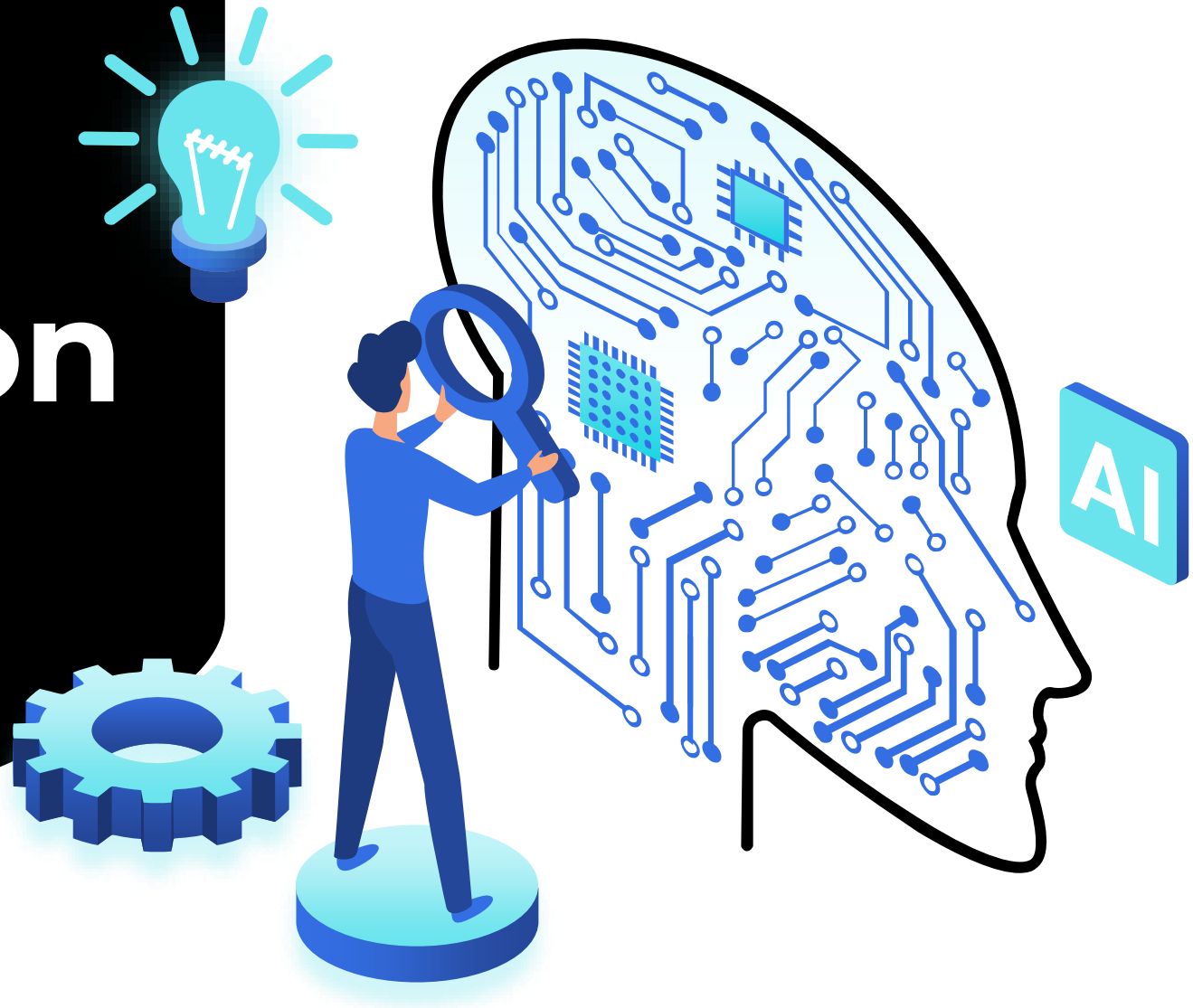
Introduction

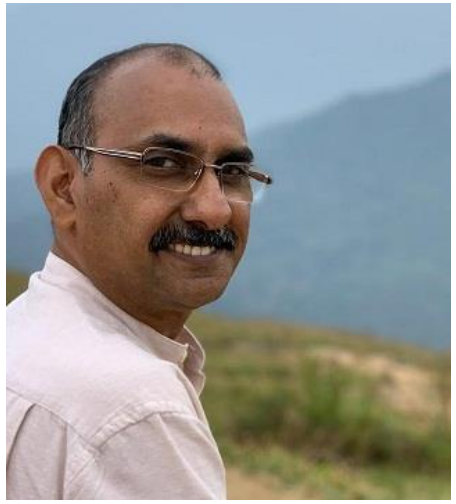
Deepu Rajan
asdrajan@ntu.edu.sg

Slides by Prof Chen-Change Loy



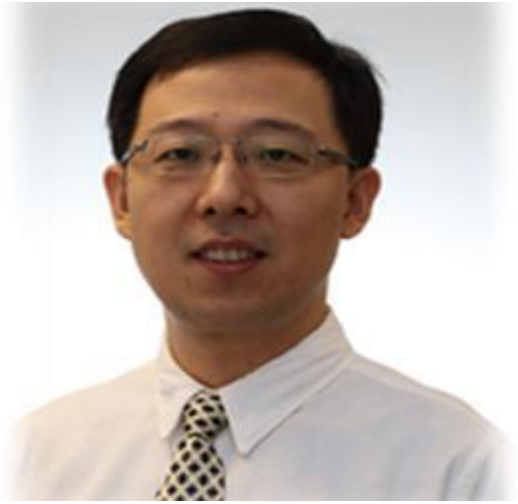
AI6126 Advanced Computer Vision

Instructors



Deepu Rajan

Email: asdrajan@ntu.edu.sg
Office: By email appointment



Shijian Lu

Email: Shijian.lu@ntu.edu.sg

Outline

- About this course
- Computer vision background and a little history
- Fundamentals of machine learning
  - Why learning?
  - Statistical learning

About This Course

Course objectives

- Understand deep learning models such as convolutional networks, transformers and autoencoders and how they are essential for different computer vision tasks like object detection, segmentation and image generation
- Get familiarized with PyTorch for developing deep learning applications
- Design and train deep learning models for solving different computer vision applications

Course hours

- Lecture + tutorial
  - Three hours per week
  - Every Friday 6:30pm – 9:30pm
- Venue
  - Face-to-face at LT3
- Course materials
  - Course notes of the particular week will be available on NTULearn before the lecture
  - The recording of the lecture will be available after each lecture. You will be able to see the videos under ‘Course Media'.

Course outline

Week	Date	Topic	Lecturer
1	16 Jan	Introduction	Deepu Rajan
2	23 Jan	Convolutional Neural Networks I	Deepu Rajan
3	30 Jan	Convolutional Neural Networks II	Deepu Rajan
4	6 Feb	Transformers	Deepu Rajan
5	13 Feb	Autoencoder	Deepu Rajan
6	20 Feb	Object Detection	Deepu Rajan
7	27 Feb	Image Segmentation + Quiz	Deepu Rajan
Recess Week	7 Mar		
8	13 Mar	Image Composition	Shijian Lu
9	20 Mar (Hari Raya Puasa)	Image style transfer	Shijian Lu
10	27 Mar	Image Editing	Shijian Lu
11	3 Apr (Good Friday)	Multimodal generation	Shijian Lu
12	10 Apr	Unsupervised domain adaptation	Shijian Lu
13	17 Apr	Unsupervised model adaptation	Shijian Lu
14	24 Apr		

Assessment

- Part 1
  - 1 Project (Individual): 30%
  - 1 Quiz: 20%
- Part 2
  - 2 assignments/projects: 25% each

Projects

- Project 1
  - Handout on 13 Feb (Week 5)
  - Deadline on 20 Mar (Week 9)

Projects

- **Python** is the recommended programming language
- PC with at least 1 GPU is recommended
- Projects are to be done individually
- Each student should submit the final report (in .pdf) and code (in a .zip file), individually, using their accounts to NTULearn before the deadline.
- Late submissions will be penalized (each day at 5% up to 3 days)
- Assessment criteria will be indicated in the handout

Quiz 1

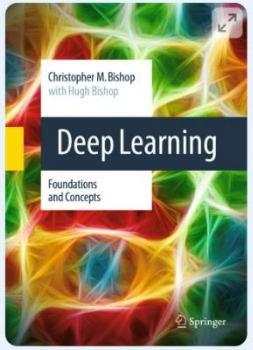
- To be conducted in Week 7 (27 February)
- The duration of the quiz will be 60 minutes and it is a **closed-book** quiz, during which you will be required to answer 3 text-entry questions (fill in the blanks) and 20 multiple-choice questions. You don't need a calculator for the quiz.
- The questions will be based on all lectures except the last lecture on 27 Feb.
- All questions will add up to 30 marks. The individual text-based questions may be of different marks.

Where to ask questions

- Post your queries on the 'Discussion Board' so we can learn from each other.
- If you have questions that you would like a more private answer, you can send an email to ai6126@e.ntu.edu.sg.
- If possible, please use the discussion forum on NTULearn instead of sending an email.

References

Home > Textbook



Deep Learning

Foundations and Concepts

Textbook | © 2024

Access provided by Nanyang Technological University Library

Download book PDF

Overview

Authors: Christopher M. Bishop , Hugh Bishop

- Foundational and conceptual approach emphasizes real-world practical value of techniques for a wide range of learners
- Companion volume to the author's standard reference text Pattern Recognition and Machine Learning
- To reinforce key ideas, end-of-chapter exercises of varying difficulty are included to promote active learning

316k Accesses 21 Citations 34 Altmetric

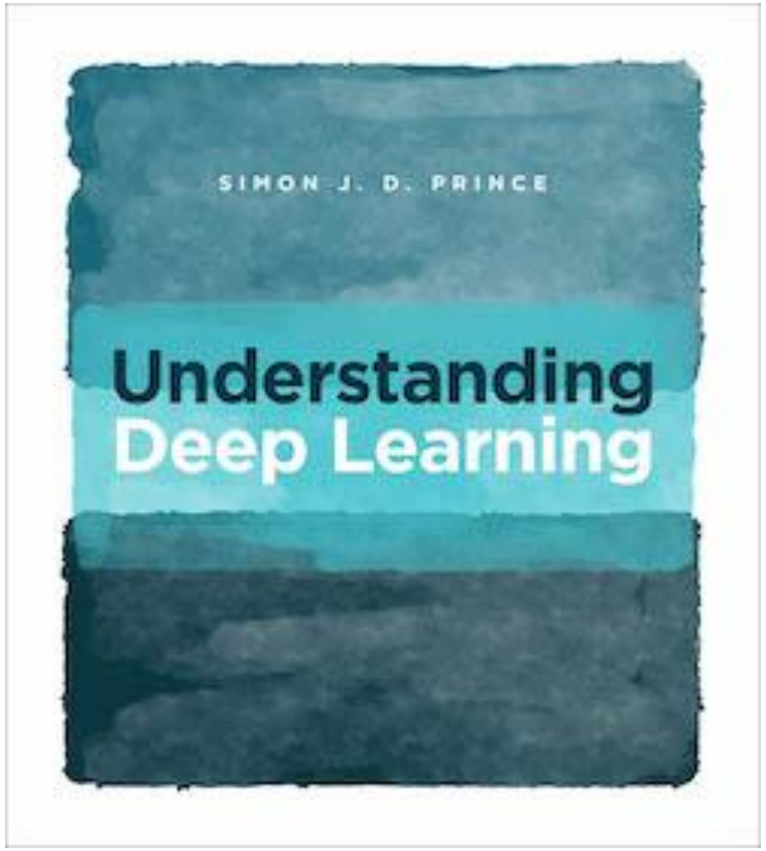
Go to Library Collection

https://libguides.ntu.edu.sg/az.php?q=SpringerLink

Search for Chris Bishop

Download the e-book for free

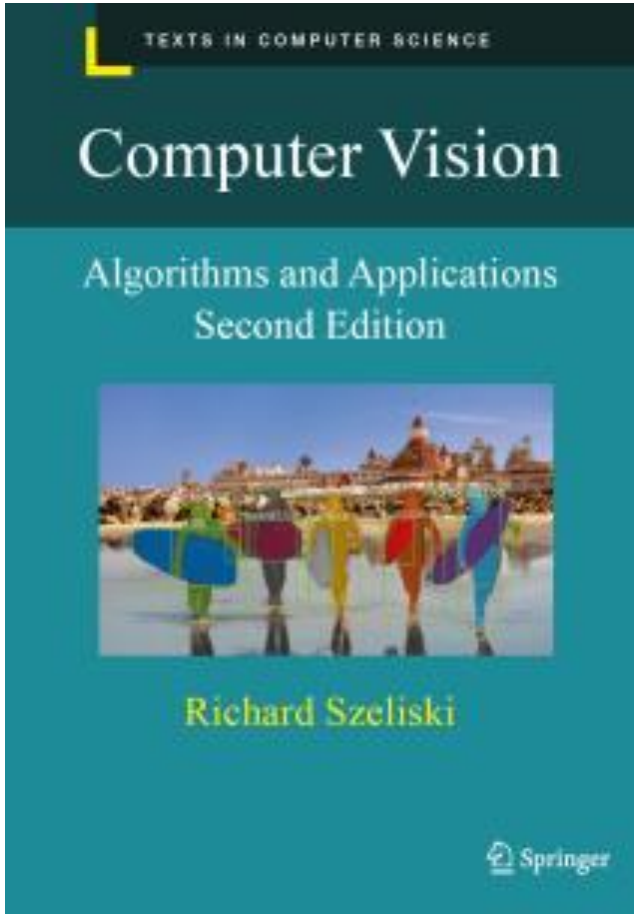
References



Understanding Deep Learning

https://udlbook.github.io/udlbook/

References



Computer Vision: Algorithms and Applications, 2nd ed.

http://szeliski.org/Book/

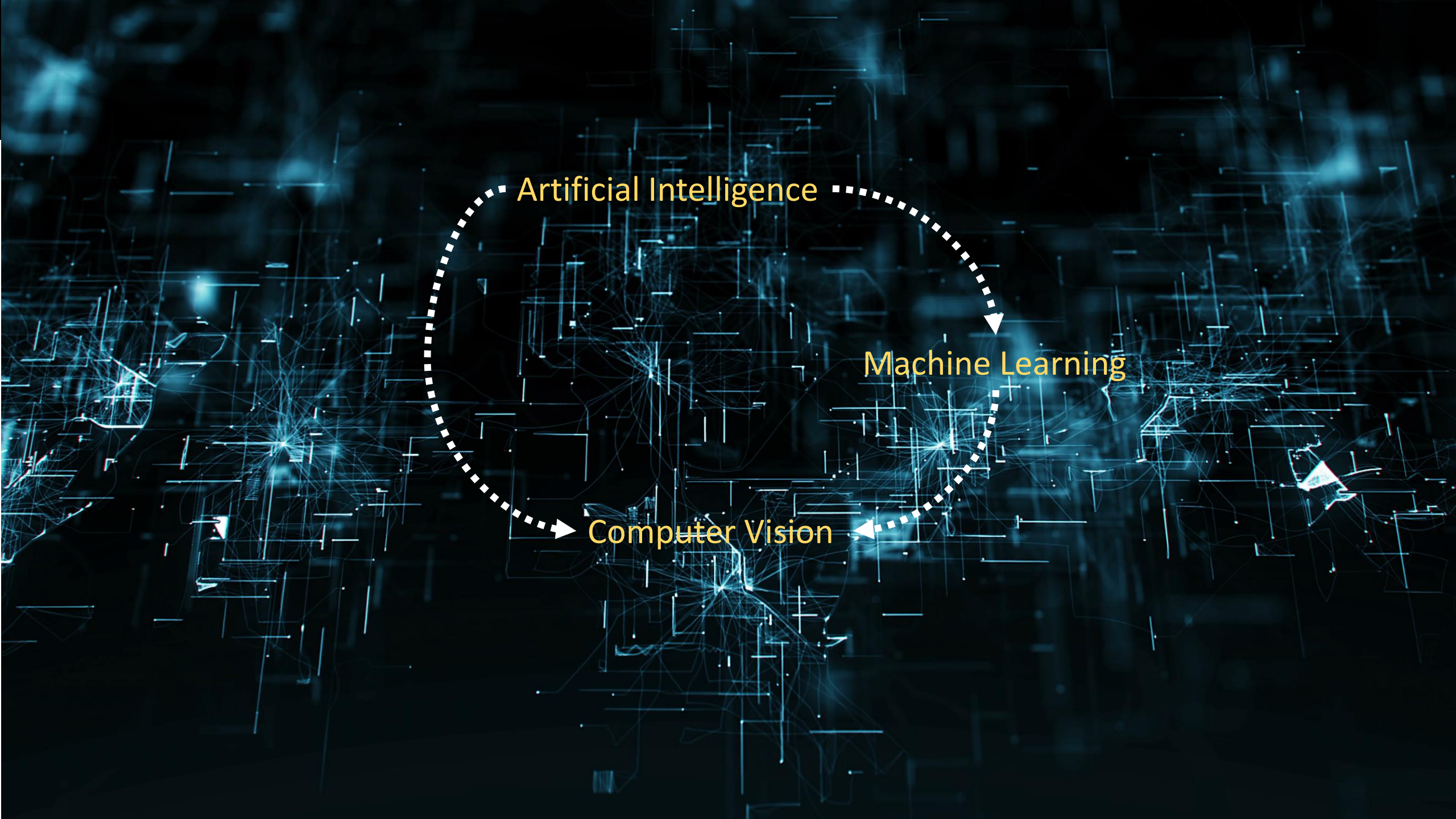
References

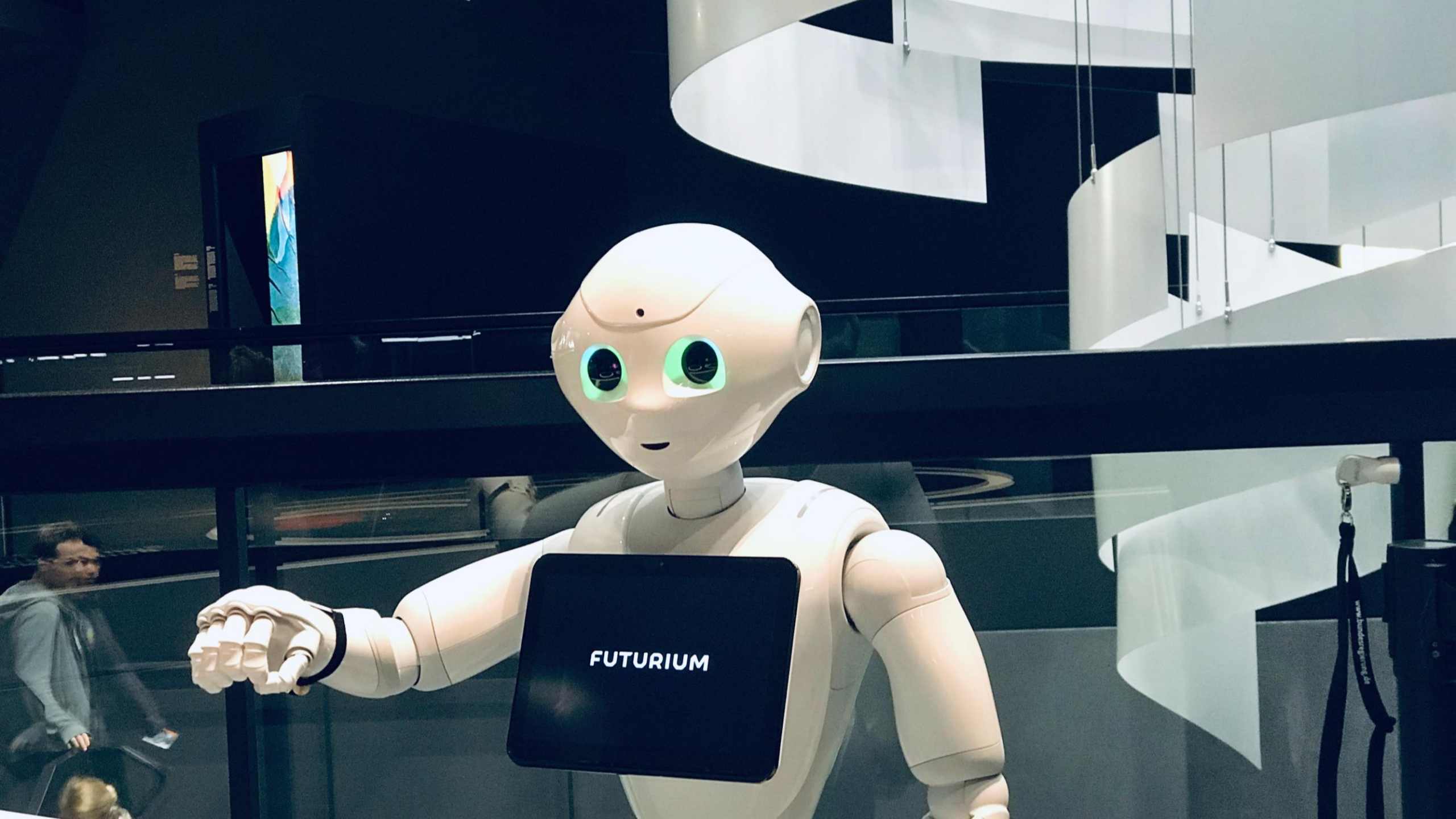
Computer Vision: Models, Learning, and Inference

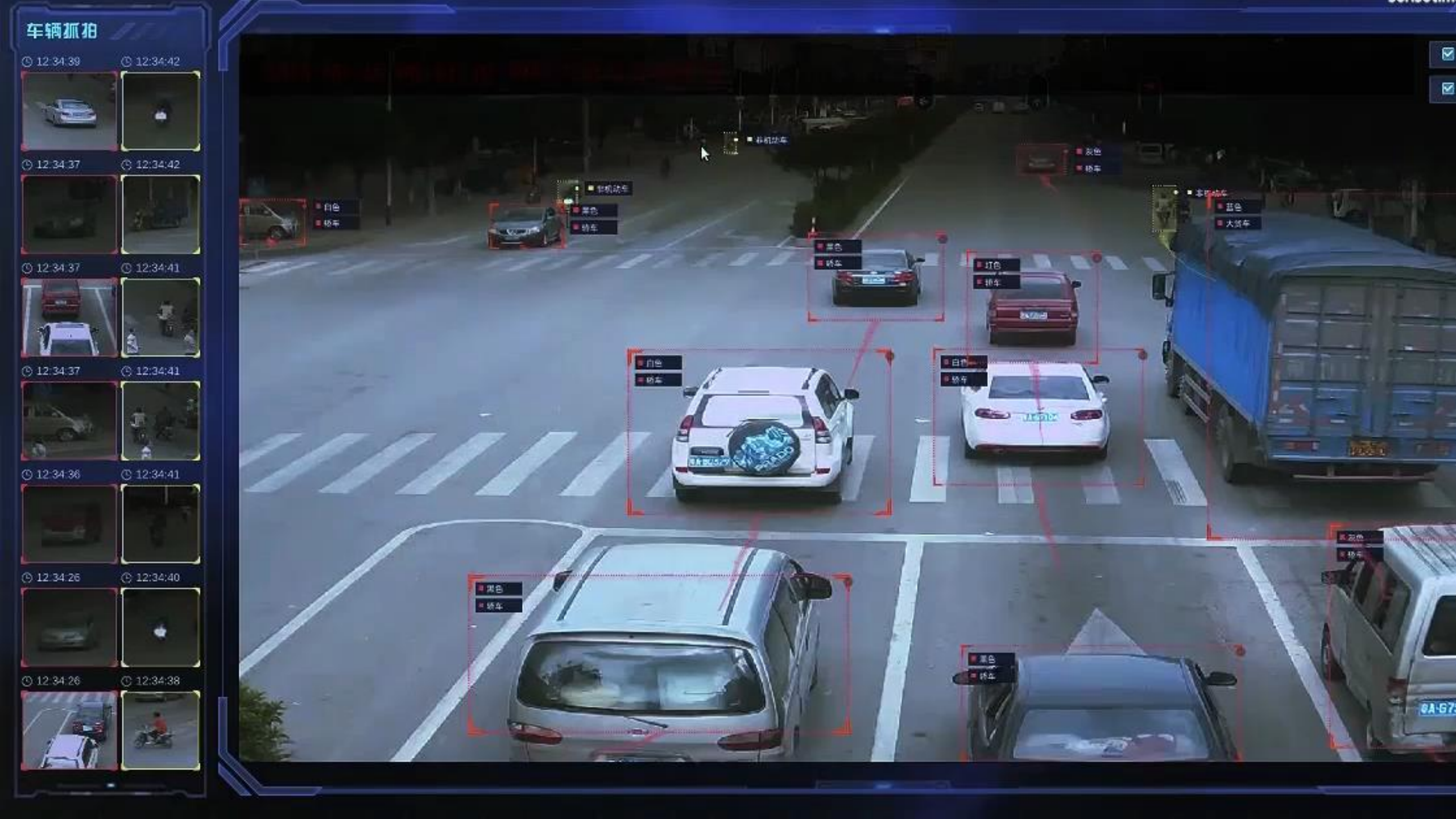
http://www.computervisionmodels.com/

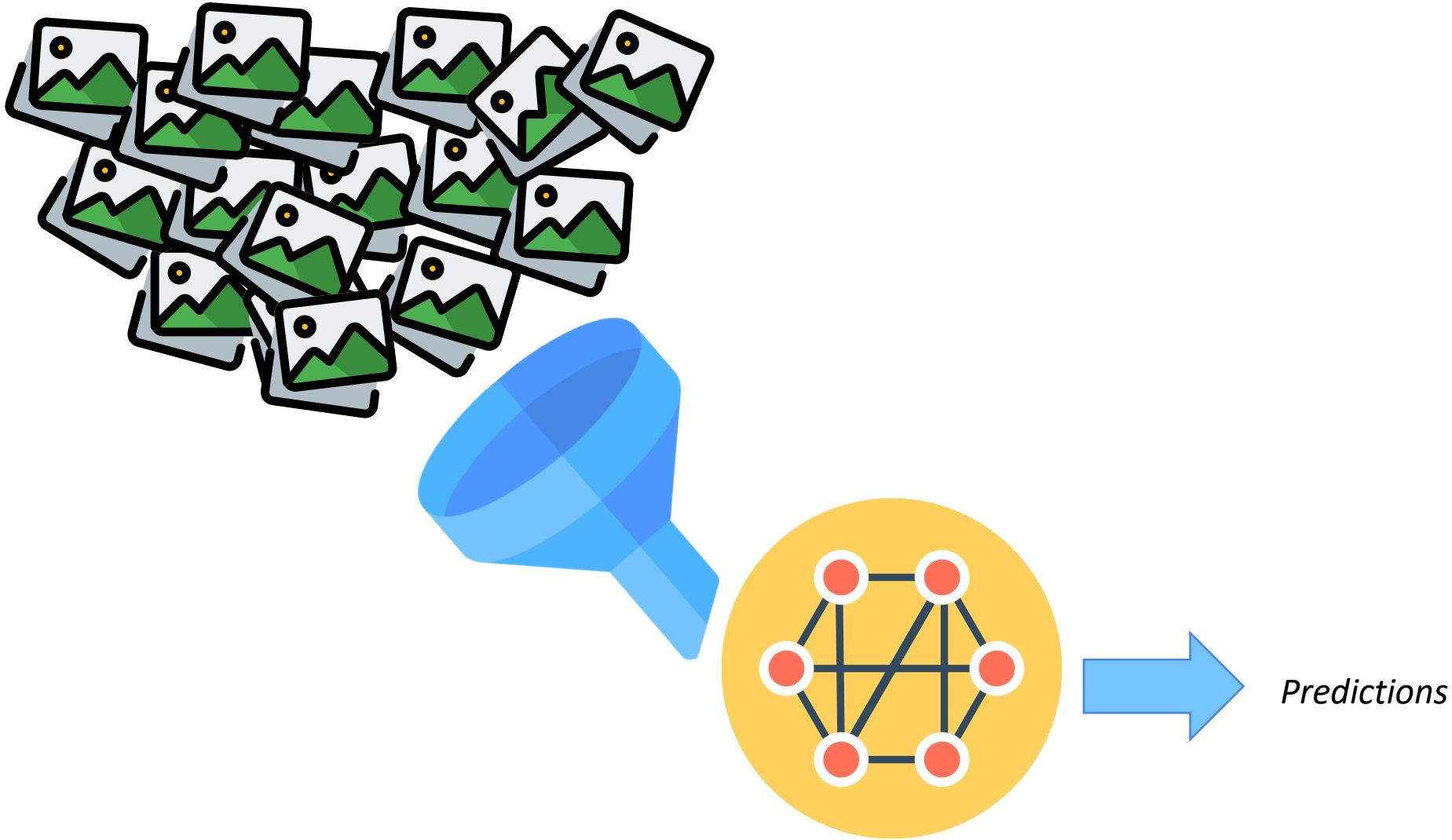
Outline

- About this course
- Computer vision background and a little history
- Fundamentals of machine learning
  - Why learning?
  - Statistical learning

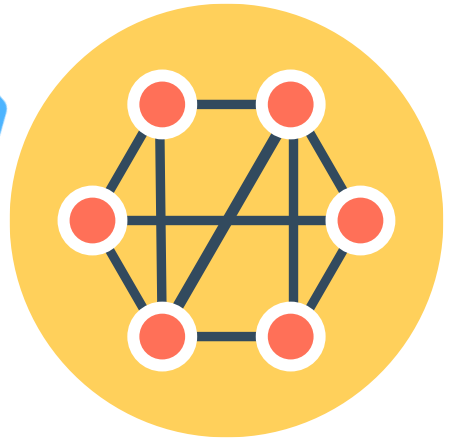

Artificial Intelligence
Machine Learning
Computer Vision

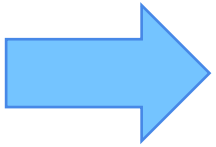

FUTURIUM


车辆抓拍
12:34:39
12:34:42
12:34:37
12:34:42
12:34:37
12:34:41
12:34:37
12:34:41
12:34:36
12:34:41
12:34:26
12:34:40
12:34:26
12:34:38
白色
轿车
黑色
轿车
灰色
轿车
红色
轿车
白色
轿车
白色
轿车
黑色
轿车
黑色
轿车
灰色
轿车
蓝色
大货车



