DEEP LEARNING FOR VISUAL RECOGNITION

Lecture 1 – Introduction

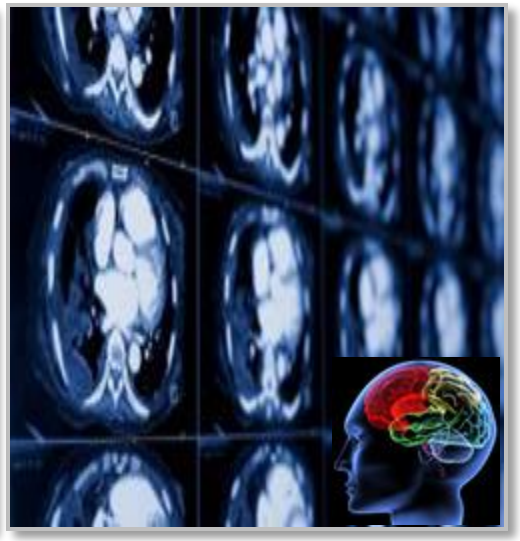
Henrik Pedersen, PhD
Part-time lecturer
Department of Computer Science
Aarhus University
hpe@cs.au.dk

The Deep Learning Revolution

Internet Services

- Image/Video classification
- Speech recognition
- Chatbots

Medicine



- Cancer cell detection
- Diabetic grading
- Drug discovery

Media & Entertainment

- Video captioning
- Content based search
- Real time translation

Security & Defense

- Face recognition
- Video surveillance
- Cyber security

Autonomous Machines

- Pedestrian detection
- Lane tracking
- Recognize traffic signs

Today’s agenda

- Practical information
- Introduction to traditional computer vision
- Introduction to deep learning

Practical information

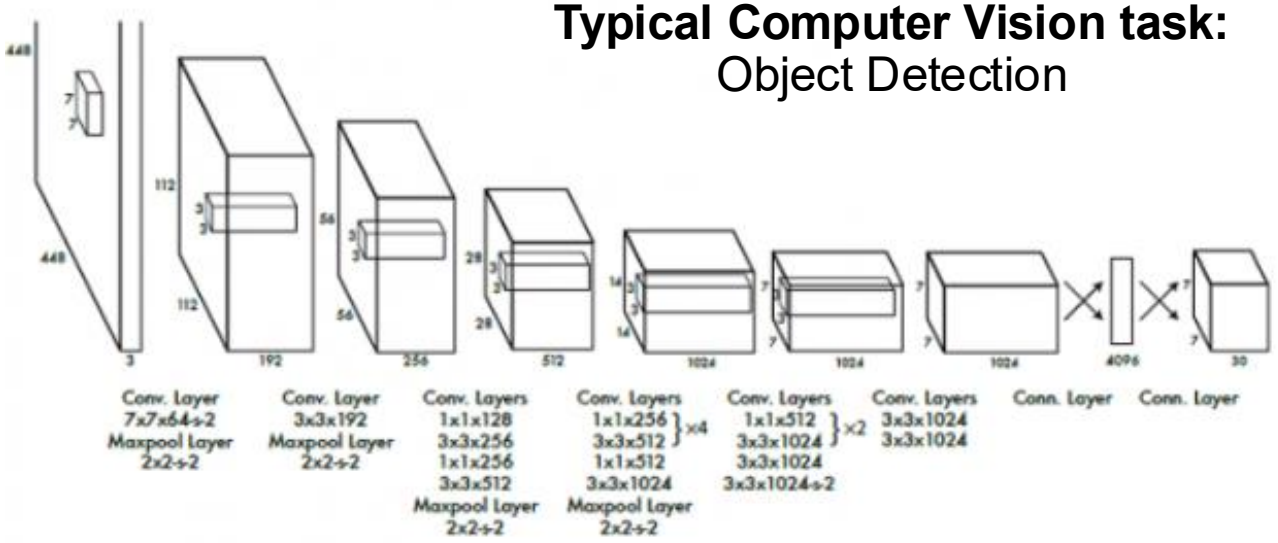
Overall structure of the course

- **Lectures** on Mondays will cover deep learning theory **bottom-up** (theory first).
- **Programming exercises** on Thursdays will teach you deep learning **top-down** (practise first).
- You will learn *how* deep learning works before learning *why* it works.
- This will allow you to start working on your course project early in the semester.

