Deep Machine Learning

Dr. Huseyin Kusetogullari

Course Responsible

- Dr. Huseyin KUSETOGULLARI
  (Associate Professor@BTH)

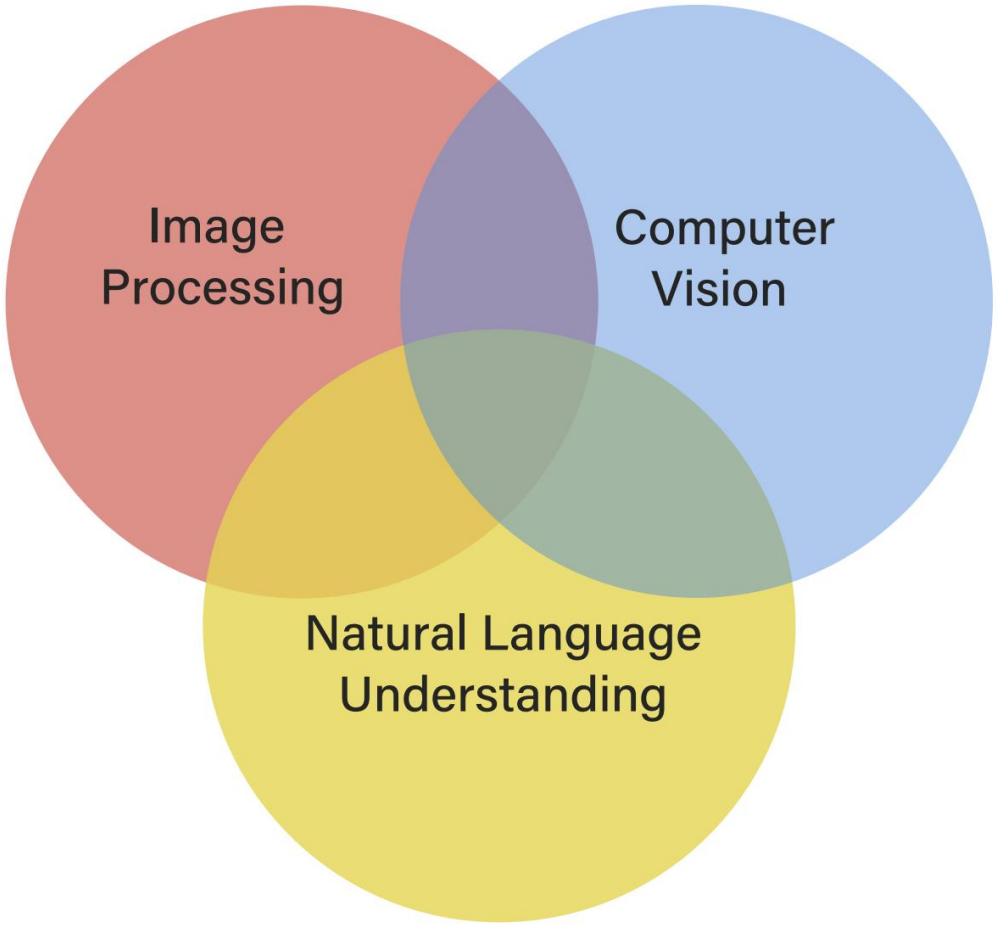
E-mail: huseyin.kusetogullari@bth.se

A little about me

- Received my PhD in 2012 from Warwick University, UK.
- Worked as Post-Doc Researcher
  - Warwick University, UK
  - Aberystwyth University, UK
  - BTH, Sweden
- Assistant Professor
  - Gediz University, Turkey
  - Skovde University, Sweden
  - Herriot- Watt University, Dubai, UAE
  - BTH, Sweden
- Associate Professor, 2024 – Present, BTH
- Master in AI/ML Program Manager 2024- Present

Research Interests

- I have been working on visual data.
- My research interests span a diverse set of topics such as object detection, segmentation and classification in images and videos.



Other Teachers

- Dr. Amir Yavariabdi

E-mail: amir.yavariabdi@bth.se

- He will teach Lectures 6, 7, 8, and 9

Course Assistants

- Tharuka Kasthuriarachchige

E-mail: tak@bth.se

Course syllabus

The course syllabus is available at https://canvas.bth.se

Please check the syllabus in detail including

learning outcomes	description of the course	assessment part and credit point	exercises	project	main book and other materials

Course syllabus

On completion of the course, you are expected to:

- explain fundamentals of deep machine learning and key subject areas
- understand and possess advanced knowledge within the area of deep machine learning
- understand real world applications of deep machine learning methods

– learn not only the theory and mathematics but also see how it is applied in industry.
– practice implementing deep learning methods in Python and in Keras, Pytorch and TensorFlow.
– likely find creative ways to apply deep learning to your work.

Course Schedule

- https://cloud.timeedit.net/bth/web/sched1/ri179912X35Z03Q6Zq6gj590y40Q6Y6n90ZgxY6Qw77l3079555l0753114Y697QQ.html

Assignments and Examination

Assignments

- **2 Assignments**
- **Will be available next week**
- **Submission deadline of both assignments is 15th May, will be available on Canvas**

Submissions

- **Implementation (if required)**
- **Report (if required)**

Project

- **Project Template can be found on Canvas.**
- **Report Submission Deadline (25th May)**
- **Can be done individually or as a team with maximum of two students.**
- **Project Presentations (10 minutes presentation and demonstration and 5 minutes Q&A)**

About Lectures

There are

- 9 lectures (will be taught on campus)
- One or more relevant research papers will be shared before lecture starts.
- 3 Labs
  - **Lab 1:** Image Classification
  - **Lab 2:** Machine Translation
  - **Lab 3:** Application of Generative Adversarial Networks (GANs)
- Project presentations
  - 30th May

3-week rule

- Don’t forget the 3-week rule in Canvas.
- Check for inactive students in Canvas, and remove inactive students.

Labs-Exercises

- There will be three different Labs:
  - **First one is to apply deep learning methods and design a deep learning model for image classification**.
  - **Second one is to use deep learning model for machine translation**.
  - **Third one is to apply Generative Adversarial Networks (GANs)**.
  - Whole or parts of implementations will be provided before lab starts.
- **The labs will be on campus and not mandatory to attend**.
- * **Recommended**: You can bring your own laptop.

Scope of this course

Artificial Neural Networks,

Understanding application of gradient descents to update the hyperparameters in NNs,

Activation functions, regularization, cost functions, optimization, and data normalization,

Deep machine learning,

CNNs: operators, drop out, convolutional layers,

Deep Recurrent, Long Short-Term Memory, and Recursive Networks,

Deep Belief Networks,

Advanced Deep classification methods: VGG, Resnet and DensNet,

Autoencoders: encoding and decoding,

Adversarial Learning and Generative Adversarial Networks (GANs),

Applications of deep learning methods in different domains, e.g., use of deep learning methods in natural language processing and computer vision

Purpose of this course

Learn how to apply deep learning methods;

Learn how to design new deep learning methods;

Understand the basic methods for problem formulation;

Understand the hyperparameters in DL;

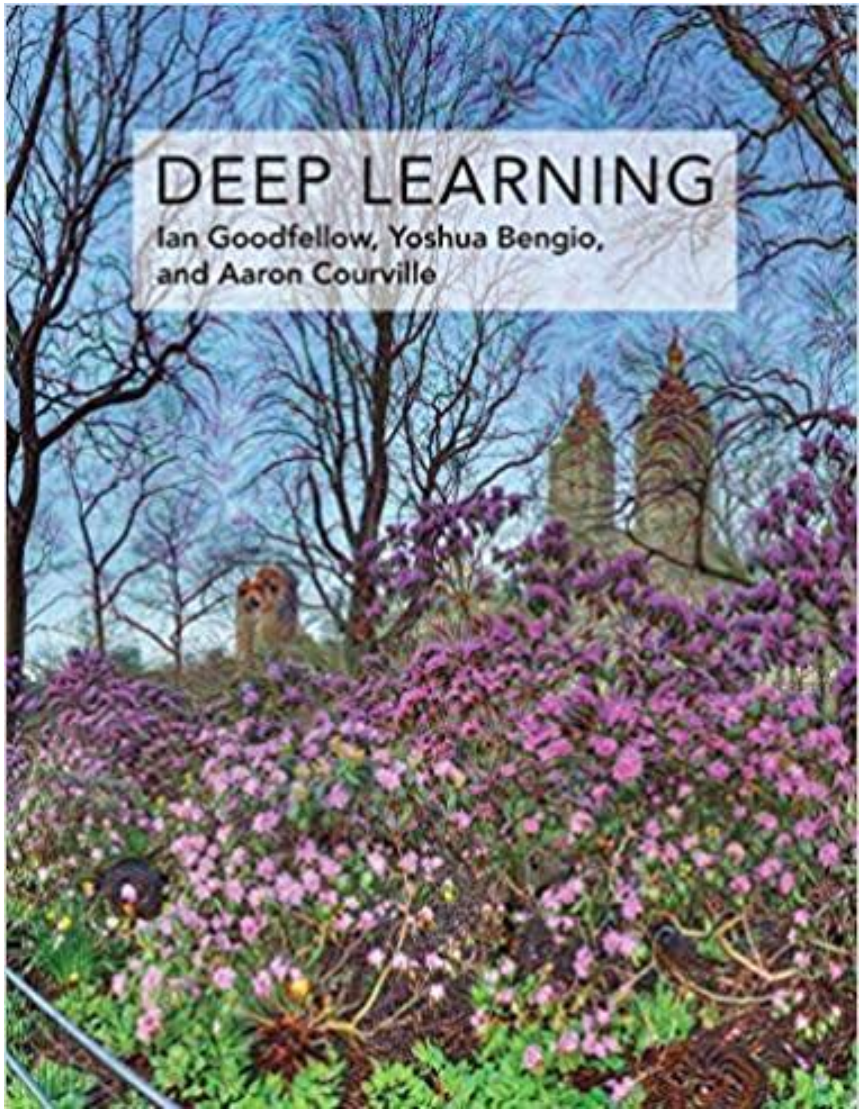
Implementing deep learning architectures;

Know some basic concepts related to optimization, prediction, pattern recognition and deep learning.

