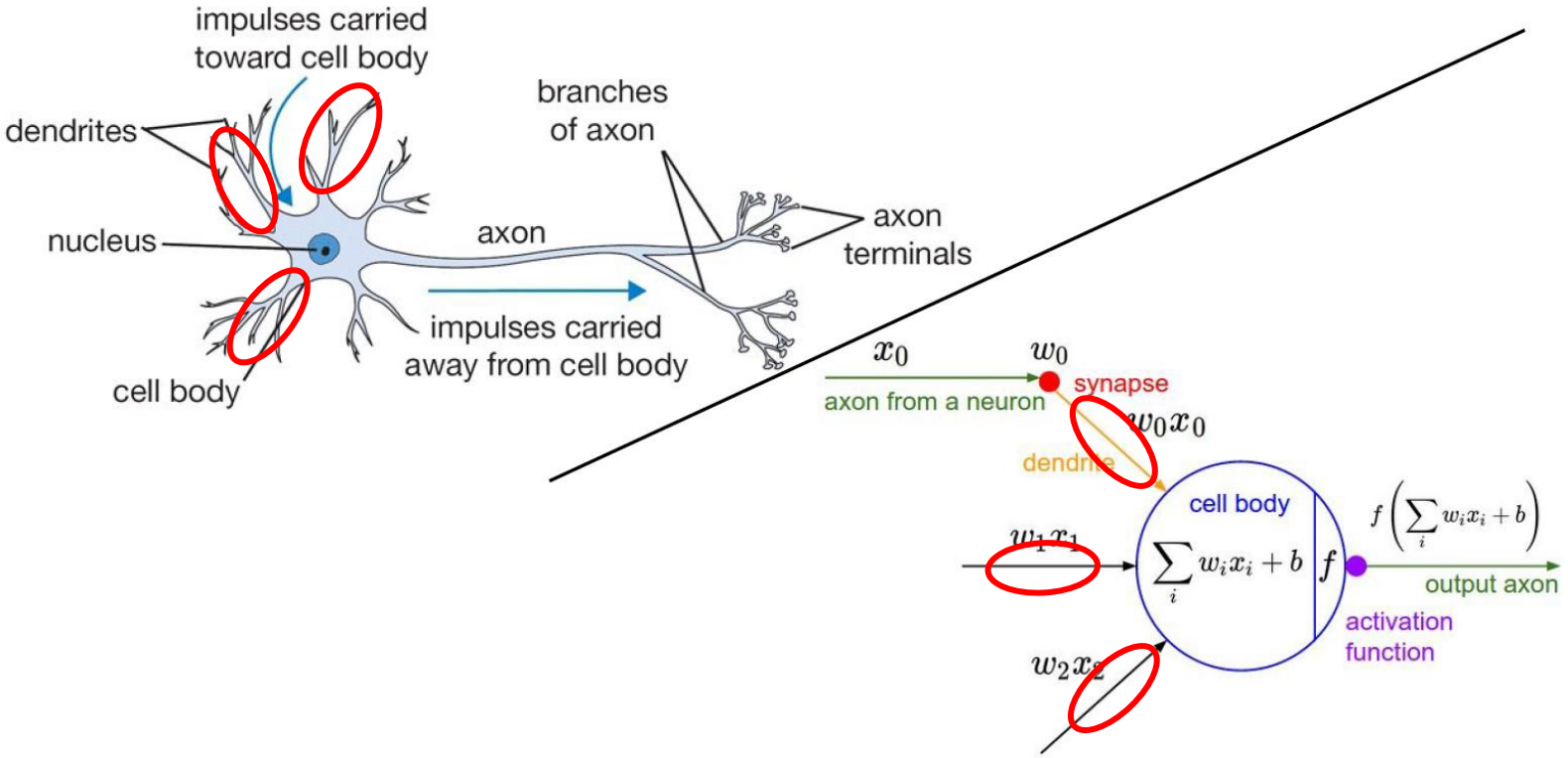
Human Brain and Neural Networks



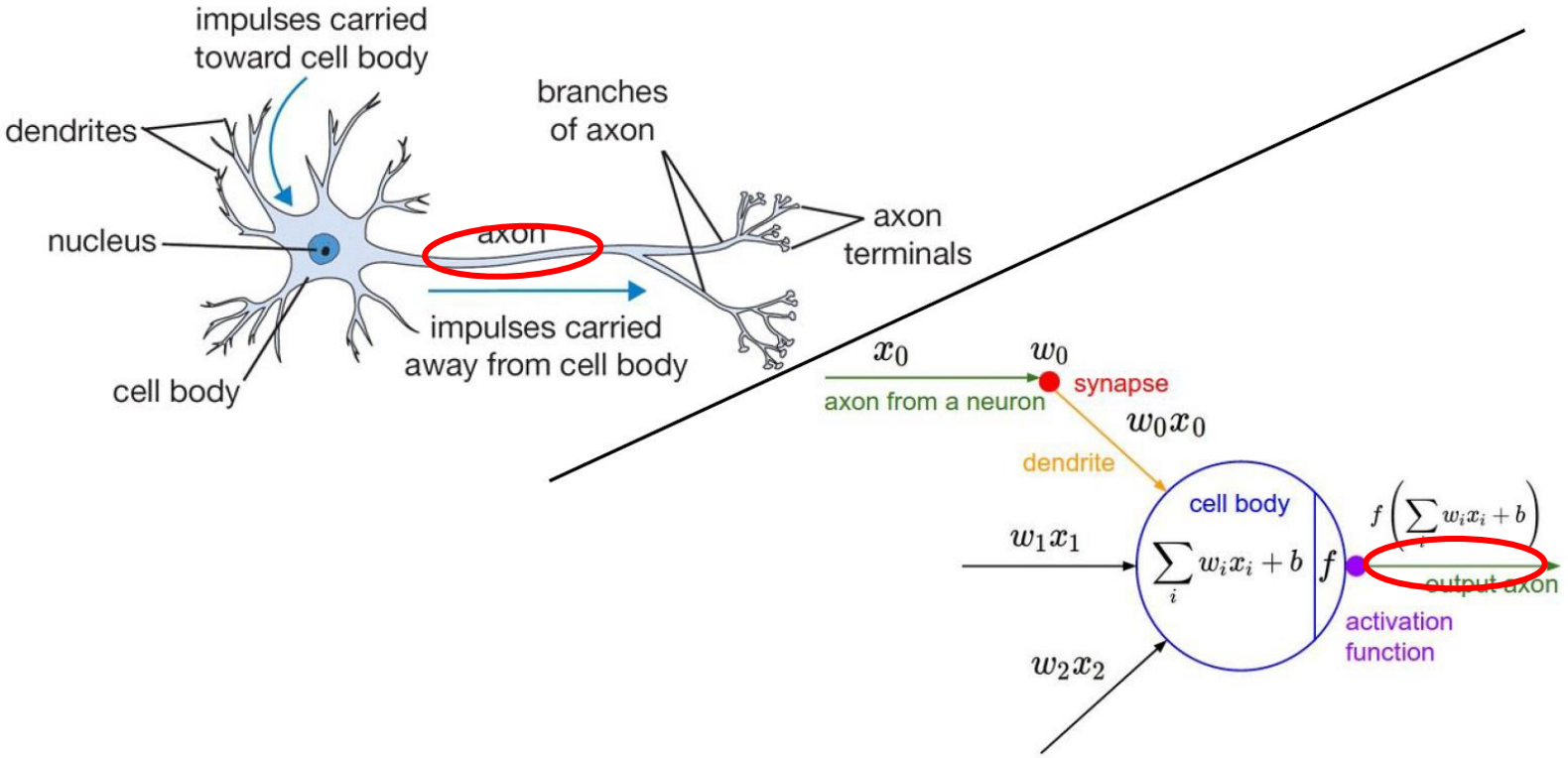
Human Brain and Neural Networks



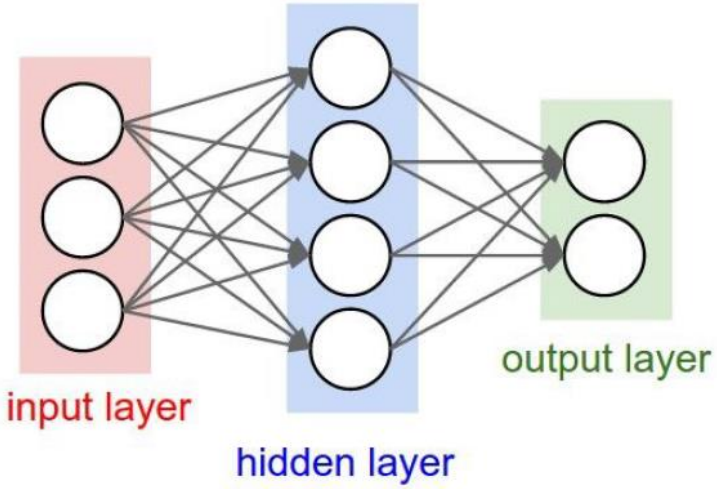
Human Brain and Neural Networks



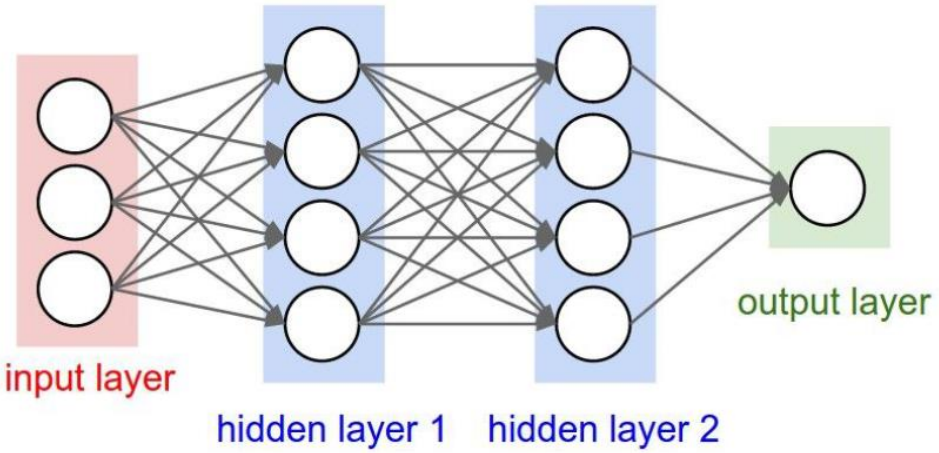
Human Brain and Neural Networks



Neural Network Architecture

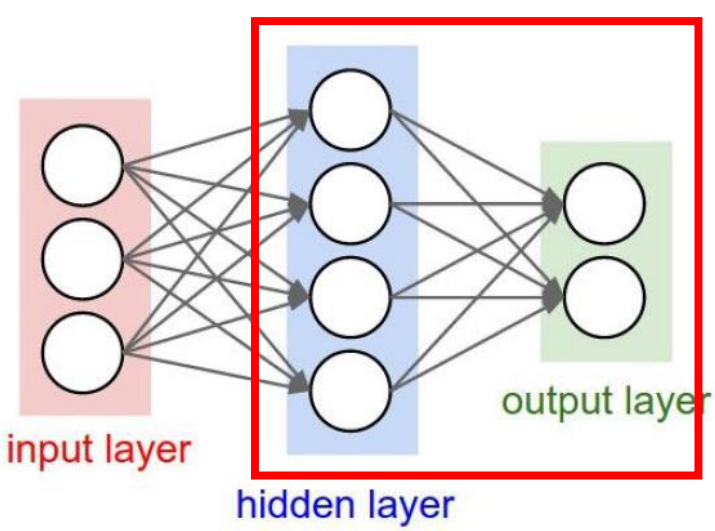


"2-layer Neural Network"
or
"1-hidden-layer Neural Network"