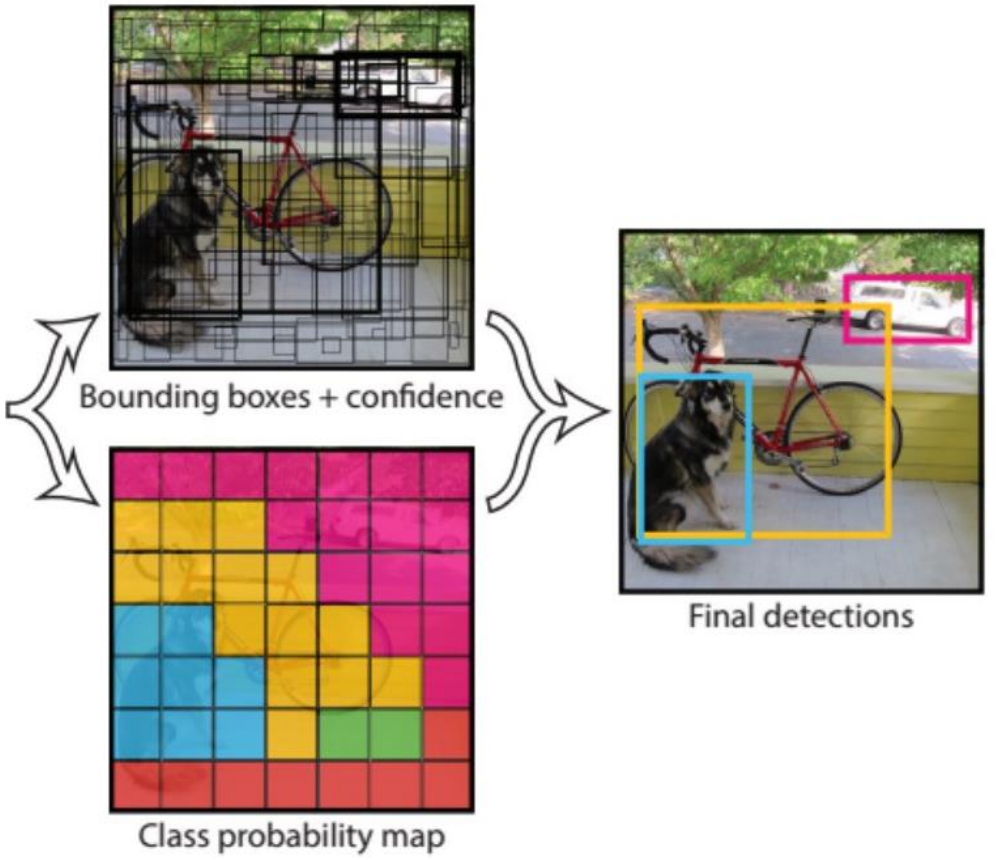
Learning goals

- Identify and describe visual recognition tasks that can be solved with deep learning.
- Describe and compare different neural networks architectures.
- Explain and compare techniques for training neural networks.
- Apply deep learning to standard visual recognition tasks and interpret the results.
- Design your own course project, implement it, perform experiments, analyse and relate the results to techniques and theories learned in class.
- Detailed learning goals are available here

Learning goals



Typical Computer Vision task:
Object Detection



1) Which neural network architecture(s) will solve the task?
2) What type of training data do you need to train the network?
3) Which training objective(s) are suitable for training the network?
4) Once trained, how do you evaluate the performance of the network?