Book

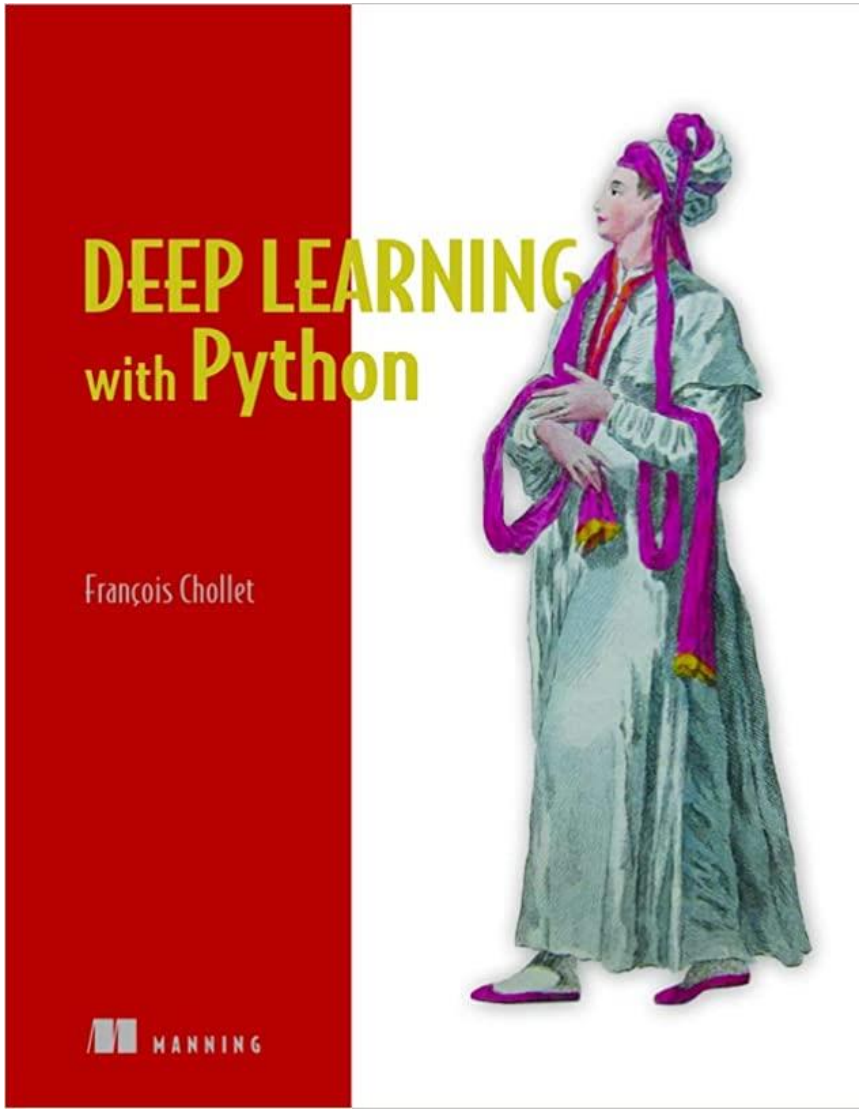
Deep Learning
Publisher: MIT Press
Authors: Ian Goodfellow, Yoshua Bengio, Aaron Courville
ISBN: 978-0262035613
Year: 2016

In addition, we will upload all the slides and other materials on the Canvas.



Book-2

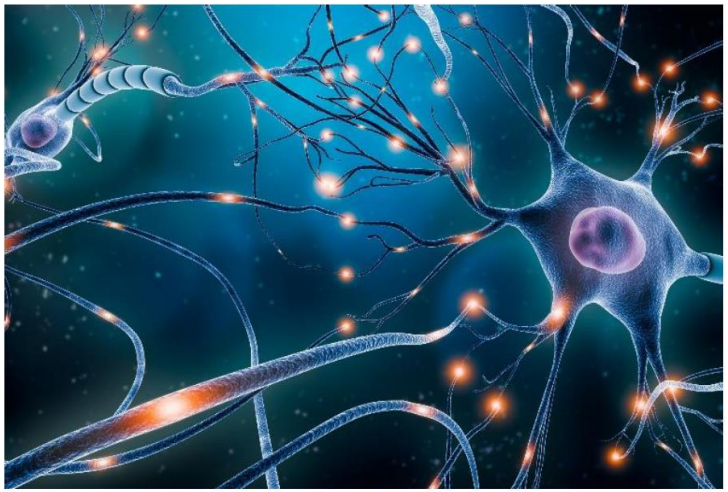
Deep Learning with Python Second Edition
Author: Francois Chollet
Publisher: Manning
ISBN: 978-1617296864
Year: 2021



Recommended research papers

- You can find recommended research papers on the course webpage in Canvas.

Human Intelligence



The Brain's Role in Processing and Decision-Making

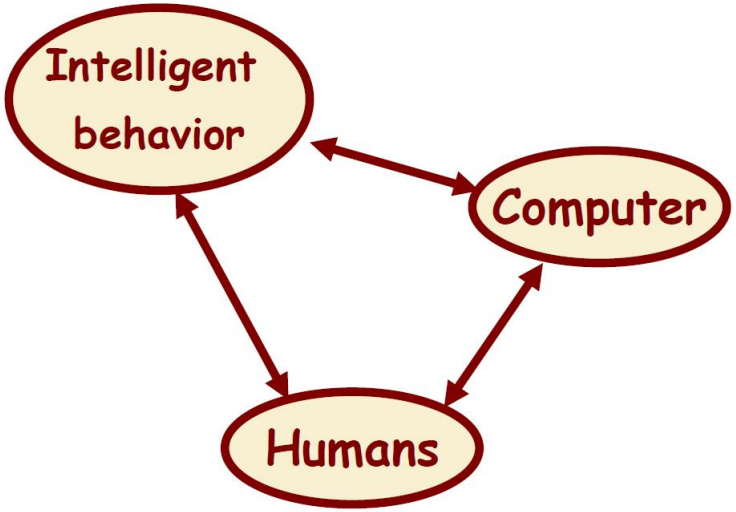
Emotions and Social Intelligence

Learning and Memory

Creativity and Problem-Solving

The human brain contains approximately **86 billion neurons [1]**. These neurons are the brain's primary cells, responsible for transmitting information through electrical and chemical signals.

[1]https://pmc.ncbi.nlm.nih.gov/articles/PMC2776484/


Intelligent behavior
Computer
Humans

AI, ML and DL

Artificial Intelligence (AI) *(The Broadest Concept)*

AI refers to the broad concept of machines being able to perform tasks that normally require human intelligence. Examples include decision-making, speech recognition, optimization, search algorithms, agents, understanding language, planning, problem-solving and many more.

Machine Learning (ML) *(A Subset of AI)*

ML is a subset of AI that involves teaching computers how to learn from data and improve their performance without being explicitly programmed for specific tasks.

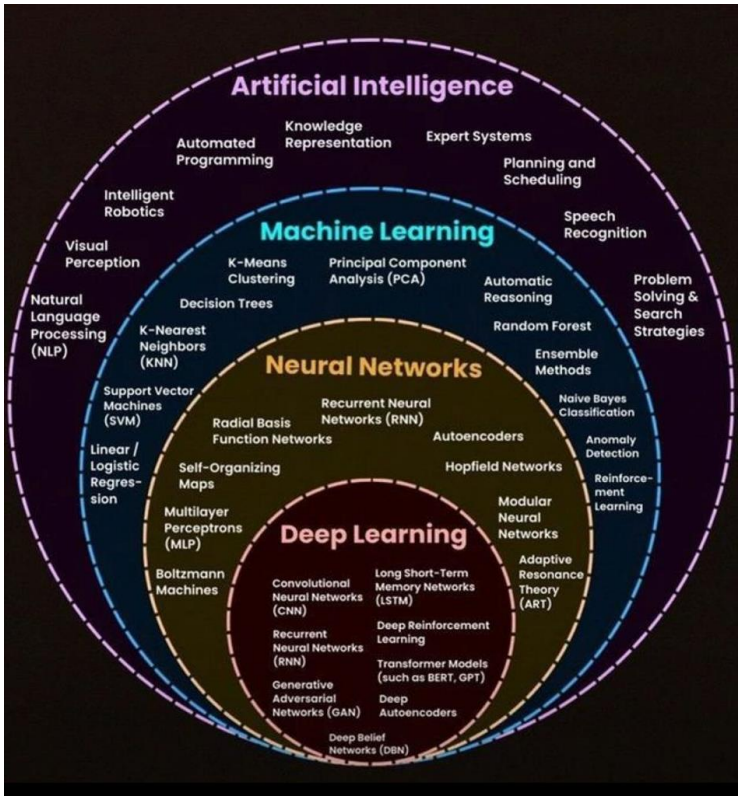
- **ML** models use algorithms to learn from data.
- **ML** relies on patterns in data to make predictions or decisions.
- Typically involves methods like *supervised learning, unsupervised learning, and reinforcement learning*.
- Can run with CPUs and GPUs

Deep Learning (DL) ***(A Subset of Machine Learning)***

DL is a specialized subset of ML that involves artificial neural networks with multiple layers (hence "deep") to learn complex patterns from vast amounts of data.