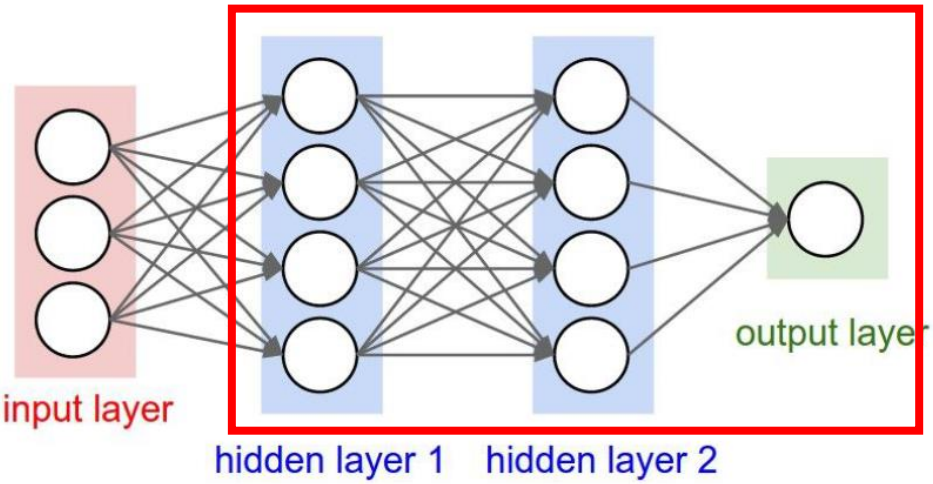


"3-layer Neural Network"
or
"2-hidden-layer Neural Network"

Neural Network Architecture

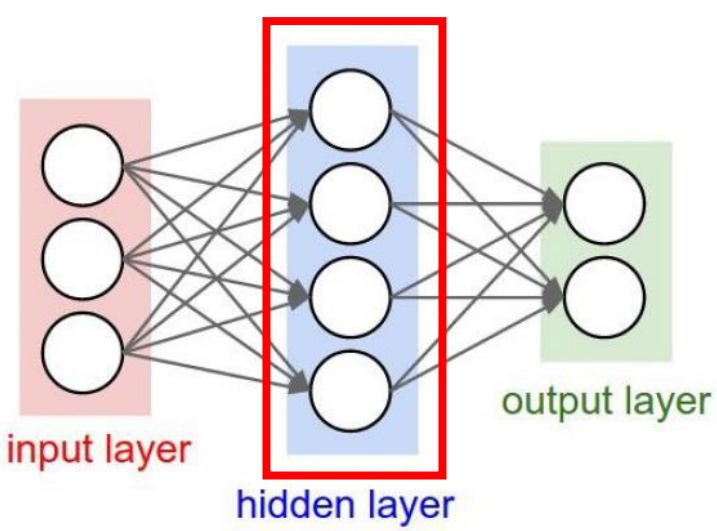


"2-layer Neural Network"
or
"1-hidden-layer Neural Network"

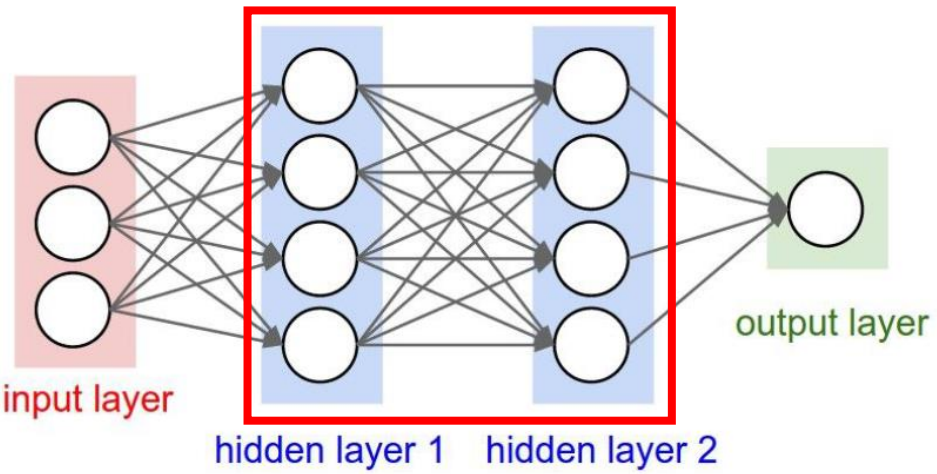


"3-layer Neural Network"
or
"2-hidden-layer Neural Network"

Neural Network Architecture

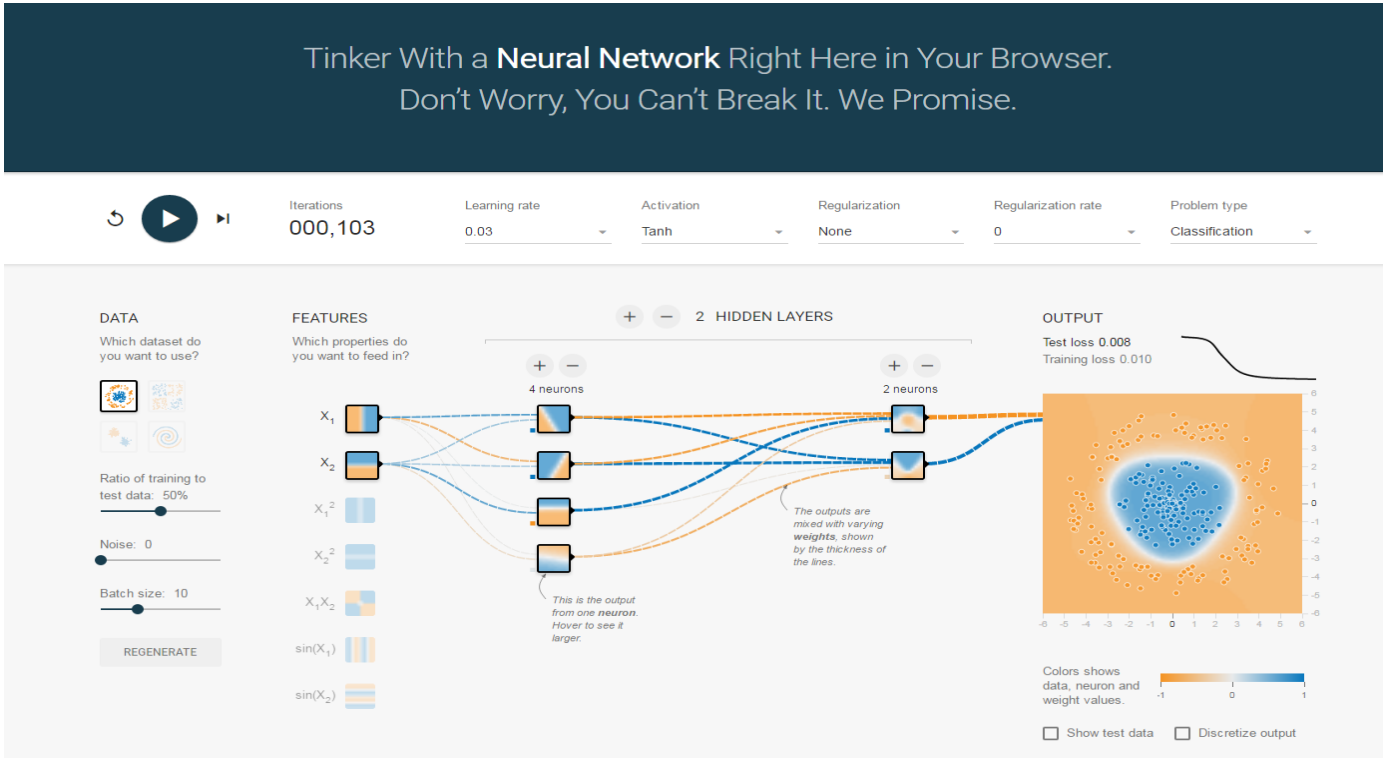


"2-layer Neural Network"
or
"1-hidden-layer Neural Network"