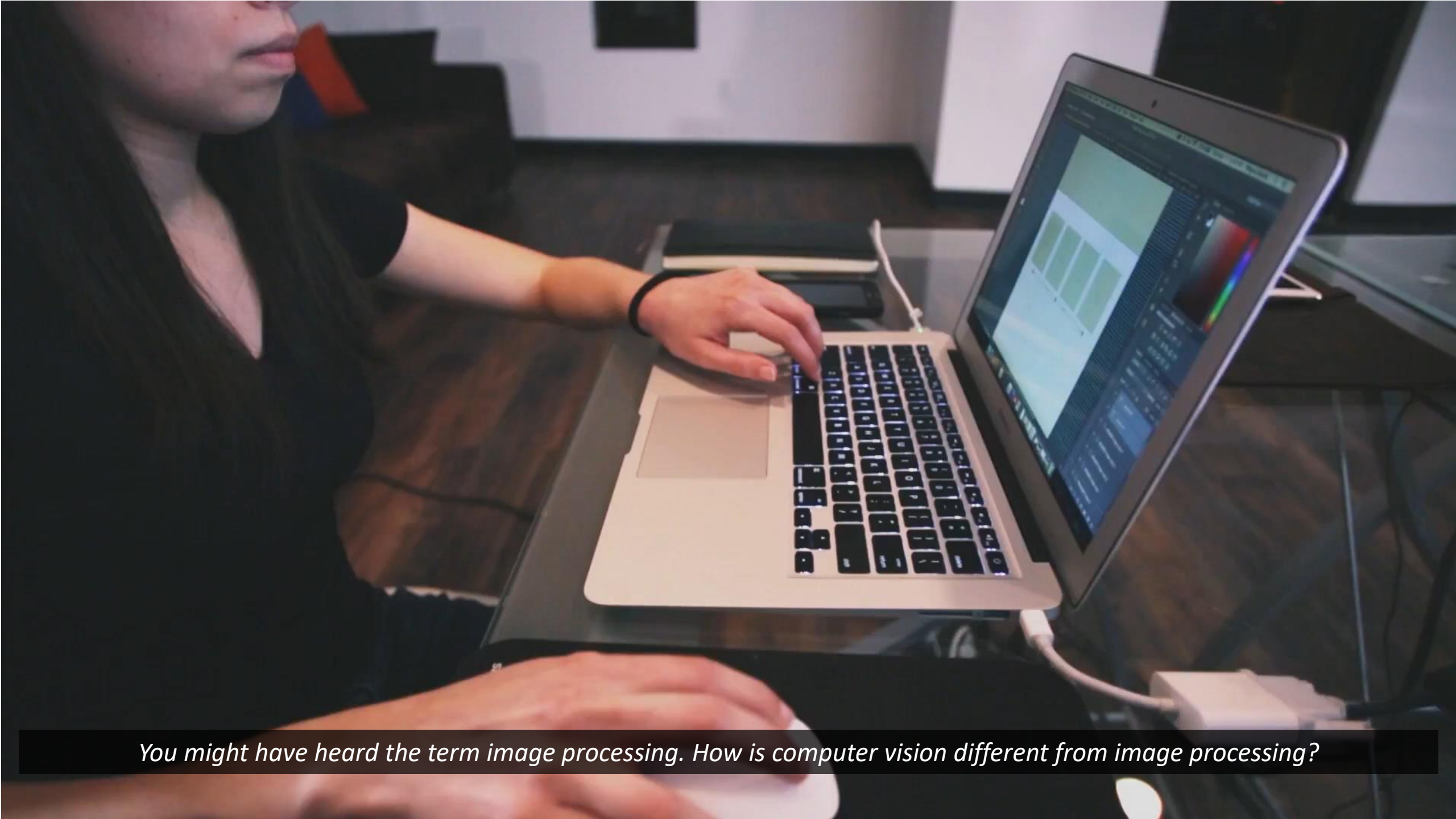
A Computer Vision Model





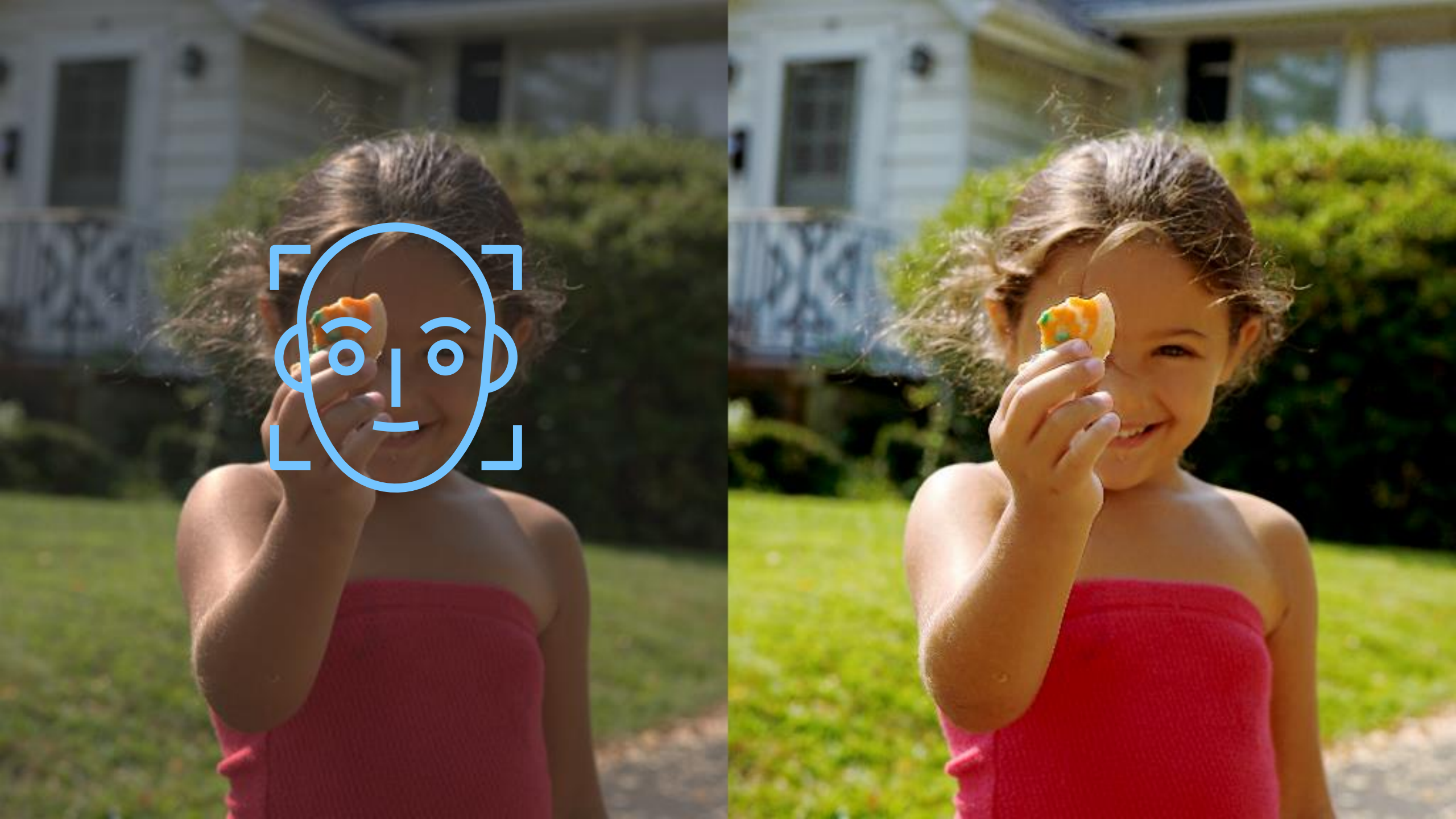
Chihuahua
or
Muffin?

A Computer Vision Model

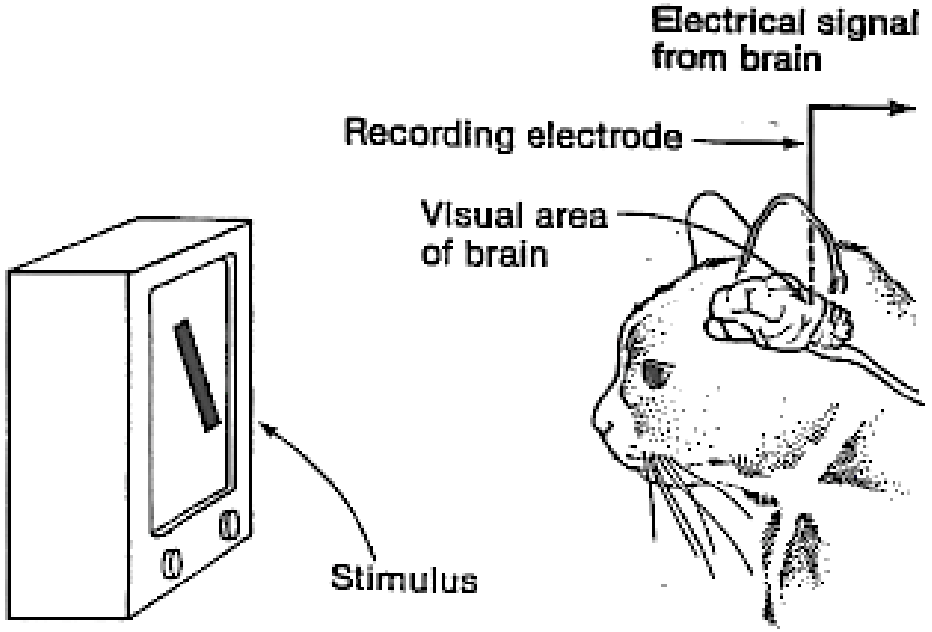


You might have heard the term image processing. How is computer vision different from image processing?

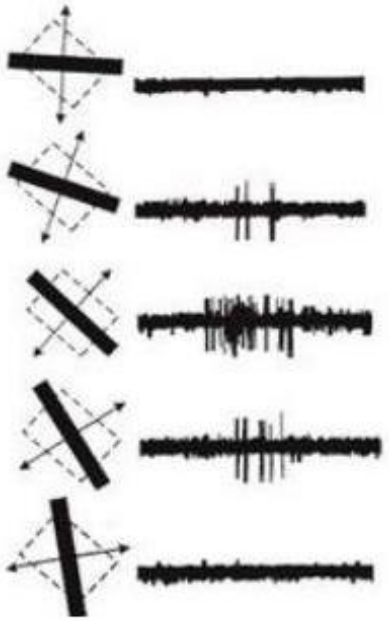
Image processing is the process of restoring or enhancing an image, by changing brightness, contrast or anything. It is just a type of digital signal processing and is not concerned with understanding the content of an image.

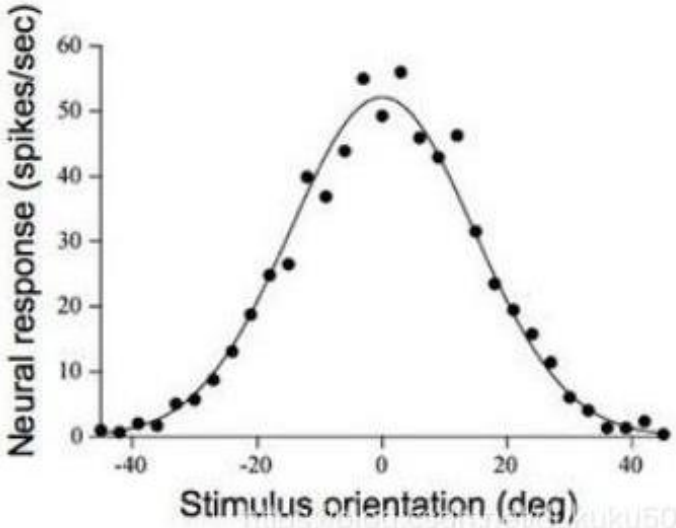


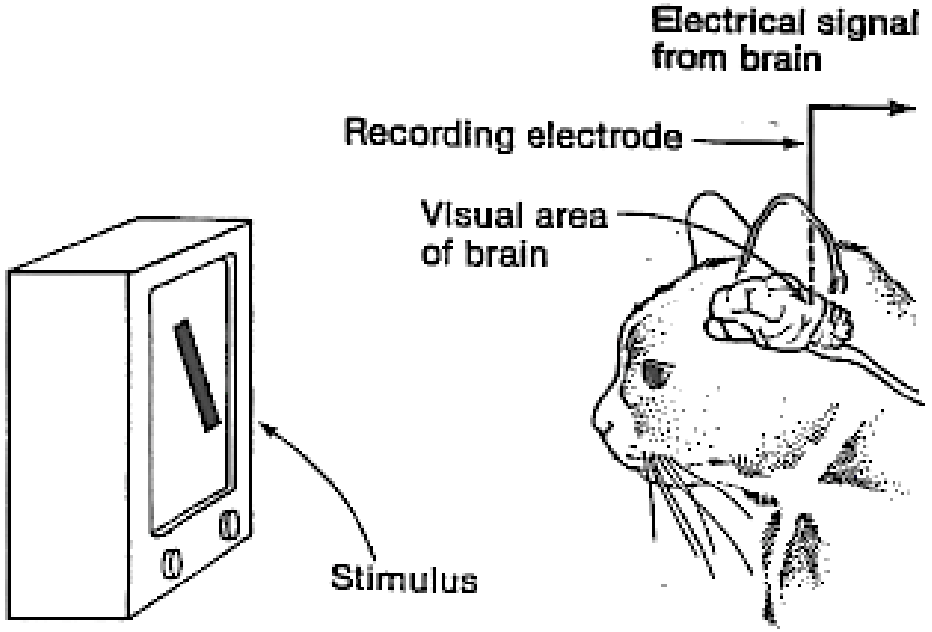
Experimentation began in 1959 when neurophysiologists Hubel and Wiesel recorded electrical activity from individual neurons in the brains of cats



They showed a cat an array of images, attempting to correlate a response in its brain.



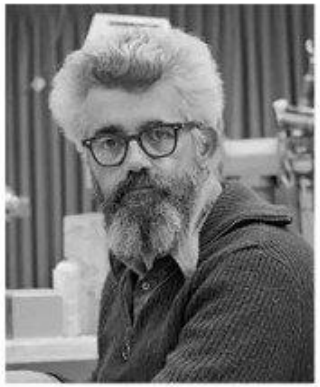




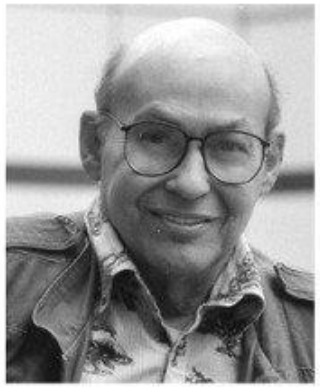
They discovered that it responded first to hard edges or lines, and scientifically, this meant that image processing starts with simple shapes like straight edges.

At about the same time, the first computer image scanning technology was developed, enabling computers to digitize and acquire images.

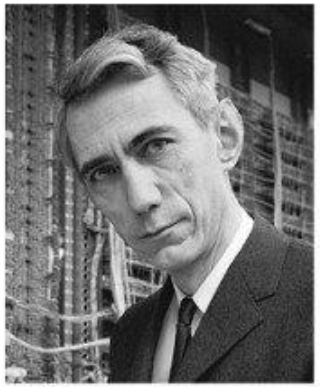
1956 Dartmouth Conference: The Founding Fathers of AI



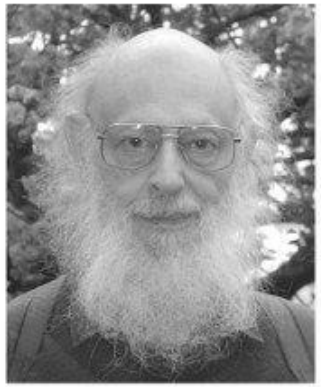
John MacCarthy



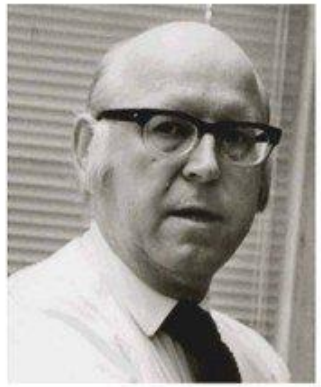
Marvin Minsky



Claude Shannon



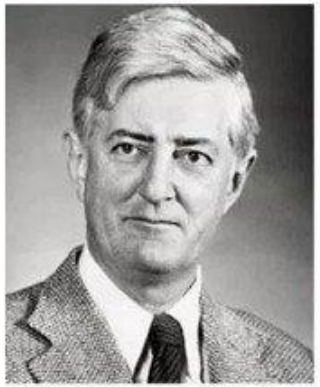
Ray Solomonoff



Alan Newell

Herbert Simon

Arthur Samuel

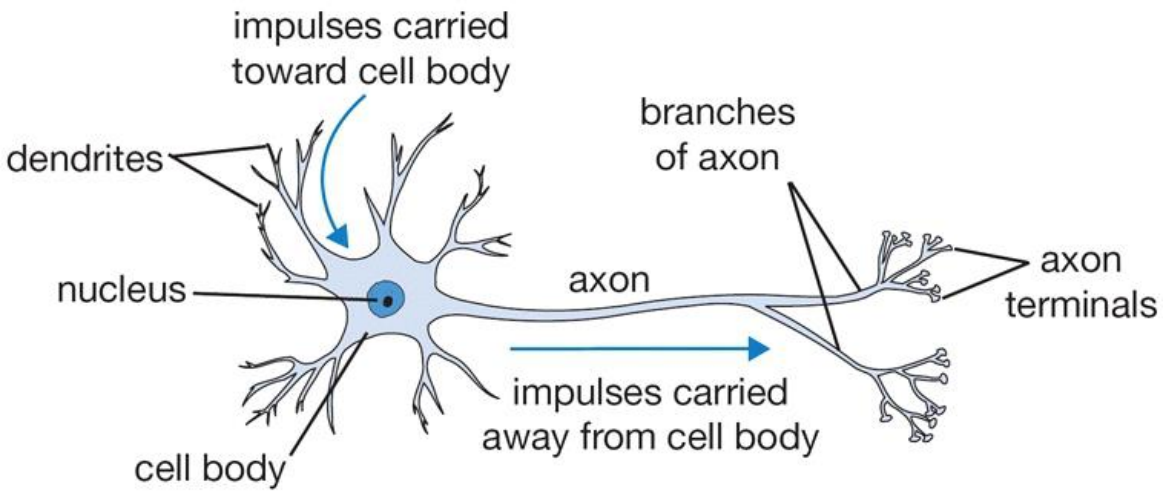


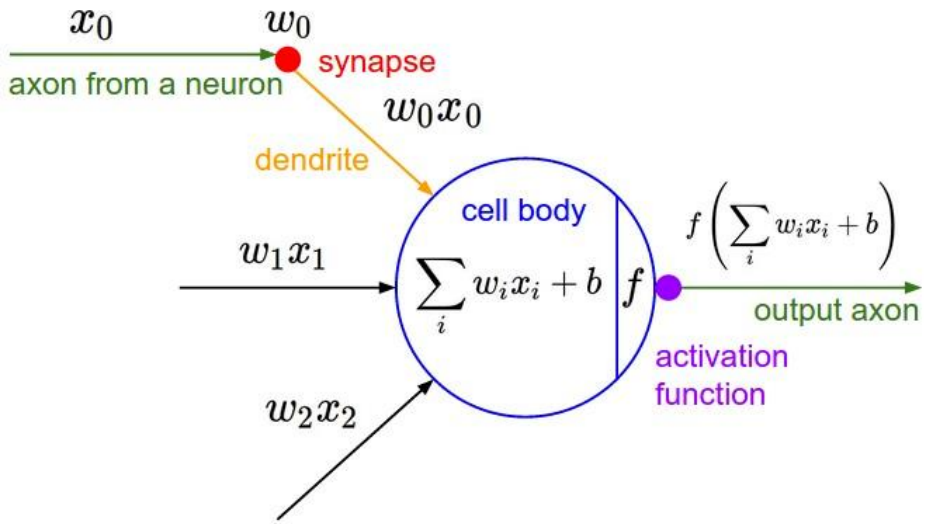
Oliver Selfridge

Nathaniel Rochester

Trenchard More

In the 1960s, AI emerged as an academic field of study, and it also marked the beginning of the AI quest to solve the human vision problem.





Neuron

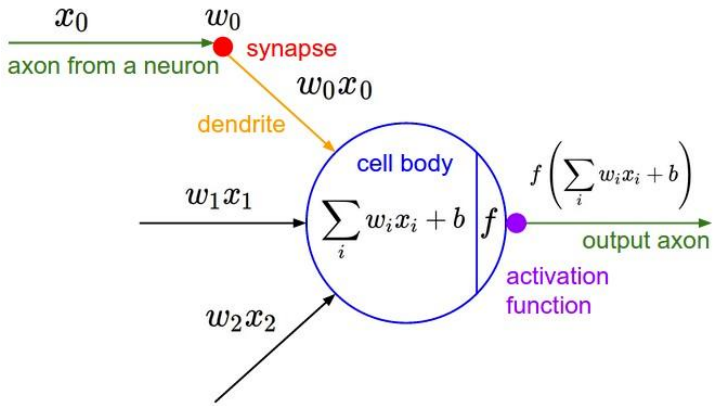
: a nerve cell

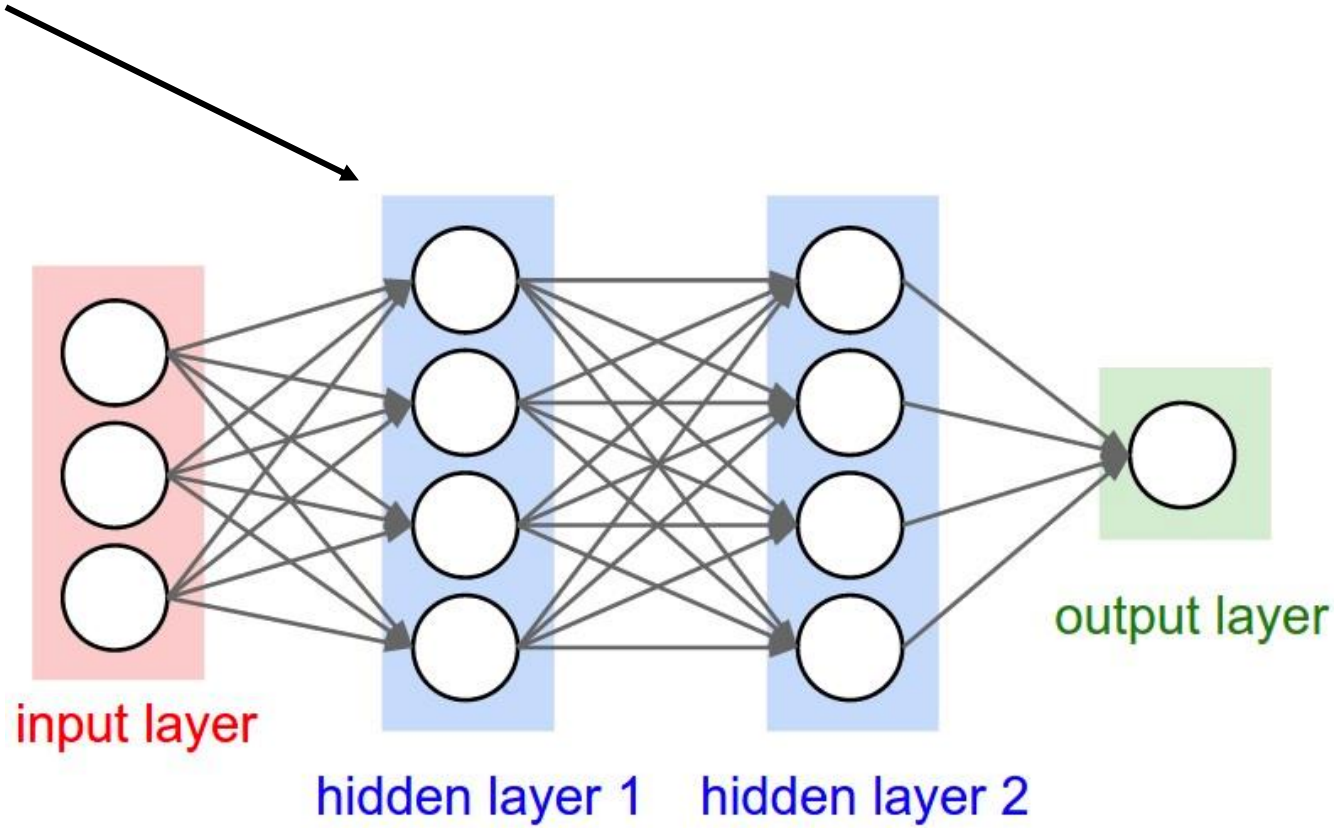
: fundamental unit of the nervous system

Perceptron (1958) : Frank Rosenblatt

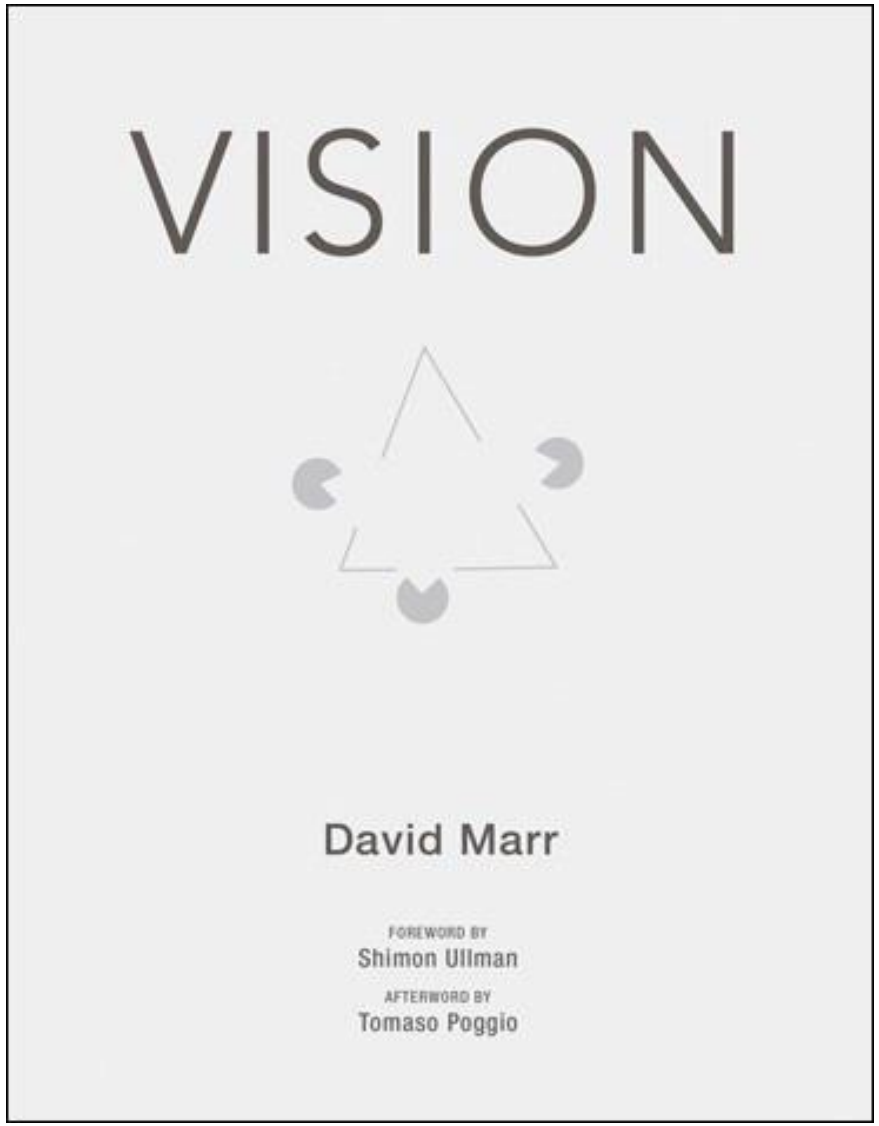
: basic building block for neural networks

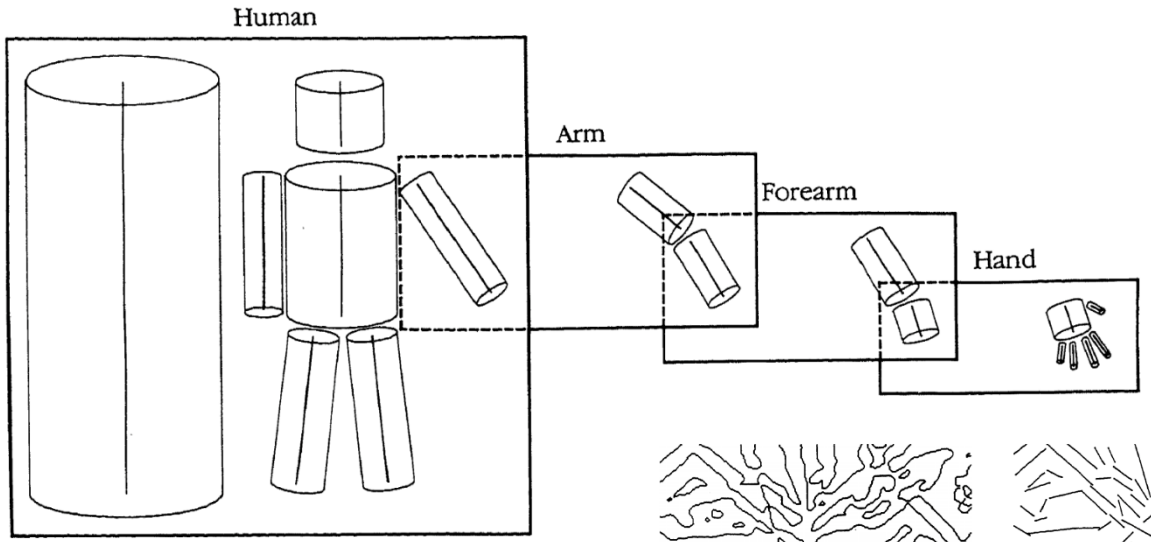
: algorithm for pattern recognition

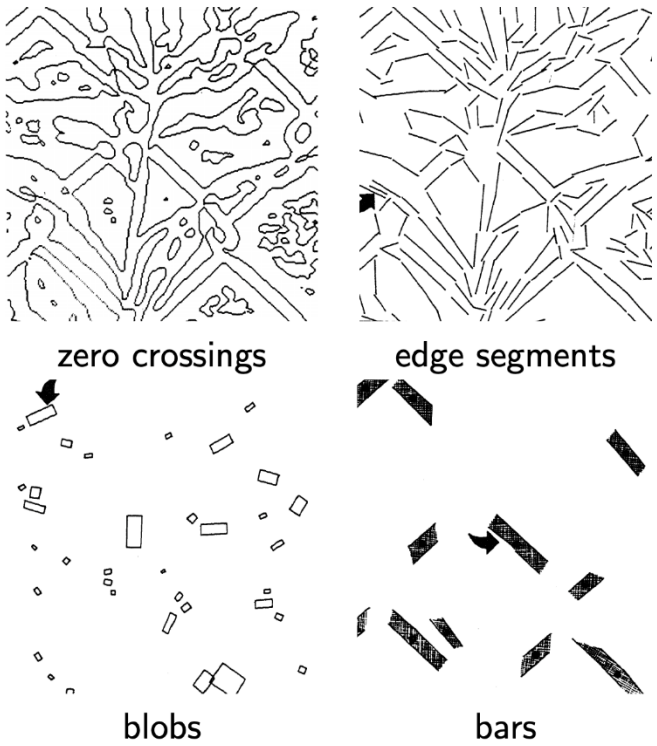




A neural network is nothing but a collection of many interconnected neurons arranged in a hierarchical manner.

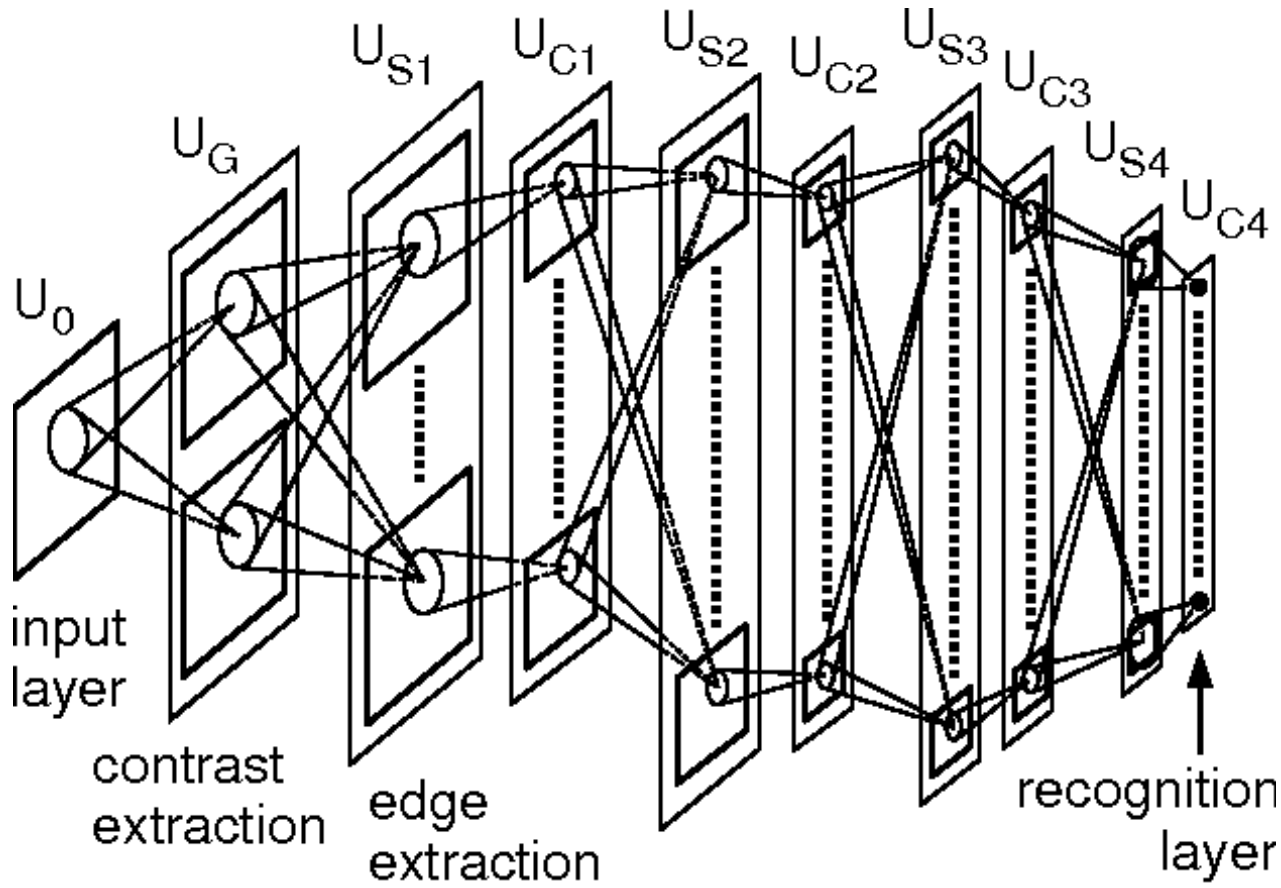






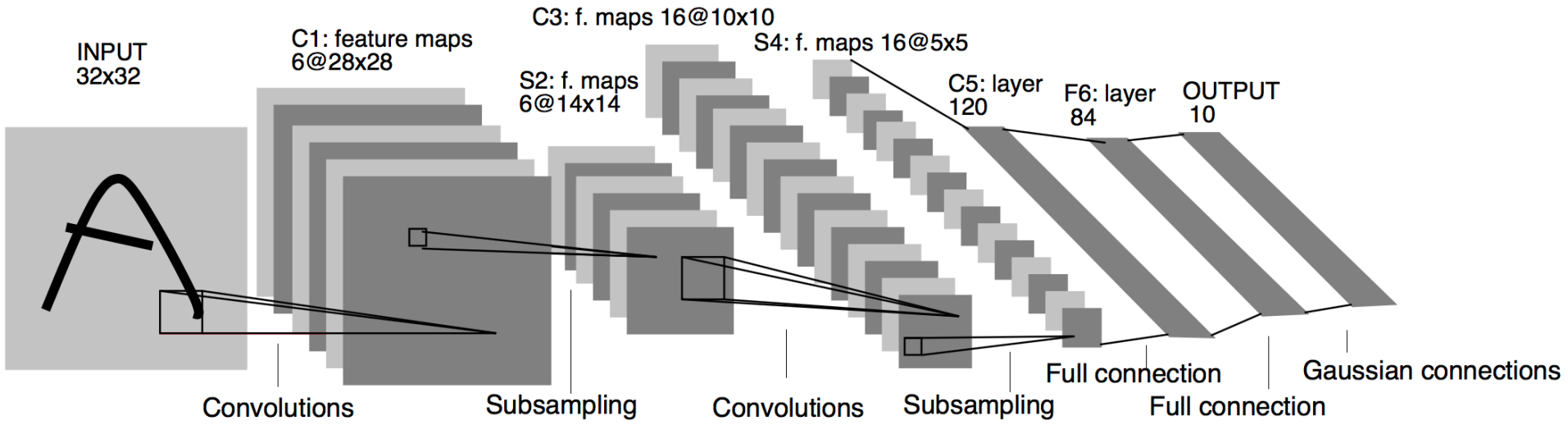
In 1982, neuroscientist David Marr established that vision works hierarchically and introduced algorithms for machines to detect edges, corners, curves and similar basic shapes.