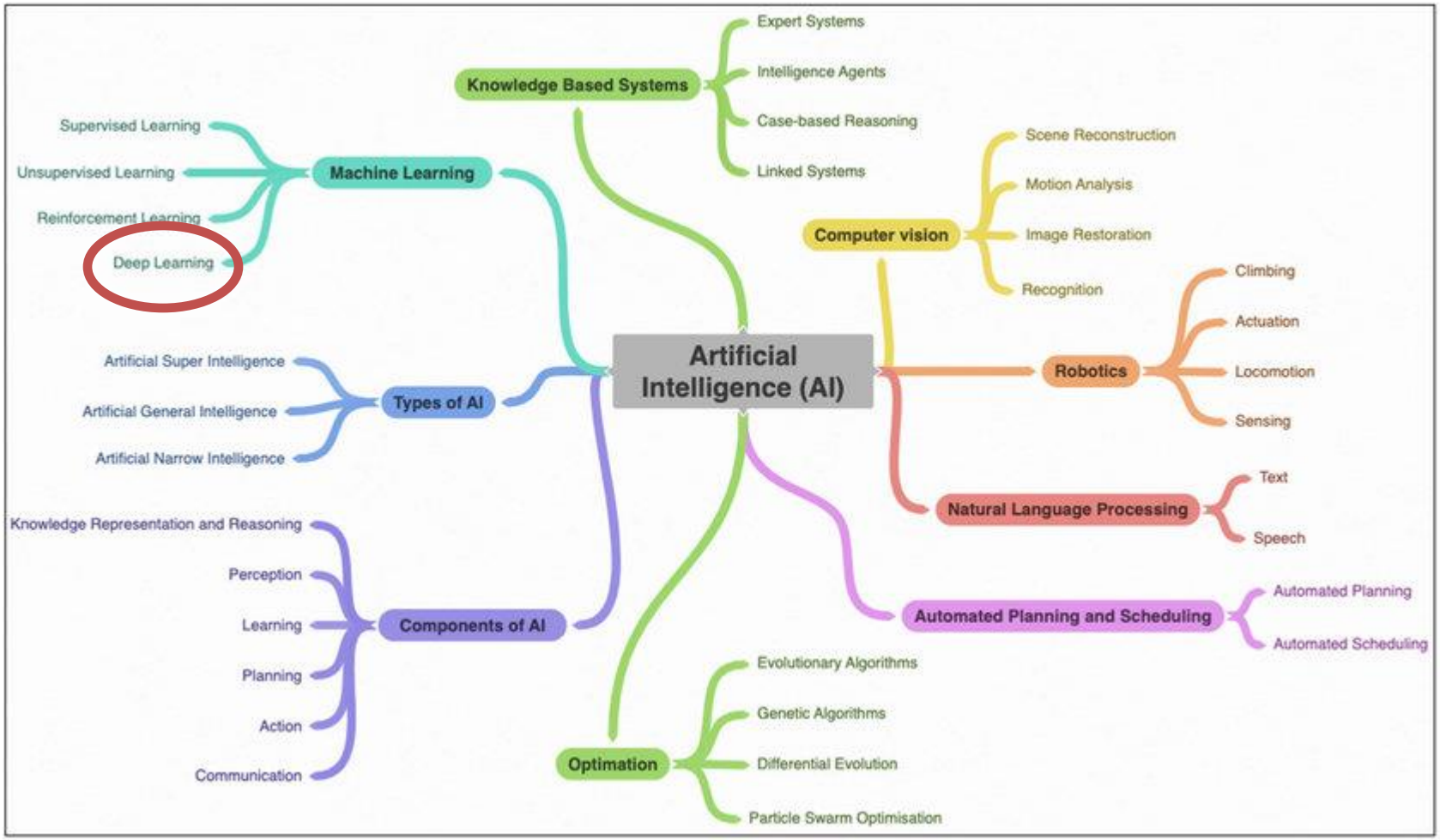
- Uses neural networks that simulate human brain structures and connections.
- Requires large amounts of data and substantial computing resources.
- Especially powerful for tasks involving unstructured data (images, audio, video, text).
- Usually need GPUs.

Artificial Intelligence (AI) *(The Broadest Concept)*



ML is a subset of AI

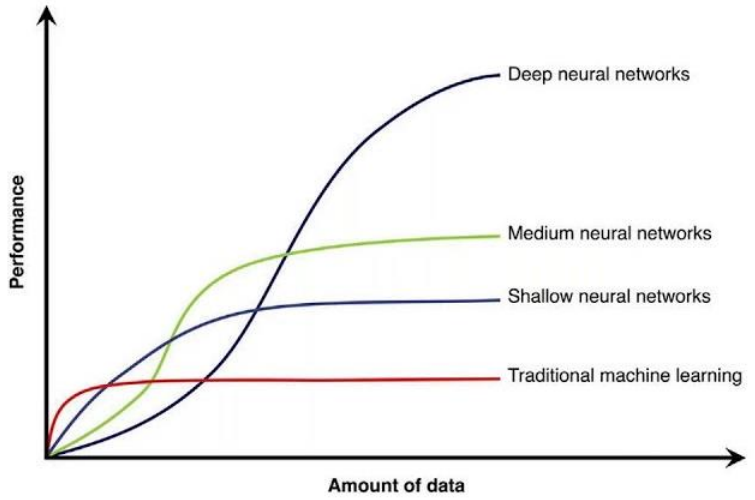
AI-tree


Artificial Intelligence (AI)
Knowledge Based Systems
Expert Systems
Intelligence Agents
Case-based Reasoning
Linked Systems
Machine Learning
Supervised Learning
Unsupervised Learning
Reinforcement Learning
Deep Learning
Computer vision
Scene Reconstruction
Motion Analysis
Image Restoration
Recognition
Robotics
Climbing
Actuation
Locomotion
Sensing
Types of AI
Artificial Super Intelligence
Artificial General Intelligence
Artificial Narrow Intelligence
Natural Language Processing
Text
Speech
Components of AI
Knowledge Representation and Reasoning
Perception
Learning
Planning
Action
Communication
Automated Planning and Scheduling
Automated Planning
Automated Scheduling
Optimation
Evolutionary Algorithms
Genetic Algorithms
Differential Evolution
Particle Swarm Optimisation

Why Deep Learning?

- **Limitations of traditional machine learning algorithms**
  - Not good at handling high dimensional data.
  - Difficult to do feature extraction and object recognition.

- **Advantages of deep learning**
  - DL is computationally expensive, but it is capable of handling high dimensional data.
  - feature extraction is done automatically.



- When the amount of data is increased, machine learning techniques are insufficient in terms of performance and deep learning gives better performance like accuracy.

ImageNet Challenge -2012

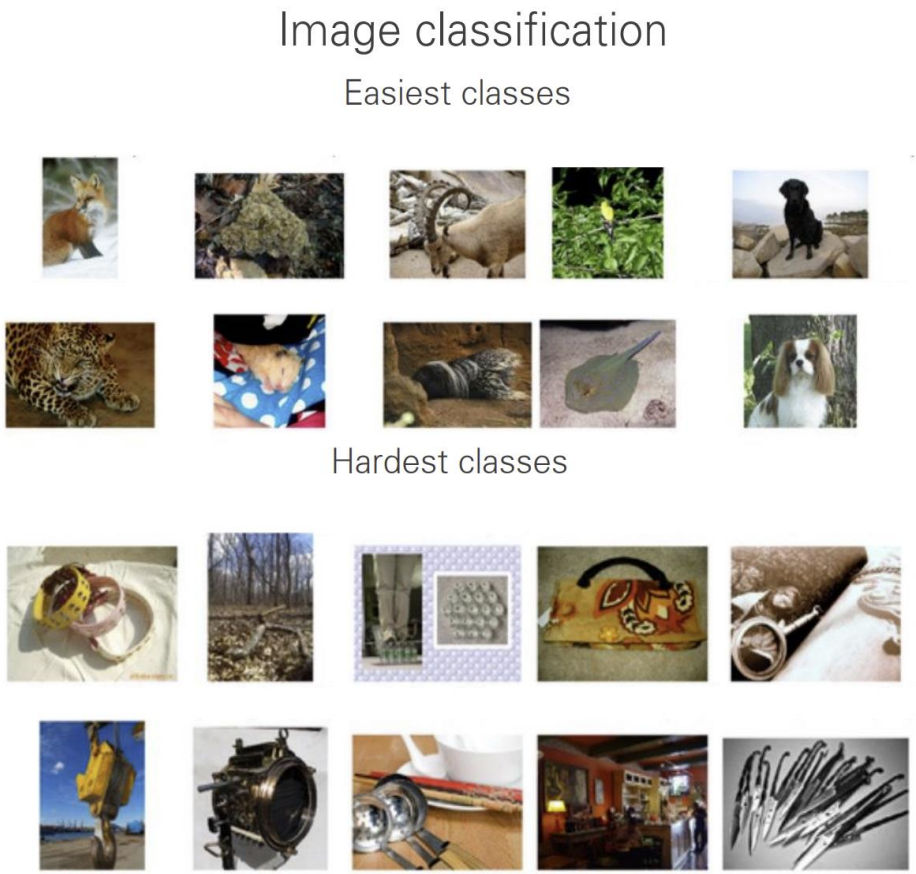
- IMAGENET Large Scale Visual Recognition Challenge (ILSVRC)
  - **1.2M** training images with **1K** categories
  - Measure top-5 classification error

Output
Scale
T-shirt
Steel drum
Drumstick
Mud turtle

✓

Output
Scale
T-shirt
Giant panda
Drumstick
Mud turtle

✗

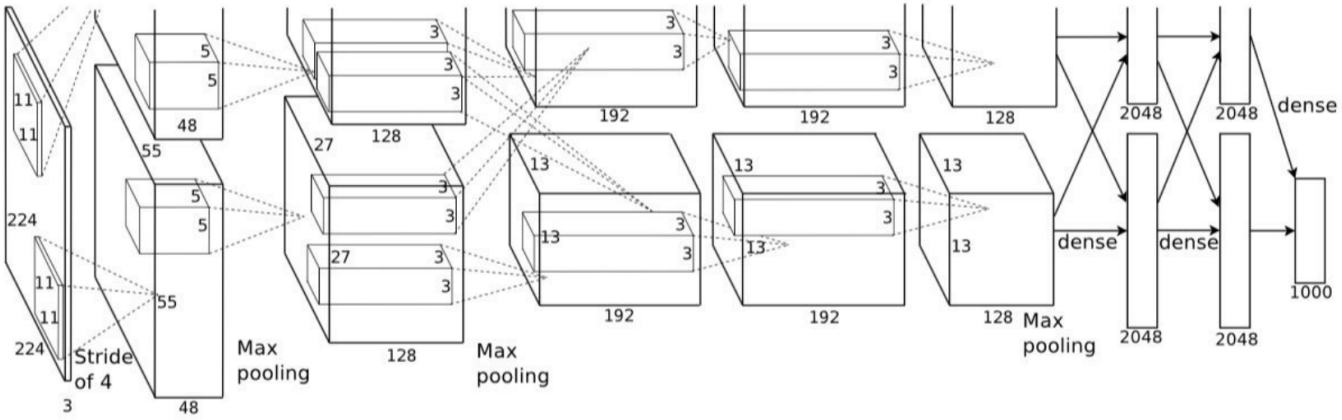


J. Deng, Wei Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei , "**ImageNet: A Large-Scale Hierarchical Image Database**", CVPR 2009.
O. Russakovsky et al., "**ImageNet Large Scale Visual Recognition Challenge**", Int. J. Comput. Vis.,, Vol. 115, Issue 3, pp 211-252, 2015.