"3-layer Neural Network"
or
"2-hidden-layer Neural Network"

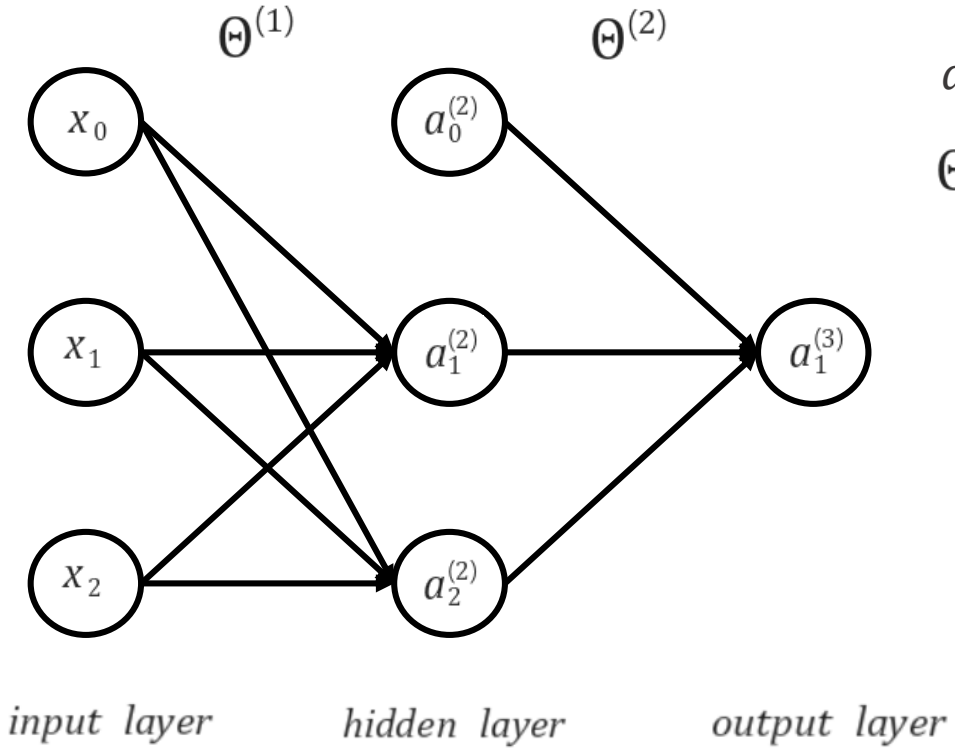
Neural Network Architecture



[A Neural Network Playground]

Forward Propagation

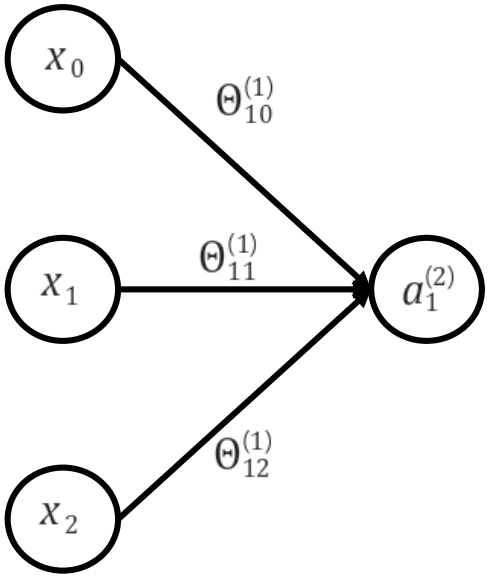
Forward Propagation



a_i^j : "activation" of i-th unit in j-th layer

$\Theta^{(j)}$: matrix of weights controlling function mapping form j-th layer to (j+1)-th layer

Forward Propagation

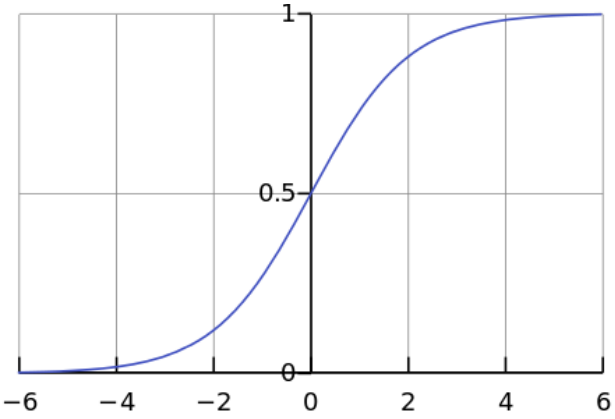


$$z_1^{(2)} = \Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2$$

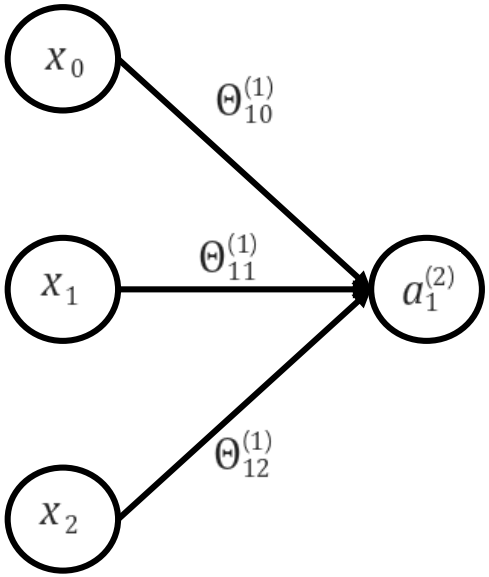
$$a_1^{(2)} = g(z_1^{(2)})$$

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

logistic sigmoid



Forward Propagation



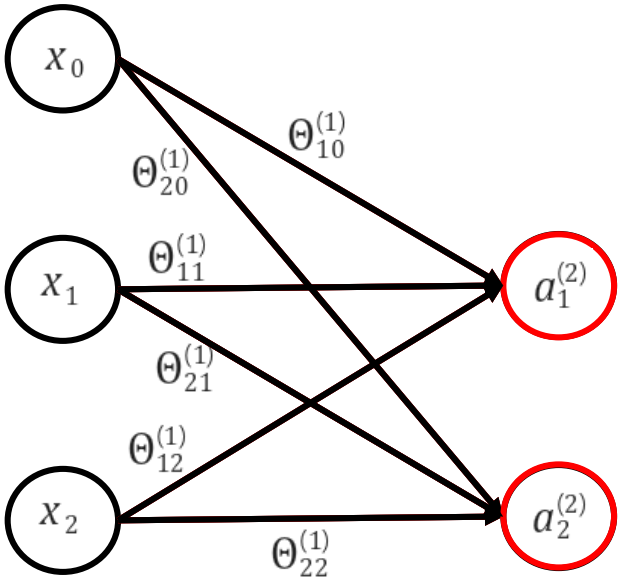
$$z_1^{(2)} = \Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2$$

$$= \begin{pmatrix} \Theta_{10}^{(1)} & \Theta_{11}^{(1)} & \Theta_{12}^{(1)} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}$$