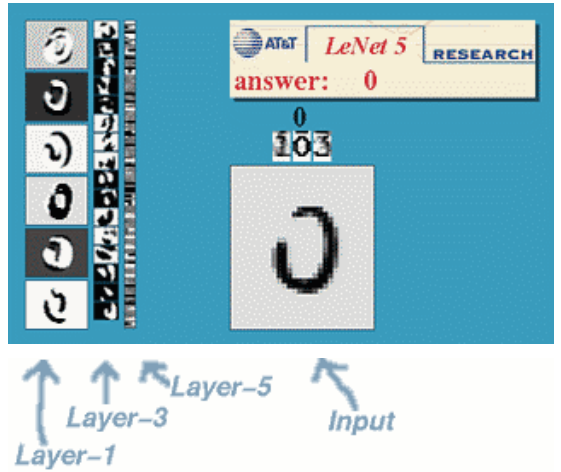
Kunihiko Fukushima

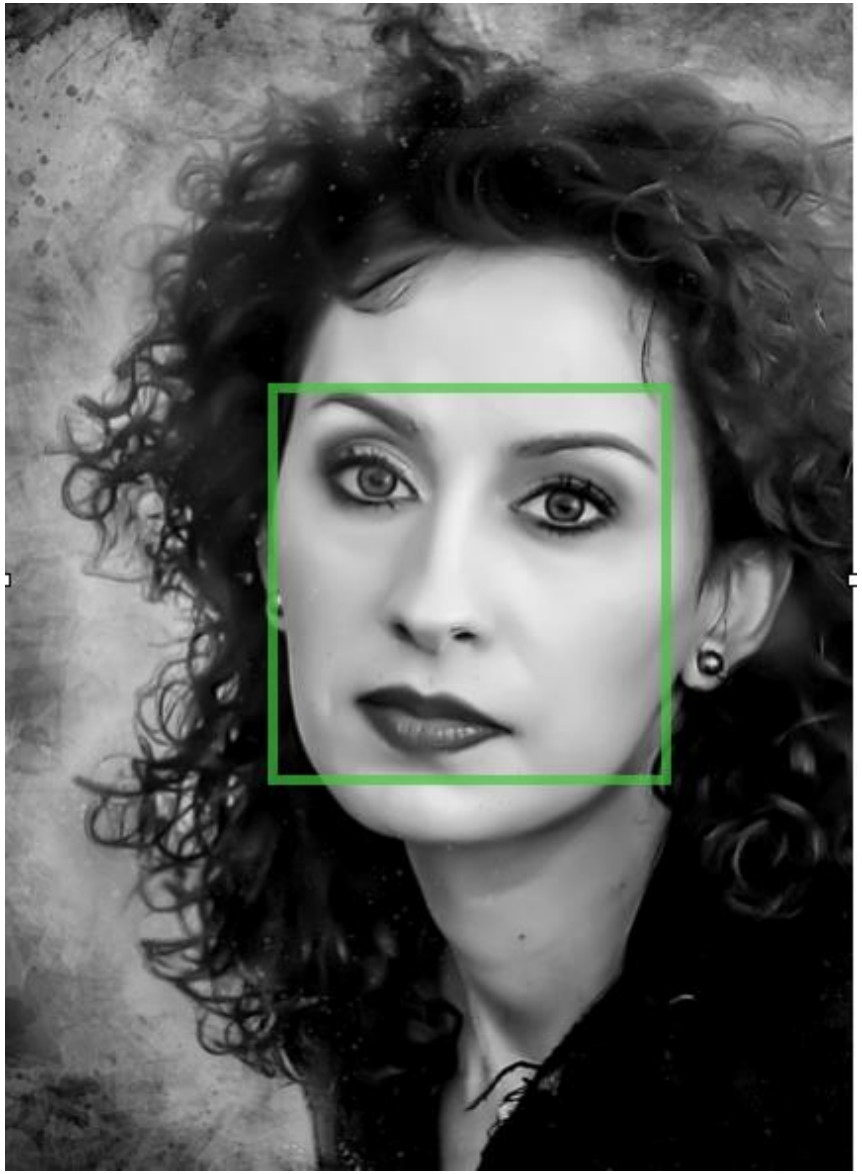


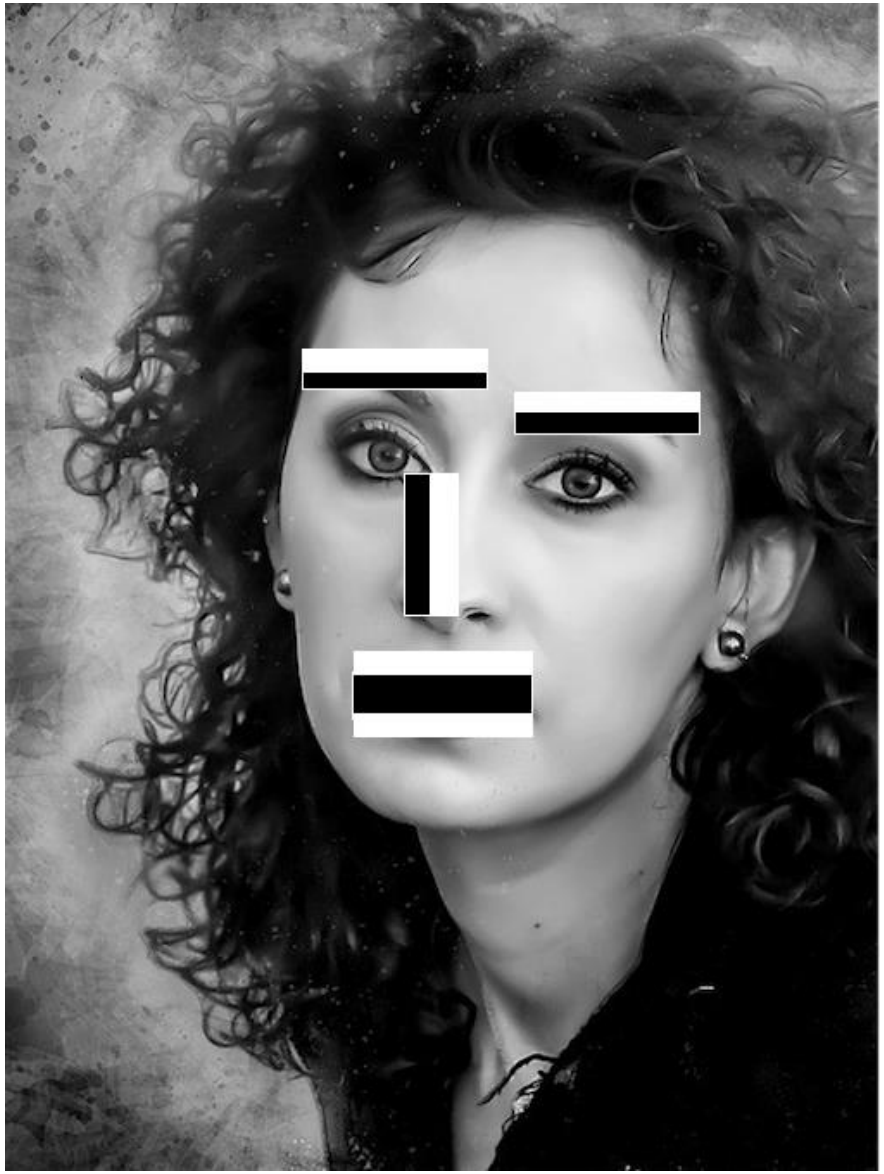
Neocognitron : network of cells to recognize patterns
: included convolutional layers

Yann LeCun 1998 : neural network in which parameters are shared spatially
: used by banks to recognize hand written numbers on cheques

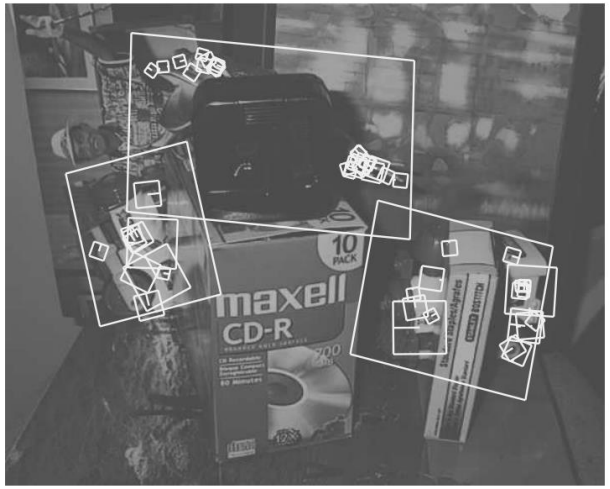




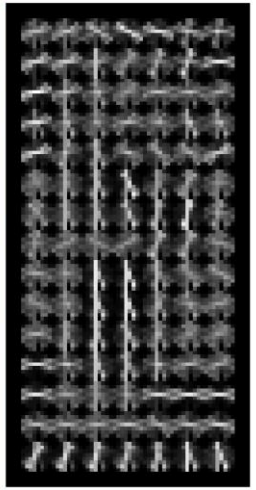


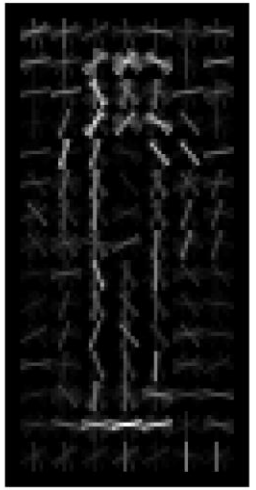


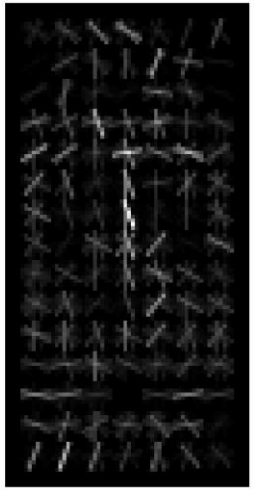
By 2001, the first real-time face detection application appeared.



SIFT







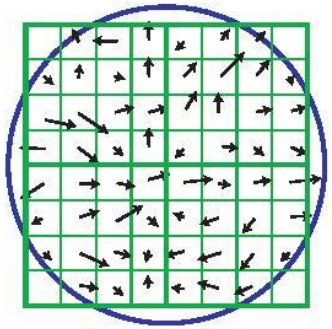
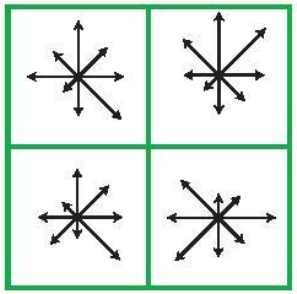
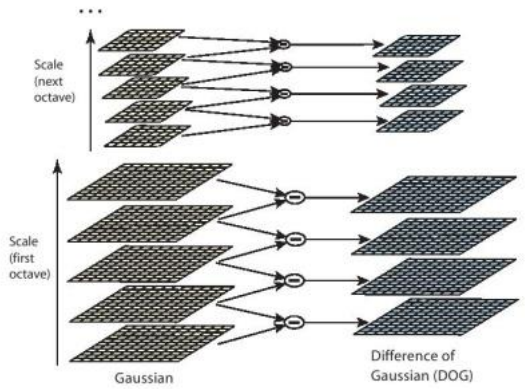
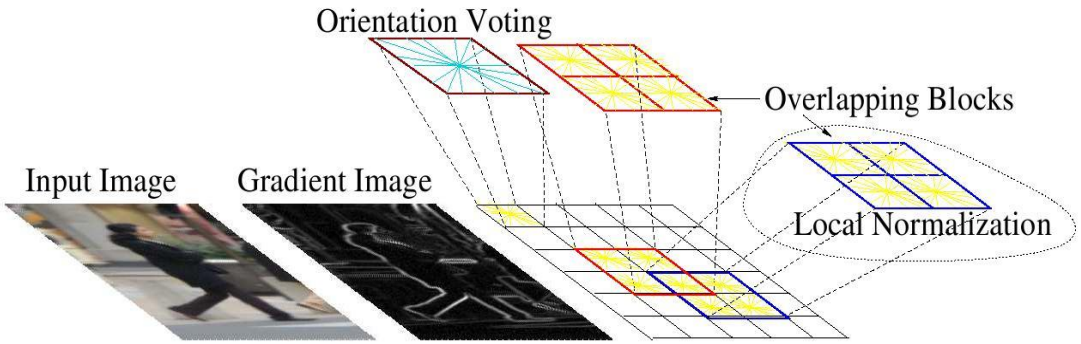


Image gradients

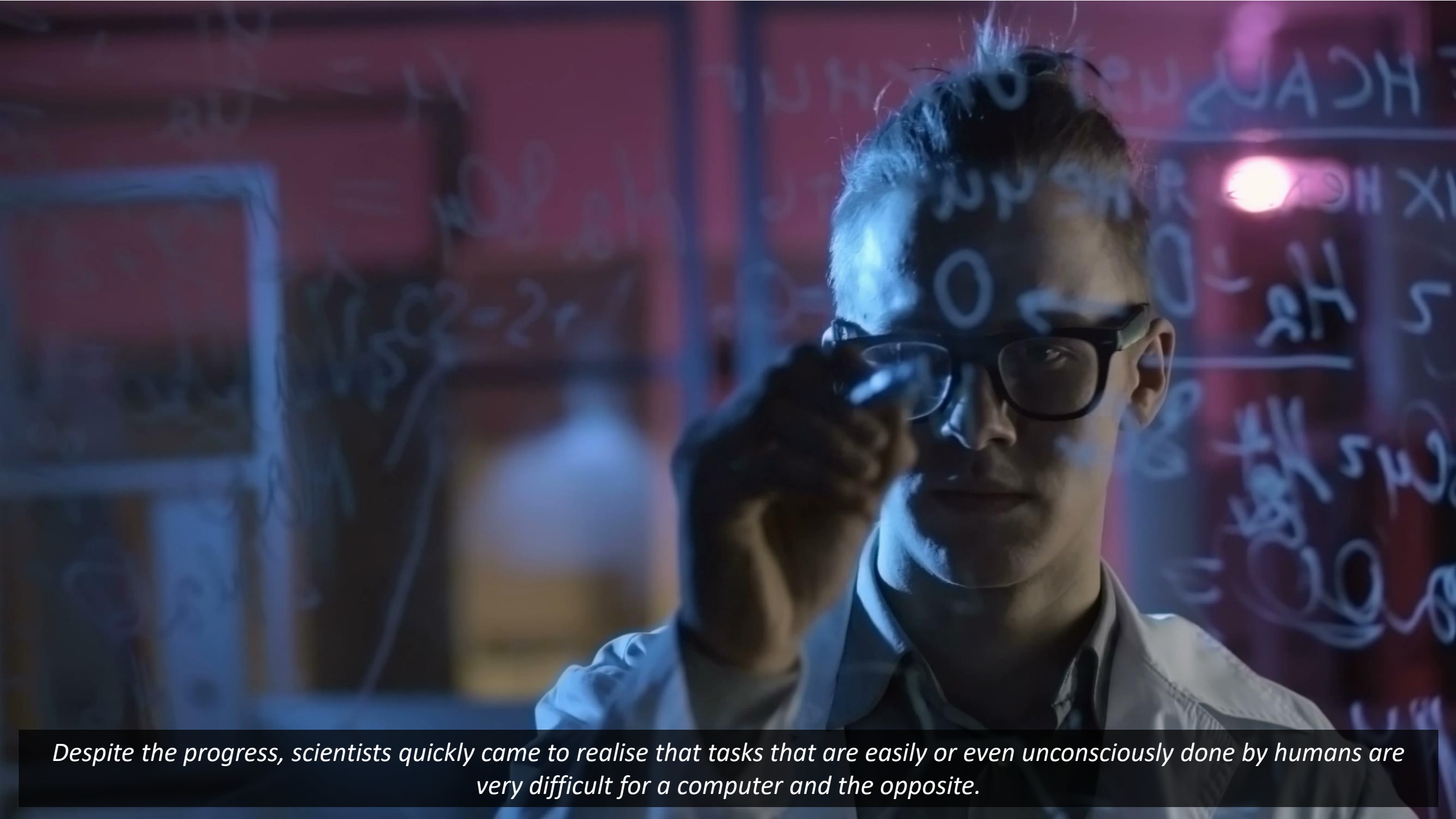


Keypoint descriptor





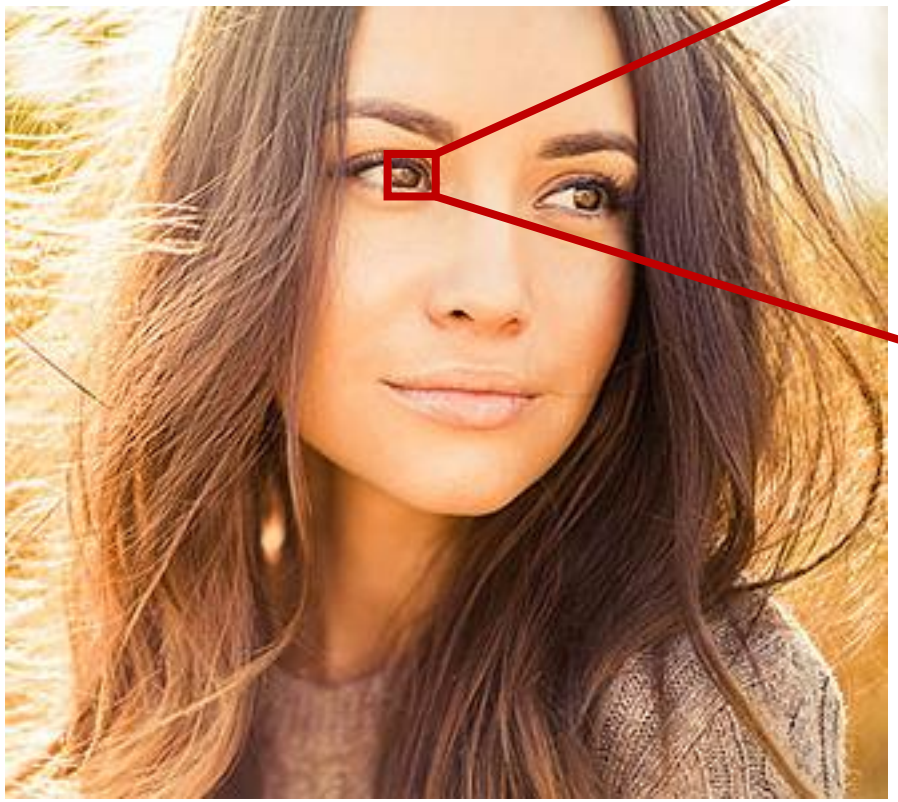
There are many more development in the field thereafter, including better methods to extract features and the use of machine learning techniques for solving tasks like object recognition and detection.

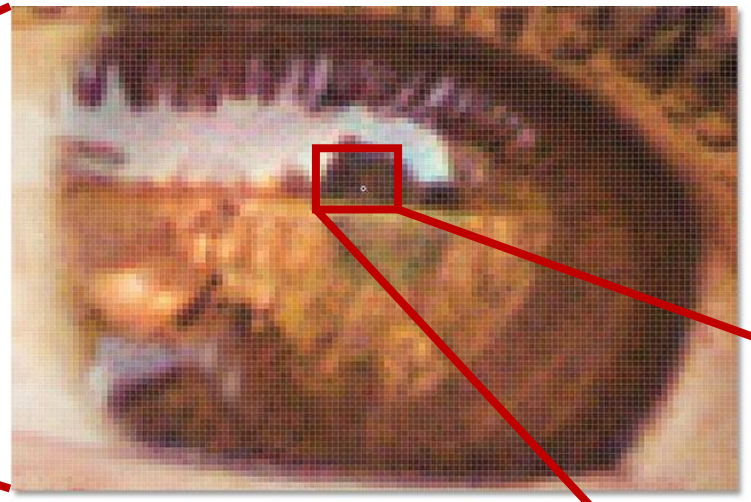


Despite the progress, scientists quickly came to realise that tasks that are easily or even unconsciously done by humans are very difficult for a computer and the opposite.

But why is it so challenging? The perception of humans comes with a pre-existing knowledge about the object and the geometry

Seeing a 2D image, one can easily distinguish between foreground and background objects and effortlessly recognise an object even if it is subject to occlusions, background clutter or deformations.





$$
\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \vdots \\ m \end{array}
\begin{array}{c}
\begin{array}{cccc} 1 & 2 & \dots & n \end{array} \\
\begin{bmatrix}
a_{11} & a_{12} & \dots & a_{1n} \\
a_{21} & a_{22} & \dots & a_{2n} \\
a_{31} & a_{32} & \dots & a_{3n} \\
\vdots & \vdots & \vdots & \vdots \\
a_{m1} & a_{m2} & \dots & a_{mn}
\end{bmatrix}
\end{array}
$$


Viewpoint Variations


Scale Variations


Illumination Variations

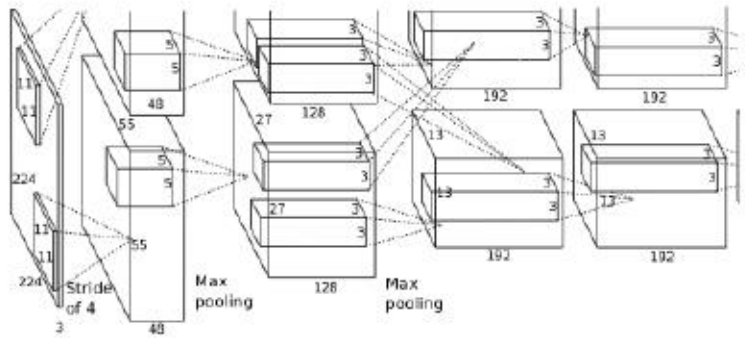


Intra-Class Variations

The Deep Learning "Computer Vision Recipe"

Big Data: ImageNet

+



Deep Convolutional Neural Network

+

Backprop on GPU

=

Learned Weights

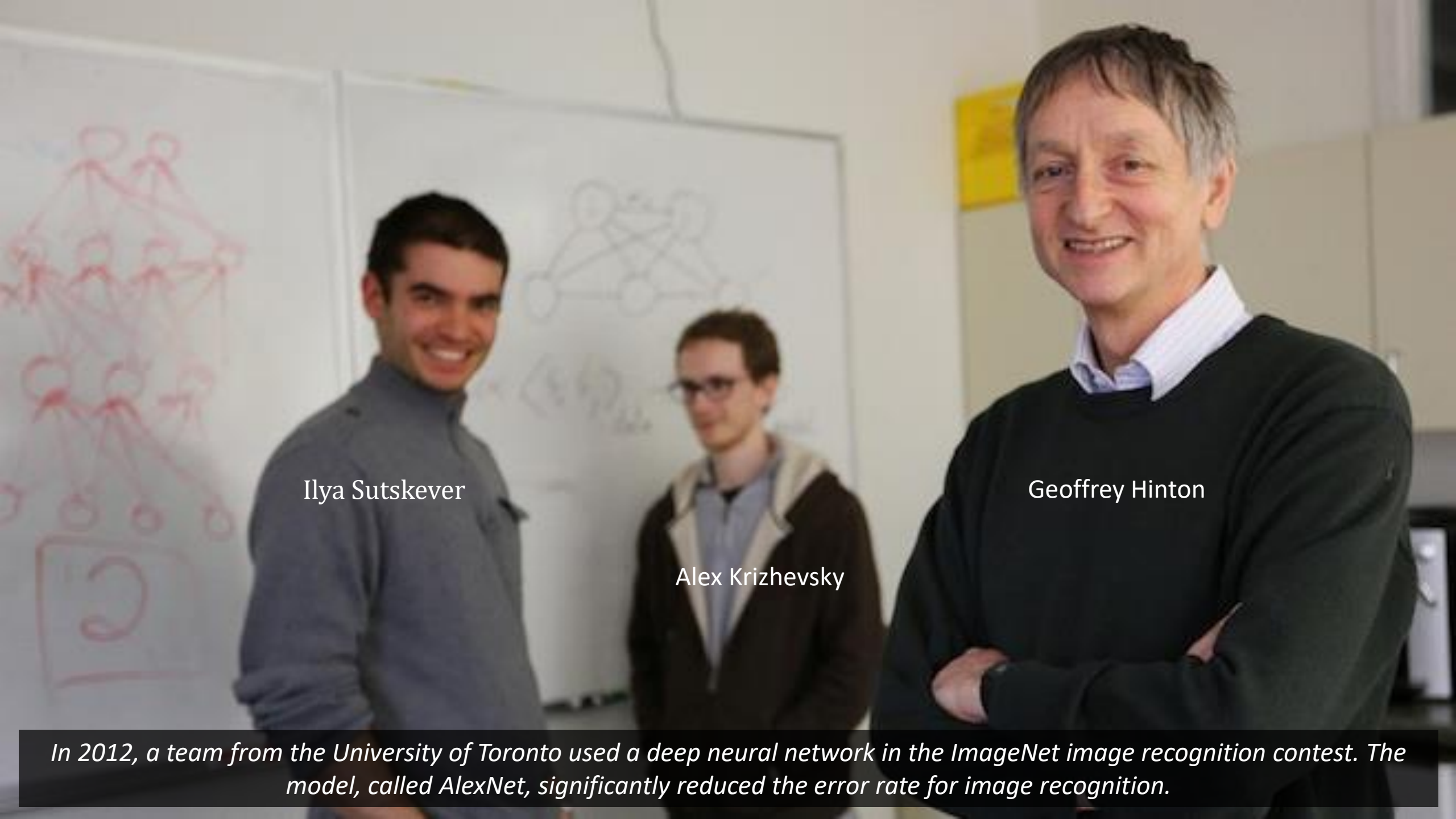
2011 - 2012



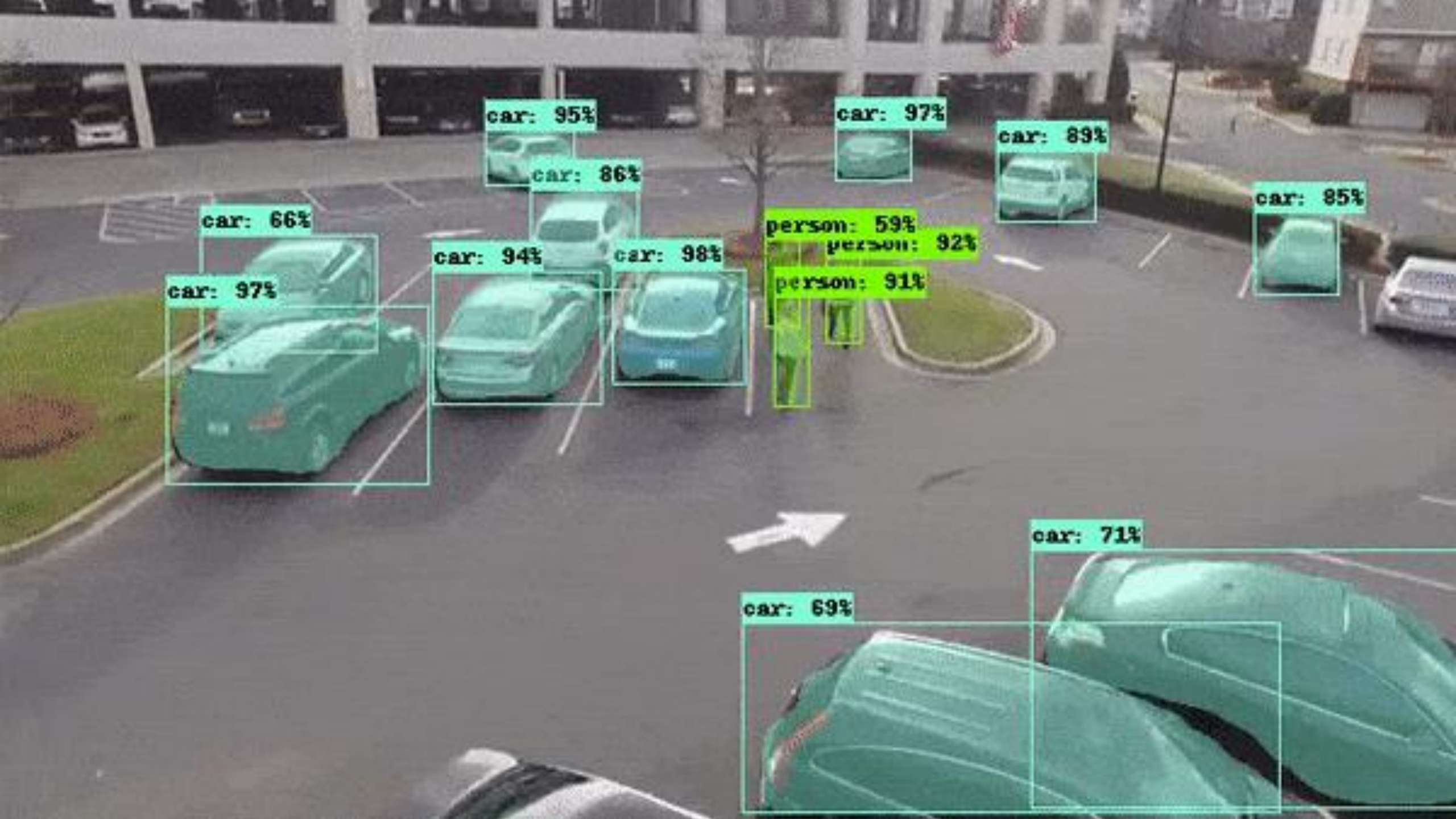


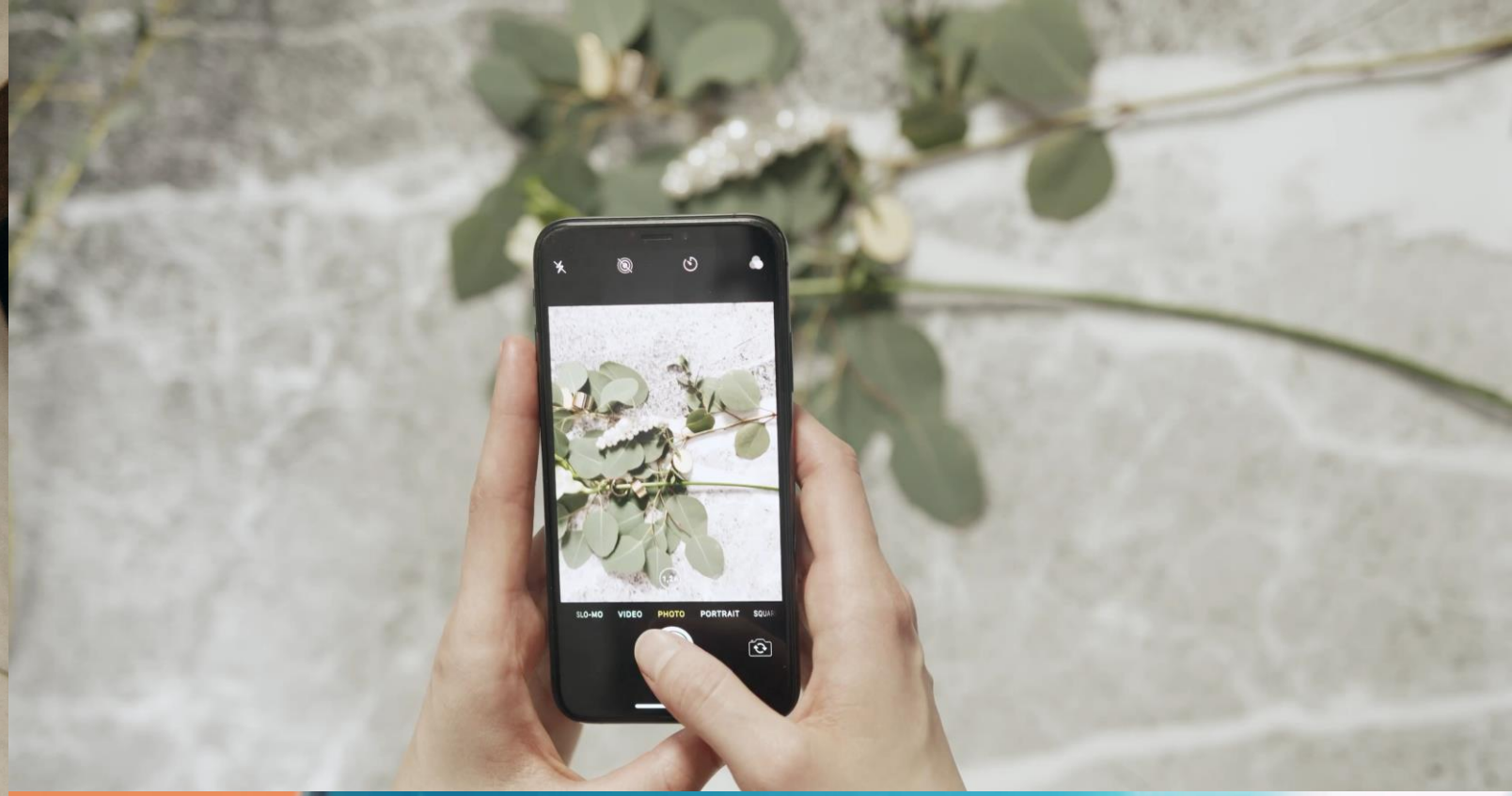



In 2010, the ImageNet dataset became available. It contained millions of tagged images across a thousand object classes and provides a foundation for training CNNs and deep learning models used today.