Resources

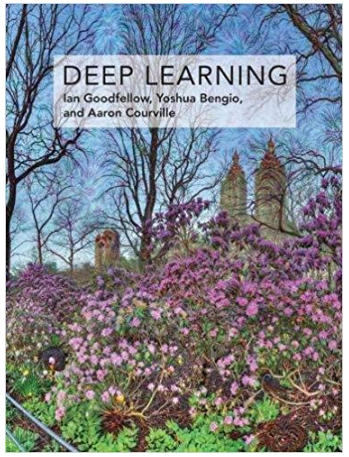
- Prerequisites
  - Basic calculus (you are expected to be able to understand the math in sections 1-3 here)
  - Basic linear algebra (at the very least, you should be able to understand chapter 2.1-2.3)
  - Book chapter covering preliminaries here (with code!)
- Brightspace
  - https://brightspace.au.dk/d2l/home/183737
  - Course schedule, syllabus, lecture slides, links to exercises, video recordings, etc.
- Slack
  - https://dlau2025.slack.com (invite link on Brightspace)
  - Group formation, questions about exercises, etc.
- GitHub repository (programming exercises)
  - https://github.com/klaverhenrik/Deep-Learning-for-Visual-Recognition-2025
- Google Colab (recommended tool to solve programming exercises)
  - https://colab.research.google.com/

Syllabus

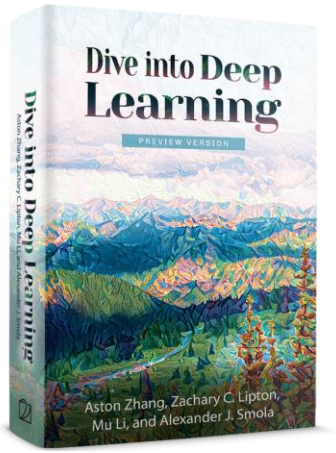
- Mostly lecture slides and research papers
- Freely available textbooks (optional)
- Lecture recordings will be uploaded at the end of the semester

https://www.deeplearningbook.org/

https://github.com/janishar/mit-deep-learning-book-pdf



https://d2l.ai/



Course project

- Your project should involve pixels of visual data in some form.
- You must train a neural network model, evaluate it and perform experiments to improve its performance.
- When designing your project, keep the deep learning workflow in mind:
  - Acquire and prepare a dataset for deep learning
  - Set up an appropriate deep learning model (i.e., a neural network) to solve the task at hand
  - Train and test your model with the correct evaluation metrics
  - Perform motivated experiments to improve your model's performance
  - Explain your results
- Groups of 2 or 3 people (use dedicated slack channel to form groups).

Exam

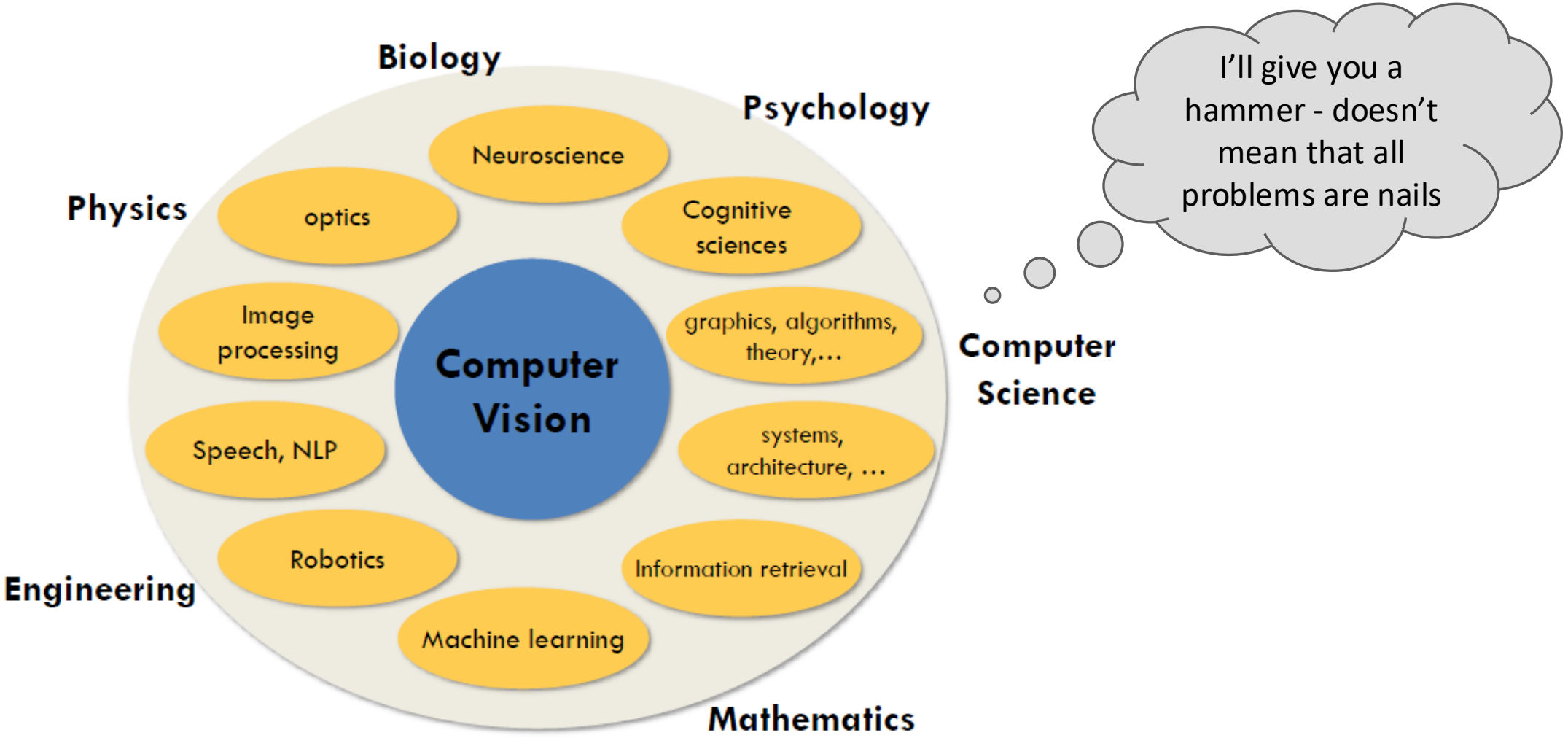
- 15 minutes oral exam without preparation (8 topics)
- Present on whiteboard (8 minutes)
  - Slides not allowed
- Questions relating to course theory (4 minutes)
- Grade reflects an overall assessment of project report and individual oral examination
  - Report weighs 50%
- Detailed learning goals are available here