$$a_1^{(2)} = g(z_1^{(2)})$$

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

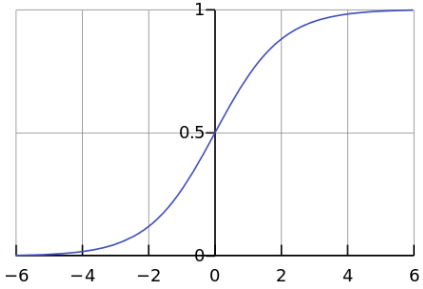
Forward Propagation



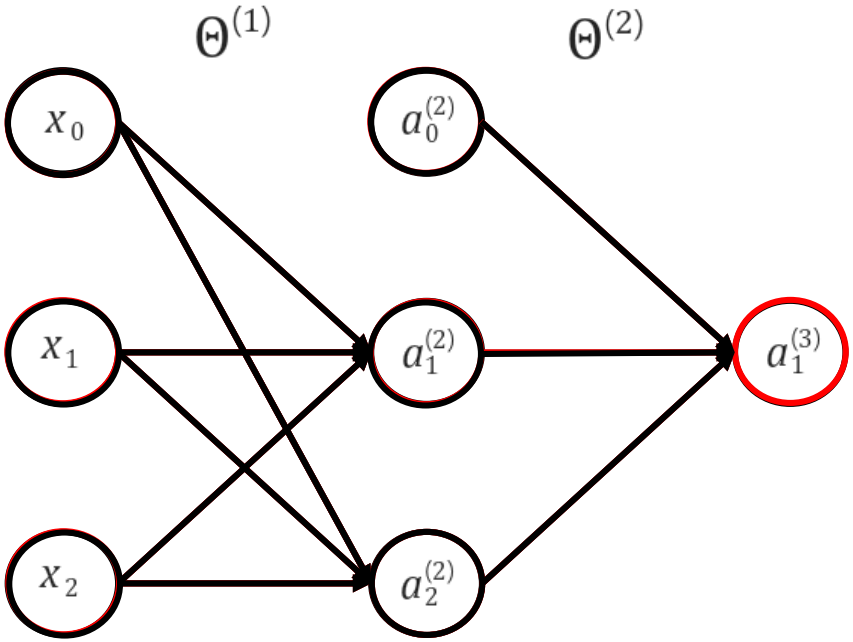
$$\begin{pmatrix} z_1^{(2)} \\ z_2^{(2)} \end{pmatrix} = \begin{pmatrix} \Theta_{10}^{(1)} & \Theta_{11}^{(1)} & \Theta_{12}^{(1)} \\ \Theta_{20}^{(1)} & \Theta_{21}^{(1)} & \Theta_{22}^{(1)} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}$$

$$\begin{pmatrix} a_1^{(2)} \\ a_2^{(2)} \end{pmatrix} = \begin{pmatrix} g(z_1^{(2)}) \\ g(z_2^{(2)}) \end{pmatrix}$$

logistic sigmoid



Forward Propagation



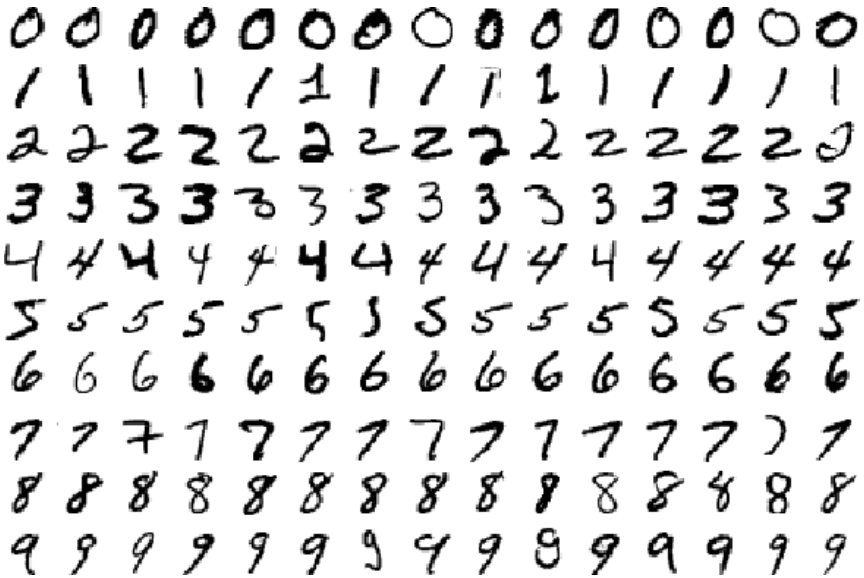
$$\begin{pmatrix} z_1^{(2)} \\ z_2^{(2)} \end{pmatrix} = \begin{pmatrix} \Theta_{10}^{(1)} & \Theta_{11}^{(1)} & \Theta_{12}^{(1)} \\ \Theta_{20}^{(1)} & \Theta_{21}^{(1)} & \Theta_{22}^{(1)} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}$$

$$\begin{pmatrix} a_1^{(2)} \\ a_2^{(2)} \end{pmatrix} = \begin{pmatrix} g(z_1^{(2)}) \\ g(z_2^{(2)}) \end{pmatrix}$$

$$z_1^{(3)} = \begin{pmatrix} \Theta_{20}^{(1)} & \Theta_{21}^{(1)} & \Theta_{22}^{(1)} \end{pmatrix} \begin{pmatrix} a_0^{(2)} \\ a_1^{(2)} \\ a_2^{(2)} \end{pmatrix}$$

$$a_1^{(3)} = g(z_1^{(3)})$$

MNIST Example

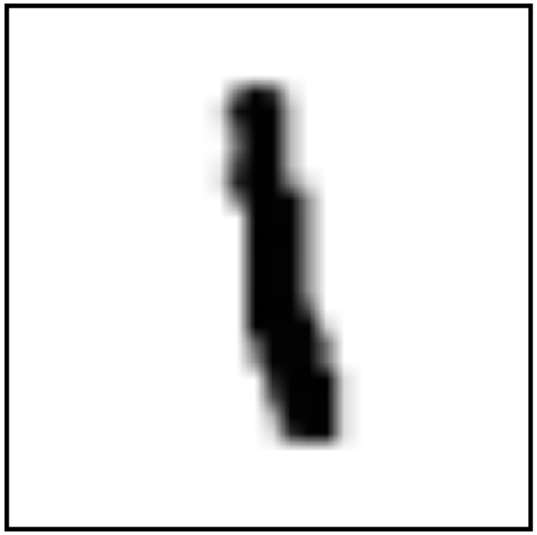


MNIST dataset of handwritten digits for 0 to 9.

It has 55,000 examples for training and 10,000 examples for testing.

The digits have been size-normalized and centered in a fixed-size image, so each sample is represented as a 2D matrix of size 28x28 with values from 0 to 1.