In 2012, a team from the University of Toronto used a deep neural network in the ImageNet image recognition contest. The model, called AlexNet, significantly reduced the error rate for image recognition.


car: 95%
car: 97%
car: 89%
car: 86%
car: 85%
car: 66%
person: 59%
person: 92%
car: 94%
car: 98%
person: 91%
car: 97%
car: 71%
car: 69%


SLO-MO
VIDEO
PHOTO
PORTRAIT

HOW WAYMO'S SELF-DRIVING CAR WORKS

One of Waymo's three lidar systems that shoots lasers so the car can see its surroundings. Waymo says this lidar can detect a helmet two-football fields away.

A forward facing camera works with 8 others stationed around the car to provide 360 degrees of vision.

Radar sensors can detect objects in rain, fog, or snow.

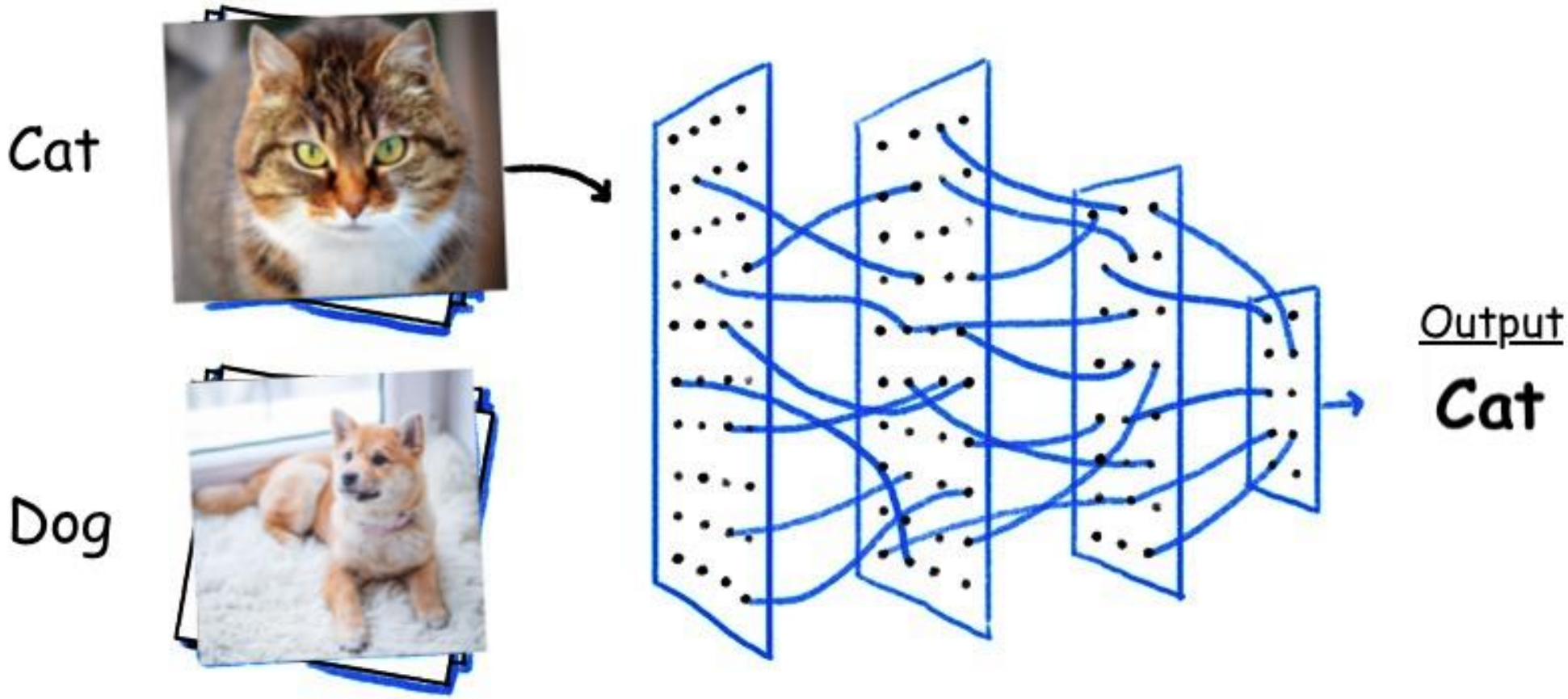
Waymo's self-driving sensors are tightly integrated into the hybrid minivan created by Fiat Chrysler.

SOURCE: Waymo

BUSINESS INSIDER


SELF DRIVING
MODE
40

Image Classification

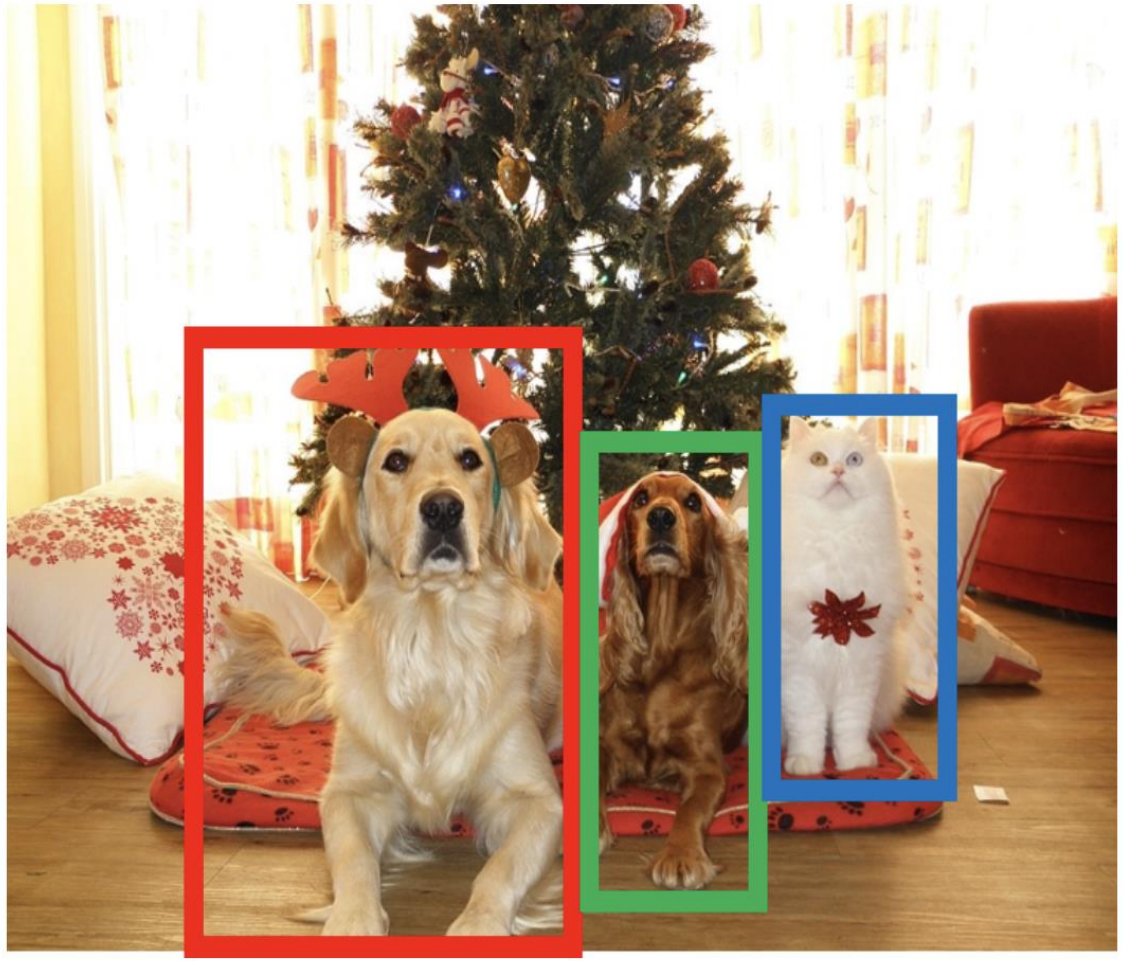


Object Detection

Input: Single RGB Image

Output: A set of detected objects;
For each object predict:

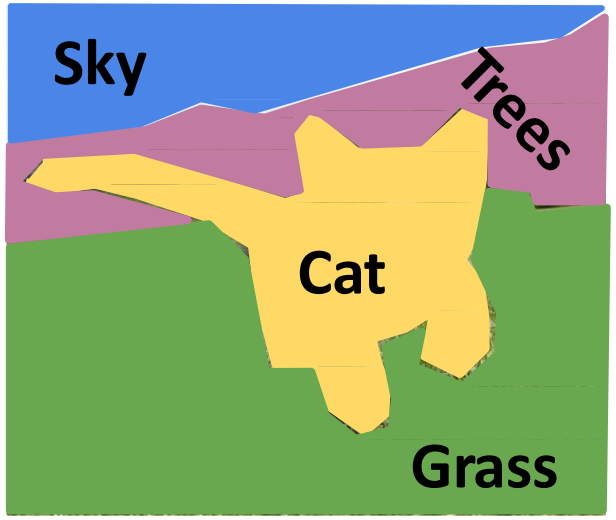
1. Category label (from fixed, known set of categories)
2. Bounding box (four numbers: x, y, width, height)

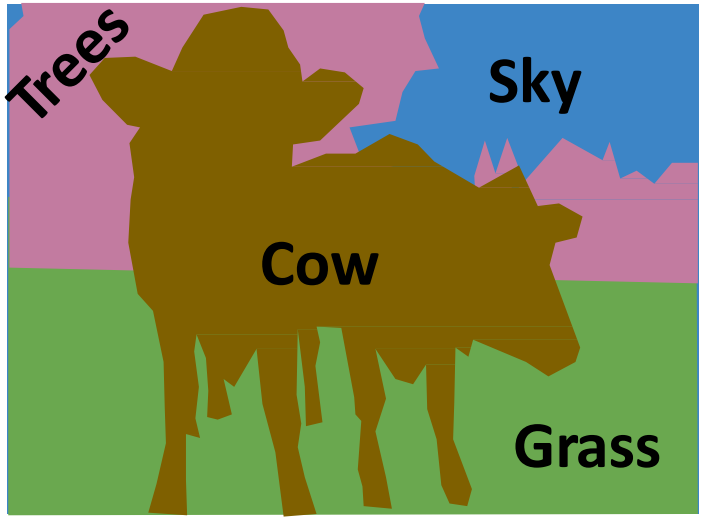


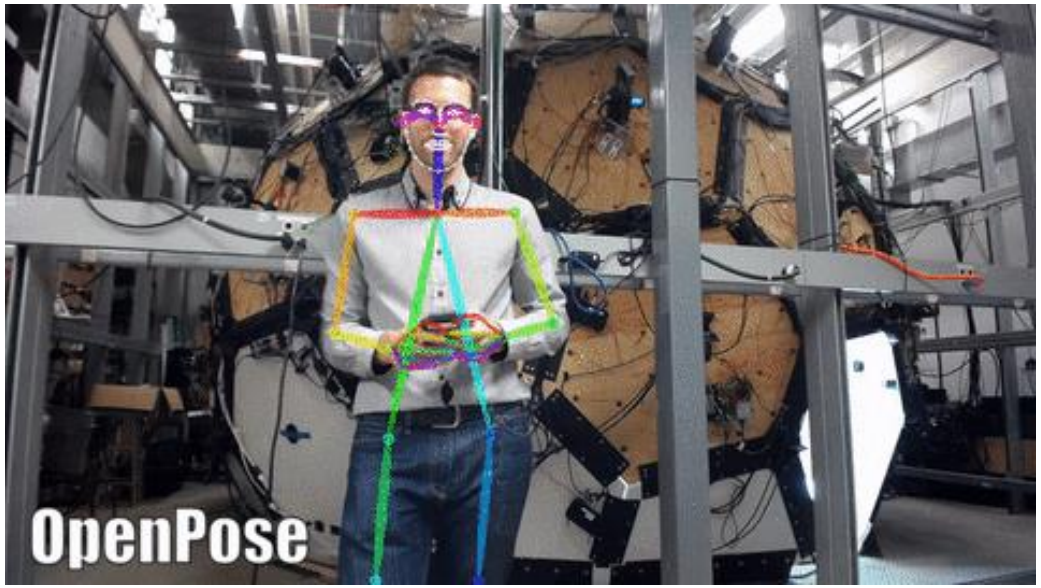
Semantic Segmentation

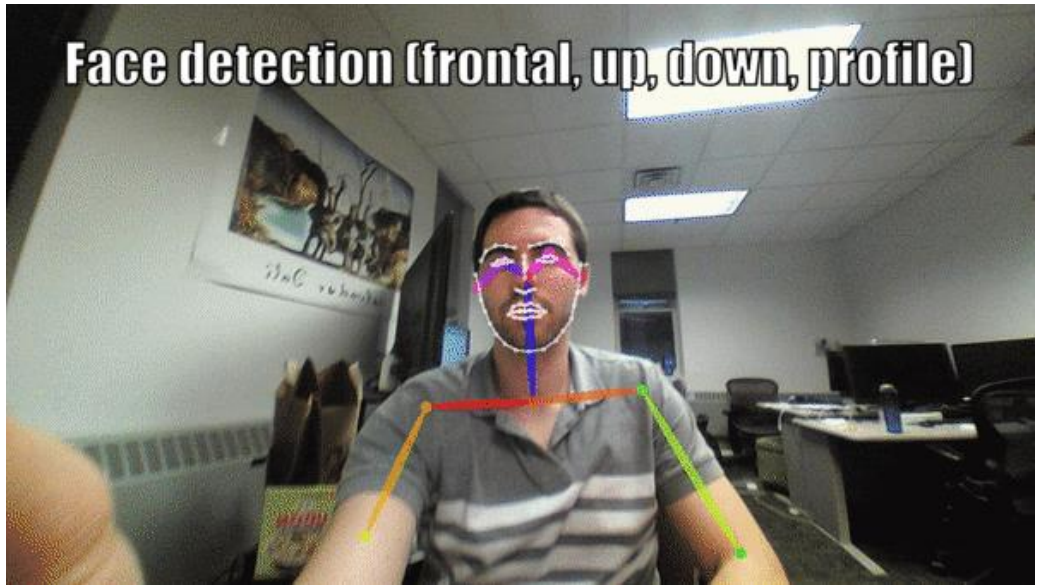
Input: Single RGB image

Output: Label each pixel in the image with a category label

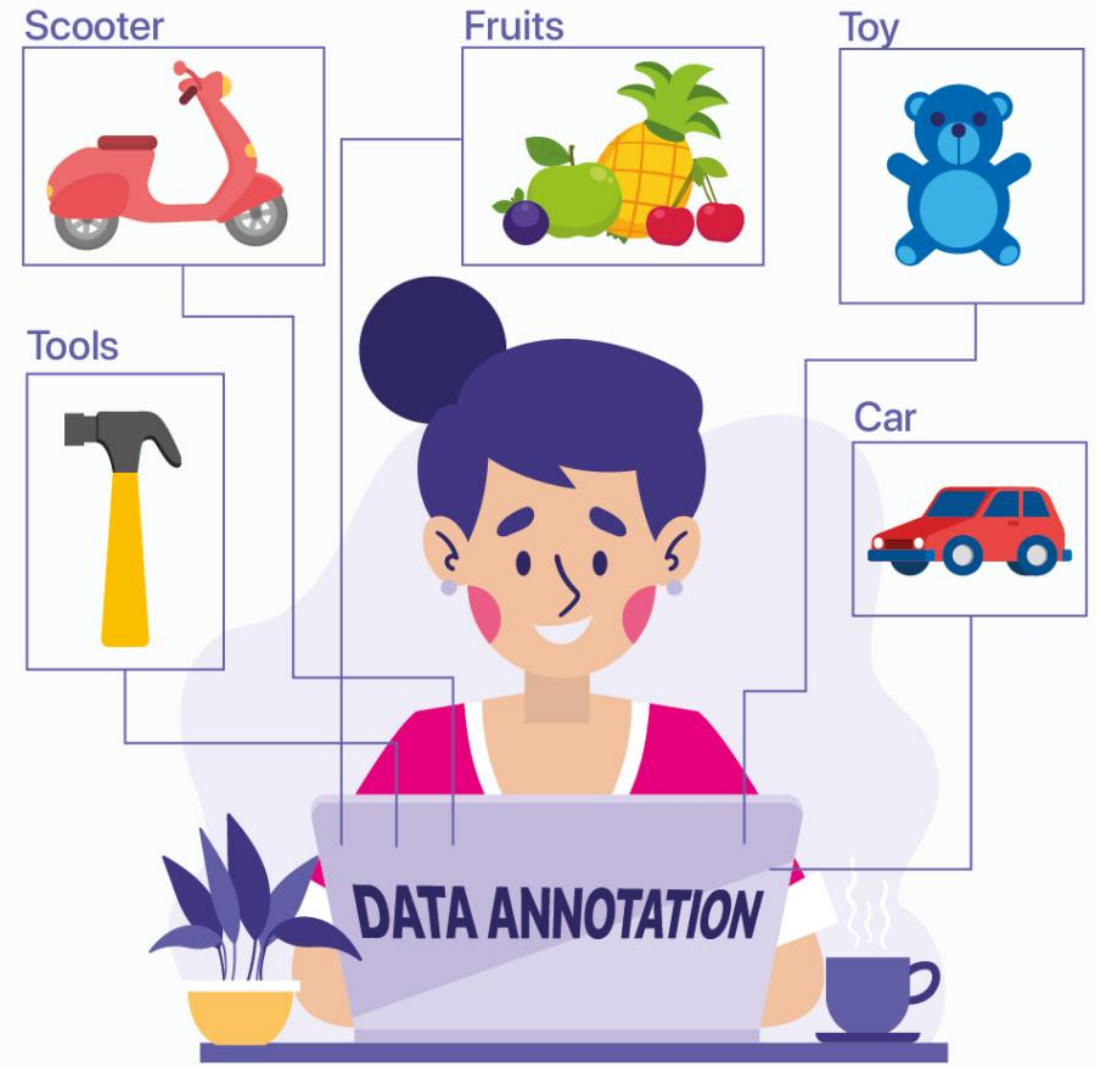








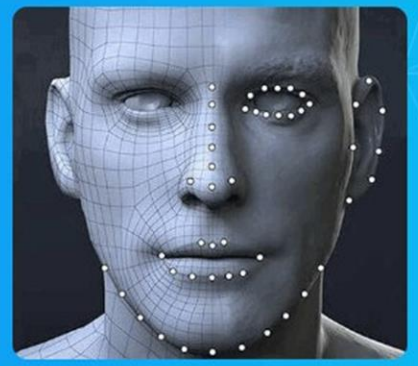
Another task is known as keypoint detection, which involves detecting people and localizing their key points simultaneously. Keypoints are spatial locations or points in the image that define what is interesting or what stands out in the image.



Lots of images, each annotated with a class label.

2D Bounding Boxes

Cuboid



Point & Landmark

Lines & Splines

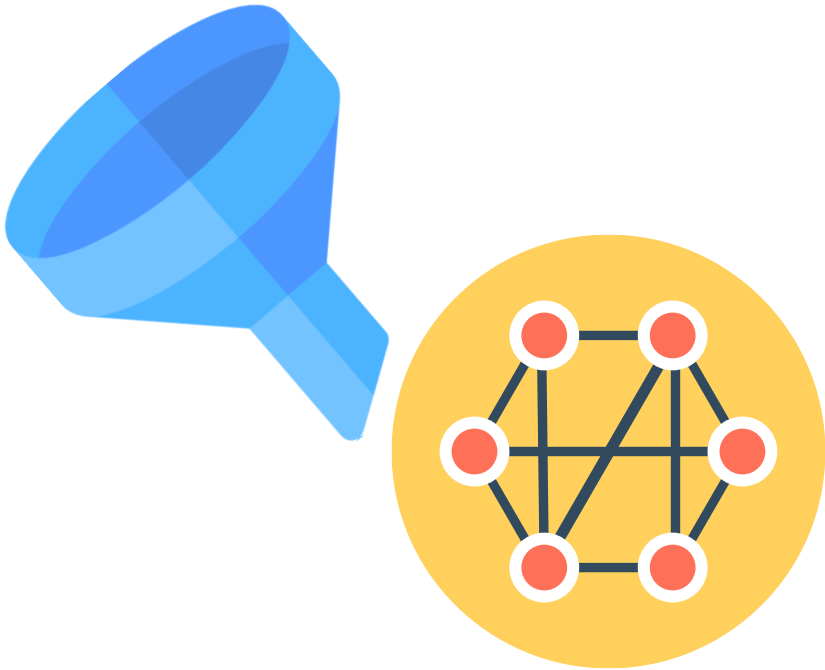
Text Annotation

Polygons

Semantic Segmentation

Video Annotation

Depending to the tasks we want to solve, we will need to prepare data with the corresponding type of annotation

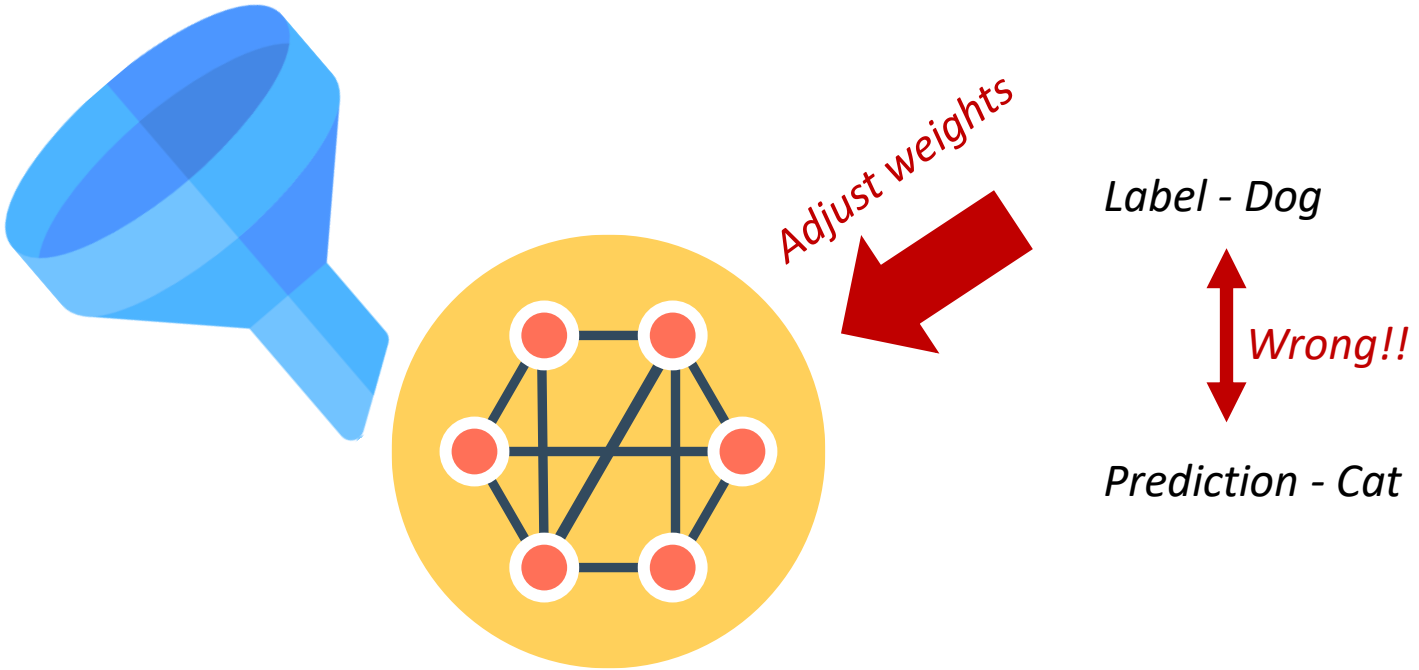


Label - Dog

Prediction - Cat

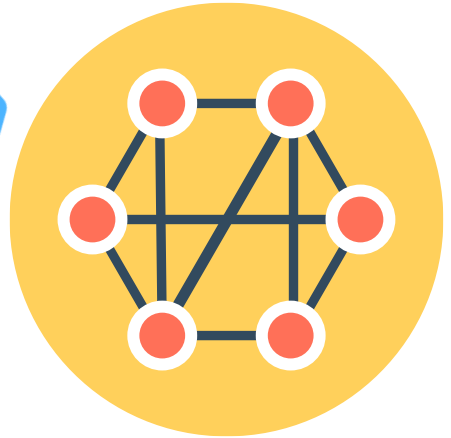
Deep convolutional network

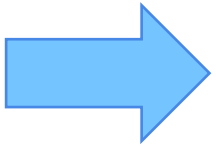
During the training, it takes an image and makes a prediction about what it is "seeing." It uses the labels as supervision and compare with its predictions.



Deep convolutional network

The neural network checks the accuracy of its predictions in a series of iterations and adjust its parameters or weights in the network when it makes a mistake.





Label - Dog

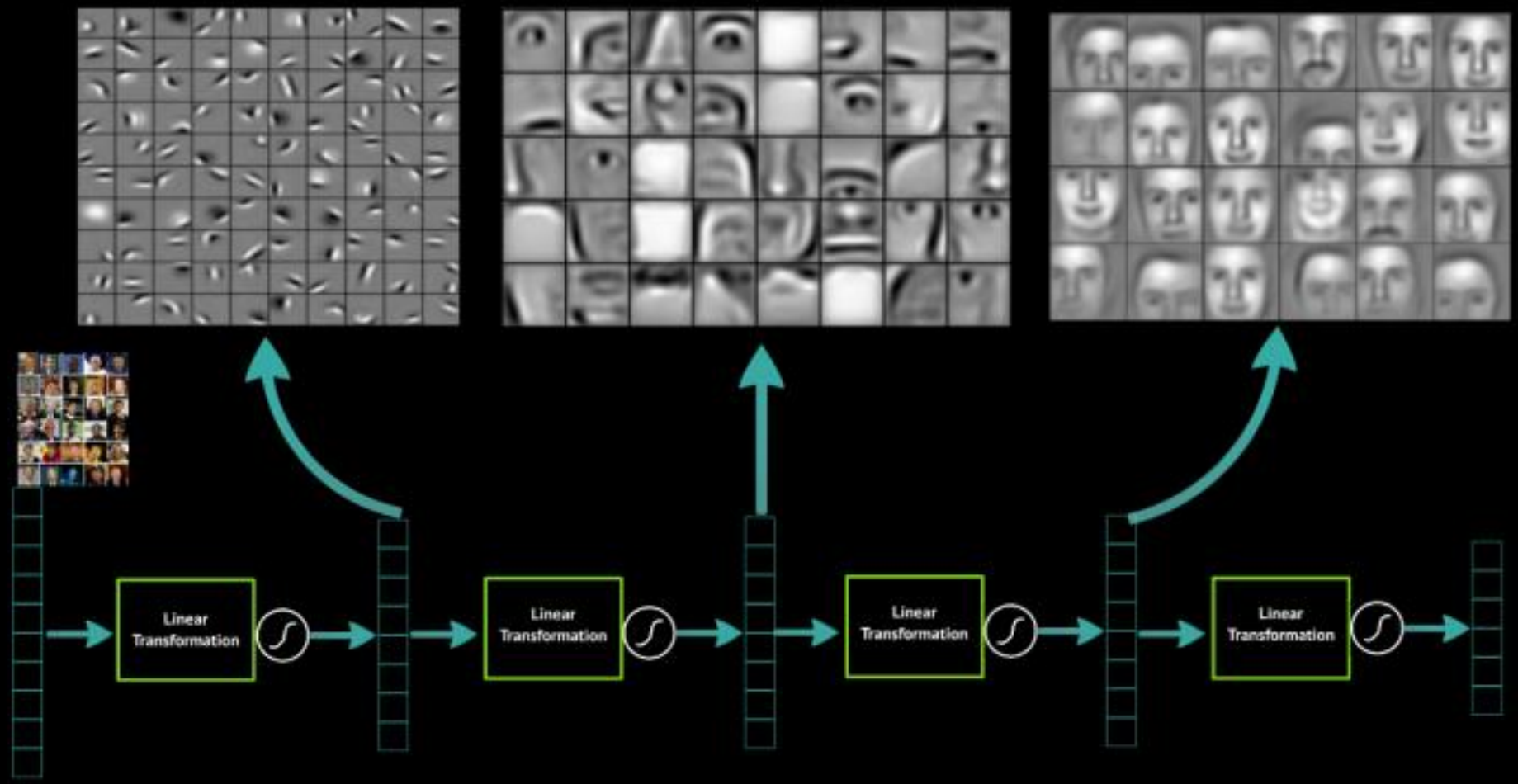
Correct!!

Prediction - Dog

Deep convolutional network

This process continues until the predictions start to come true. It can then recognize the objects accurately.