ILSVRC 2012 Competition

2012 Teams	%Error
Supervision (Toronto)	15.3
ISI (Tokyo)	26.1
VGG (Oxford)	26.9
XRCE/INRIA	27.0
UvA (Amsterdam)	29.6
INRIA/LEAR	33.4

CNN based, non-CNN based



- The success of AlexNet, a deep convolutional network
  - 7 hidden layers (not counting some max pooling layers)
  - 60M parameters
- Combined several tricks
  - ReLU activation function, data augmentation, dropout

A. Krizhevsky, I. Sutskever, G.E. Hinton "**ImageNet Classification with Deep Convolutional Neural Networks**", NeurIPS 2012

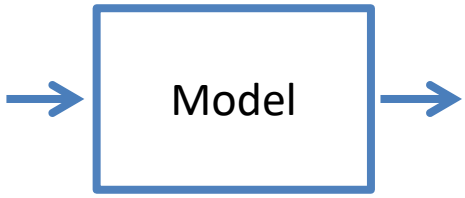
Supervised Deep Learning (DL)

The goal of Supervised DL is to learn from data

---->

Technically, combines statistics and computational tools (optimization)

Example (supervised learning) tasks



Cat or Dog

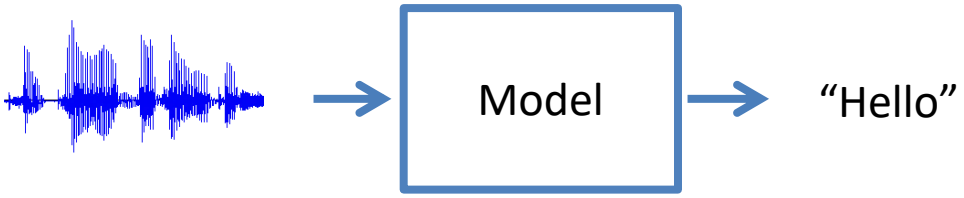
Image recognition / classification

Dataset:

- **Training**
- **Validation**
- **Testing**

Cont.

- The goal of DL is to learn from data
  - Technically, combines statistics and computational tools (optimization)
- Example (supervised learning) tasks



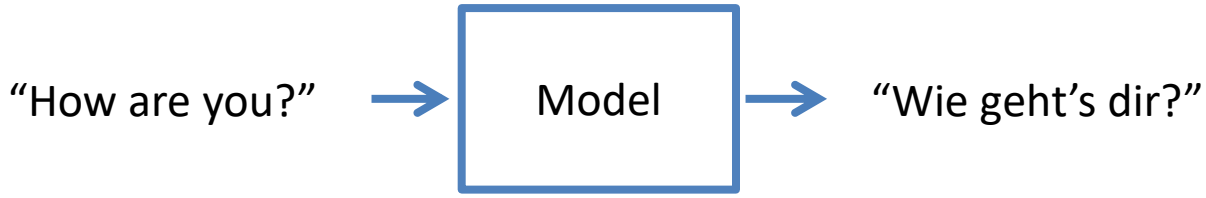
Speech recognition

Dataset:

Training
Validation
Testing

Cont.

- The goal of DL is to learn from data
  - Technically, combines statistics and computational tools (optimization)
- Example (supervised learning) tasks



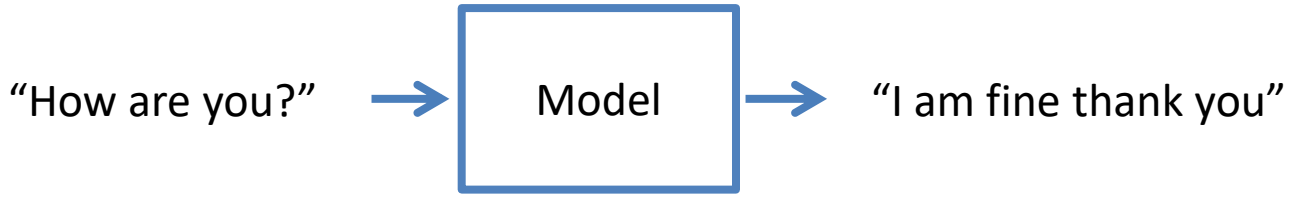
Machine translation

Dataset:

- **Training**
- **Validation**
- **Testing**

Cont.

- The goal of DL is to learn from data
  - Technically, combines statistics and computational tools (optimization)
- Example (supervised learning) tasks



Conversational agent / chatbot

Dataset:

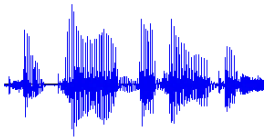
- **Training**
- **Validation**
- **Testing**

Cont.

- The goal of DL is to learn from data
  - Technically, combines statistics and computational tools (optimization)
- Example (supervised learning) tasks

Dataset:
Training
Validation
Testing

"How are you?"



Model

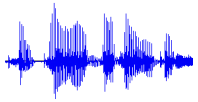
Cat or Dog

"Wie geht's dir?"

"I am fine thank you"

Cont.

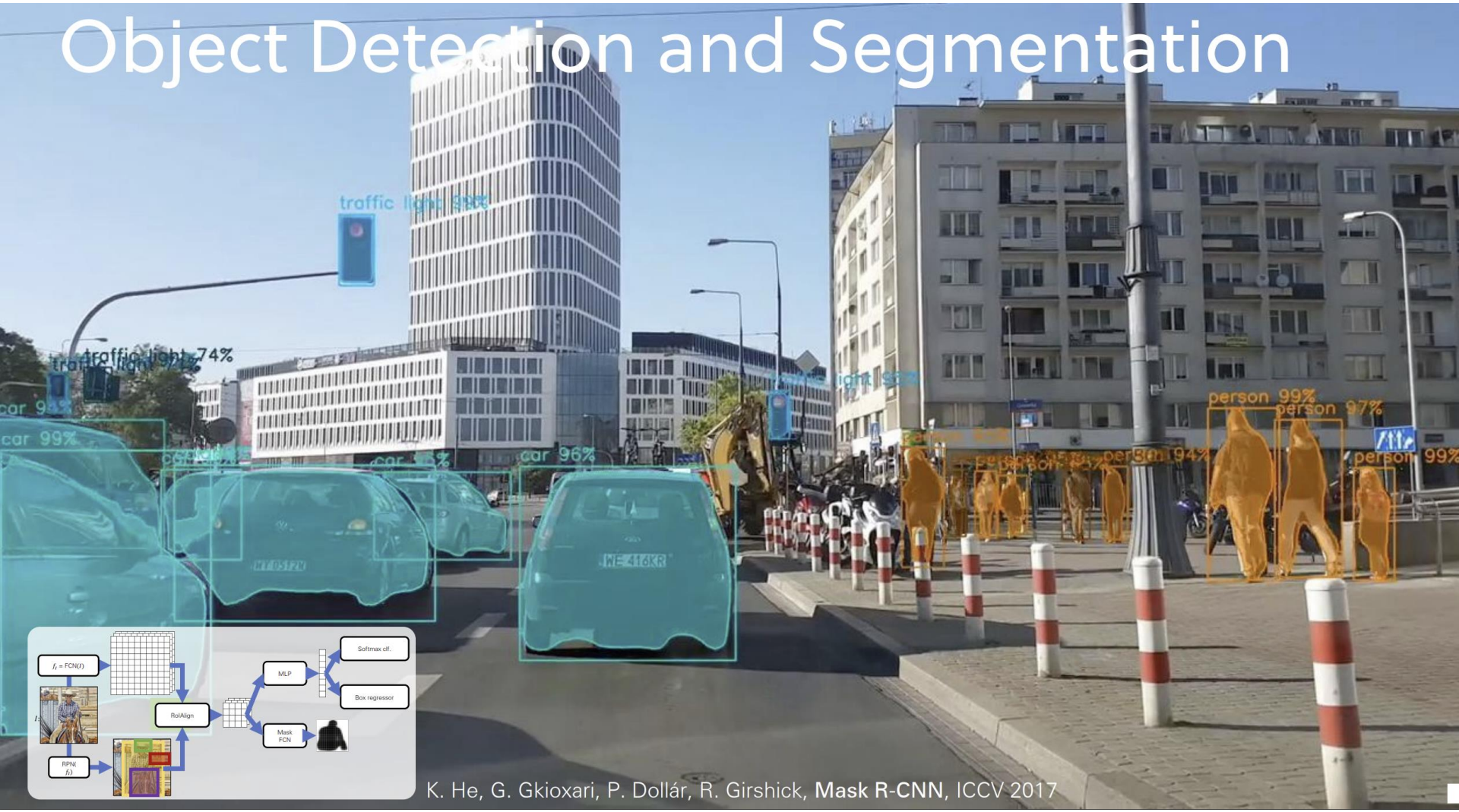
- The goal of DL is to learn from data ---->
  - Technically, combines statistics and computational tools (optimization)
- Example (supervised learning) tasks
  - Image recognition
  - Speech recognition
  - Machine translation
- Other form of learning
  - Unsupervised learning



“Hello” → “Bonjour”

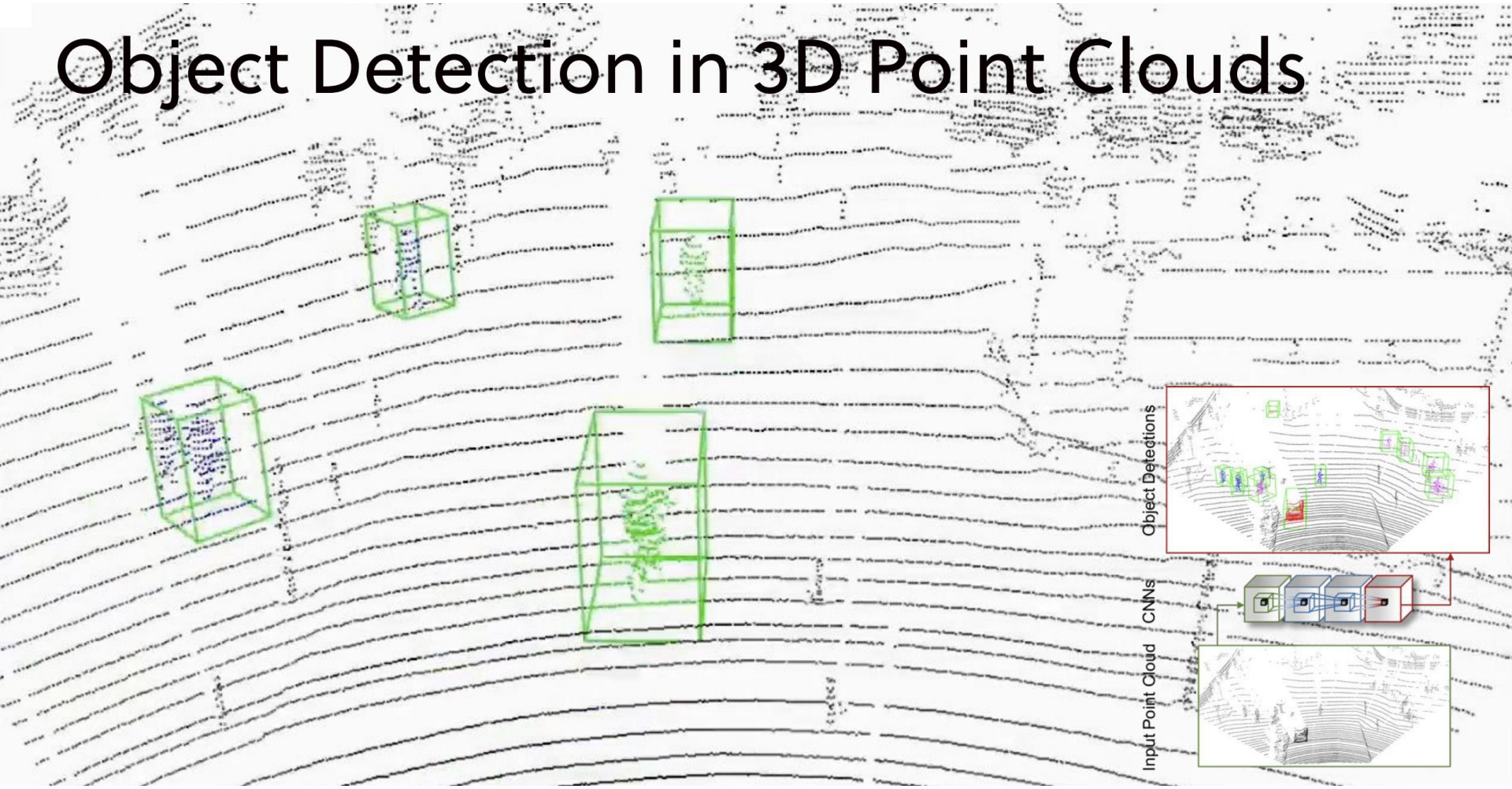
Dataset:
- Training
- Validation
- Testing