MNIST Example



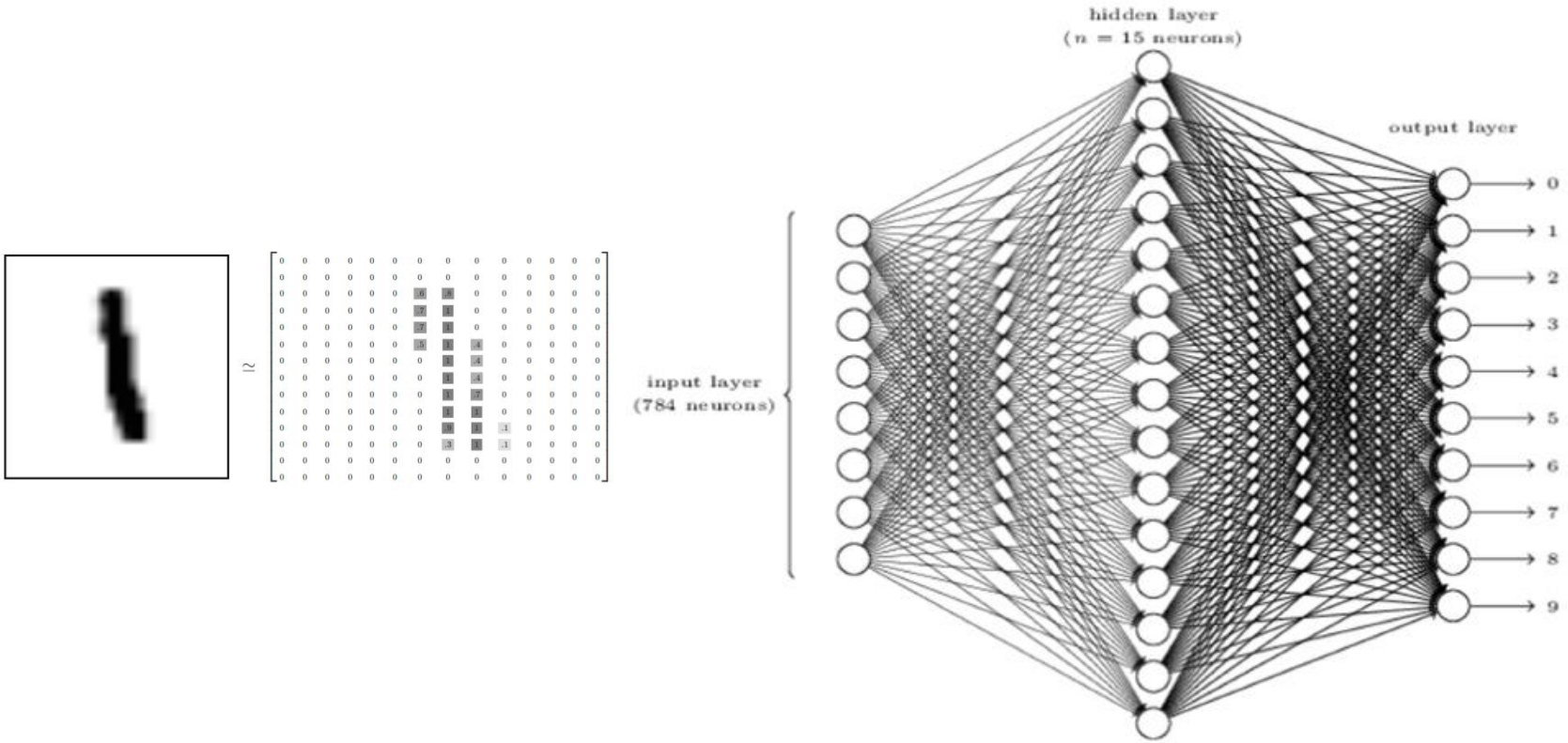
≈

28

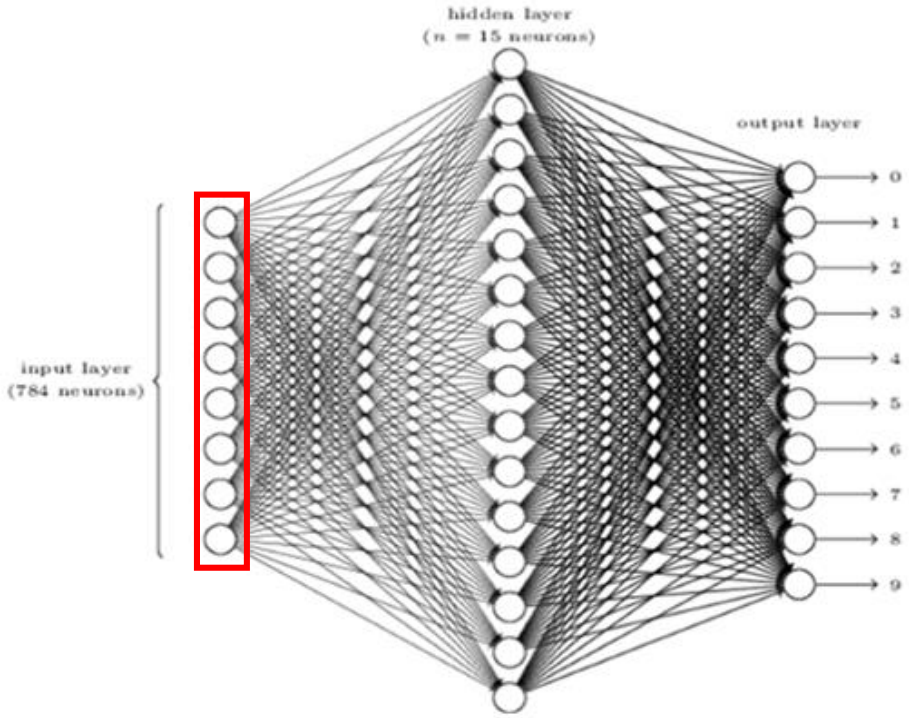
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	.6	.8	0	0	0	0	0	0
0	0	0	0	0	0	.7	1	0	0	0	0	0	0
0	0	0	0	0	0	.7	1	0	0	0	0	0	0
0	0	0	0	0	0	.5	1	.4	0	0	0	0	0
0	0	0	0	0	0	0	1	.4	0	0	0	0	0
0	0	0	0	0	0	0	1	.4	0	0	0	0	0
0	0	0	0	0	0	0	1	.7	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	.9	1	.1	0	0	0	0
0	0	0	0	0	0	0	.3	1	.1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

28

MNIST Example



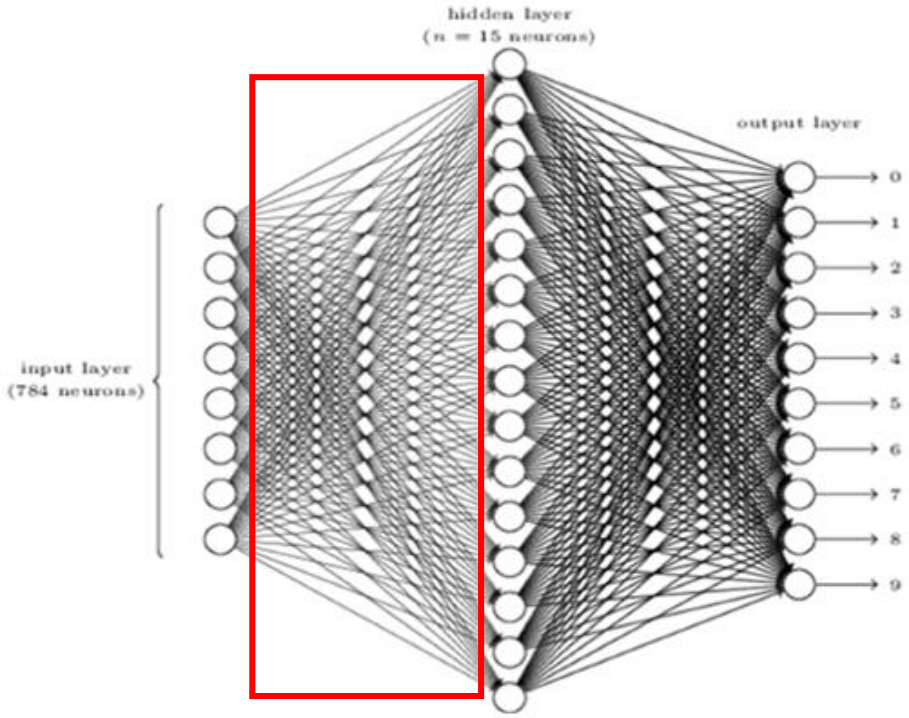
MNIST Example




```
x: 784 x 1
w1: 15 x (784 + 1)
w2: 10 x (15 + 1)


import numpy as np

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def neural_network(x, w1, w2):
    a1 = np.concatenate([x, [[1]]], 0)
    z2 = np.dot(w1, a1)
    a2 = sigmoid(z2)
    a2 = np.concatenate([a2, [[1]]], 0)
    z3 = np.dot(w2, a2)
    a3 = sigmoid(z3)
    return a3
```

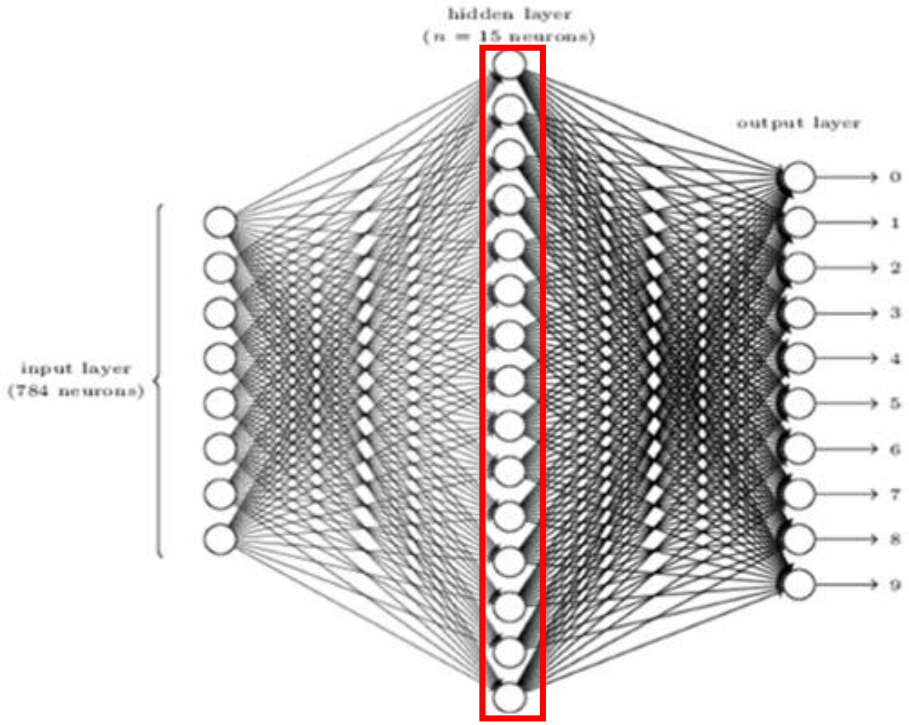
785 x 1

MNIST Example




```
x: 784 x 1                    a1: 785 x 1
w1: 15 x (784 + 1)
w2: 10 x (15 + 1)


import numpy as np

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def neural_network(x, w1, w2):
    a1 = np.concatenate([x, [[1]]], 0)
    z2 = np.dot(w1, a1)
    a2 = sigmoid(z2)
    a2 = np.concatenate([a2, [[1]]], 0)
    z3 = np.dot(w2, a2)
    a3 = sigmoid(z3)
    return a3
```

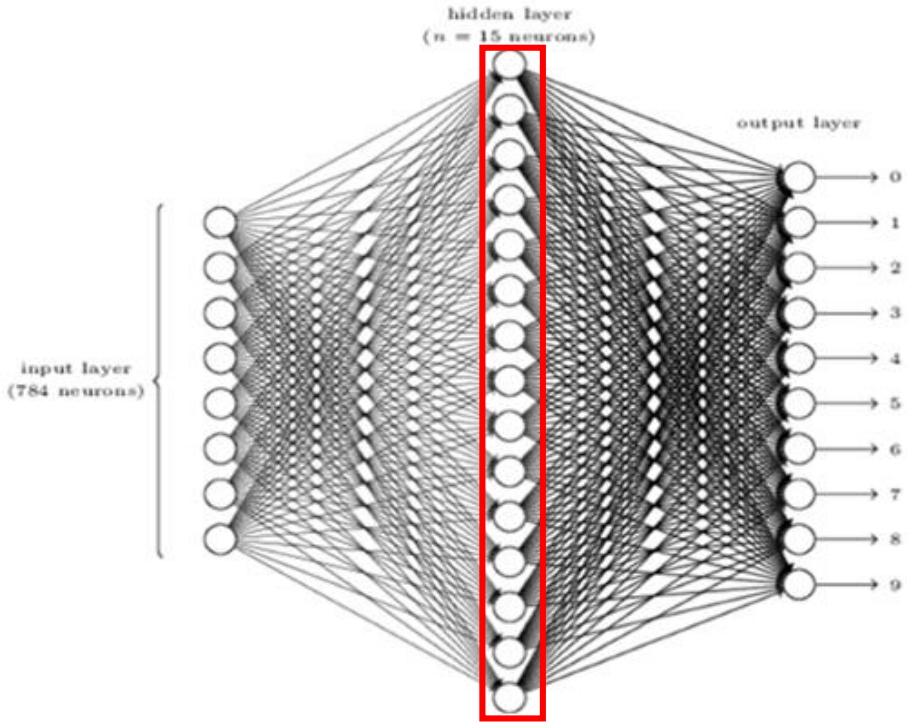
MNIST Example




```
x: 784 x 1                    a1: 785 x 1
w1: 15 x (784 + 1)
w2: 10 x (15 + 1)


import numpy as np

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def neural_network(x, w1, w2):
    a1 = np.concatenate([x, [[1]]], 0)
    z2 = np.dot(w1, a1)
    a2 = sigmoid(z2)
    a2 = np.concatenate([a2, [[1]]], 0)
    z3 = np.dot(w2, a2)
    a3 = sigmoid(z3)
    return a3
```