Deep Learning learns layers of features



Much like a human making out an image at a distance, a deep convolutional network first discerns hard edges and simple shapes in the shallower layers, then it starts to recognize parts and attributes in the deeper layers of the network

Video Inpainting

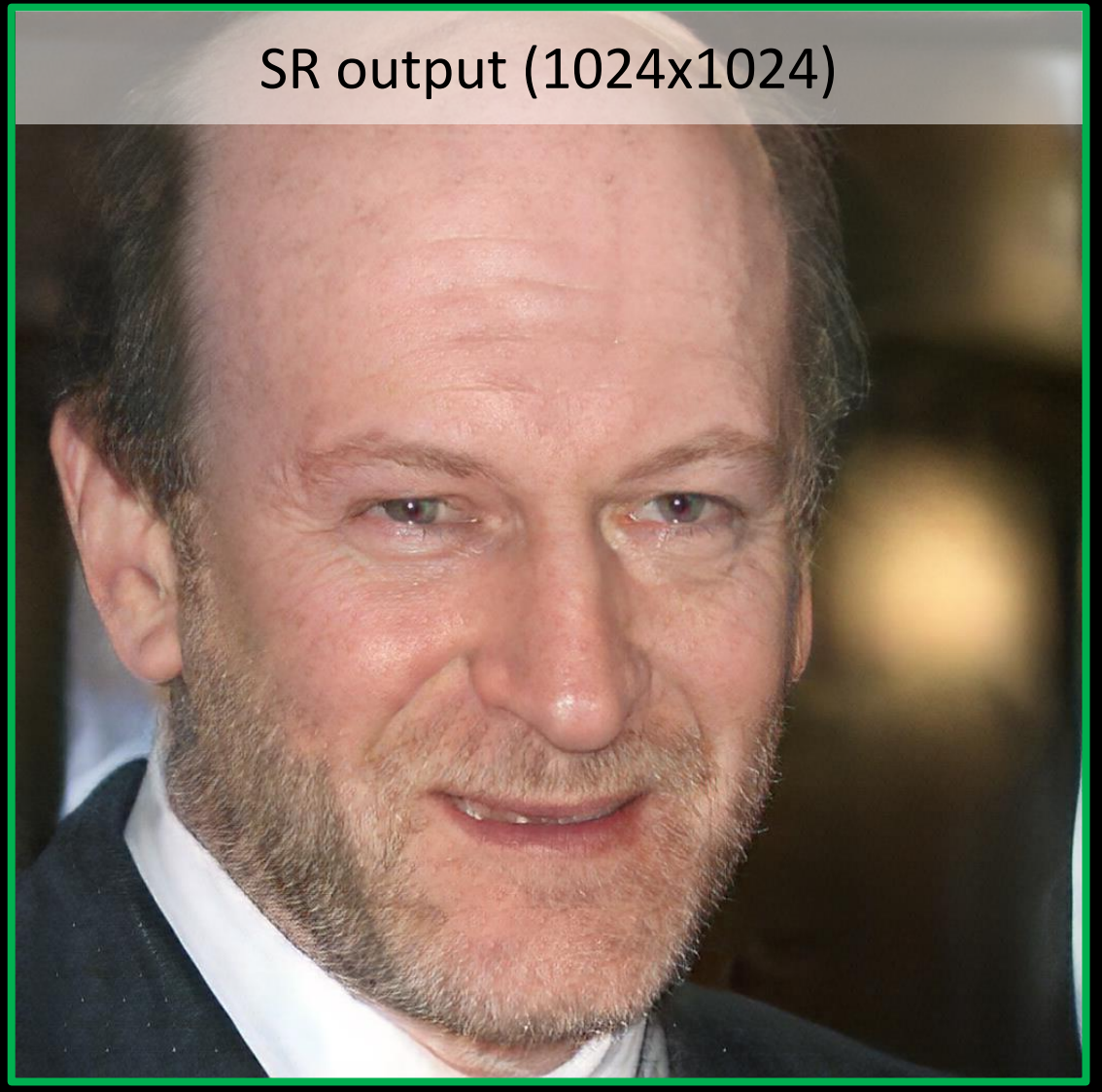
Blurry and
low-resolution

Enhanced
image

Super-resolution

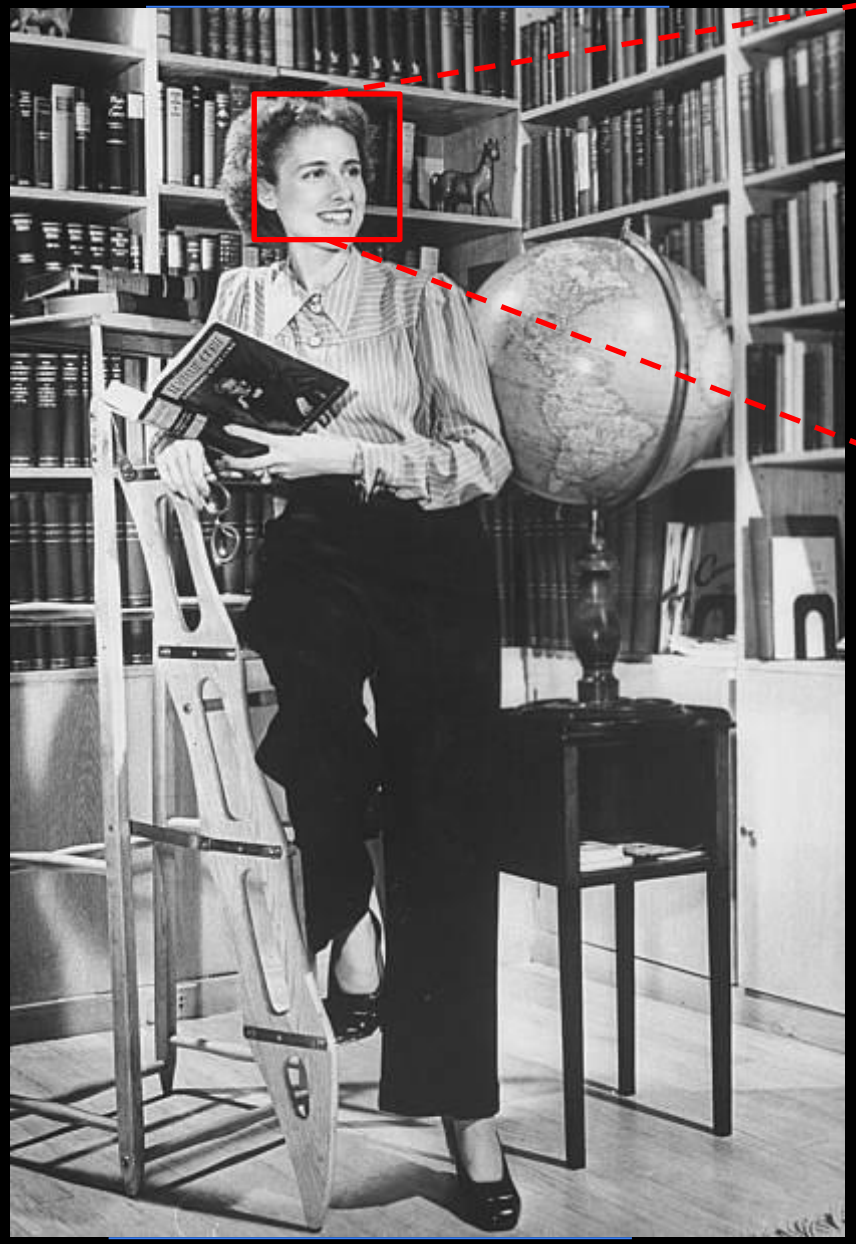
LR input (heavily compressed)

SR output (1024x1024)



LR input (heavily compressed)

SR output (1024x1024)



LR input (74x74)

SR output (1024x1024)


LR input (134x134)


SR output (1024x1024)


Low-Resolution Input
Output of BasicVSR++
Of course, we can do it for videos as well

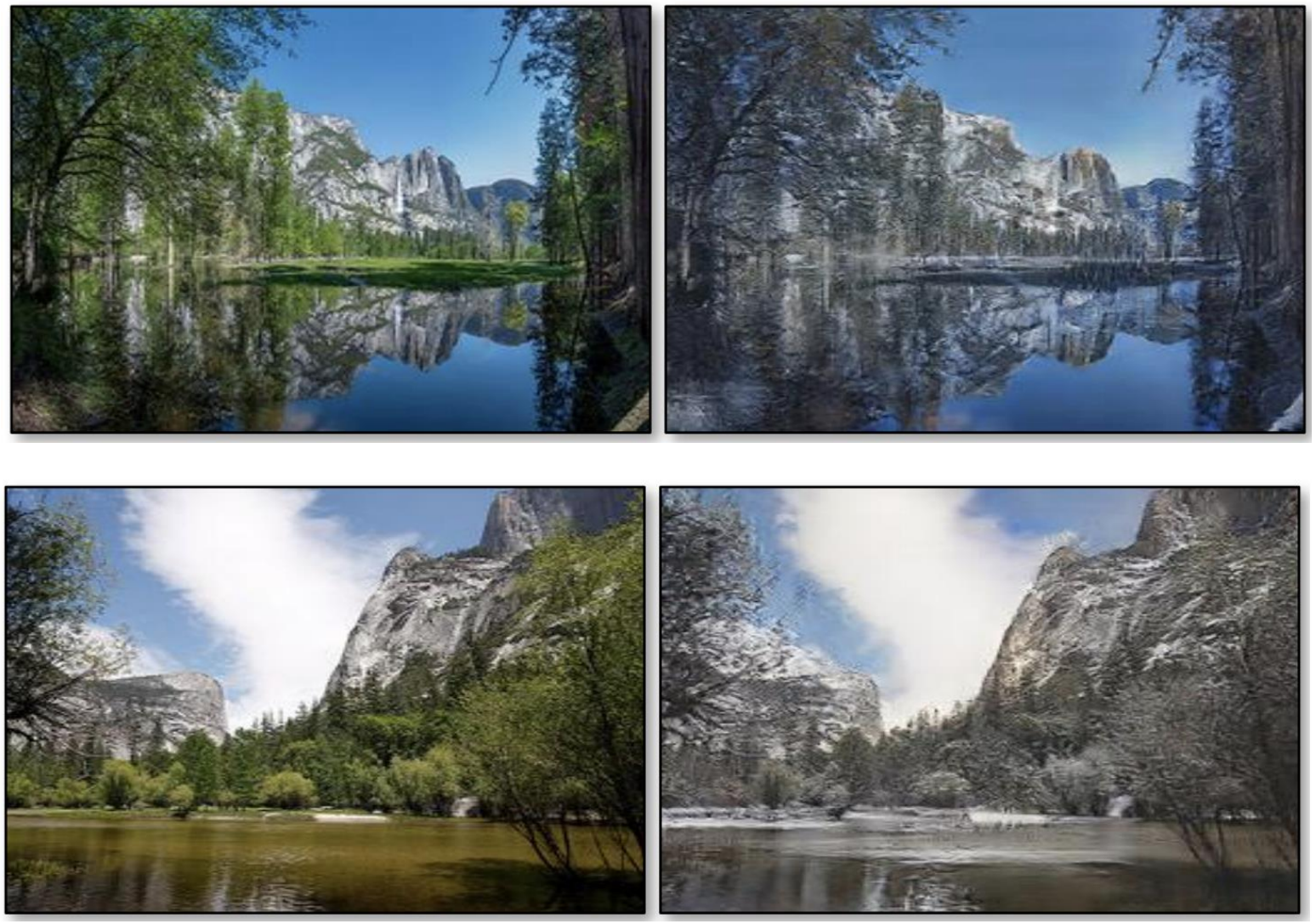
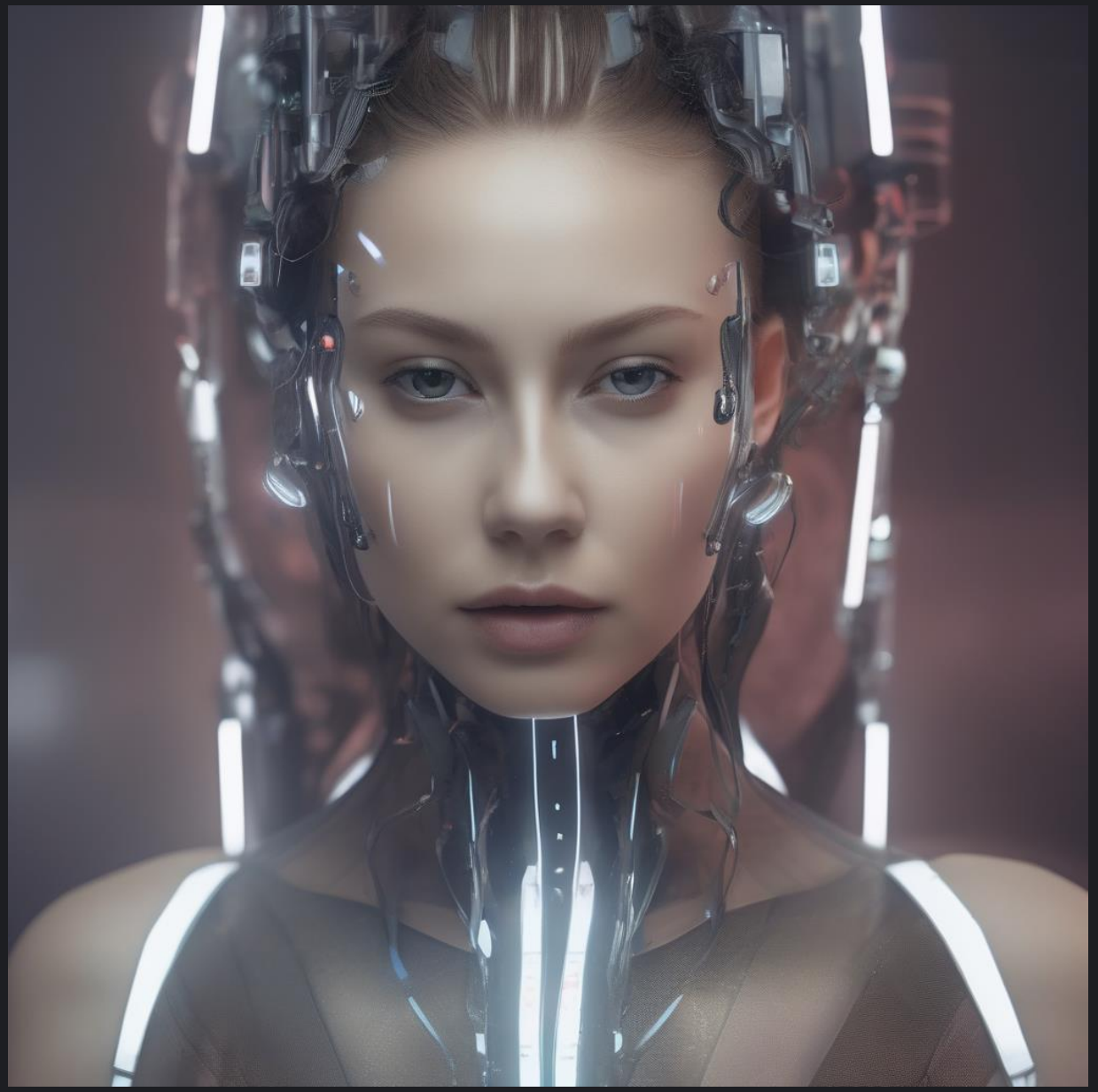
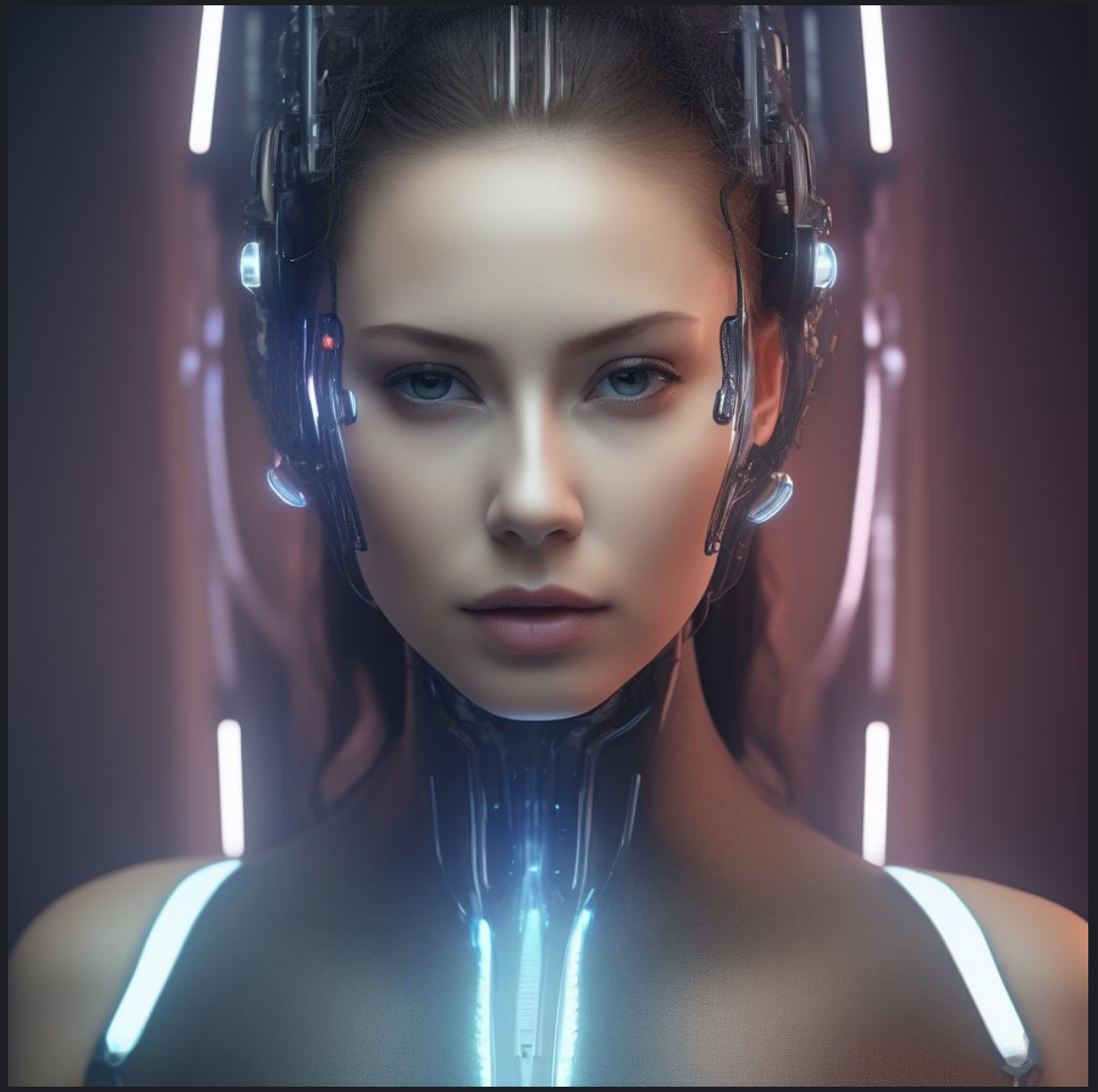


Image-to-image translation

Not just face images, now we can generate synthetic images of different kinds, like cats.



SDXL



SDXL + FreeU

SDXL

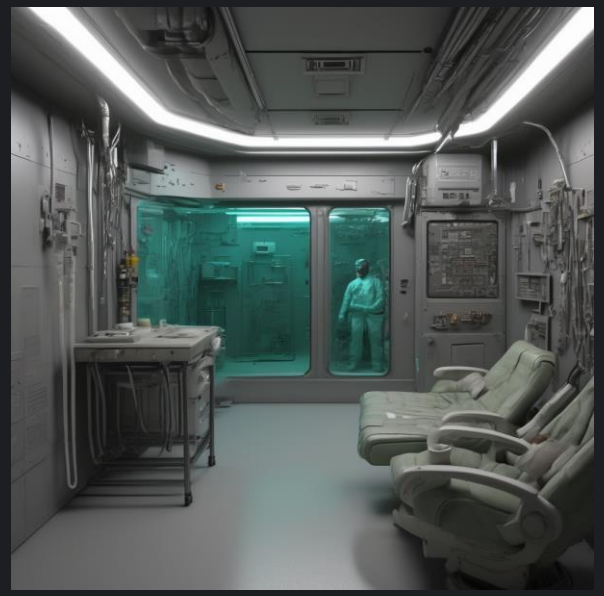
SDXL + FreeU

SDXL

SDXL + FreeU

SDXL

SDXL + FreeU



A cat eating food out of a bowl, in style of Van Gogh

A boat sailing leisurely along the Seine River with the Eiffel Tower in background by Vincent van Gogh

Cinematic shot of Van Gogh's selfie, Van Gogh style

Vincent van Gogh is painting in the room

A corgi's head depicted as an explosion of a nebula, high quality

A robot dj is playing the turntable, in heavy raining futuristic tokyo rooftop cyberpunk night, sci-fi, fantasy

A fantasy landscape, trending on artstation, 4k, high resolution

Iron Man flying in the sky

Yoda playing guitar on the stage

A space shuttle launching into orbit, with flames and smoke billowing out from the engines

A jellyfish floating through the ocean, with biolum inescent tentacles

A panda drinking coffee in a cafe in Paris

2023 arXiv LAVIE: High-Quality Video Generation with Cascaded Latent Diffusion Models

Style Transfer

Input Video

Reference Style

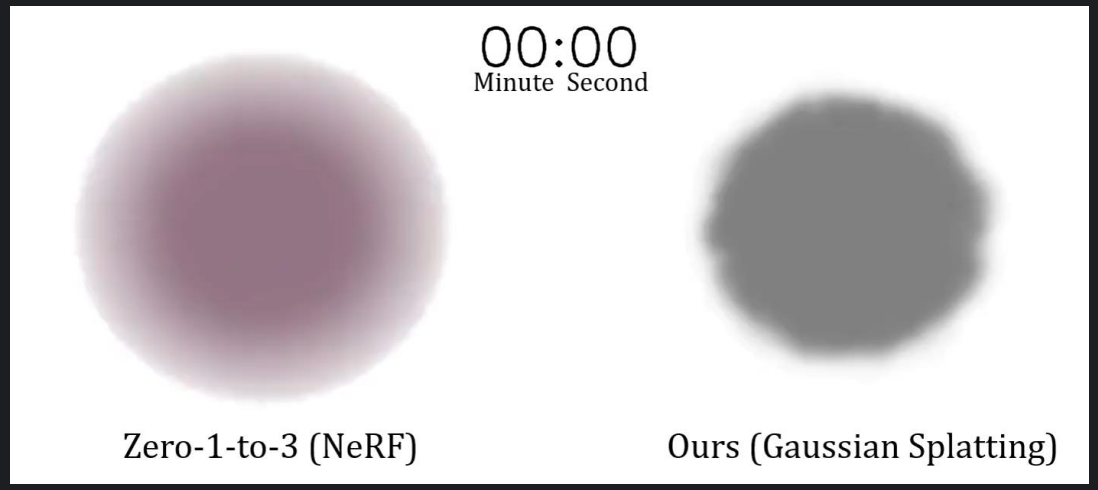
Our Toonification Result

Hair Editing

Age Editing

Age Editing

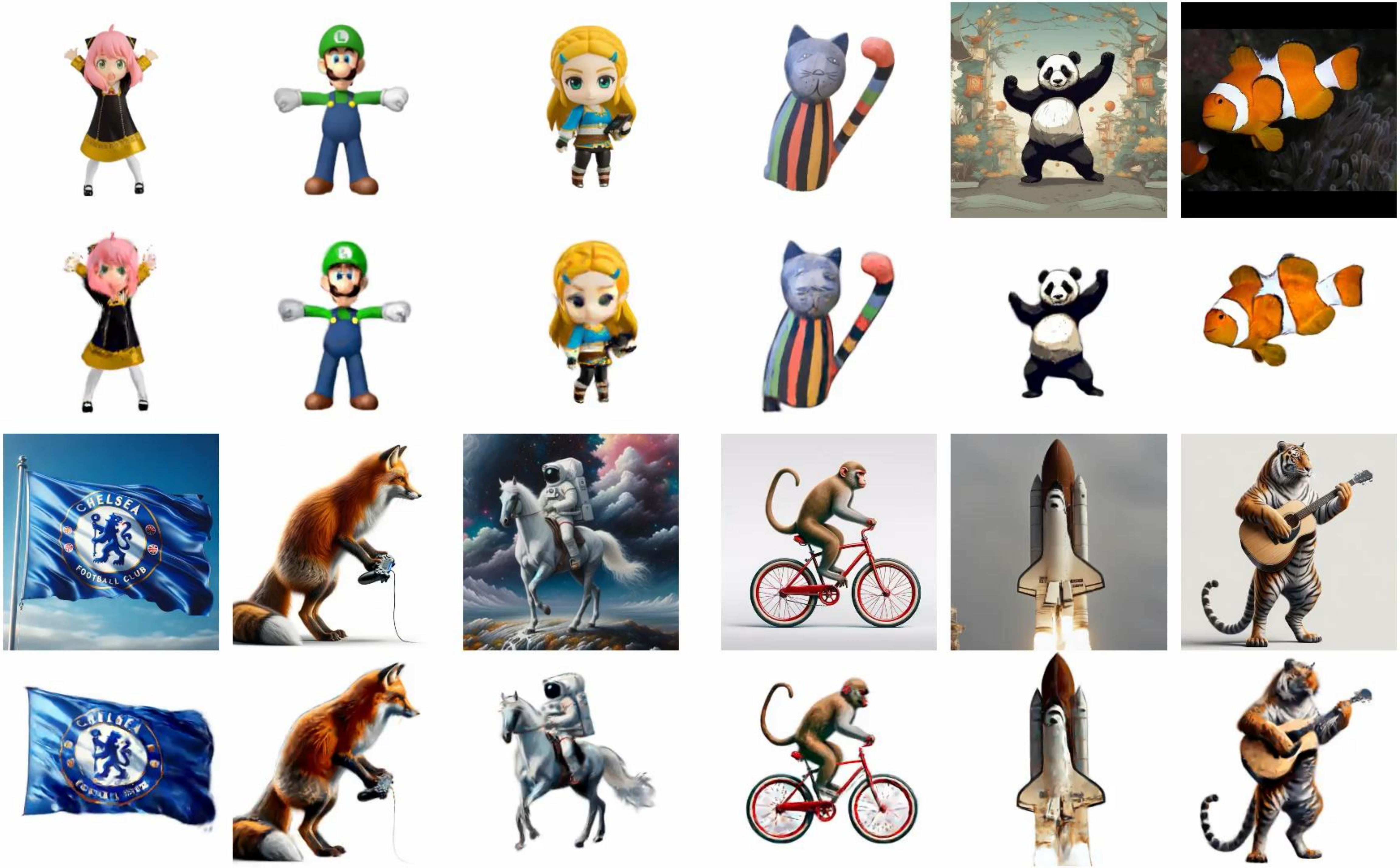
Video Style Transfer



DreamGaussian

DreamGaussian produces high-quality textured meshes in just 2 minutes from a single-view image

10 times acceleration compared to existing methods


CHELSEA
FOOTBALL CLUB

LLaVA OneVision

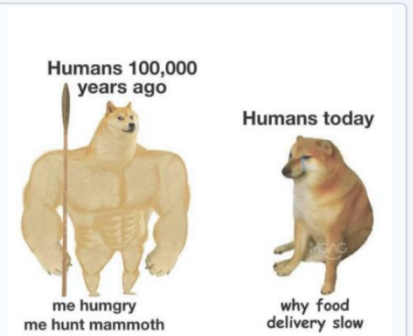
Code | Checkpoints | Data | LLaVA-NeXT Series Blog | OneVision-72B Demo

Note: `AnyResMax9` strategy is only applied for single image-text interactions (best for high-resolution images). Multi-image and multi-turn image-text conversations will use default tokenization (729 tokens per image).

llava-onevision-qwen2-7b-ov

Parameters

Chatbot



Why this image funny?

The image is a humorous meme that plays on the contrast between the physical abilities and lifestyle of humans in the past and the modern era. On the left, it shows a muscular, prehistoric human with the text "Humans 100,000 years ago" and "me humgry me hunt mammoth," suggesting that ancient humans were physically strong and had to hunt for their food. On the right, it shows a less muscular, modern human with the text "Humans today" and "why food delivery slow," implying that contemporary humans are less physically active and reliant on food delivery services, which can sometimes be slow. The humor comes from the exaggerated portrayal of the difference between the two time periods and the relatable frustration of waiting for food delivery in the present day.

Translate into Chinese

https://llava-onevision.lmms-lab.com/

Image Credits

OpenPose

MIT News,
How the Brain Distinguishes between Objects

Juan Hernandez,
Making AI Interpretable with Generative Adversarial Networks

Ethan Yanjia Li,
10 Papers You Should Read to Understand Image Classification in the Deep Learning Era

Outline

- About this course
- Computer vision background and a little history
- Fundamentals of machine learning
  - Why learning?
  - Statistical learning

Fundamentals of Machine Learning

Credit: Gilles Louppe , "Lecture 1: Fundamentals of machine learning”, 2019.

Outline

Set the fundamentals of machine learning

- Why learning?
- Statistical learning
  - Supervised learning
  - Empirical risk minimization (using polynomial regression as an example)
  - Underfitting and overfitting
  - Bias-variance dilemma
  - Deep double descent

What do you see? How do we do that?!

Exceptional generalization ability

Sheepdog or mop?

Exceptional discriminative ability

Chihuahua or muffin?

Exceptional discriminative ability

Why learning?

The automatic extraction of semantic information from raw signal is at the core of many applications, such as

- image recognition
- speech processing
- natural language processing
- robotic control
- ... and many others.

How can we write a computer program that implements that?

Why learning?

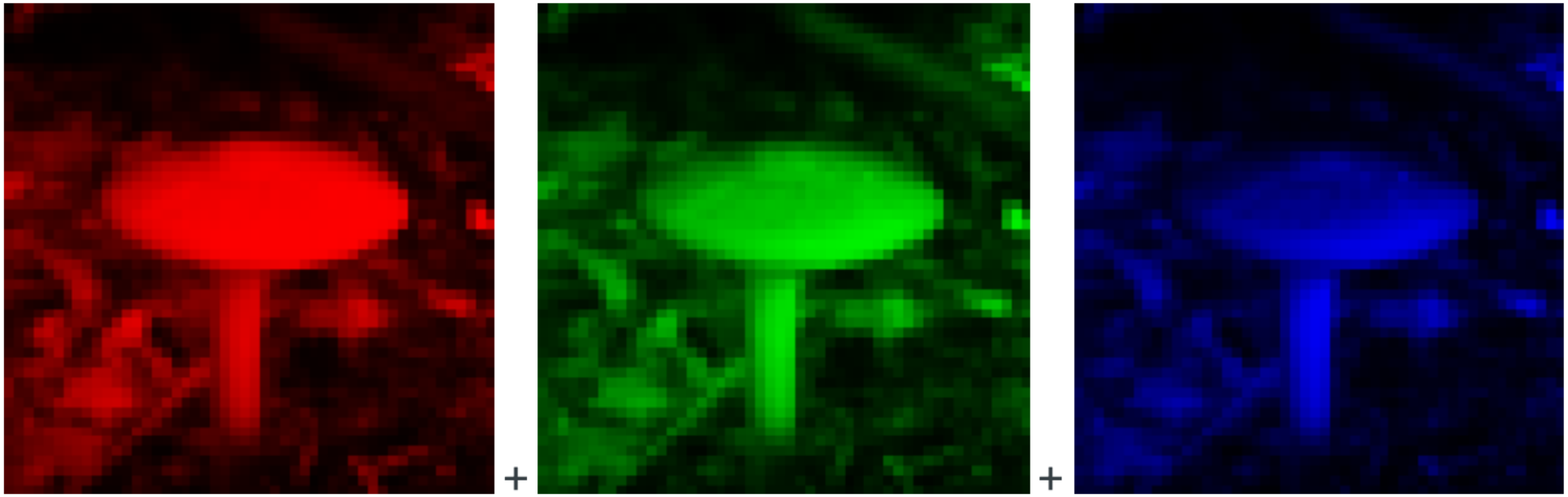
The (human) brain is so good at interpreting visual information that the gap between raw data and its semantic interpretation is difficult to assess intuitively:

This is a mushroom.

Why learning?

This is a mushroom.

Why learning?



This is a mushroom.

Why learning?

```
array([[[0.03921569, 0.03529412, 0.02352941, 1.        ],
        [0.2509804 , 0.1882353 , 0.20392157, 1.        ],
        [0.4117647 , 0.34117648, 0.37254903, 1.        ],
        ...,
        [0.20392157, 0.23529412, 0.17254902, 1.        ],
        [0.16470589, 0.18039216, 0.12156863, 1.        ],
        [0.18039216, 0.18039216, 0.14117648, 1.        ]],

       [[0.1254902 , 0.11372549, 0.09411765, 1.        ],
        [0.2901961 , 0.2509804 , 0.24705882, 1.        ],
        [0.21176471, 0.2       , 0.20392157, 1.        ],
        ...,
        [0.1764706 , 0.24705882, 0.12156863, 1.        ],
        [0.10980392, 0.15686275, 0.07843138, 1.        ],
        [0.16470589, 0.20784314, 0.11764706, 1.        ]],

       [[0.14117648, 0.12941177, 0.10980392, 1.        ],
        [0.21176471, 0.1882353 , 0.16862746, 1.        ],
        [0.14117648, 0.13725491, 0.12941177, 1.        ],
        ...,
        [0.10980392, 0.15686275, 0.08627451, 1.        ],
        [0.0627451 , 0.08235294, 0.05098039, 1.        ],
        [0.14117648, 0.2       , 0.09803922, 1.        ]],

       ...,
```

This is a mushroom.???

Why learning?

Extracting semantic information requires models of high complexity, which cannot be designed by hand.

However, one can write a program that learns the task of extracting semantic information.

Techniques used in practice consist of:

- defining a parametric model with high capacity,
- optimizing its parameters, by "making it work" on the training data.

Supervised learning

Consider an unknown joint probability distribution $P(X, Y)$.

Assume training data $(\mathbf{x}_i, y_i) \sim P\ (X, Y)$,
with $\boldsymbol{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}, i = 1, \ldots, N$.

- In most cases,
  - $\mathbf{x}_i$ is a p-dimensional vector of features or descriptors,
  - y_i is a scalar (e.g., a category or a real value).
- The training data is generated i.i.d.
- The training data can be of any finite size N.
- In general, we do not have any prior information about $P(X, Y)$.

Notes:
*i.i.d. (***independent and identically distributed***) -*
each random variable has the same probability distribution as the others and all are mutually independent

This assumption is often made for training datasets to imply that all samples stem from the same generative process and that the generative process is assumed to have no memory of past generated samples.