Object Detection and Segmentation



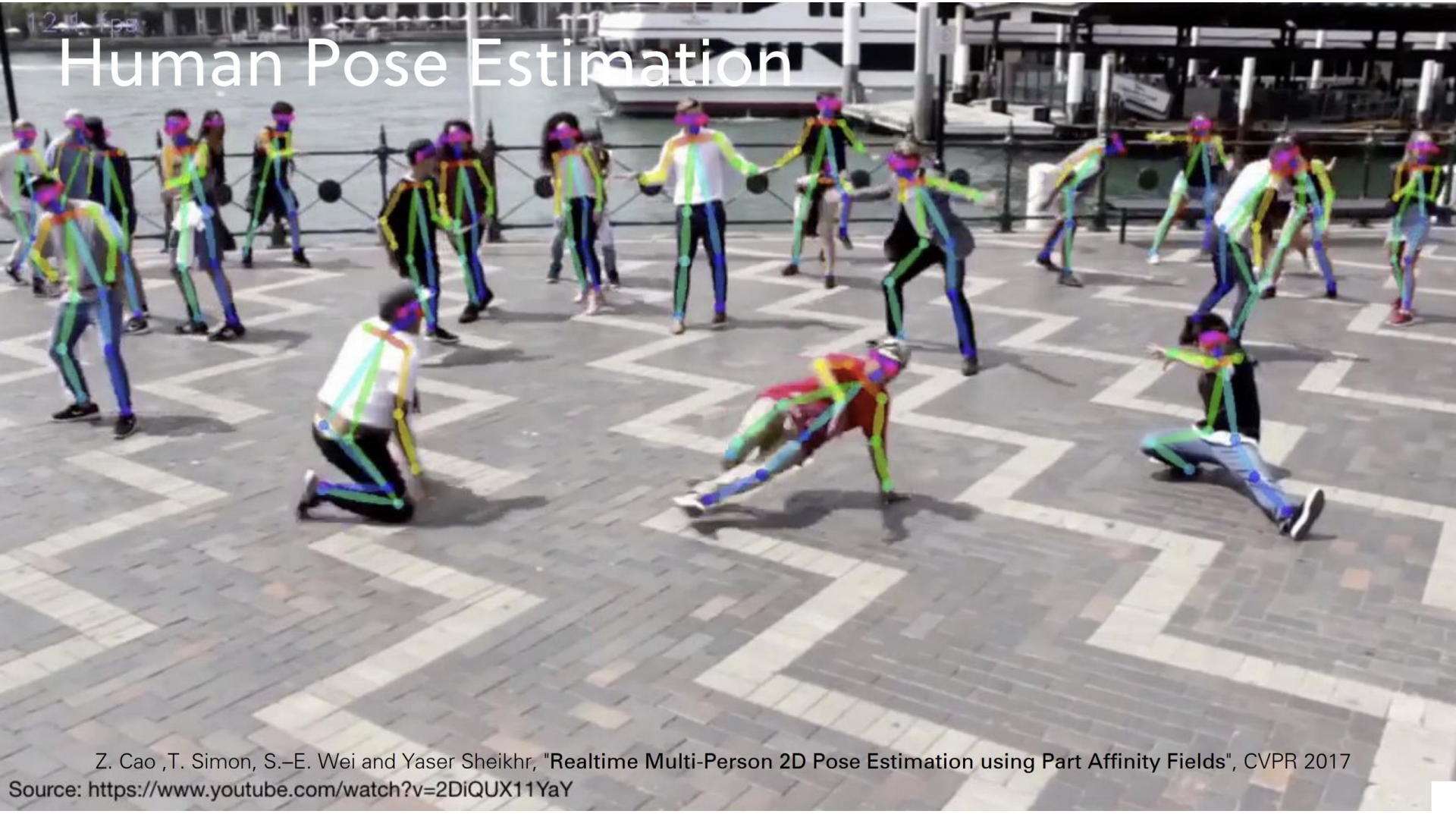
K. He, G. Gkioxari, P. Dollár, R. Girshick, **Mask R-CNN**, ICCV 2017

Object Detection in 3D Point Clouds



M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner. **Vote3Deep: Fast Object Detection in 3D Point Clouds Using Efficient Convolutional Neural Networks**. ICRA 2017

Human Pose Estimation



Z. Cao ,T. Simon, S.–E. Wei and Yaser Sheikhr, "**Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields**", CVPR 2017

Source: https://www.youtube.com/watch?v=2DiQUX11YaY

Image Synthesis

2014

2015

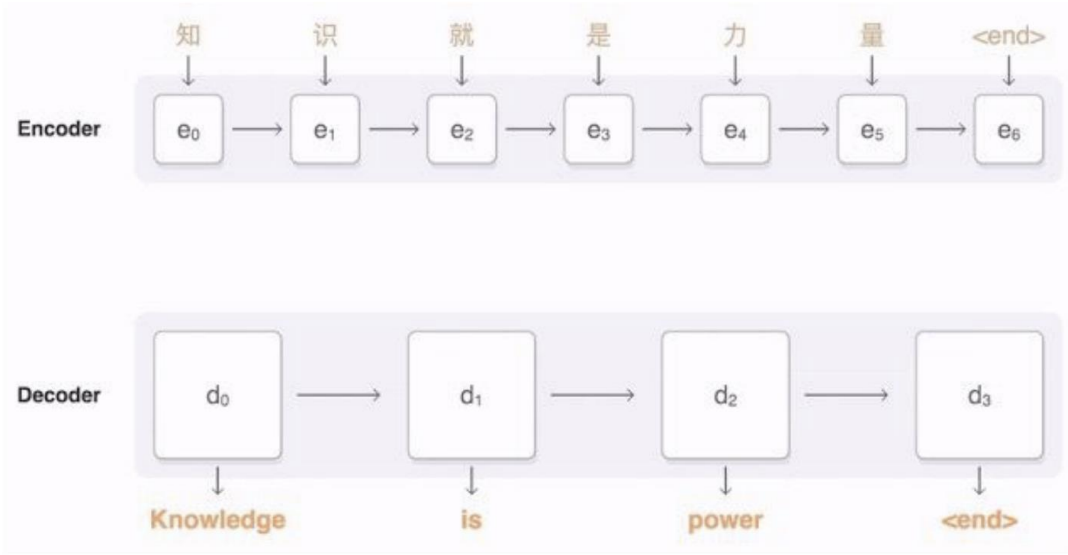
2016

2017

2018

Ian J. Goodfellow et al., " **Generative Adversarial Networks**", NIPS 2014
A. Radford et al., " **Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks**", NIPS 2015
M.-Y. Liu, O. Tuzel, " **Coupled Generative Adversarial Networks**", NIPS 2016
T. Karras, T. Aila, S. Laine, J. Lehtinen, " **Progressive Growing of GANs for Improved Quality, Stability, and Variation**", ICLR 2018
T. Karras, S. Laine, T. Aila, " **A Style-Based Generator Architecture for Generative Adversarial Networks**", arXiv 2018

Machine Translation

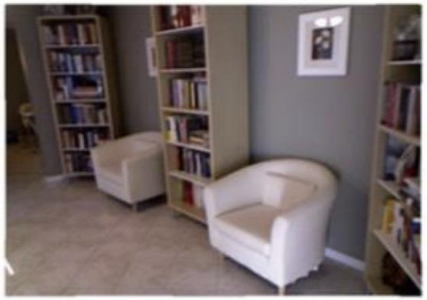


D. Bahdanau, K. Cho, Y. Bengio. **Neural Machine Translation by Jointly Learning to Align and Translate**. ICLR 2015

Visual Question Answering

COCOQA 33827
What is the color of the cat?
Ground truth: black
IMG+BOW: black (0.55)
2-VIS+LSTM: black (0.73)
BOW: gray (0.40)

COCOQA 33827a
What is the color of the couch?
Ground truth: red
IMG+BOW: red (0.65)
2-VIS+LSTM: black (0.44)
BOW: red (0.39)



DAQUAR 1522
How many chairs are there?
Ground truth: two
IMG+BOW: four (0.24)
2-VIS+BLSTM: one (0.29)
LSTM: four (0.19)

DAQUAR 1520
How many shelves are there?
Ground truth: three
IMG+BOW: three (0.25)
2-VIS+BLSTM: two (0.48)
LSTM: two (0.21)

COCOQA 14855
Where are the ripe bananas sitting?
Ground truth: basket
IMG+BOW: basket (0.97)
2-VIS+BLSTM: basket (0.58)
BOW: bowl (0.48)

COCOQA 14855a
What are in the basket?
Ground truth: bananas
IMG+BOW: bananas (0.98)
2-VIS+BLSTM: bananas (0.68)
BOW: bananas (0.14)

DAQUAR 585
What is the object on the chair?
Ground truth: pillow
IMG+BOW: clothes (0.37)
2-VIS+BLSTM: pillow (0.65)
LSTM: clothes (0.40)