15 x 1

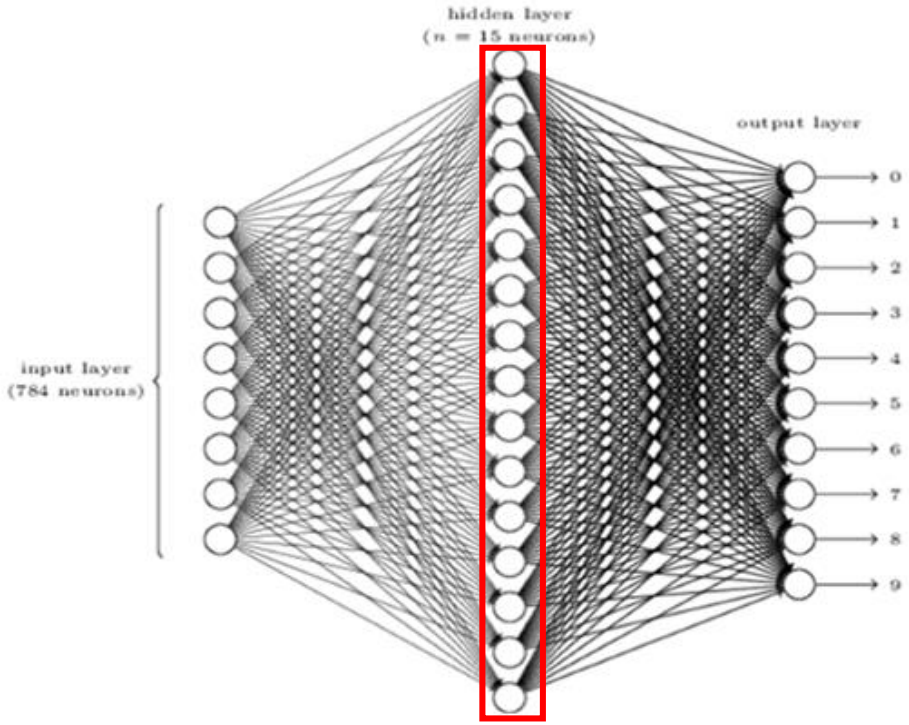
MNIST Example




```
x: 784 x 1                    a1: 785 x 1
w1: 15 x (784 + 1)
w2: 10 x (15 + 1)


import numpy as np

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def neural_network(x, w1, w2):
    a1 = np.concatenate([x, [[1]]], 0)
    z2 = np.dot(w1, a1)
    a2 = sigmoid(z2)
    a2 = np.concatenate([a2, [[1]]], 0)
    z3 = np.dot(w2, a2)
    a3 = sigmoid(z3)
    return a3
```

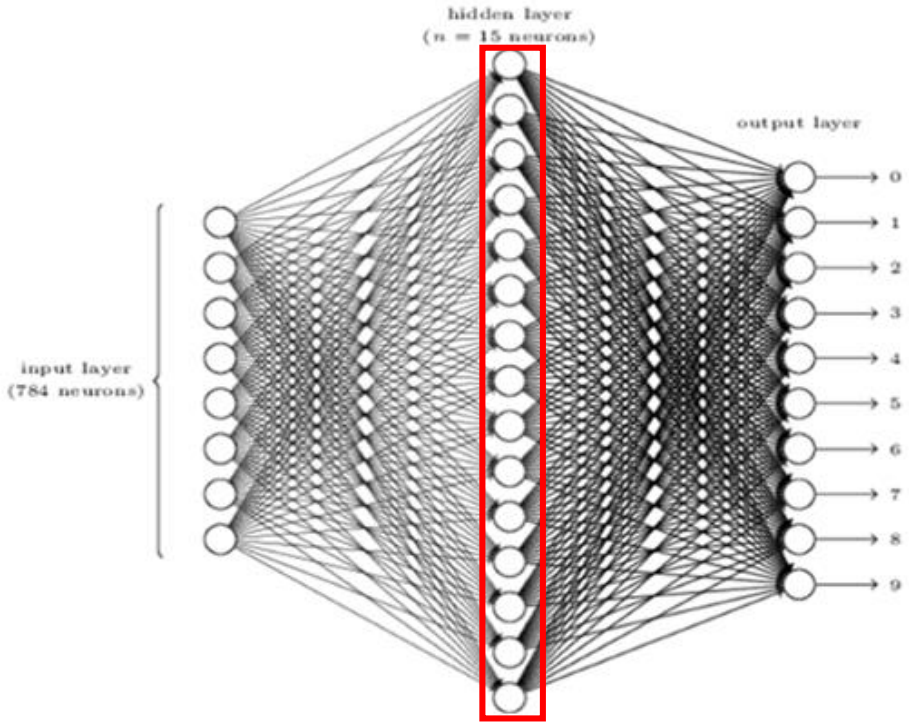
15 x 1

MNIST Example




```
x: 784 x 1                    a1: 785 x 1
w1: 15 x (784 + 1)
w2: 10 x (15 + 1)


import numpy as np

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def neural_network(x, w1, w2):
    a1 = np.concatenate([x, [[1]]], 0)
    z2 = np.dot(w1, a1)
    a2 = sigmoid(z2)                          16 x 1
    a2 = np.concatenate([a2, [[1]]], 0)
    z3 = np.dot(w2, a2)
    a3 = sigmoid(z3)
    return a3
```

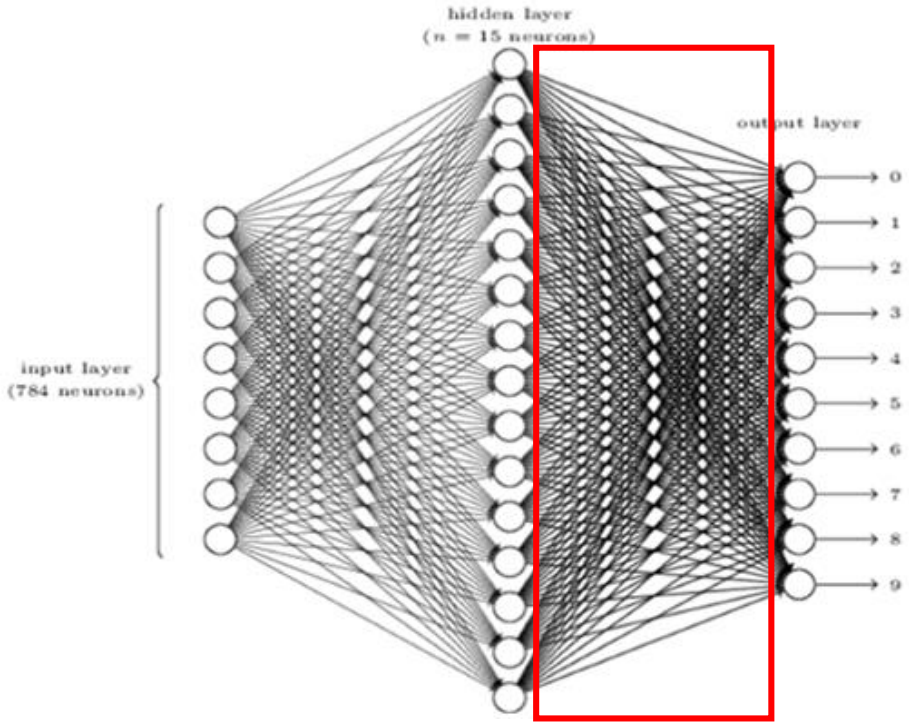
MNIST Example




```
x: 784 x 1                  a1: 785 x 1
w1: 15 x (784 + 1)          a2: 16 x 1
w2: 10 x (15 + 1)
```

```
import numpy as np

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def neural_network(x, w1, w2):
    a1 = np.concatenate([x, [[1]]], 0)
    z2 = np.dot(w1, a1)
    a2 = sigmoid(z2)
    a2 = np.concatenate([a2, [[1]]], 0)
    z3 = np.dot(w2, a2)
    a3 = sigmoid(z3)
    return a3
```