Supervised learning

Inference

Supervised learning is usually concerned with the two following inference problems:

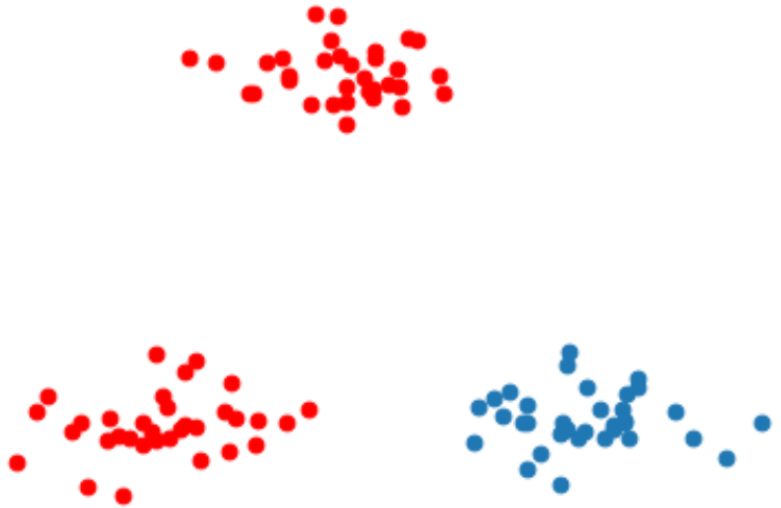
Classification: Given $(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y} = \mathbb{R}^p \times \{1, \dots, C\}$, for $i = 1, \dots, N$, we want to estimate for any new $\mathbf{x}$,

$$\underset{y}{\operatorname{argmax}}\, P(Y = y \mid X = \mathbf{x}) .$$

Regression: Given$(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y} = \mathbb{R}^p \times \mathbb{R}$, for $i = 1, \dots, N$, we want to estimate for any new $\mathbf{x}$,

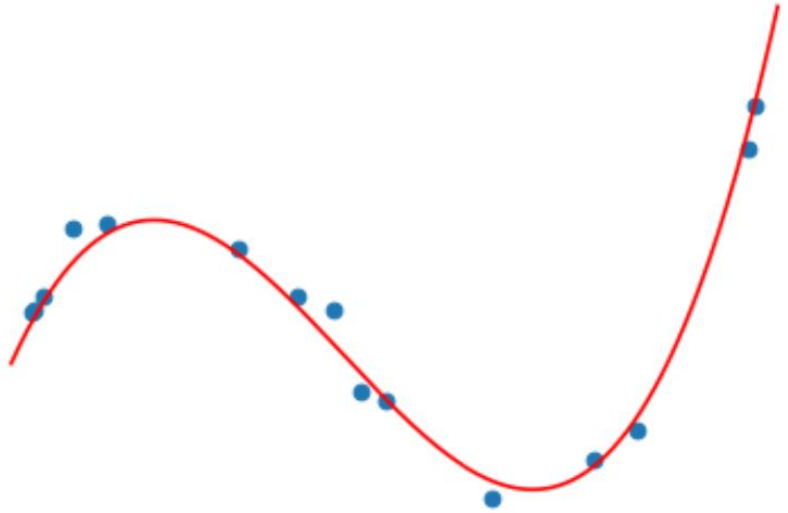
$$\mathbb{E}\left[Y \mid X = \mathbf{x}\right].$$

Supervised learning



Classification consists in identifying
a decision boundary between objects of distinct classes.

Supervised learning



Regression aims at estimating relationships among (usually continuous) variables.

Empirical risk minimization

Our objective is to train a good model.

How to define whether a model is good or bad?

We do not know the how well an algorithm will work in practice (the expected risk) because we do not know the true distribution of the data the algorithm will work on.

What we can do is to measure the model's performance of a known set of training data (the empirical risk / training error).

Empirical risk minimization

Consider a function $f : \mathcal{X} \to \mathcal{Y}$ produced by some learning algorithm. The predictions of this function can be evaluated through a loss

$$\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R},$$

such that $\ell(y, f(\mathbf{x})) \geq 0$ measures how close the prediction $f(\boldsymbol{x})$ from y is.

Examples of loss functions

Classification: $\ell(y, f(\mathbf{x})) = \mathbf{1}_{y \neq f(\mathbf{x})}$

Regression: $\ell(y, f(\mathbf{x})) = \big(y - f(\mathbf{x})\big)^2$

Empirical risk minimization

Let $\mathcal{F}$ denote the hypothesis space, i.e. the set of all functions f that can be produced by the chosen learning algorithm.

We are looking for a function $f \in \mathcal{F}$ with a small expected risk (or generalization error)

$$R(f) = \mathbb{E}_{(\mathbf{x},y)\sim P(X,Y)}\big[\ell\big(y, f(\mathbf{x})\big)\big].$$

This means that for a given data generating distribution $P(X, Y)$ and for a given hypothesis space $\mathcal{F}$, the optimal model is

$$f_* = \underset{f\in\mathcal{F}}{\operatorname{argmin}}\, R(f).$$

Empirical risk minimization

Unfortunately, since $P(X, Y)$ is unknown, the expected risk cannot be evaluated and the optimal model cannot be determined.

However, if we have i.i.d. training data $\mathbf{d} = \{(\mathbf{x}_i, y_i) \mid i = 1, \ldots, N\}$, we can compute an estimate, the empirical risk (or training error)

$$\hat{R}(f, \mathbf{d}) = \frac{1}{N} \sum_{(\mathbf{x}_i, y_i) \in \mathbf{d}} \ell(y_i, f(\mathbf{x}_i)) .$$

This estimate can be used for finding an approximation of f_*. This results into the empirical risk minimization principle:

$$f_*^{\mathbf{d}} = \operatorname*{argmin}_{f \in \mathcal{F}} \hat{R}(f, \mathbf{d}) .$$

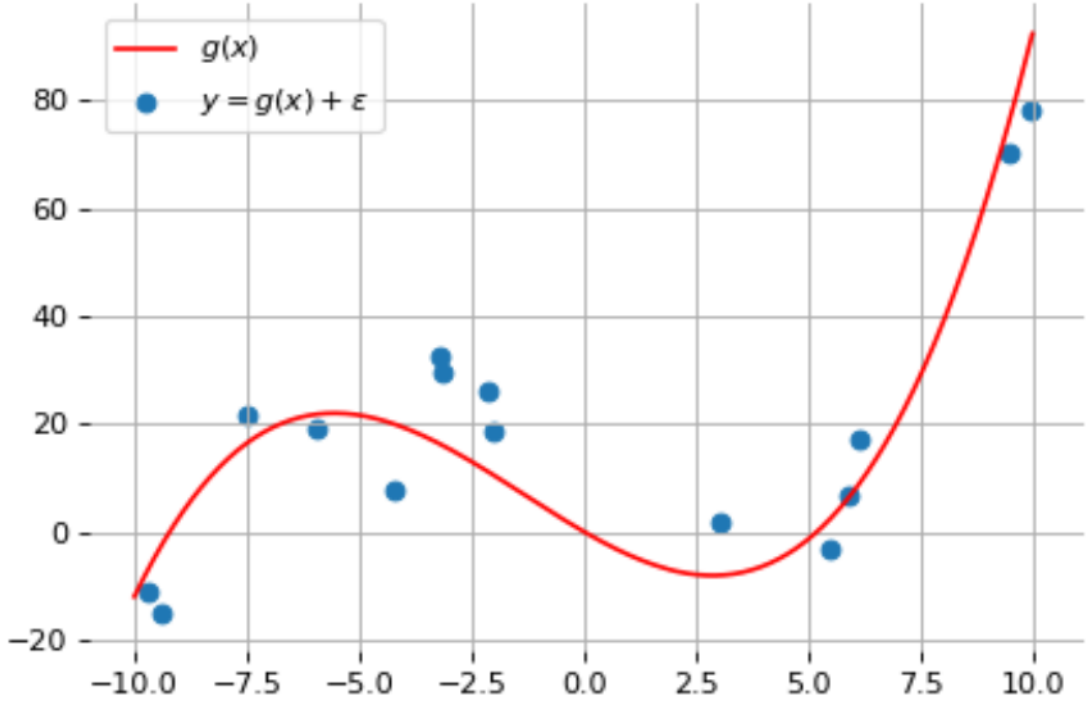
Empirical risk minimization

Most machine learning algorithms, including neural networks, implement empirical risk minimization.

Under regularity assumptions, empirical risk minimizers converge:

$$\lim_{N\to\infty} f_*^{\mathbf{d}} = f_*$$

Polynomial regression



Consider the joint probability distribution $P(X, Y)$ induced by the data generating process

$$(x, y) \sim P(X, Y) \Leftrightarrow x \sim U[-10; 10], \varepsilon \sim \mathcal{N}(0, \sigma^2), y = g(x) + \varepsilon$$

where $x \in \mathbb{R}, y \in \mathbb{R}$ and g is an unknown polynomial of degree 3.

Polynomial regression

Our goal is to find a function f that makes good predictions on average over $P(X, Y)$.

Consider the hypothesis space $f \in \mathcal{F}$ of polynomials of degree 3 defined through their parameters $\mathbf{w} \in \mathbb{R}^4$ such that

$$\hat{y} \triangleq f(x; \mathbf{w}) = \sum_{d=0}^{3} w_d x^d$$

Polynomial regression

For this regression problem, we use the squared error loss

$$\ell(y, f(x, \mathbf{w})) = \big(y - f(x, \mathbf{w})\big)^2$$

to measure how wrong the predictions are.

Therefore, our goal is to find the best value $\mathbf{w}_*$ such

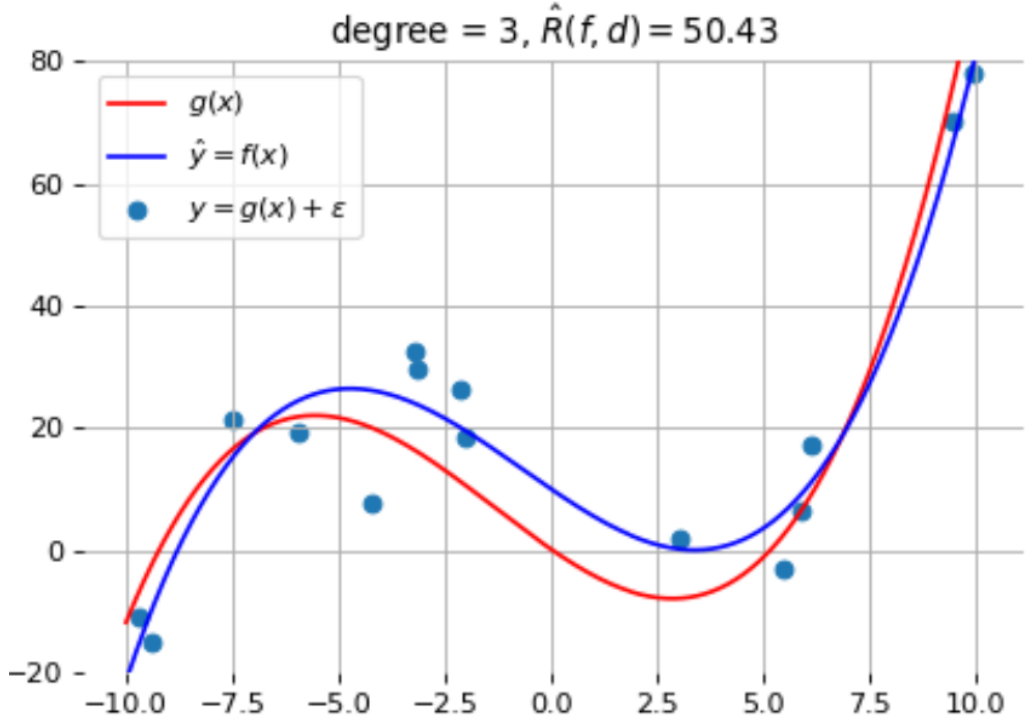
$$\begin{aligned}\mathbf{w}_* &= \operatorname*{argmin}_{\mathbf{w}} R(\mathbf{w}) \\ &= \operatorname*{argmin}_{\mathbf{w}} \mathbb{E}_{(\mathbf{x},y)\sim P(X,Y)} \Big[\big(y - f(x, \mathbf{w})\big)^2\Big]\end{aligned}$$

Polynomial regression

Given a large enough training set $\mathbf{d} = \{(x_i, y_i) \mid i = 1, \ldots, N\}$, the empirical risk minimization principle tells us that a good estimate $\mathbf{w}_*^{\mathbf{d}}$ of $\mathbf{w}_*$ can be found by minimizing the empirical risk:

$$
\begin{aligned}
\mathbf{w}_*^{\mathbf{d}} &= \arg\min_{\mathbf{w}} \hat{R}(\mathbf{w}, \mathbf{d}) \\
&= \arg\min_{\mathbf{w}} \frac{1}{N} \sum_{(x_i, y_i) \in \mathbf{d}} (y_i - f(x_i; \mathbf{w}))^2 \\
&= \arg\min_{\mathbf{w}} \frac{1}{N} \sum_{(x_i, y_i) \in \mathbf{d}} (y_i - \sum_{d=0}^{3} w_d x_i^d)^2 \\
&= \arg\min_{\mathbf{w}} \frac{1}{N} \left\| \underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \ldots \\ y_N \end{pmatrix}}_{\mathbf{y}} - \underbrace{\begin{pmatrix} x_1^0 \ldots x_1^3 \\ x_2^0 \ldots x_2^3 \\ \ldots \\ x_N^0 \ldots x_N^3 \end{pmatrix}}_{\mathbf{X}} \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{pmatrix} \right\|^2
\end{aligned}
$$

Polynomial regression



This is ordinary least squares regression, for which the solution is known analytically:

$$\mathbf{w}_*^{\mathbf{d}} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$$

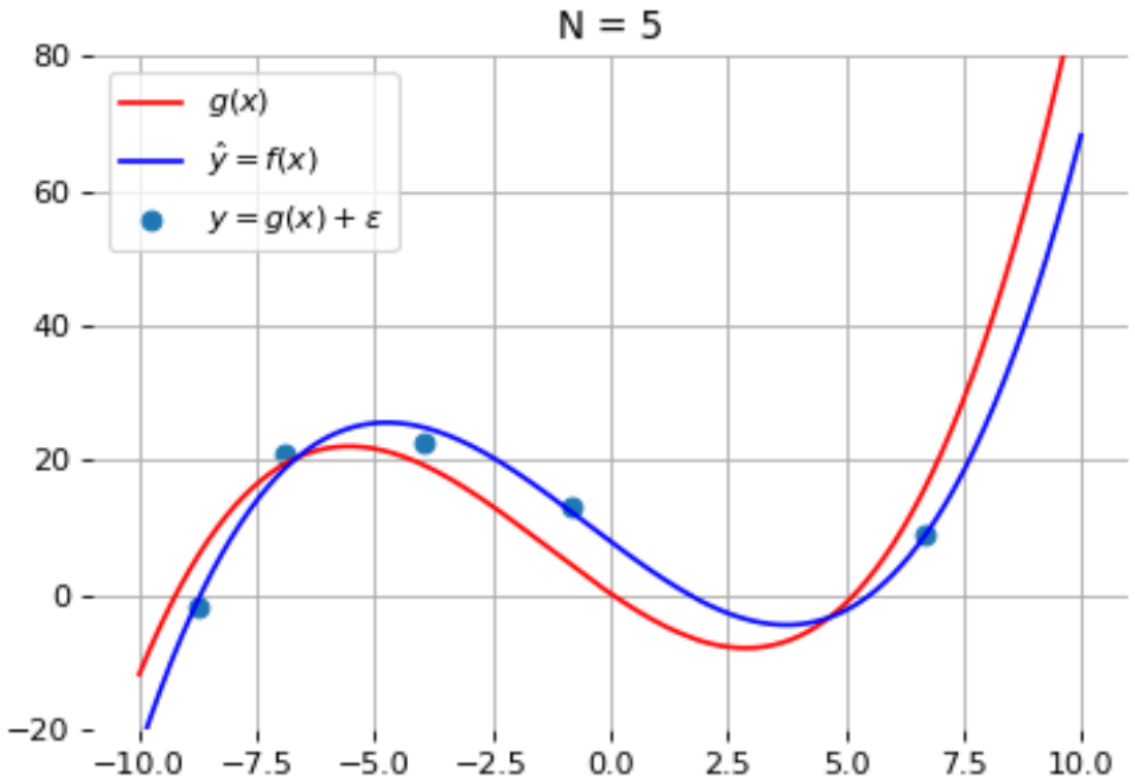
The expected risk minimizer $\mathbf{w}_*$ within our hypothesis space is g itself.

Therefore, on this toy problem, we can verify that

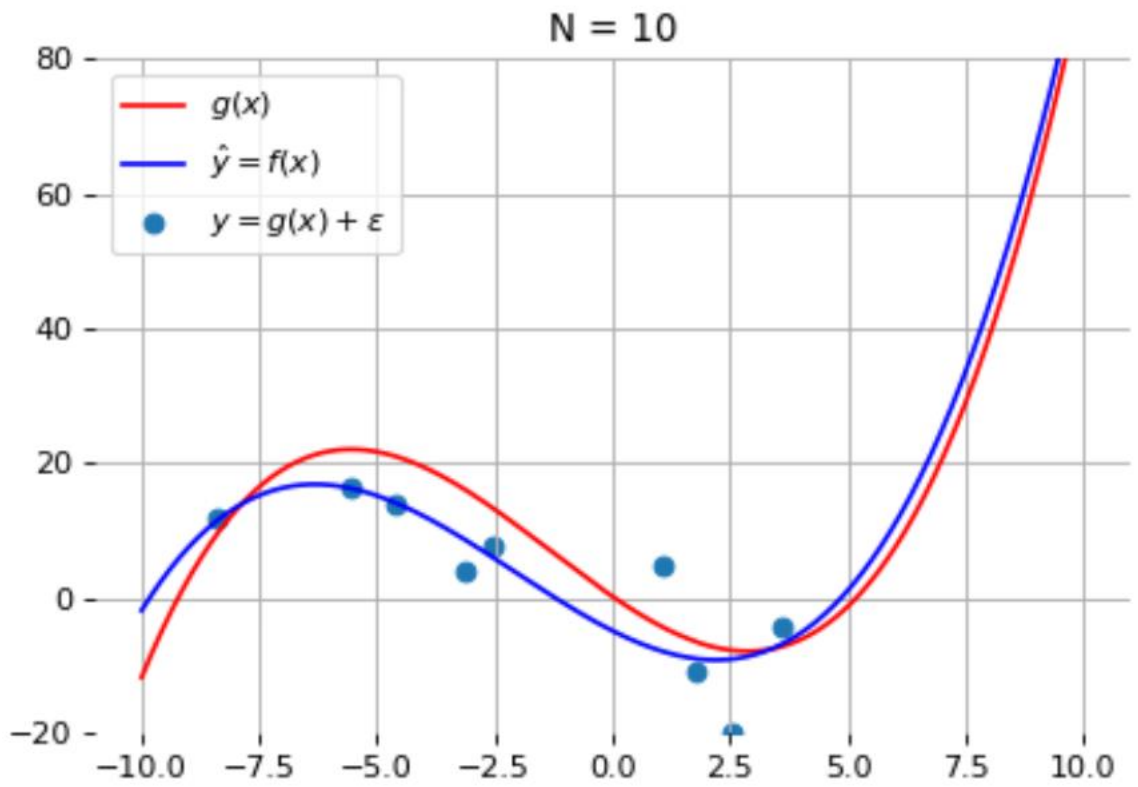
$f\left(x; \mathbf{w}_*^{\mathbf{d}}\right) \to f\left(x; \mathbf{w}_*\right) = g(x)$ as $N \to \infty$

Polynomial regression

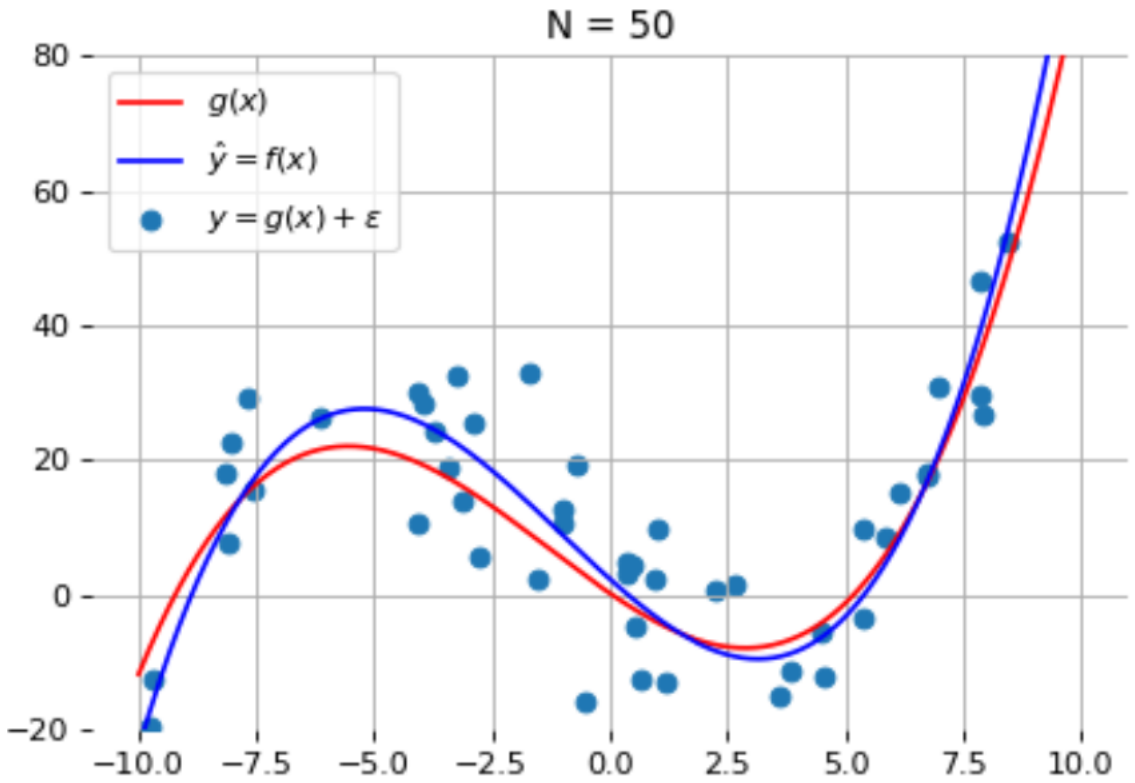
Let’s see what happen if we have different number of training samples



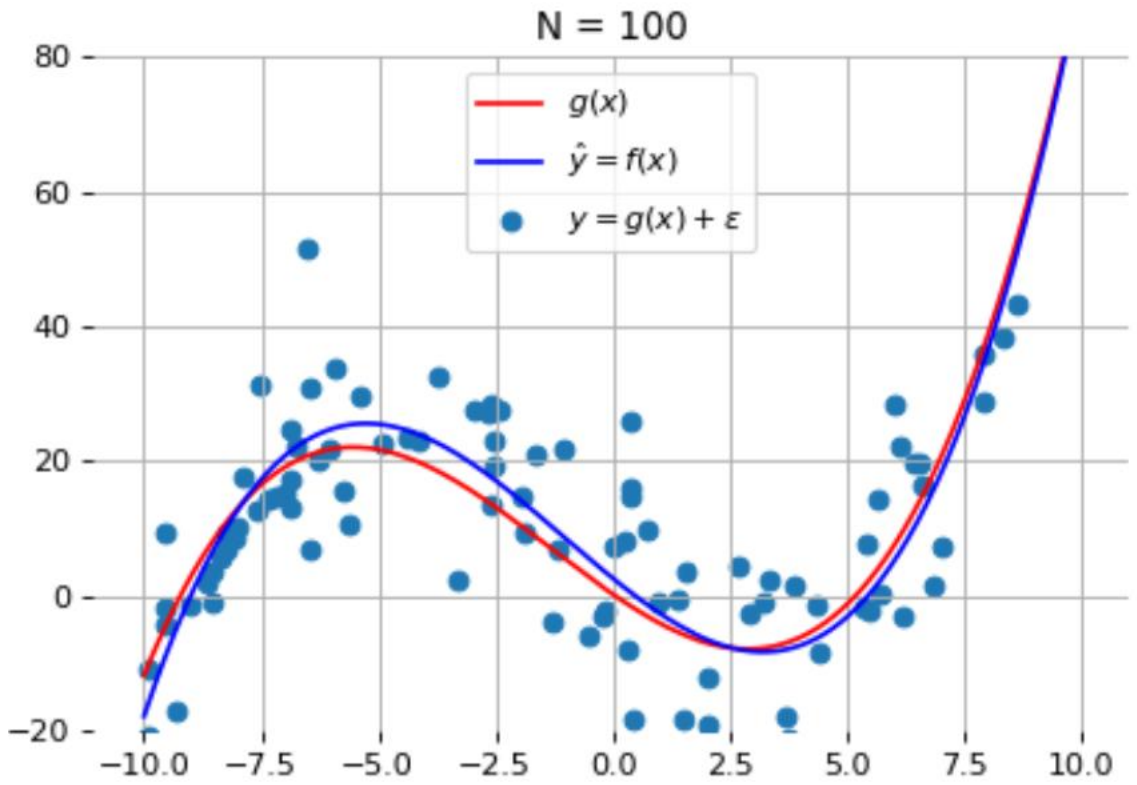
Polynomial regression



Polynomial regression



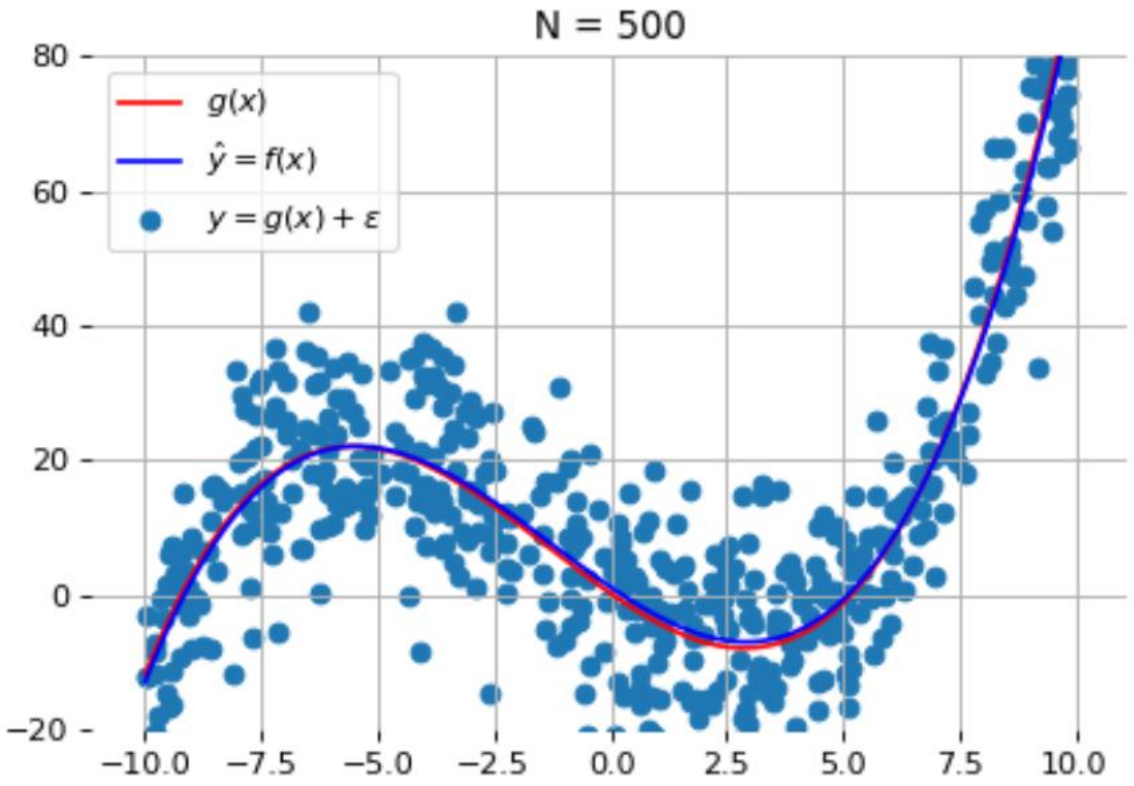
Polynomial regression



Polynomial regression

More training samples help us to find a better function!

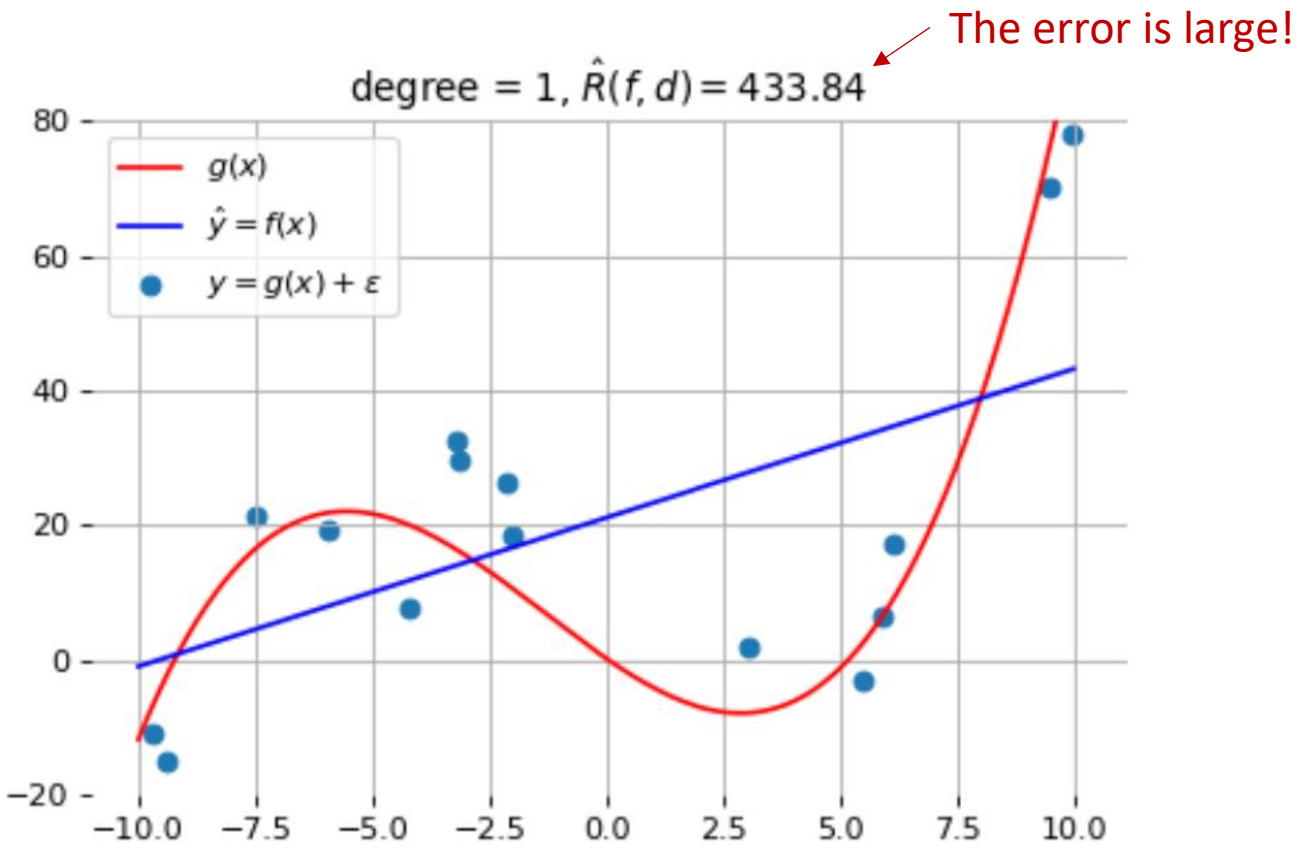
$f\left(x; \mathbf{w}_*^{\mathbf{d}}\right) \to f(x; \mathbf{w}_*) = g(x)$ as $N \to \infty$



Under-fitting and over-fitting

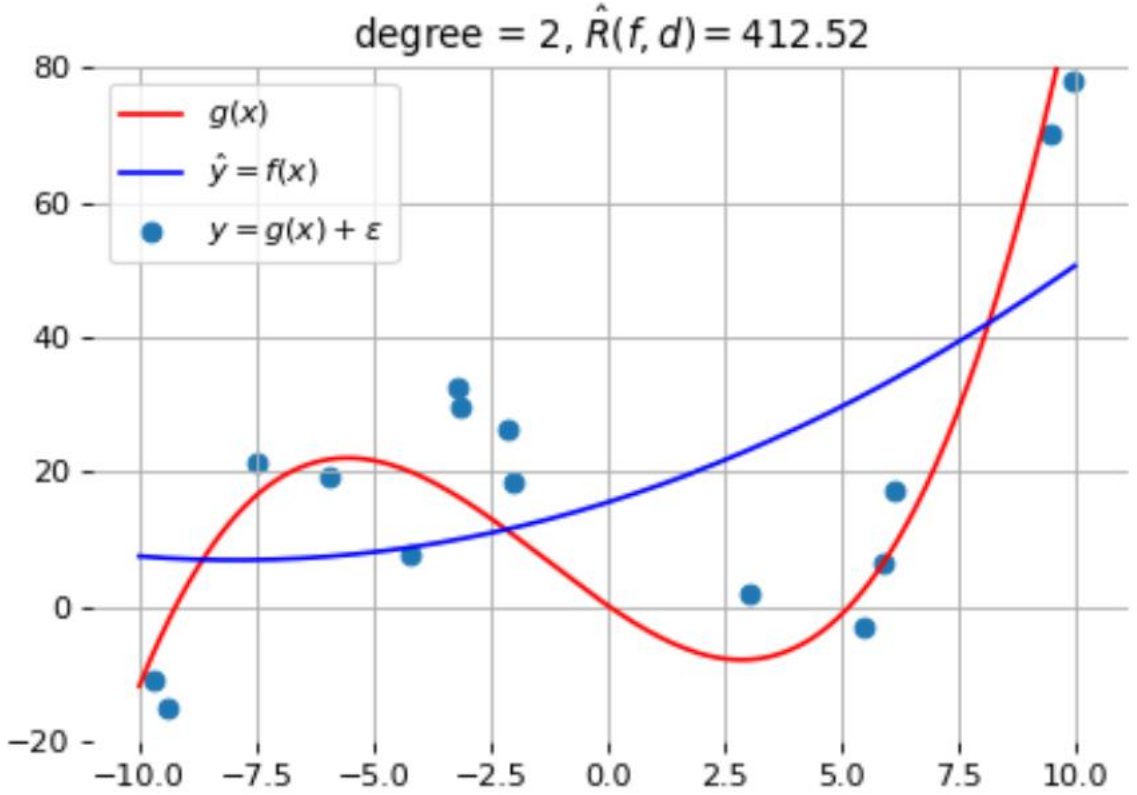
What if we consider a hypothesis space $\mathcal{F}$ in which candidate functions f are either too "simple" or too "complex" with respect to the true data generating process?