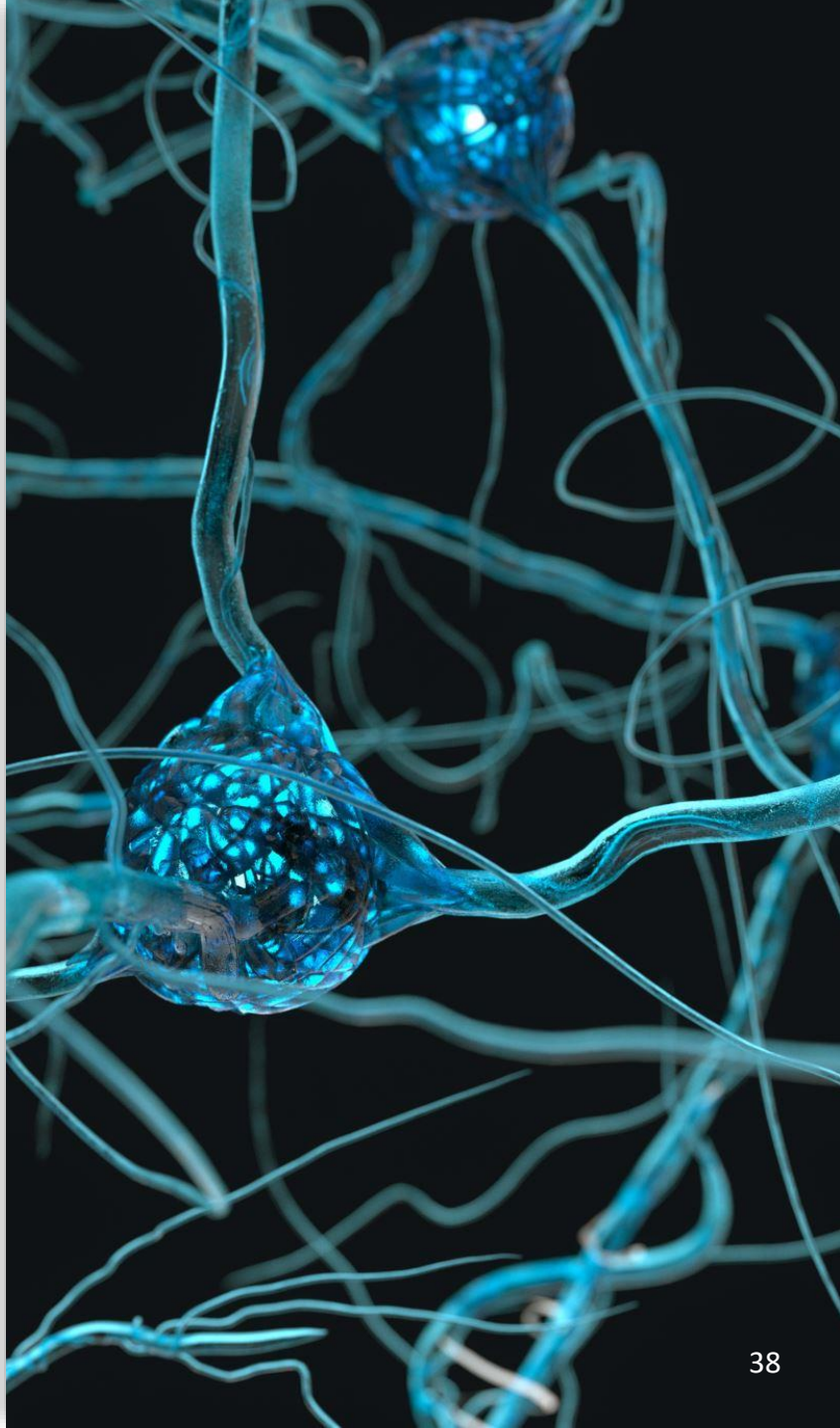
DAQUAR 585a
Where is the pillow found?
Ground truth: chair
IMG+BOW: bed (0.13)
2-VIS+BLSTM: chair (0.17)
LSTM: cabinet (0.79)

M. Ren, R. Kiros, and R. Zemel. **Exploring Models and Data for Image Question Answering**. NeurIPS 2015

Year	Breakthroughs in AI	Datasets (First Available)	Algorithms (First Proposed)
1994	Human-level spontaneous speech recognition	Spoken Wall Street Journal articles and other texts (1991)	Hidden Markov Model (1984)
1997	IBM Deep Blue defeated Garry Kasparov	700,000 Grandmaster chess games, aka "The Extended Book" (1991)	Negascout planning algorithm (1983)
2005	Google's Arabic-and Chinese-to-English translation	1.8 trillion tokens from Google Web and News pages (collected in 2005)	Statistical machine translation algorithm (1988)
2011	IBM Watson became the world Jeopardy! champion	8.6 million documents from Wikipedia, Wiktionary, and Project Gutenberg (updated in 2010)	Mixture-of-Experts (1991)
2014	Google's GoogLeNet object classification at near-human performance	ImageNet corpus of 1.5 million labeled images and 1,000 object categories (2010)	Convolutional Neural Networks (1989)
2015	Google's DeepMind achieved human parity in playing 29 Atari games by learning general control from video	Arcade Learning Environment dataset of over 50 Atari games (2013)	Q-learning (1992)

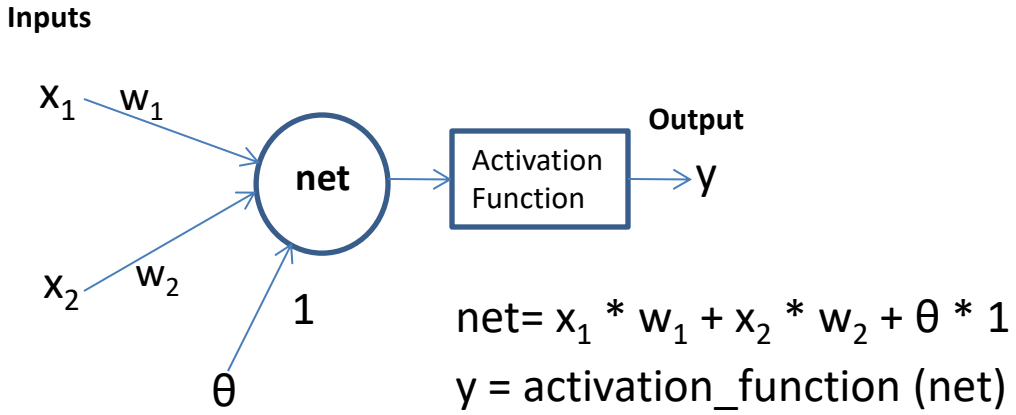
What is Deep Learning?

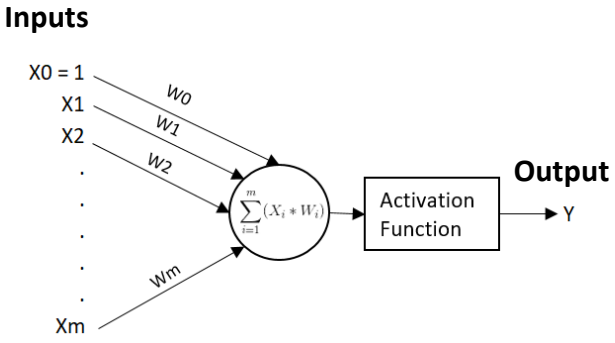
- **Neural Networks**: Deep learning uses neural networks, composed of nodes (also known as neurons), organized into layers. Each node processes information, applies weights, biases, and activation functions, and passes outputs to subsequent layers.
- **Depth**: The term "deep" refers to having many layers of neurons—typically, a deep neural network might have dozens or even hundreds of layers.
- **Learning from Data**: Deep learning networks learn by adjusting their internal parameters (weights and biases) based on large amounts of data, usually through a process known as backpropagation and gradient descent.
- **Automatic Feature Extraction**: Unlike traditional machine learning, where features often have to be handcrafted, deep learning automatically discovers meaningful representations directly from raw data, making it exceptionally powerful for complex tasks like image and speech recognition.



Perceptron

- Is the simplest form of a neural network,
- Perceptron is a linear classifier,
- Consists of a single neuron with adjustable weights and bias,
- A neuron receives several signals from its input links, computes a new activation level and sends it, as an output signal through the output link(s),
- The input signal can be raw data or outputs of other neurons,
- The output signal can be either a final solution to the problem or an input to other neurons.



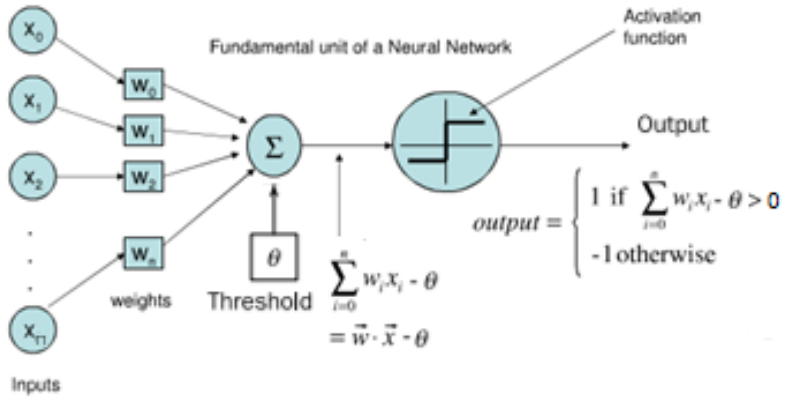


How does a perceptron learn its classification tasks?

- By updating weights and biases iteratively

- By reducing/minimizing the error or difference between the actual and desired outputs of the perceptron.