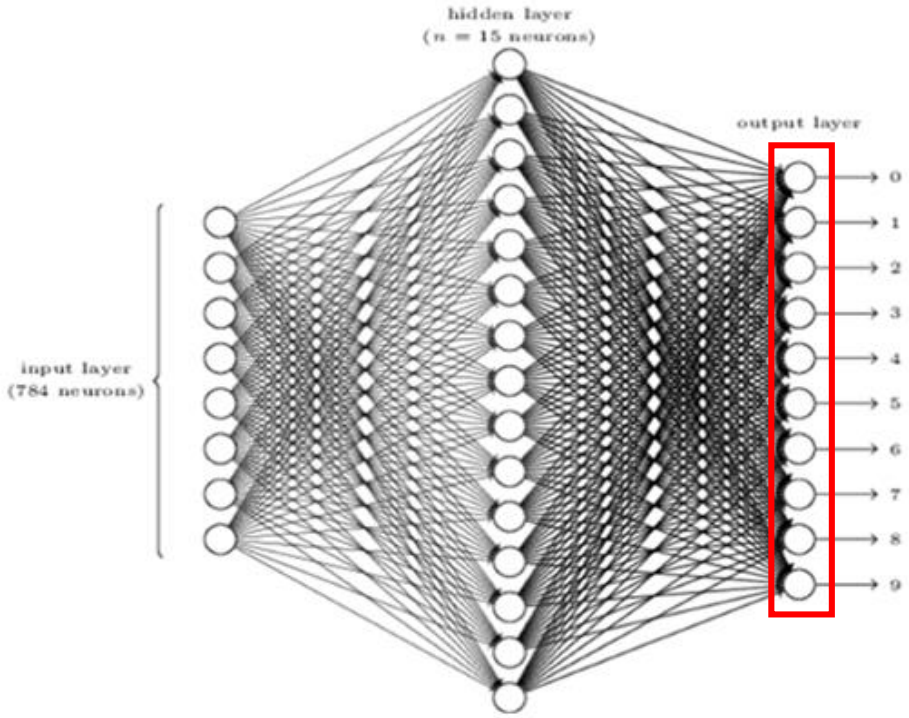
MNIST Example




```
x: 784 x 1                  a1: 785 x 1
w1: 15 x (784 + 1)          a2: 16 x 1
w2: 10 x (15 + 1)
```

```python
import numpy as np

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def neural_network(x, w1, w2):
    a1 = np.concatenate([x, [[1]]], 0)
    z2 = np.dot(w1, a1)
    a2 = sigmoid(z2)
    a2 = np.concatenate([a2, [[1]]], 0)
    z3 = np.dot(w2, a2)
    a3 = sigmoid(z3)
    return a3
```

MNIST Example

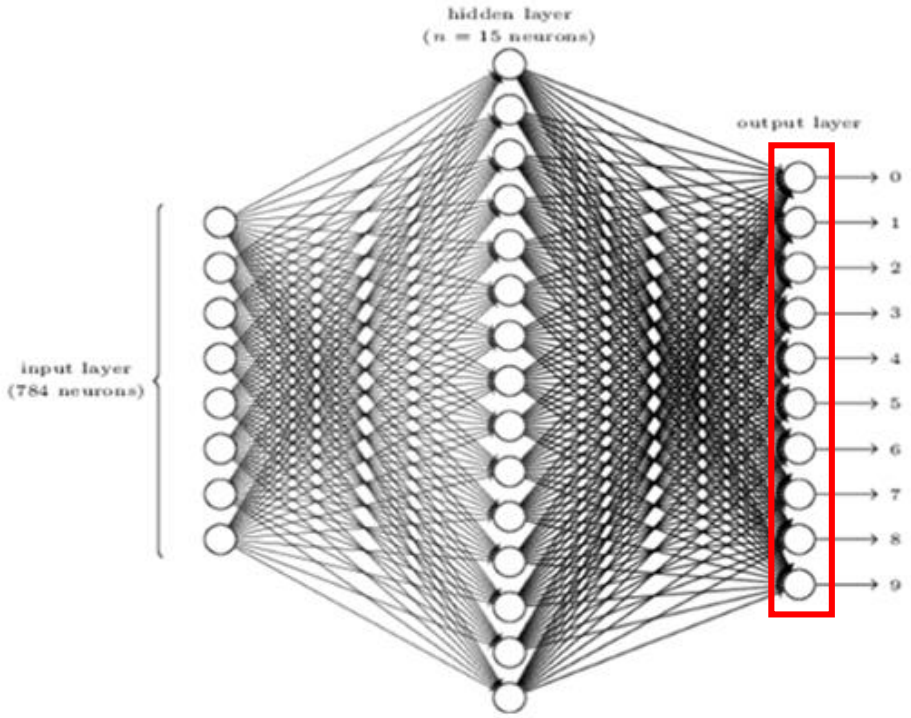



```
x: 784 x 1                  a1: 785 x 1
w1: 15 x (784 + 1)          a2: 16 x 1
w2: 10 x (15 + 1)
```

```
import numpy as np

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def neural_network(x, w1, w2):
    a1 = np.concatenate([x, [[1]]], 0)
    z2 = np.dot(w1, a1)
    a2 = sigmoid(z2)
    a2 = np.concatenate([a2, [[1]]], 0)
    z3 = np.dot(w2, a2)
    a3 = sigmoid(z3)
    return a3
```

10 x 1

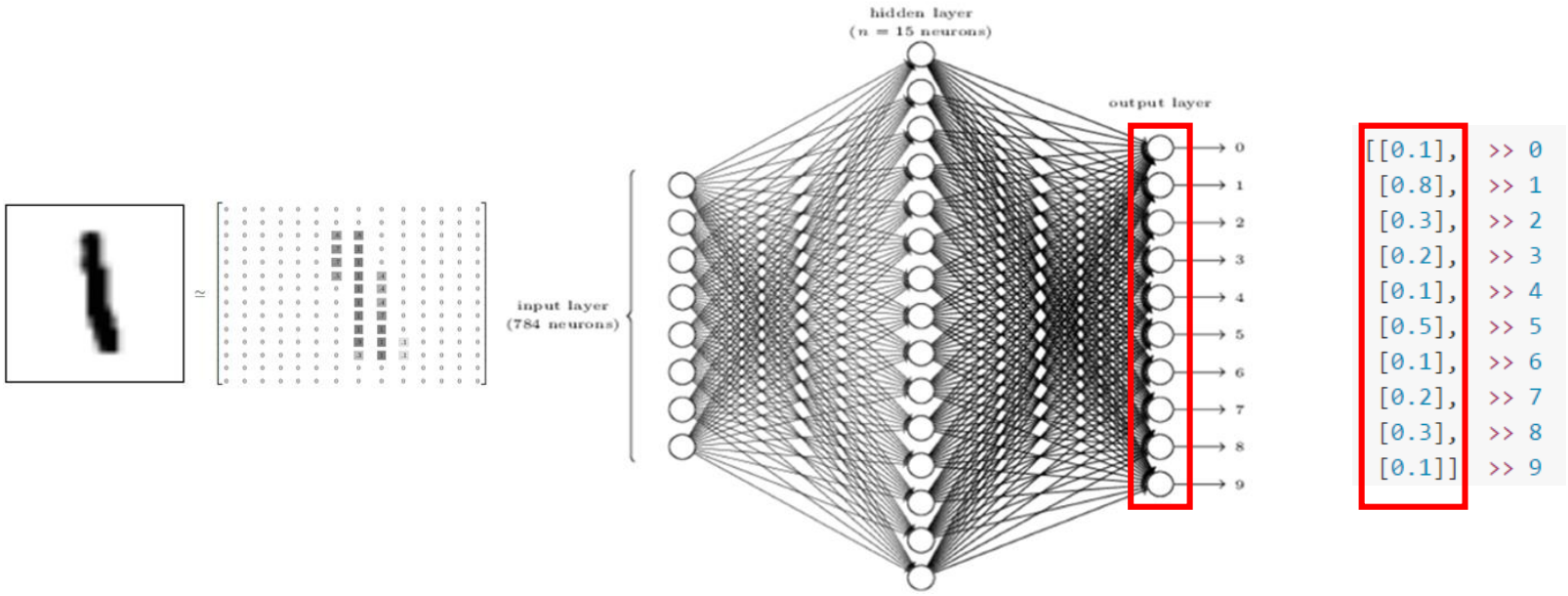
MNIST Example




```
x: 784 x 1                  a1: 785 x 1
w1: 15 x (784 + 1)          a2: 16 x 1
w2: 10 x (15 + 1)


import numpy as np

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def neural_network(x, w1, w2):
    a1 = np.concatenate([x, [[1]]], 0)
    z2 = np.dot(w1, a1)
    a2 = sigmoid(z2)
    a2 = np.concatenate([a2, [[1]]], 0)
    z3 = np.dot(w2, a2)
    a3 = sigmoid(z3)
    return a3
```