Under-fitting and over-fitting



$\mathcal{F}$ = polynomials of degree 1

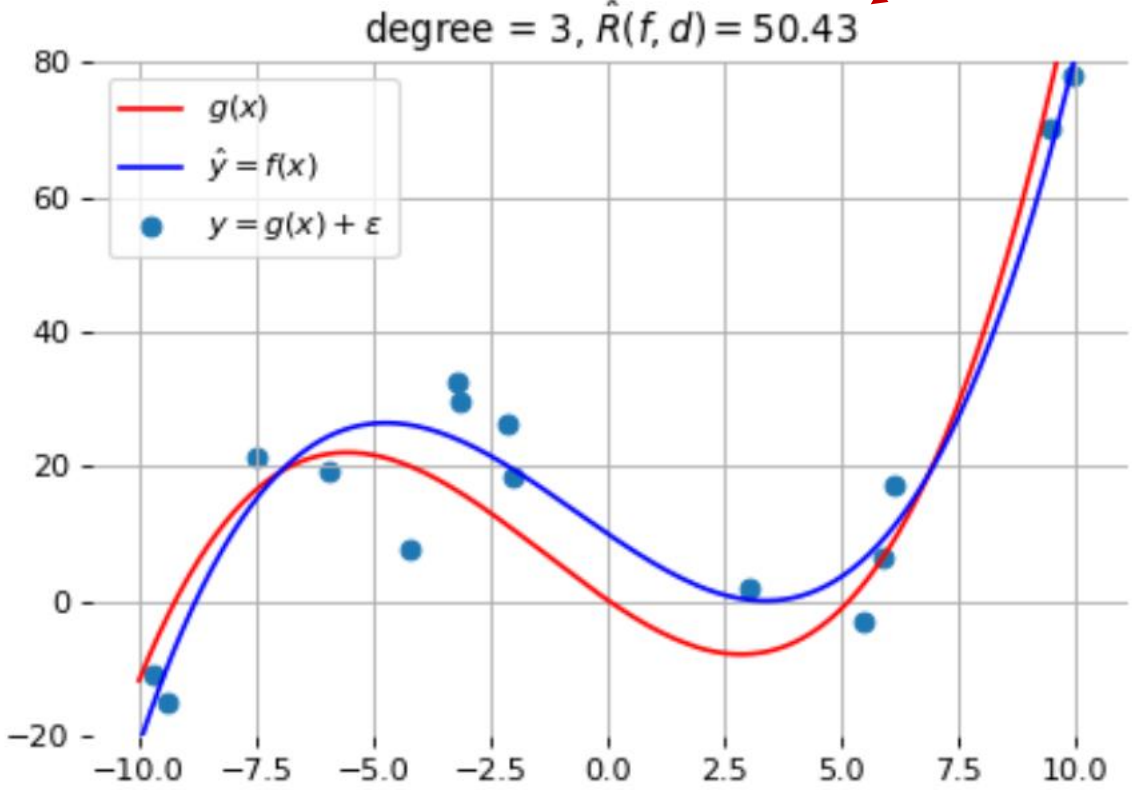
Under-fitting and over-fitting



$\mathcal{F}$ = polynomials of degree 2

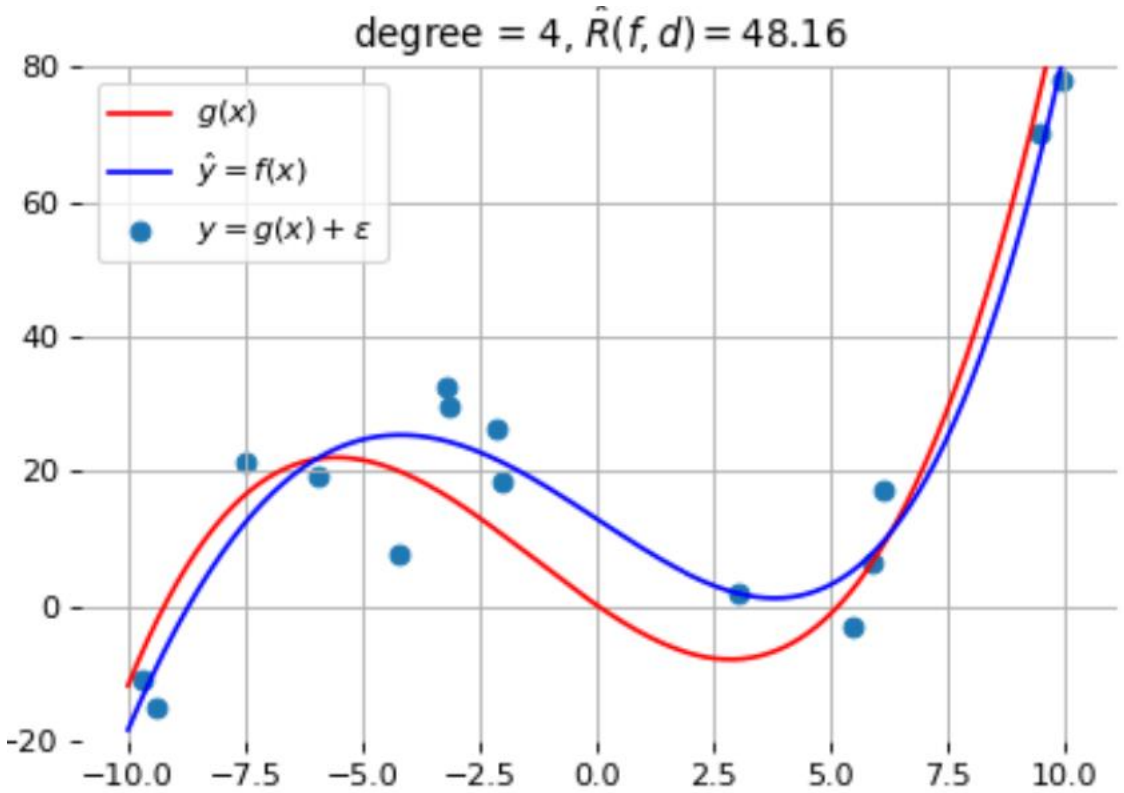
Under-fitting and over-fitting

The error is reducing



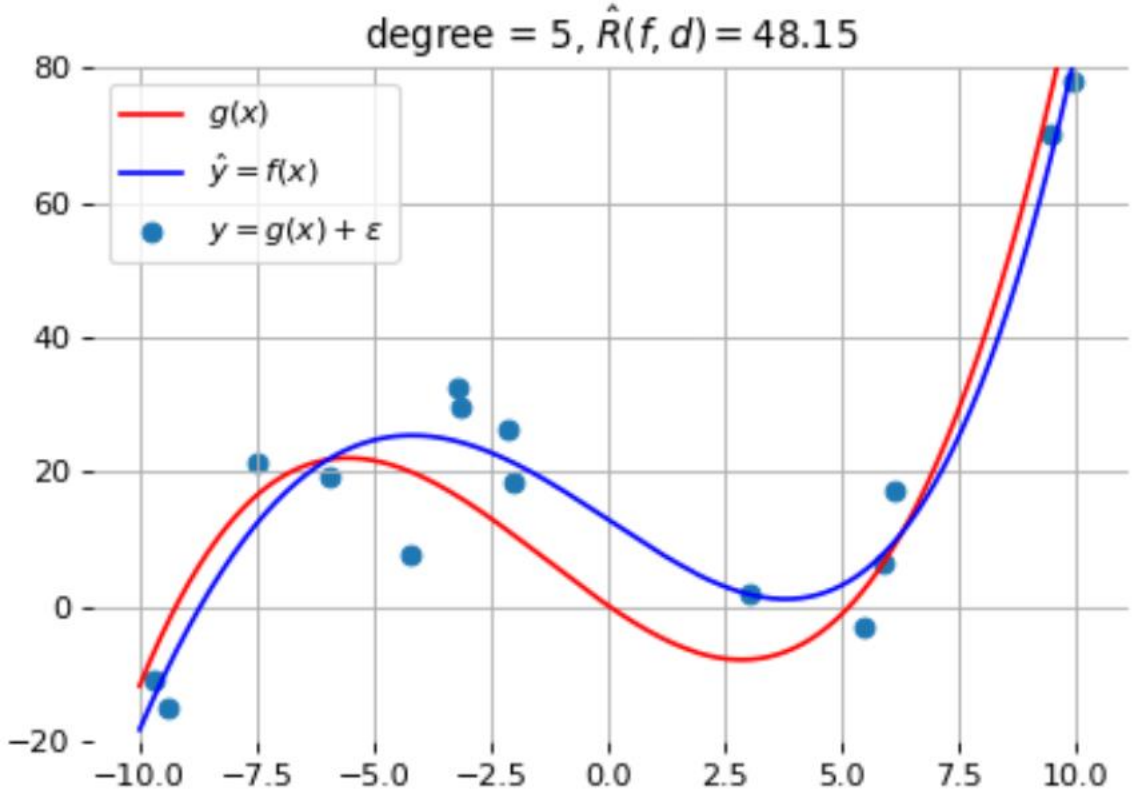
$\mathcal{F}$ = polynomials of degree 3

Under-fitting and over-fitting



$\mathcal{F}$ = polynomials of degree 4

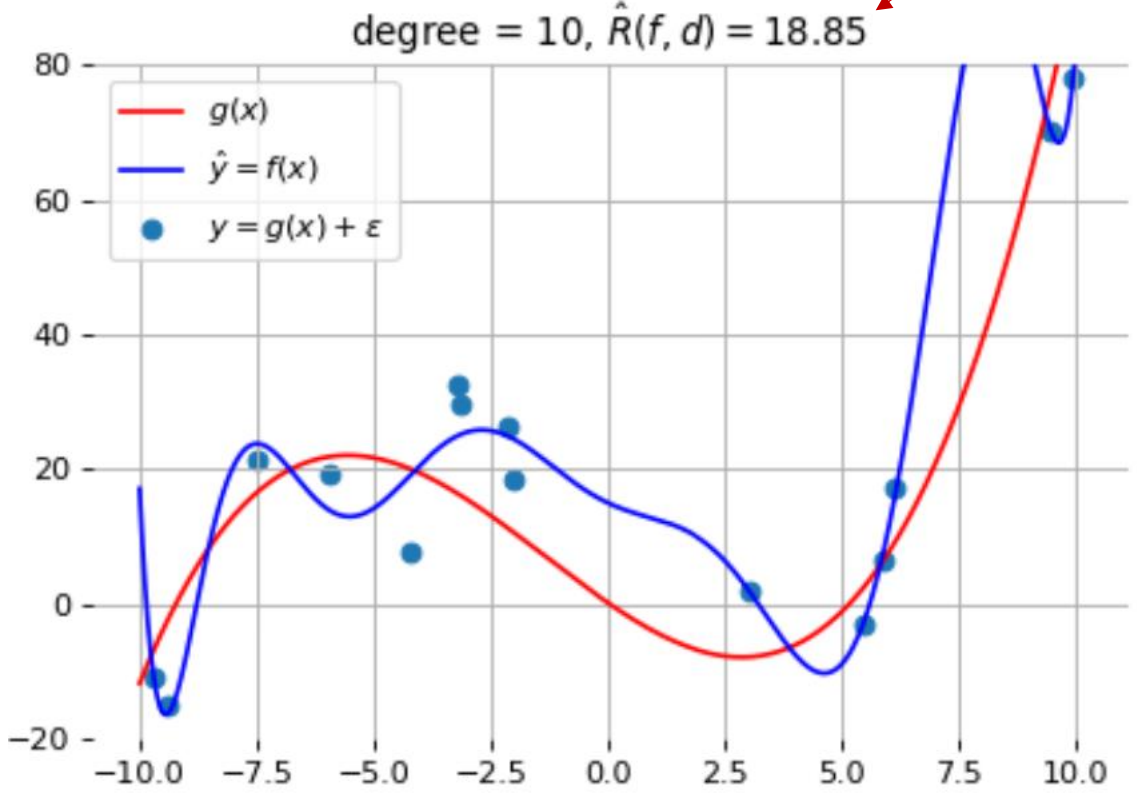
Under-fitting and over-fitting



$\mathcal{F}$ = polynomials of degree 5

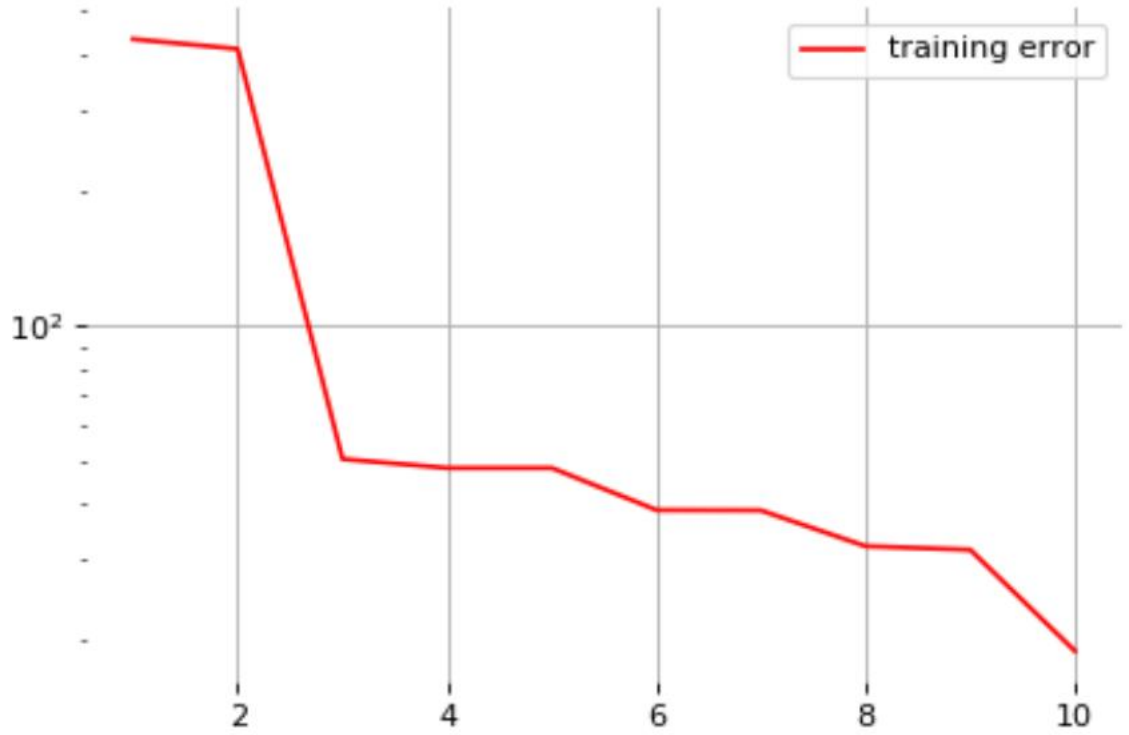
Under-fitting and over-fitting

The error is small. Is this a good solution?



$\mathcal{F}$ = polynomials of degree 10

Under-fitting and over-fitting



Degree d of the polynomial VS. error.

Under-fitting and over-fitting

Degree d of the polynomial VS. error.

Under-fitting and over-fitting

Let $\mathcal{Y}^{\mathcal{X}}$ be the set of all functions $f : \mathcal{X} \to \mathcal{Y}$.

We define the Bayes risk as the minimal expected risk over all possible functions,

$$R_B = \min_{f \in \mathcal{Y}^{\mathcal{X}}} R(f),$$

and call Bayes model the model f_B that achieves this minimum.

No model f can perform better than f_B .

Under-fitting and over-fitting

The capacity of an hypothesis space induced by a learning algorithm intuitively represents the ability to find a good model $f \in \mathcal{F}$ for any function, regardless of its complexity.

In practice, capacity can be controlled through hyper-parameters of the learning algorithm. For example:

- The degree of the family of polynomials;
- The number of layers in a neural network;
- The number of training iterations;
- Regularization terms.

Under-fitting and over-fitting

- If the capacity of $\mathcal{F}$ is too low, then $f_B \notin \mathcal{F}$ and $R(f) - R_B$ is large for any $f \in \mathcal{F}$, including f_* and $f_*^{\mathbf{d}}$. Such models *f* are said to underfit the data.

- If the capacity of $\mathcal{F}$ is too high, then $f_B \in \mathcal{F}$ or $R(f_*) - R_B$ is small. However, because of the high capacity of the hypothesis space, the empirical risk minimizer $f_*^{\mathbf{d}}$ could fit the training data arbitrarily well such that

$$R(f_*^{\mathbf{d}}) \geq R_B \geq \hat{R}(f_*^{\mathbf{d}}, \mathbf{d}) \geq 0 .$$

Expected risk Bayes risk Empirical risk

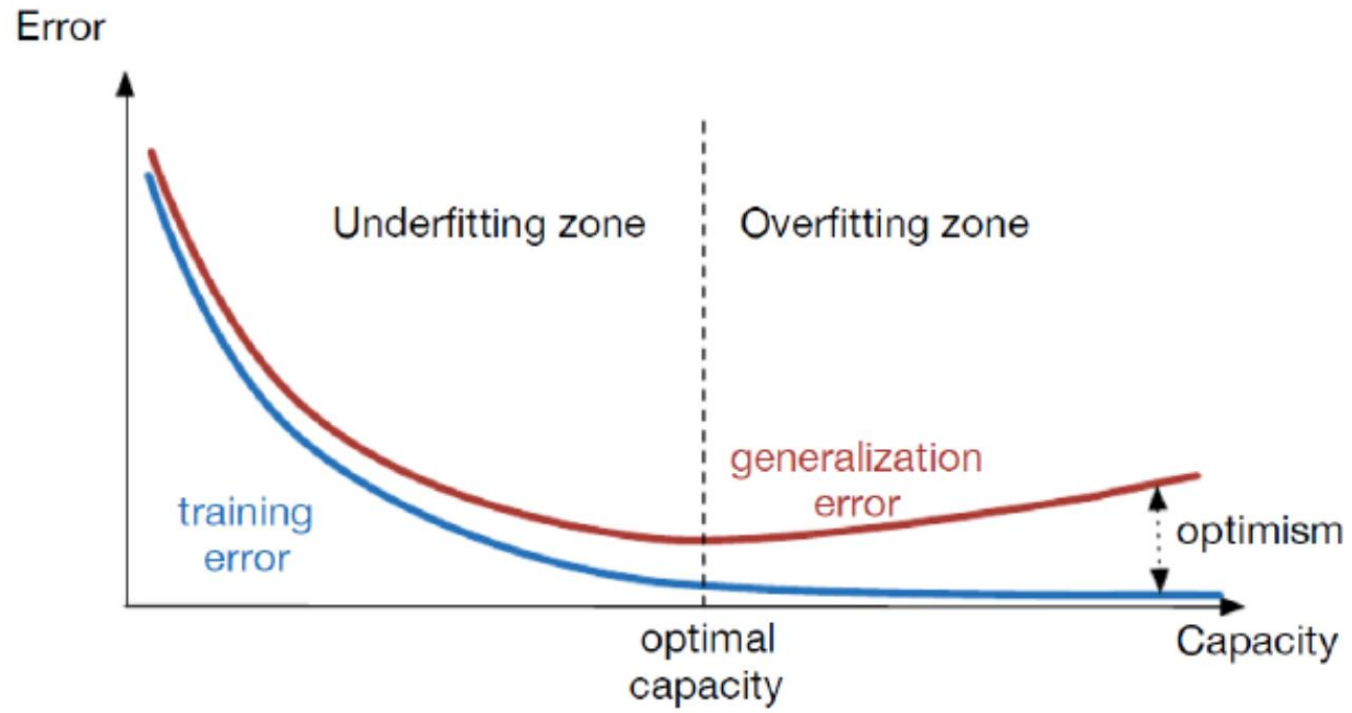
- In this situation, $f_*^{\mathbf{d}}$ becomes too specialized with respect to the true data generating process and a large reduction of the empirical risk (often) comes at the price of an increase of the expected risk of the empirical risk minimizer $R(f_*^{\mathbf{d}})$. In this situation, $f_*^{\mathbf{d}}$ is said to overfit the data.

Under-fitting and over-fitting

Can you draw the generalization error and training error curves?

Under-fitting and over-fitting

Therefore, our goal is to adjust the capacity of the hypothesis space such that the expected risk of the empirical risk minimizer gets as low as possible.



Under-fitting and over-fitting

When over-fitting

$$R(f_*^{\mathbf{d}}) \geq R_B \geq \hat{R}(f_*^{\mathbf{d}}, \mathbf{d}) \geq 0 .$$

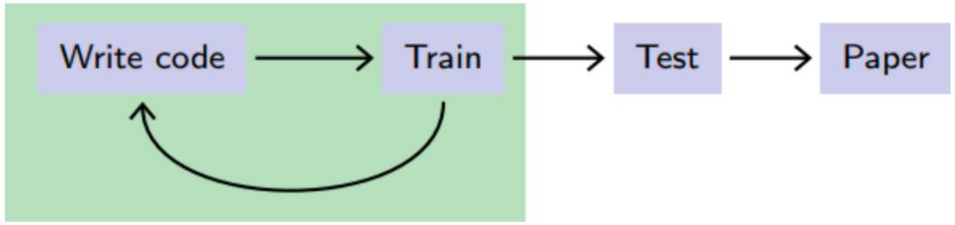
This indicates that the empirical risk $\hat{R}(f_*^{\mathbf{d}}, \mathbf{d})$ is a poor estimator of the expected risk $R(f_*^{\mathbf{d}})$.

Nevertheless, an unbiased estimate of the expected risk can be obtained by evaluating $f_*^{\mathbf{d}}$ on data $\mathbf{d}_{\text{test}}$ independent from the training samples $\mathbf{d}$:

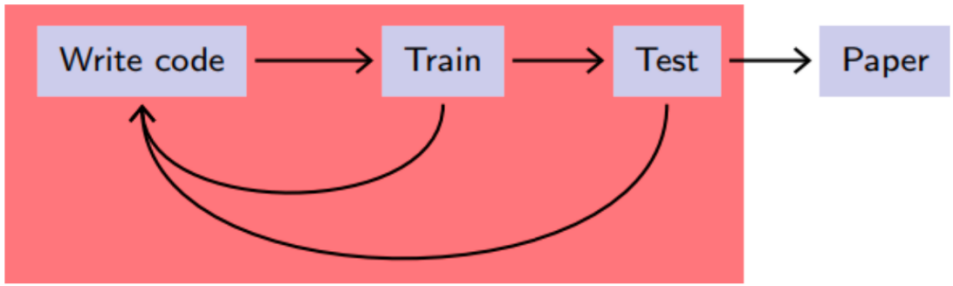
$$\hat{R}(f_*^{\mathbf{d}}, \mathbf{d}_{\text{test}}) = \frac{1}{N} \sum_{(\mathbf{x}_i, y_i) \in \mathbf{d}_{\text{test}}} \ell(y_i, f_*^{\mathbf{d}}(\mathbf{x}_i)) .$$

This test error estimate can be used to evaluate the actual performance of the model. However, it should not be used, at the same time, for model selection.

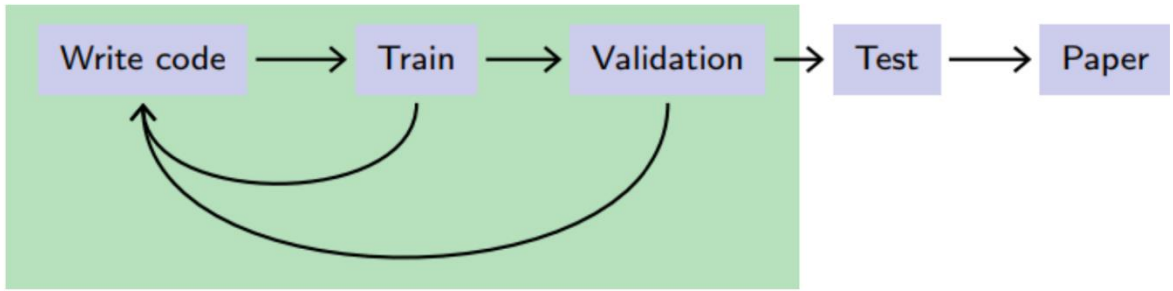
Evaluation protocol



There may be over-fitting, but it does not bias the final performance evaluation.



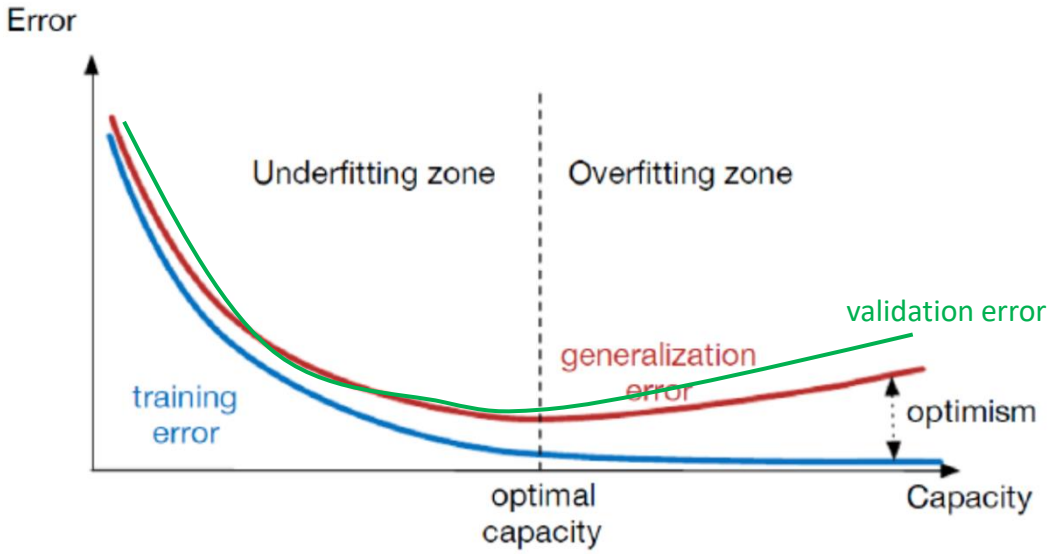
This should be avoided at all costs!



Instead, keep a separate validation set for tuning the hyper-parameters.

Under-fitting and over-fitting

- Early stopping is a common regularization technique to avoid over-fitting
- Stop optimization after some number of gradient steps, when the validation error starts increasing, even if optimization has not converged yet
- Work well in practice in deep learning to control overfitting



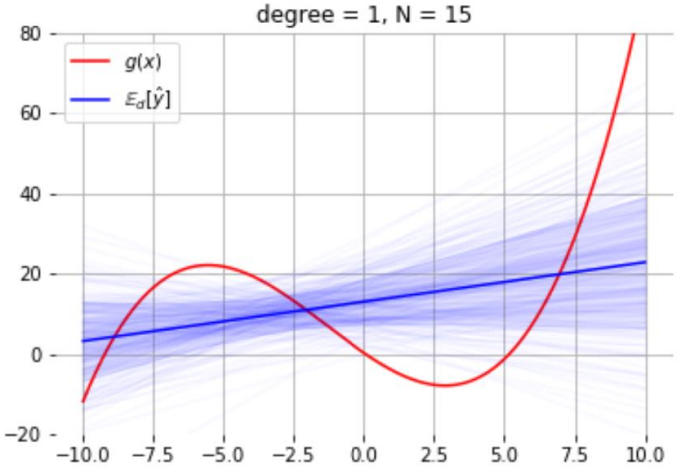
Bias-variance dilemma

- Whenever we discuss model prediction, it's important to understand prediction errors (bias and variance).
- There is a **tradeoff** between a model's ability to minimize bias and variance.
- Gaining a proper understanding of these errors would help us not only to build accurate models but also to avoid the mistake of overfitting and underfitting.

Bias-variance dilemma

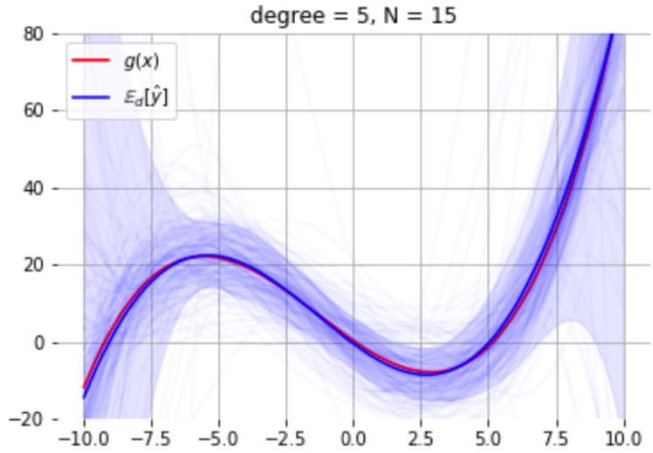
Bias

- Bias is the difference between the average prediction of our model and the correct value which we are trying to predict.
- Model with high bias pays very little attention to the training data and oversimplifies the model.
- It always leads to high error on training and test data.



Variance

- Variance is the variability of model prediction for a given data point or a value which tells us spread of our data.
- Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn't seen before.
- As a result, such models perform very well on training data but has high error rates on test data.



Bias-variance dilemma

Bias-variance decomposition: Consider a fixed point unseen sample x and the prediction $\hat{Y} = f_*^{\mathbf{d}}(x)$ of the empirical risk minimizer at x

It can be shown that the expected error on the unseen sample x

$$\mathbb{E}_{\mathbf{d}}\left[\left(y - f_*^{\mathbf{d}}(x)\right)^2\right] = \sigma^2 + \mathrm{Bias}_{\mathbf{d}}(f_*^{\mathbf{d}}(x))^2 + \mathrm{Var}_{\mathbf{d}}(f_*^{\mathbf{d}}(x))$$

where

- The first term is a irreducible noise term with zero mean and variance σ^2
- The second term is a bias term that measures the discrepancy between the average model and the Bayes model $\mathrm{Bias}_{\mathbf{d}}\left(f_*^{\mathbf{d}}(x)\right) = \mathbb{E}_{\mathbf{d}}\left[f_*^{\mathbf{d}}(x)\right] - f_B(x)$
- The third term is a variance term that quantifies the variability of the predictions $\mathrm{Var}_{\mathbf{d}}\left(f_*^{\mathbf{d}}(x)\right) = \mathbb{E}_{\mathbf{d}}\left[(f_*^{\mathbf{d}}(x))^2\right] - \mathbb{E}_{\mathbf{d}}\left[f_*^{\mathbf{d}}(x)\right]^2$

Bias-variance dilemma

Bias-variance trade-off - Question

If our model is too simple and has very few parameters, it will results in high/low bias and high/low variance

If our model has large number of parameters then it’s going to have high/low bias and high/low variance

Bias-variance dilemma

Bias-variance trade-off

- Reducing the capacity makes $f_*^{\mathbf{d}}$ fit the data less on average, which increases the bias term.
- Increasing the capacity makes $f_*^{\mathbf{d}}$ vary a lot with the training data, which increases the variance term.