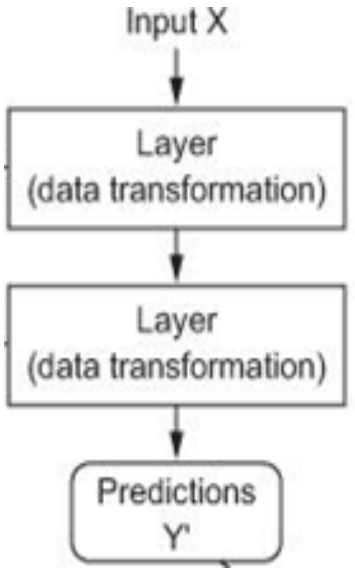
- Iteration based

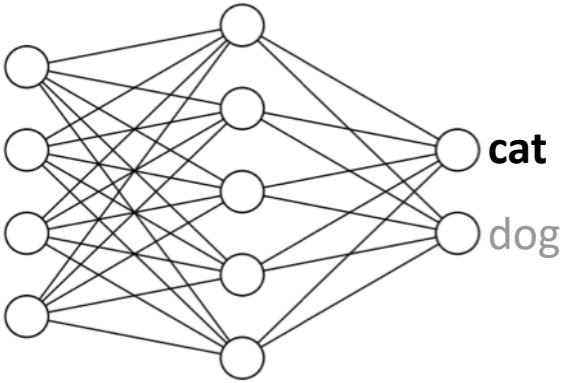


Input data and targets

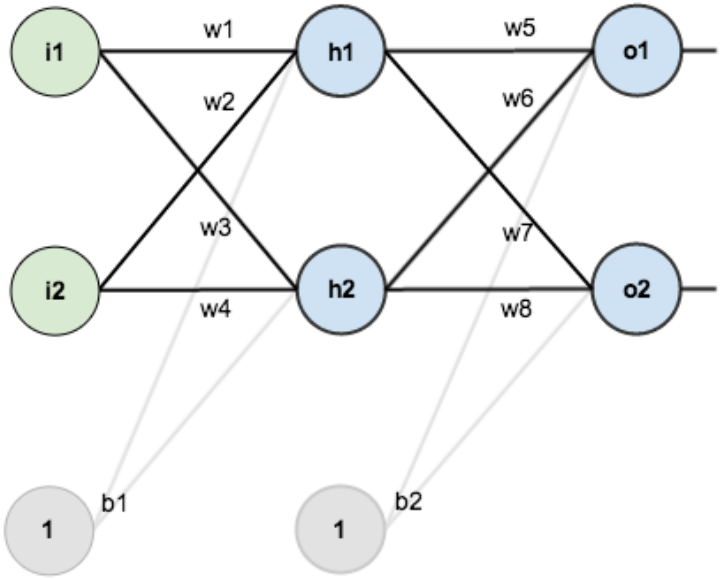
The network maps the input data X to predictions Y’

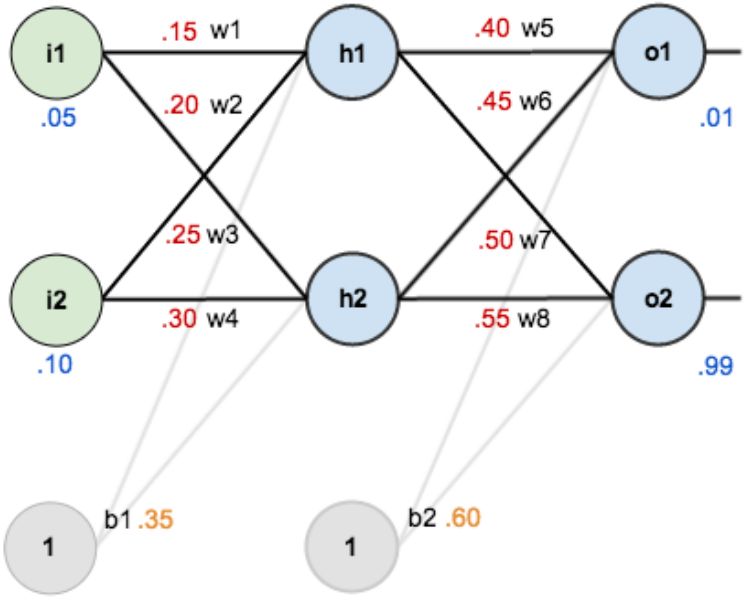
During training, the predictions Y’ are compared to true targets Y using the loss function





A Step by Step Backpropagation Example





here are the initial weights, the biases, and training inputs/outputs

Here's how we calculate the total net input for h_1:

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

We then squash it using the logistic function to get the output of h_1:

$$out_{h1} = \frac{1}{1+e^{-net_{h1}}} = \frac{1}{1+e^{-0.3775}} = 0.593269992$$

Carrying out the same process for h_2 we get:

$$out_{h2} = 0.596884378$$

We repeat this process for the output layer neurons, using the output from the hidden layer neurons as inputs.

Here's the output for o_1:

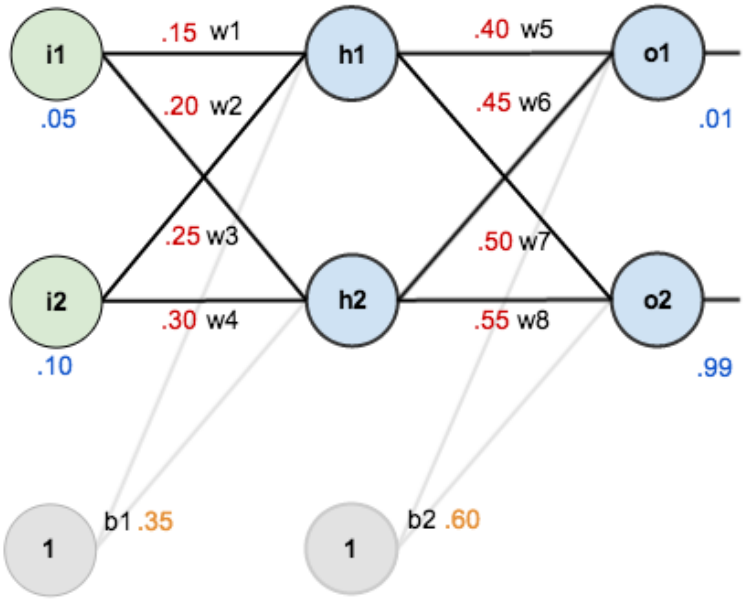
$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$net_{o1} = 0.4 * 0.593269992 + 0.45 * 0.596884378 + 0.6 * 1 = 1.105905967$$

$$out_{o1} = \frac{1}{1+e^{-net_{o1}}} = \frac{1}{1+e^{-1.105905967}} = 0.75136507$$

And carrying out the same process for o_2 we get:

$$out_{o2} = 0.772928465$$



Calculating the Total Error

We can now calculate the error for each output neuron using the squared error function and sum them to get the total error:

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

For example, the target output for o_1 is 0.01 but the neural network output 0.75136507, therefore its error is:

$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$$

Repeating this process for o_2 (remembering that the target is 0.99) we get:

$$E_{o2} = 0.023560026$$

The total error for the neural network is the sum of these errors:

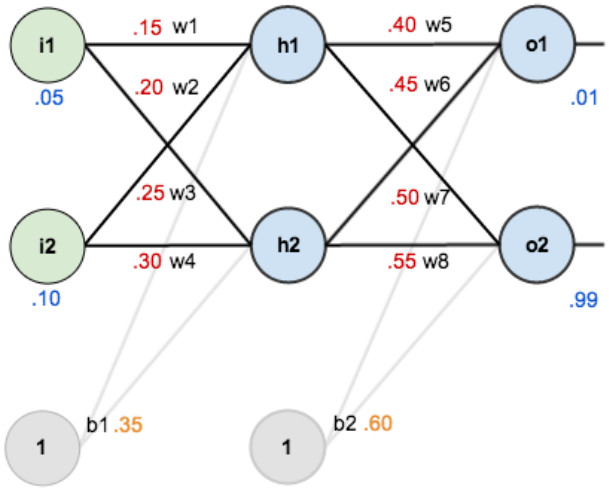
$$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$$

The Backwards Pass

Our goal with backpropagation is to **update each of the weight and bias** in the network so that they cause the actual output to be closer the target output, thereby **minimizing the error** for each output neuron and the network as a whole.

How do we apply gradient descent in neural networks?

Cont.



Output Layer

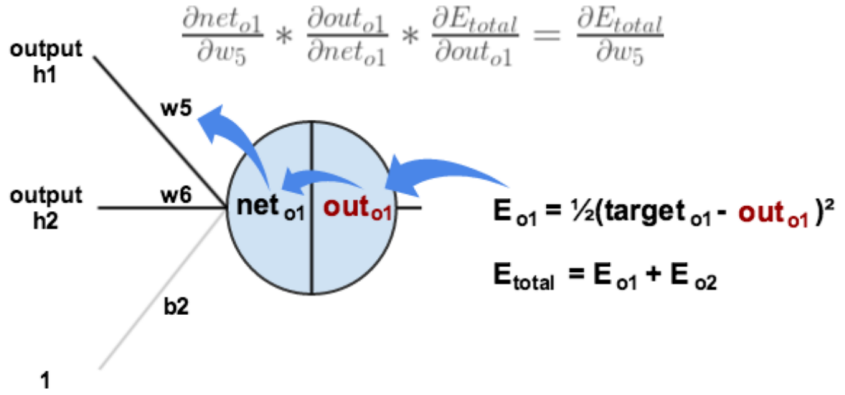
Consider w_5. We want to know how much a change in w_5 affects the total error, aka $\frac{\partial E_{total}}{\partial w_5}$.

> $\frac{\partial E_{total}}{\partial w_5}$ is read as "the partial derivative of E_{total} with respect to w_5". You can also say "the gradient with respect to w_5".

By applying the chain rule we know that:

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

Visually, here's what we're doing:



First, how much does the total error change with respect to the output?

$$E_{total} = \frac{1}{2}(target_{o1} - out_{o1})^2 + \frac{1}{2}(target_{o2} - out_{o2})^2$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = 2 * \frac{1}{2}(target_{o1} - out_{o1})^{2-1} * -1 + 0$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1}) = -(0.01 - 0.75136507) = 0.74136507$$

Next, how much does the output of o_1 change with respect to its total net input?

The partial derivative of the logistic function is the output multiplied by 1 minus the output:

$$out_{o1} = \frac{1}{1+e^{-net_{o1}}}$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) = 0.75136507(1 - 0.75136507) = 0.186815602$$

Finally, how much does the total net input of $o1$ change with respect to w_5?