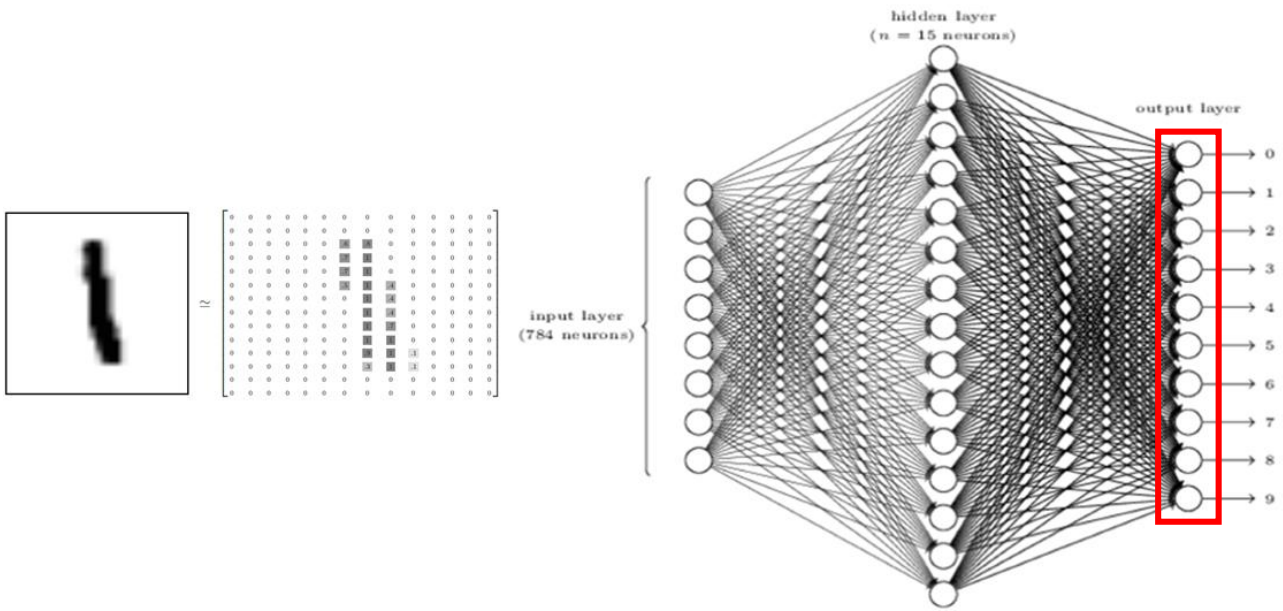
10 x 1

MNIST Example




```
[[0.1],  >> 0
 [0.8],  >> 1
 [0.3],  >> 2
 [0.2],  >> 3
 [0.1],  >> 4
 [0.5],  >> 5
 [0.1],  >> 6
 [0.2],  >> 7
 [0.3],  >> 8
 [0.1]]  >> 9
```

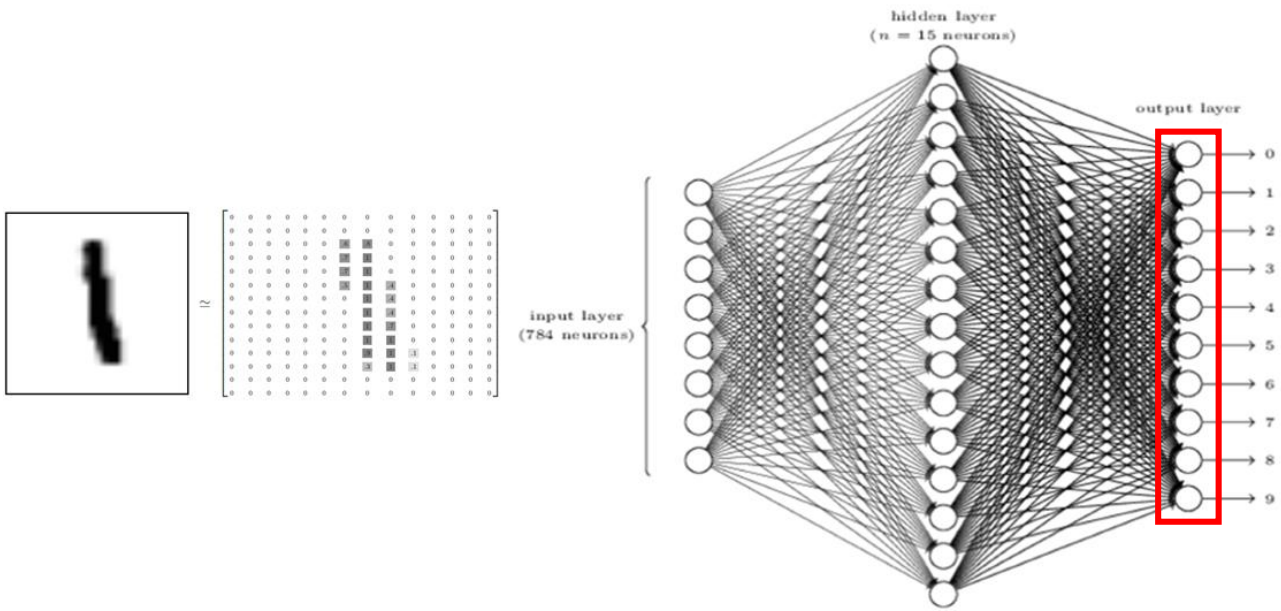
MNIST Example



good case !

```
[[0.1],  >> 0
 [0.8],  >> 1
 [0.3],  >> 2
 [0.2],  >> 3
 [0.1],  >> 4
 [0.5],  >> 5
 [0.1],  >> 6
 [0.2],  >> 7
 [0.3],  >> 8
 [0.1]]  >> 9
```

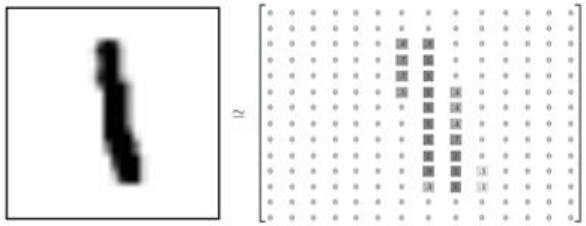
MNIST Example




```
[[0.1],  >> 0
 [0.2],  >> 1
 [0.3],  >> 2
 [0.2],  >> 3
 [0.1],  >> 4
 [0.8],  >> 5
 [0.1],  >> 6
 [0.2],  >> 7
 [0.3],  >> 8
 [0.1]]  >> 9
```

bad case !

MNIST Example



prediction

```
[[0.1],  >> 0
 [0.2],  >> 1
 [0.3],  >> 2
 [0.2],  >> 3
 [0.1],  >> 4
 [0.8],  >> 5
 [0.1],  >> 6
 [0.2],  >> 7
 [0.3],  >> 8
 [0.1]]  >> 9
```

−

target

```
[[0],  >> 0
 [1],  >> 1
 [0],  >> 2
 [0],  >> 3
 [0],  >> 4
 [0],  >> 5
 [0],  >> 6
 [0],  >> 7
 [0],  >> 8
 [0]]  >> 9
```

^{2} =

```
[[0.01], >> 0
 [0.64],  >> 1
 [0.09],  >> 2
 [0.04],  >> 3
 [0.01],  >> 4
 [0.64],  >> 5
 [0.1],   >> 6
 [0.04],  >> 7
 [0.09],  >> 8
 [0.01]]  >> 9
```

```
squared error: 1.67
```

we should minimize this error

Cost Function

Assume we have 1 data and dimension of output vector is n.

Regression

Mean Squared Error

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

```
[[0.1],  >> 0
 [0.2],  >> 1
 [0.3],  >> 2
 [0.2],  >> 3
 [0.1],  >> 4
 [0.8],  >> 5
 [0.1],  >> 6
 [0.2],  >> 7
 [0.3],  >> 8
 [0.1]]  >> 9
```

```
[[0],  >> 0
 [1],  >> 1
 [0],  >> 2
 [0],  >> 3
 [0],  >> 4
 [0],  >> 5
 [0],  >> 6
 [0],  >> 7
 [0],  >> 8
 [0]]  >> 9
```

Classification

Sigmoid Cross Entropy Loss

$$\sum_{i=1}^{n} -y_i \log \hat{y}_i - (1-y_i)\log(1-\hat{y}_i)$$

Softmax Cross Entropy Loss

$$\sum_{i=1}^{n} -y_i \log \hat{y}_i$$

Cost Function

Assume we have 1 data and dimension of output vector is n.

Regression

Mean Squared Error

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

Classification

Sigmoid Cross Entropy Loss

$$\sum_{i=1}^{n} -y_i \log \hat{y}_i - (1-y_i)\log(1-\hat{y}_i)$$

when using sigmoid function in last layer

Softmax Cross Entropy Loss

$$\sum_{i=1}^{n} -y_i \log \hat{y}_i$$

Cost Function

Assume we have 1 data and dimension of output vector is n.

Regression

Mean Squared Error

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

Classification

Sigmoid Cross Entropy Loss

$$\sum_{i=1}^{n} -y_i \log \hat{y}_i - (1-y_i)\log(1-\hat{y}_i)$$

Softmax Cross Entropy Loss

$$\sum_{i=1}^{n} -y_i \log \hat{y}_i$$

when using softmax function in last layer (Don't worry about this, we will cover this later)

Until Next Class

- Until this Thursday's class, watch 2017 cs231n Lectures:
  - Lecture 1 (optional): https://youtu.be/vT1JzLTH4G4
  - Lecture 2: https://youtu.be/OoUX-nOEjG0
  - Lecture 3: https://youtu.be/h7iBpEHGVNc
- Please post your questions regarding the above lectures at www.slido.com (access code: kudl).
- We will start with the quiz from the next class.