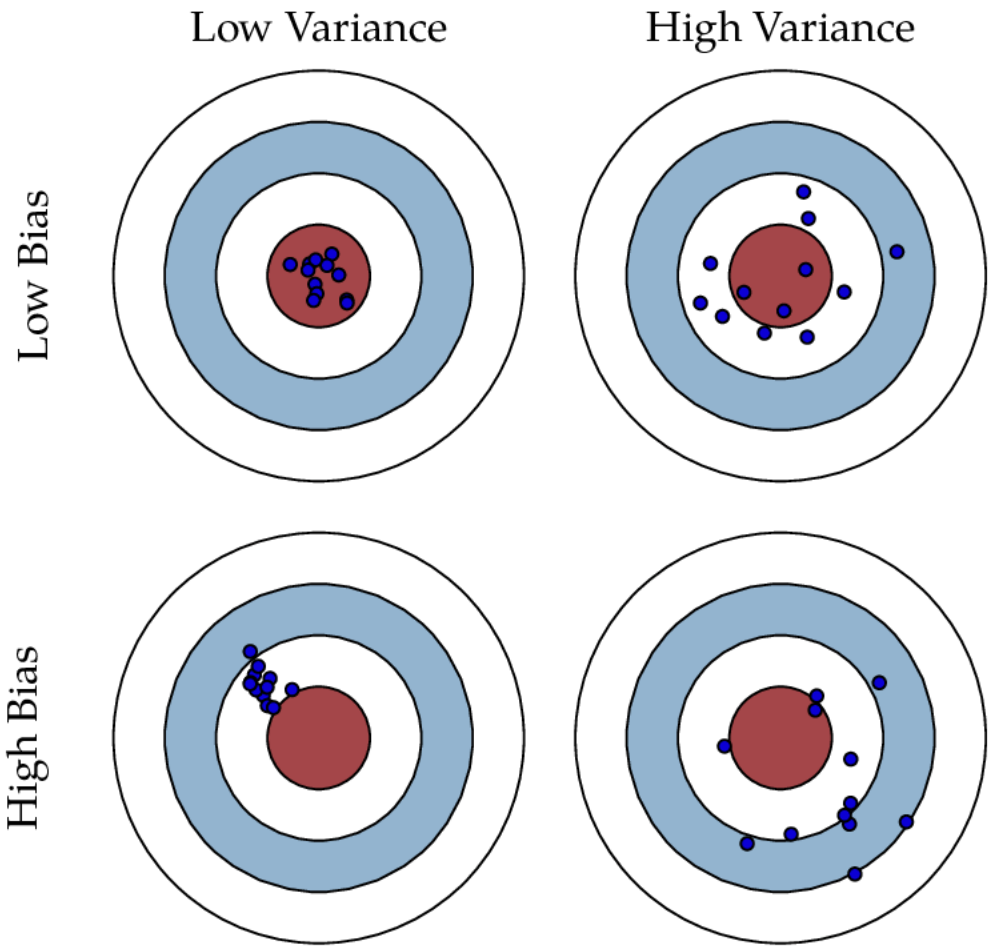
Bias-variance dilemma

Bias and variance explained using bulls-eye diagram

Which example indicates underfitting?

Which example indicates overfitting?

How to mitigate these problems?

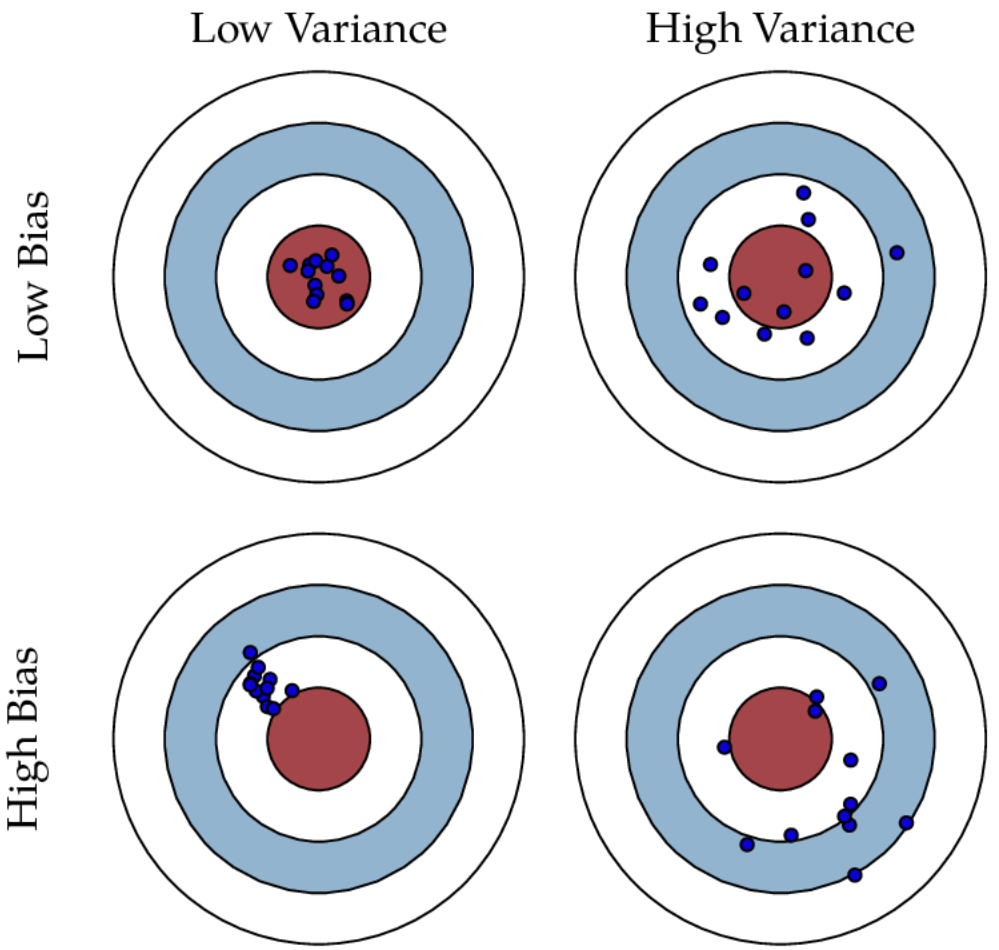


Bias-variance dilemma

Bias and variance explained using bulls-eye diagram

How to mitigate underfitting?

- This happens when a model unable to capture the underlying pattern of the data.
- These models usually have high bias and low variance.
- It happens when we have insufficient amount of data to build an accurate model or when we try to use a overly simple model to capture the complex patterns in data
- Increase the model complexity, add more meaningful features

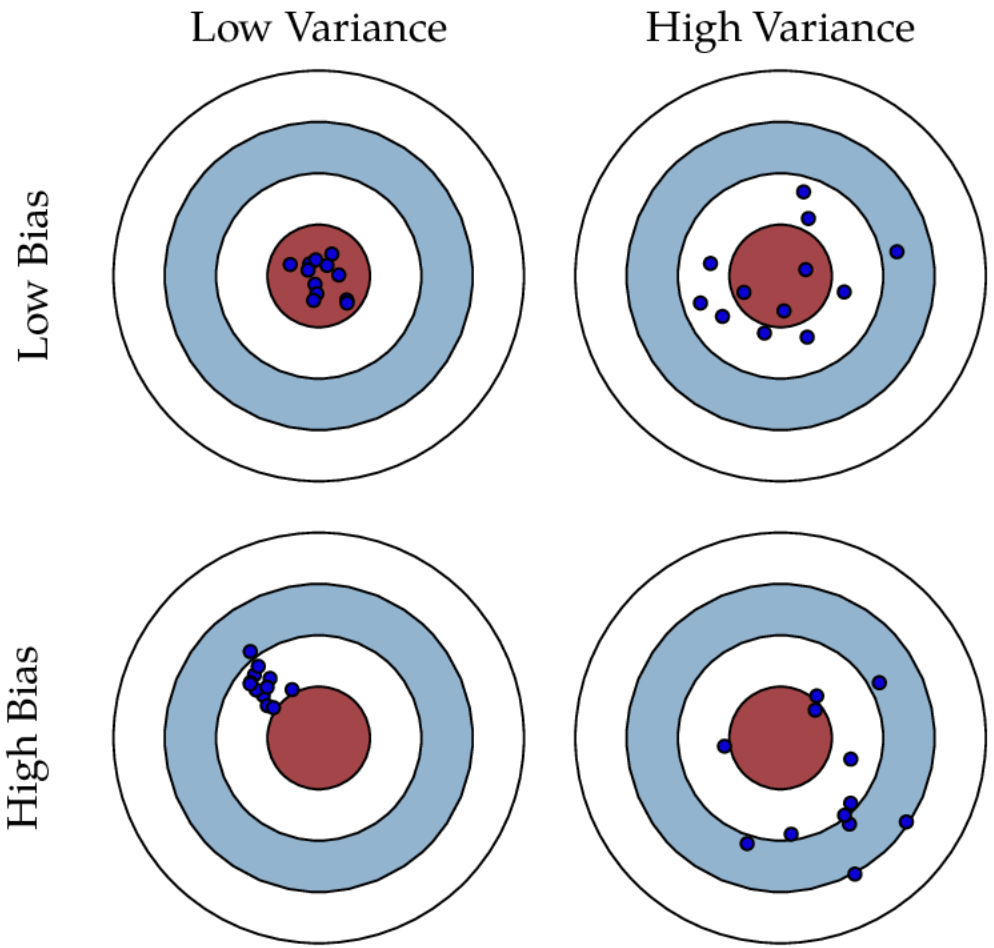


Bias-variance dilemma

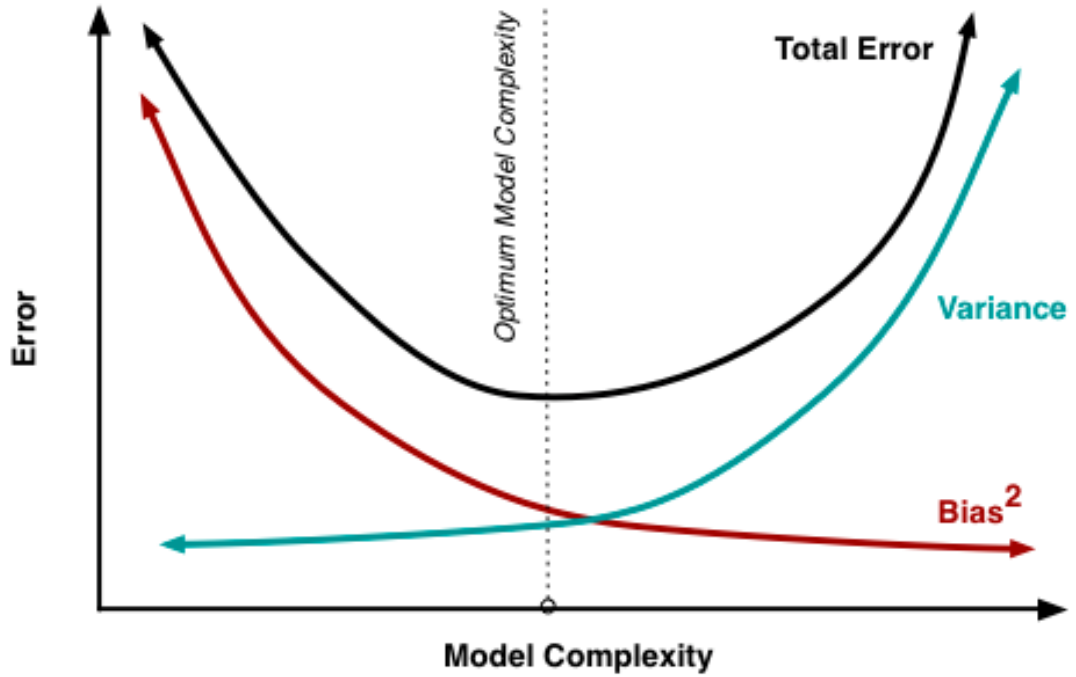
Bias and variance explained using bulls-eye diagram

How to mitigate overfitting?

- This happens when our model is too complex and too specialized on a small number of training data.
- These models usually have low bias and high variance.
- Increase the size of the data, remove outliers in data, reduce the complexity of the model, reduce the feature dimension



Bias-variance dilemma

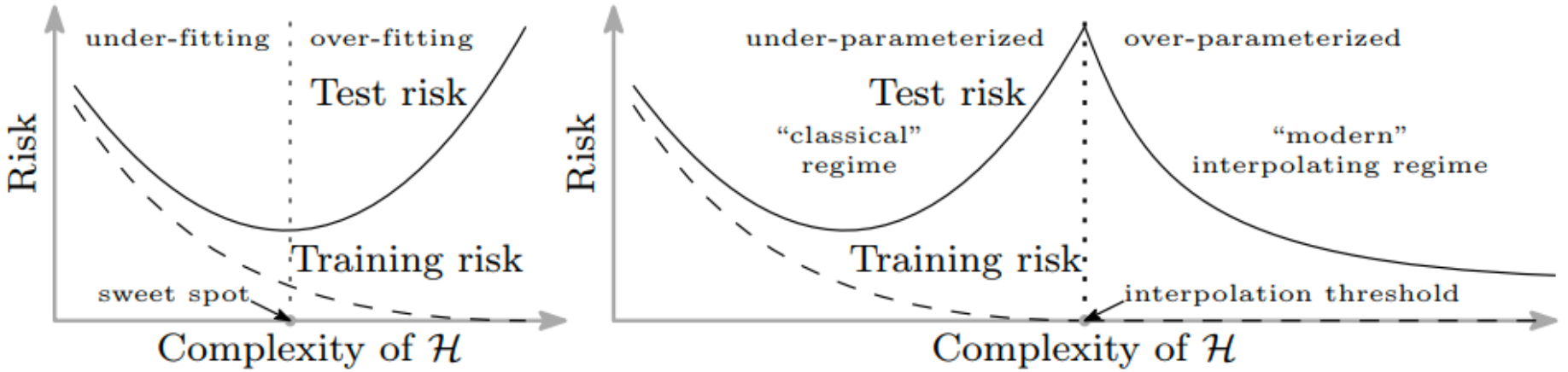


An optimal balance of bias and variance would never overfit or underfit the model.

What happen to neural networks with million of parameters?

Credit: Xavier Bresson CS3244 : Machine Learning - Lecture 6 : Regularization Techniques

Deep double descent



(a) U-shaped "bias-variance" risk curve

(b) "double descent" risk curve

Figure 1: Curves for training risk (dashed line) and test risk (solid line). (a) The classical *U-shaped risk curve* arising from the bias-variance trade-off. (b) The *double descent risk curve*, which incorporates the U-shaped risk curve (i.e., the "classical" regime) together with the observed behavior from using high complexity function classes (i.e., the "modern" interpolating regime), separated by the interpolation threshold. The predictors to the right of the interpolation threshold have zero training risk.

Reconciling modern machine learning practice and the bias-variance trade-off

Deep double descent

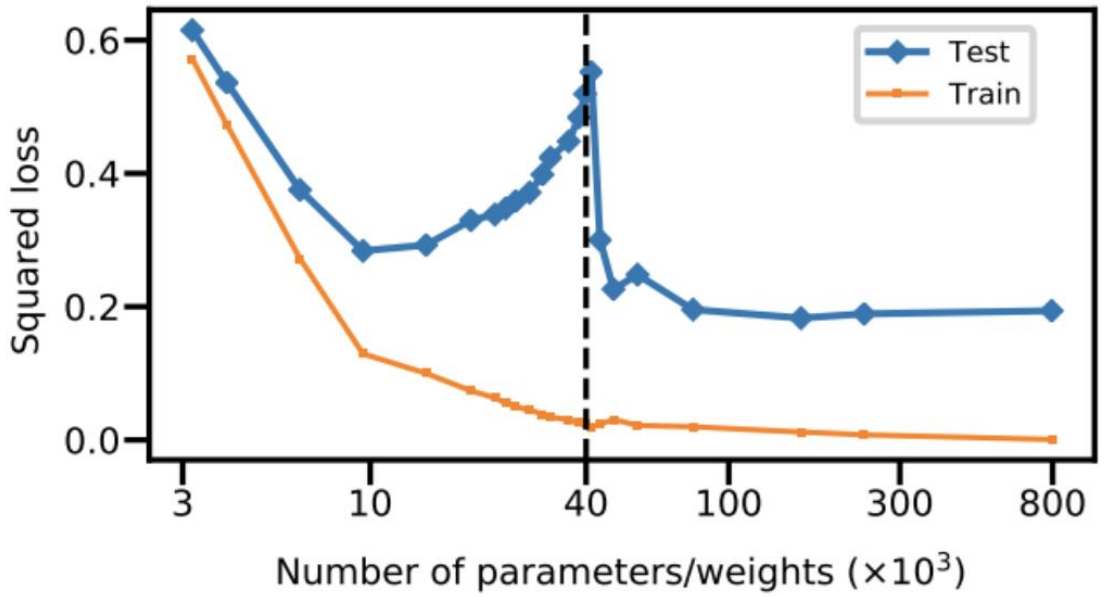
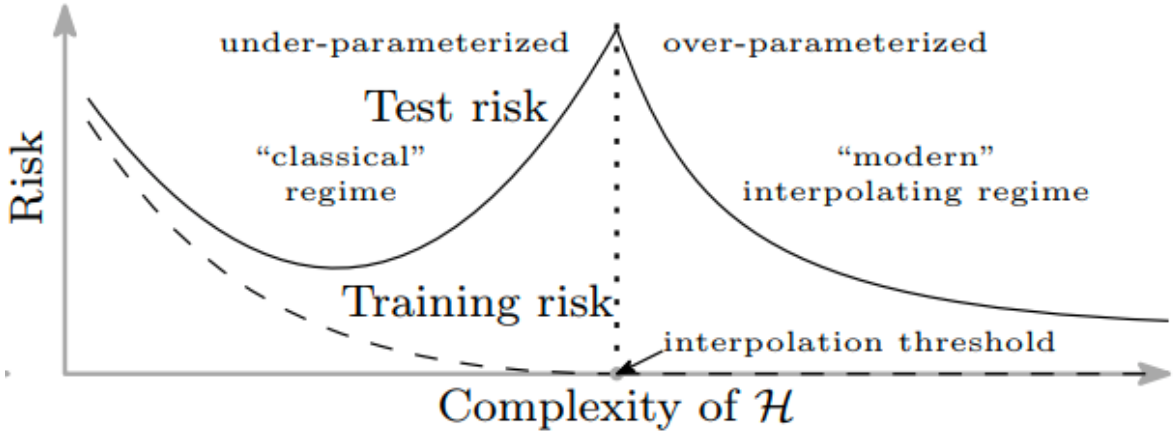


Figure 4: **Double descent risk curve for fully connected neural network on MNIST.** Training and test risks of network with a single layer of H hidden units, learned on a subset of MNIST ($n = 4 \cdot 10^3$, $d = 784$, $K = 10$ classes). The number of parameters is $(d+1) \cdot H + (H+1) \cdot K$. The interpolation threshold (black dotted line) is observed at $n \cdot K$.

Deep double descent



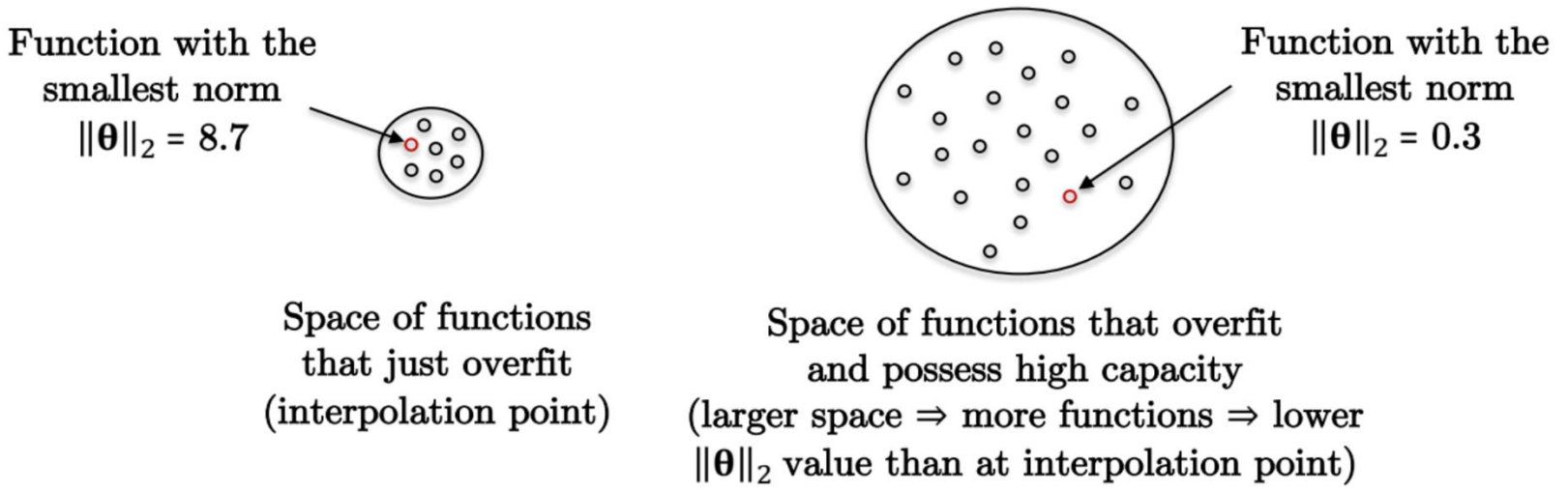
Let p be the number of parameters of the learner and n the number of training data points

- Under-parametrized functions are defined by $p \ll n$
- Over-parametrized functions are defined by $p \gg n$
- Interpolation threshold is the minimal capacity needed to overfit the training set, $p = n$

In the modern ML regime, over-fitting is considered beneficial, and over-parametrized functions with high model complexity lead to successful generalization.

Deep double descent

- When $p = n$, the model possesses just enough parameters to over-fit all the training data. However, it also exhibits a significant variance, making it unable to generalize
- When $p \gg n$, the model has much greater parameters than the number of training data. In this regime, the learner $f_{\boldsymbol{\theta}}(x)$ continues to over-fit but critically, the L_2norm of its parameters $\|\boldsymbol{\theta}\|_2$ is significantly minimized by SGD, effectively reducing the model capacity (regularization effect).



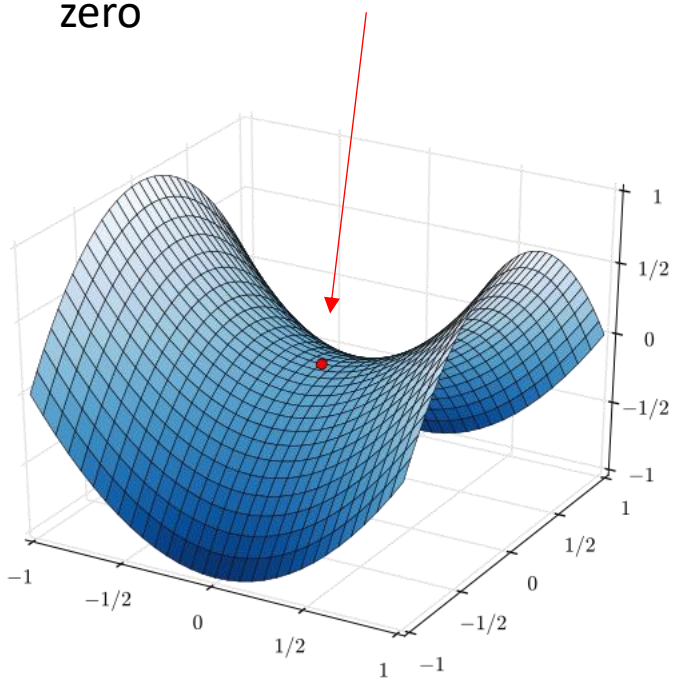
Credit: Xavier Bresson CS3244 : Machine Learning - Lecture 6 : Regularization Techniques

Deep double descent

SGD is critical in deep learning for several reasons

- It helps to leave saddle points in the loss landscape during optimization.
- It finds better local (or even global) minima, allowing successful generalization.
- It is necessary for the double descent phenomenon to emerge.

A point on the surface of the graph of a function where the slopes (derivatives) in orthogonal directions are all zero



A saddle is a supportive structure for a rider of an animal, fastened to an animal's back by a girth

Deep double descent

The double descent regularization only emerges with exceedingly large networks

- The critical threshold to enable double descent is $p^* = O(n.k)$, where k is the number of classes
- ImageNet : $n = 10^6$ (1.3M images), $k = 10^3$ (1k classes) $\Rightarrow p^* = 10^9$

  ResNet-152 has p =60.2M (107) parameters$\ll 10^9$

- ViT: $n = 10^9$ (4B images), $k = 10^4$ (30k classes) $\Rightarrow p^* = 10^{13}$

  ViT-22B has p =22B (10^{10}) parameters $\ll 10^{13}$

Deep double descent

- The double descent learning mechanism is applicable to both non-linear and linear ML models.
  - This includes techniques such as decision trees, kernel methods, and deep learning.

- The phenomenon is independent of the nature of the datasets involved.

- One important ML principle is that more data provides better results.
  - Both theory and empirical experiments align on this principle
  - However, this trend continually increases the critical threshold p^* of required network parameters for the double descent phenomenon to manifest.

Neural scaling law

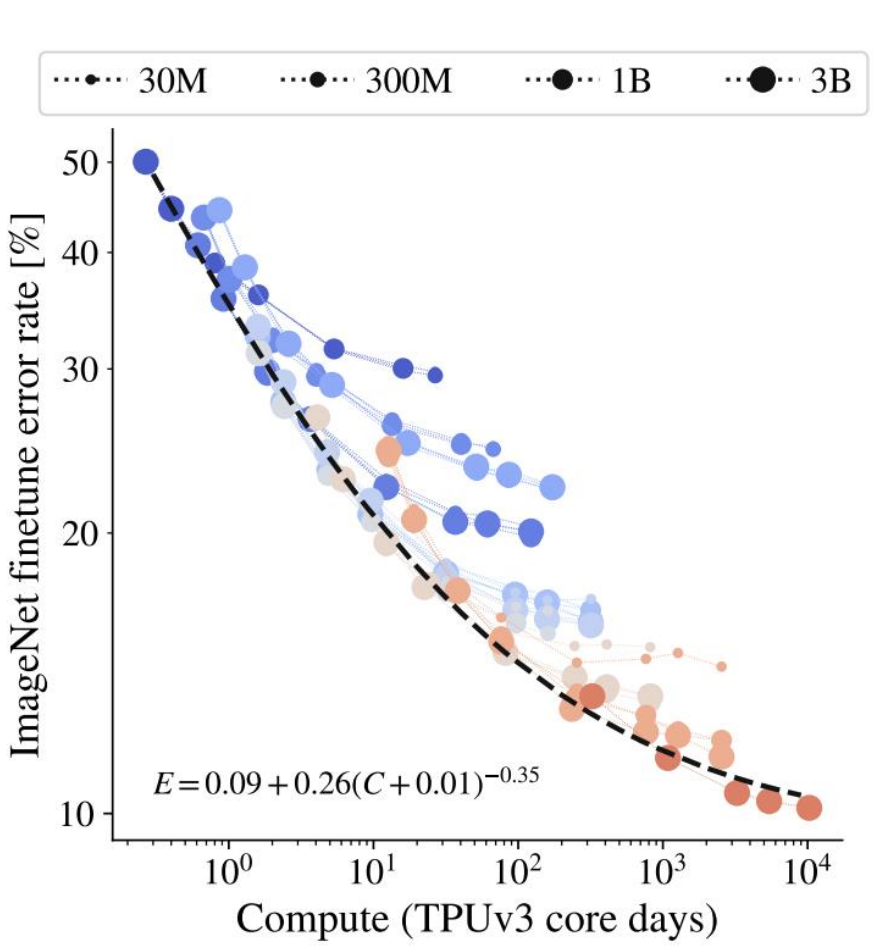
- Neural scaling law - empirical relationships between
  - **Model size**: Number of parameters.
  - **Dataset size**: Amount of training data.
  - **Compute**: Training steps or FLOPs (floating-point operations per second).

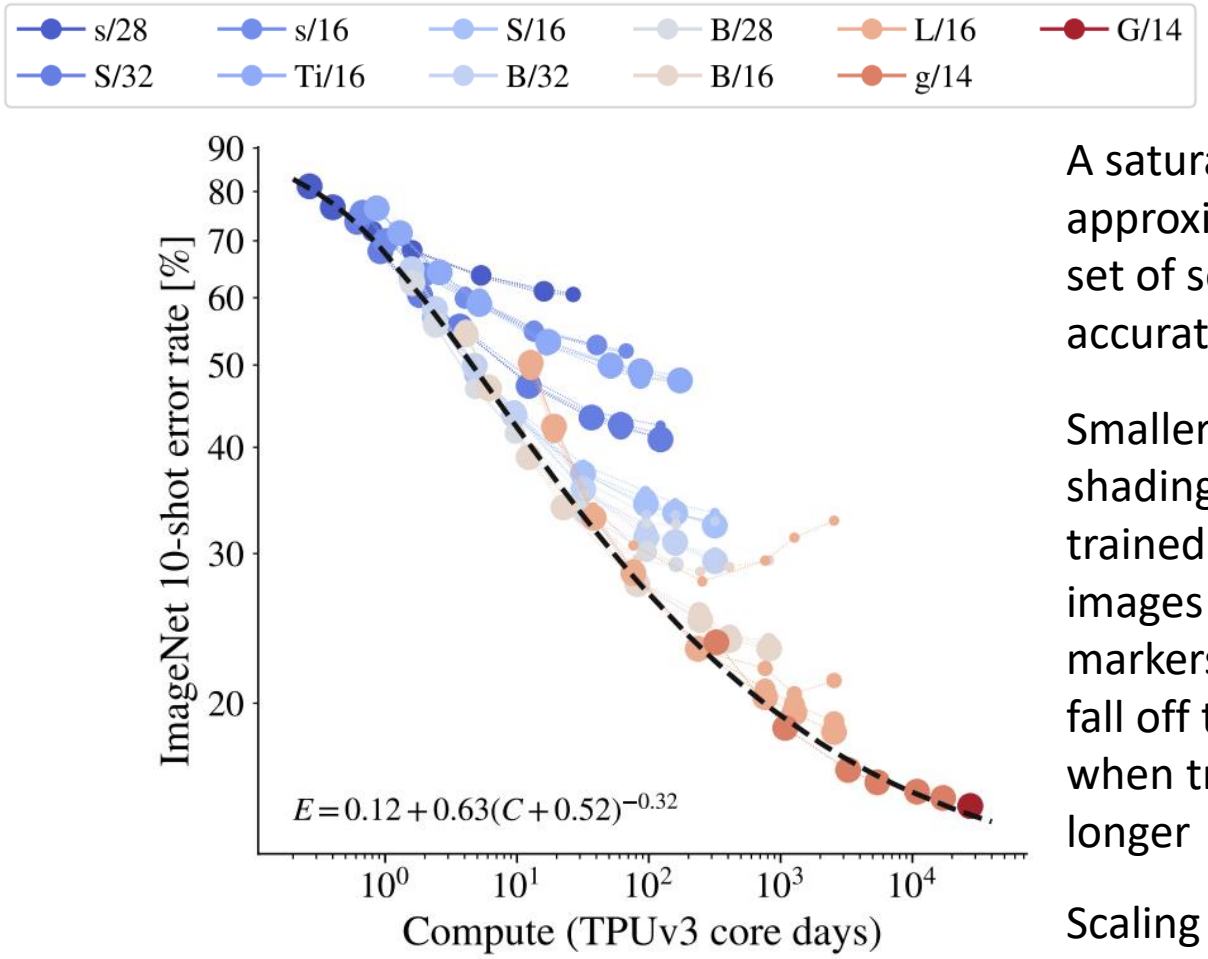
- Performance (e.g., loss or accuracy) improves predictably with increases of the above factors.

- Often follows a power law $L = A \times N^{-\alpha} + B$

L: Loss
N: Scaling variable (e.g., model size or dataset size)
α, A, B: Empirically determined constants, α is the scaling exponent that determines the impact of variable

Neural scaling law





Representation quality, measured as ImageNet finetune and linear 10-shot error rate, as a function of total training compute

A saturating power-law approximates the best set of solutions fairly accurately

Smaller models (blue shading), or models trained on fewer images (smaller markers), saturate and fall off the frontier when trained for longer

Scaling up compute, model and data together improves representation quality

[Scaling vision transformers]

Neural scaling law

- Observations
  - Resources should be proportionally allocated across model size, dataset size, and compute.
  - Performance gains taper off if additional data is not diverse or useful.
  - Large models trained on diverse datasets exhibit strong transfer learning capabilities.
  - Larger models may develop new capabilities not present in smaller ones (e.g., language understanding or reasoning)

Summary

Set the fundamentals of machine learning

- Why learning?
- Statistical learning
  - Supervised learning
  - Empirical risk minimization (using polynomial regression as an example)
  - Underfitting and overfitting
  - Bias-variance dilemma
  - Deep double descent
  - Neural scaling law

Next lecture

- Basic components in CNN
- CNN architectures
- Training a classifier
- Optimizers