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial w_5} = 1 * out_{h1} * w_5^{(1-1)} + 0 + 0 = out_{h1} = 0.593269992$$

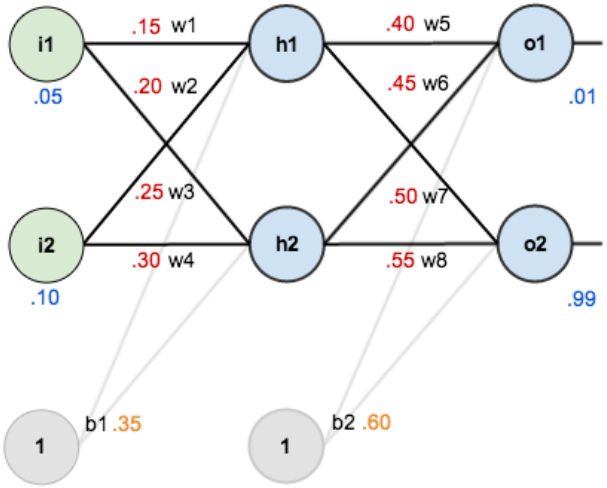
Putting it all together:

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.74136507 * 0.186815602 * 0.593269992 = 0.082167041$$

To decrease the error, we then subtract this value from the current weight (optionally multiplied by some learning rate, eta, which we'll set to 0.5):

$$w_5^+ = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 * 0.082167041 = 0.35891648$$



We can repeat this process to get the new weights w_6, w_7, and w_8:

$$w_6^+ = 0.408666186$$

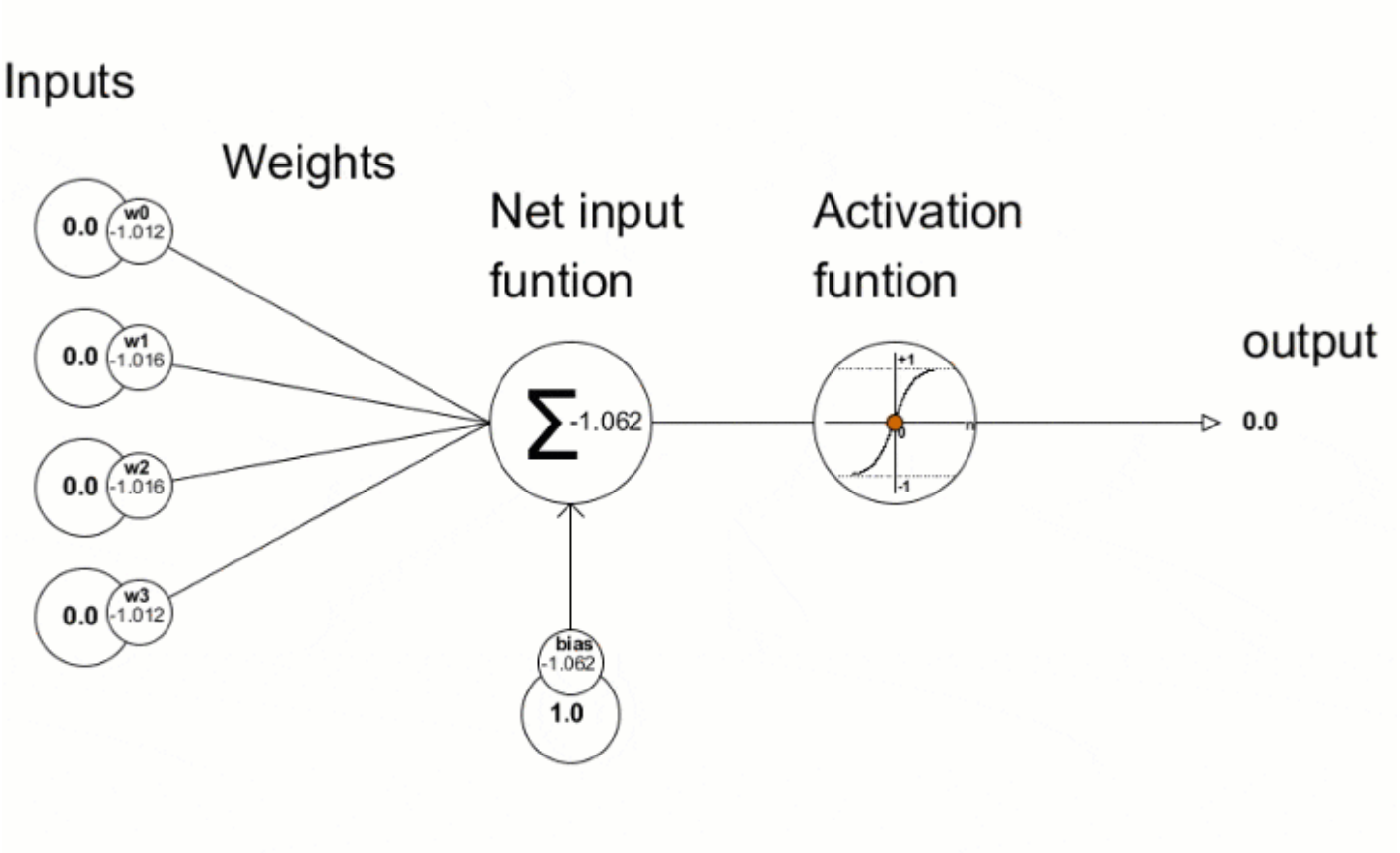
$$w_7^+ = 0.511301270$$

$$w_8^+ = 0.561370121$$

Implementation

```
import numpy as np

# Sigmoid function and its derivative
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def sigmoid_derivative(x):
    return x * (1 - x)

# Initialize inputs, targets, weights, biases, and learning rate
x = np.array([0.5, 0.8])  # Inputs
t = np.array([0.6, 0.5])  # Targets
w_h = np.array([[0.1, 0.3], [0.2, 0.4]])  # Weights for hidden layer
b_h = np.array([0.1, 0.2])  # Biases for hidden layer
w_o = np.array([[0.1, 0.3], [0.2, 0.4]])  # Weights for output layer
b_o = np.array([0.1, 0.2])  # Biases for output layer
learning_rate = 0.1

# Training loop for three iterations
for iteration in range(3):
    # Forward pass: Hidden layer
    z_h = np.dot(x, w_h) + b_h
    h = sigmoid(z_h)

    # Forward pass: Output layer
    z_o = np.dot(h, w_o) + b_o
    o = sigmoid(z_o)

    # Backpropagation: Output layer error
    error_o = t - o
    delta_o = error_o * sigmoid_derivative(o)

    # Backpropagation: Hidden layer error
    error_h = delta_o.dot(w_o.T)
    delta_h = error_h * sigmoid_derivative(h)

    # Gradients for weight and bias updates
    gradient_w_o = h.reshape(-1, 1).dot(delta_o.reshape(1, -1))
    gradient_b_o = delta_o
    gradient_w_h = x.reshape(-1, 1).dot(delta_h.reshape(1, -1))
    gradient_b_h = delta_h

    # Update weights and biases
    w_o += learning_rate * gradient_w_o
    b_o += learning_rate * gradient_b_o
    w_h += learning_rate * gradient_w_h
    b_h += learning_rate * gradient_b_h

# Display the updated weights and biases after three iterations
print("Weights from inputs to hidden layer (w_h):")
print(w_h)
print("Biases for hidden layer (b_h):")
print(b_h)
print("Weights from hidden layer to outputs (w_o):")
print(w_o)
print("Biases for output layer (b_o):")
print(b_o)
```

Neural Network Demonstration



Hidden Layer

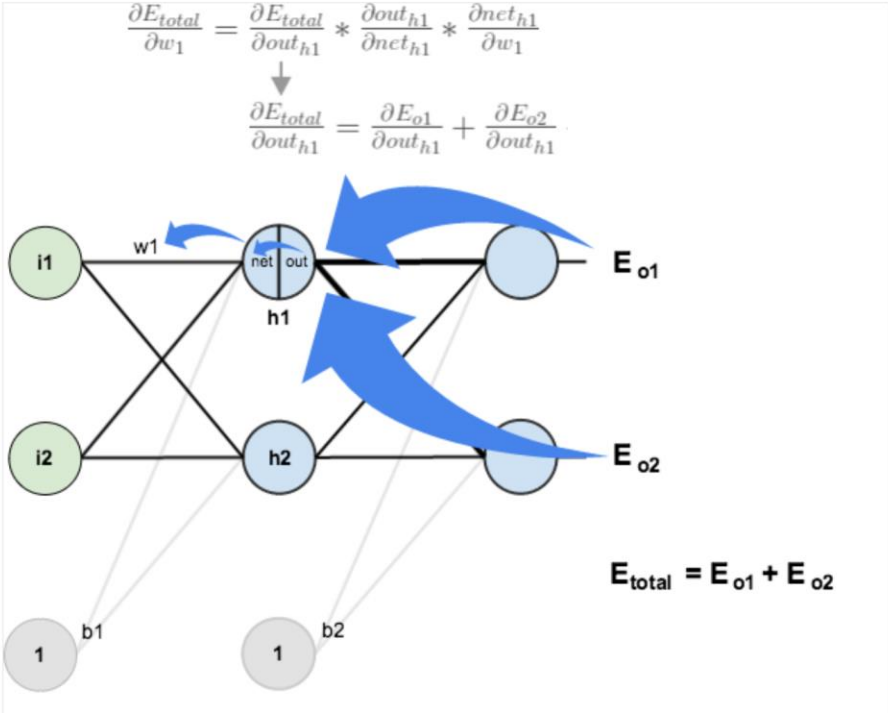
Hidden Layer

Next, we'll continue the backwards pass by calculating new values for w_1, w_2, w_3, and w_4.

Big picture, here's what we need to figure out:

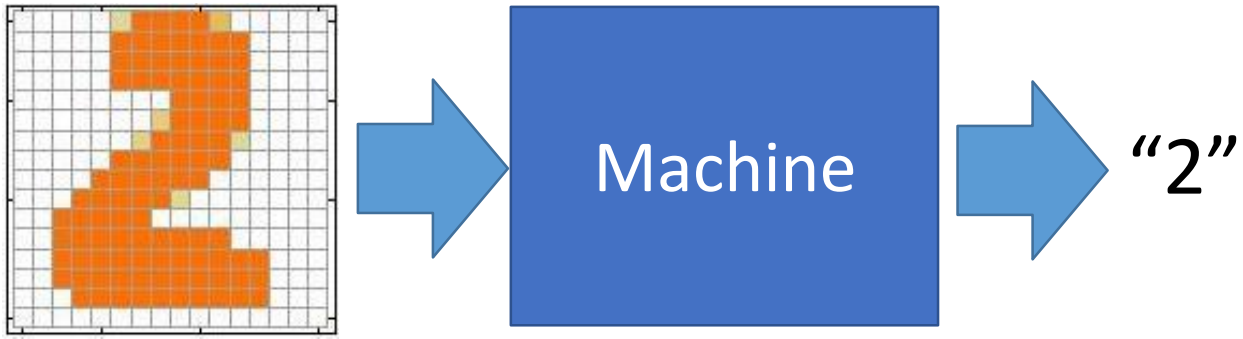
$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

Visually:



Example Application

- Handwriting Digit Recognition



Listen a song generated by AI ☺

https://aimusic.fm/community/

Thank you for listening!!!