Questions?

Traditional computer vision

QUICK OVERVIEW

An interdisciplinary field



Levels of computer vision

HIGH LEVEL

Recognition, reconstruction, understanding and integration.

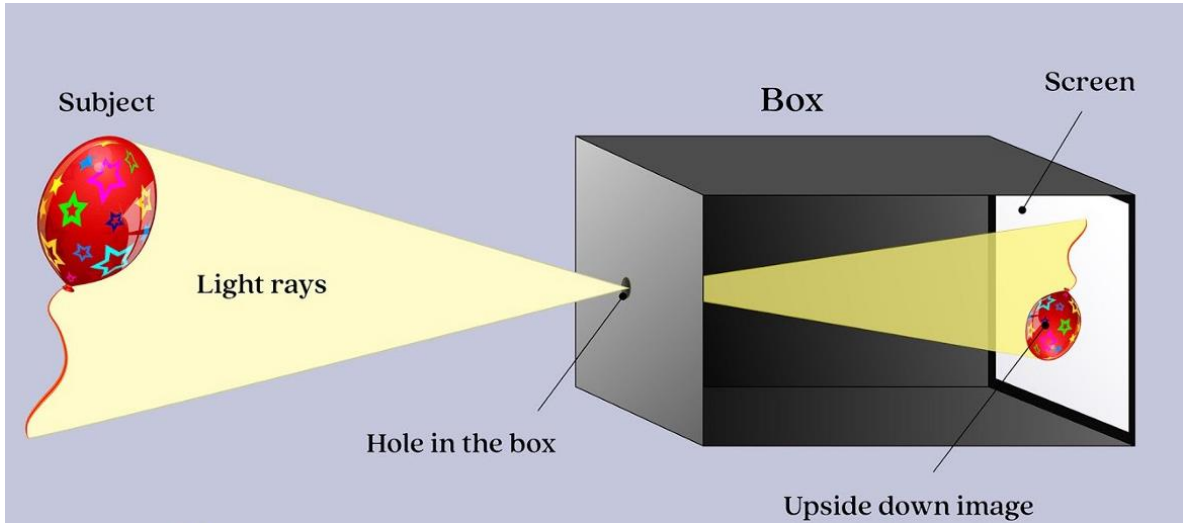
MEDIUM LEVEL

Mainly feature extraction: shape, motion and segmentation.

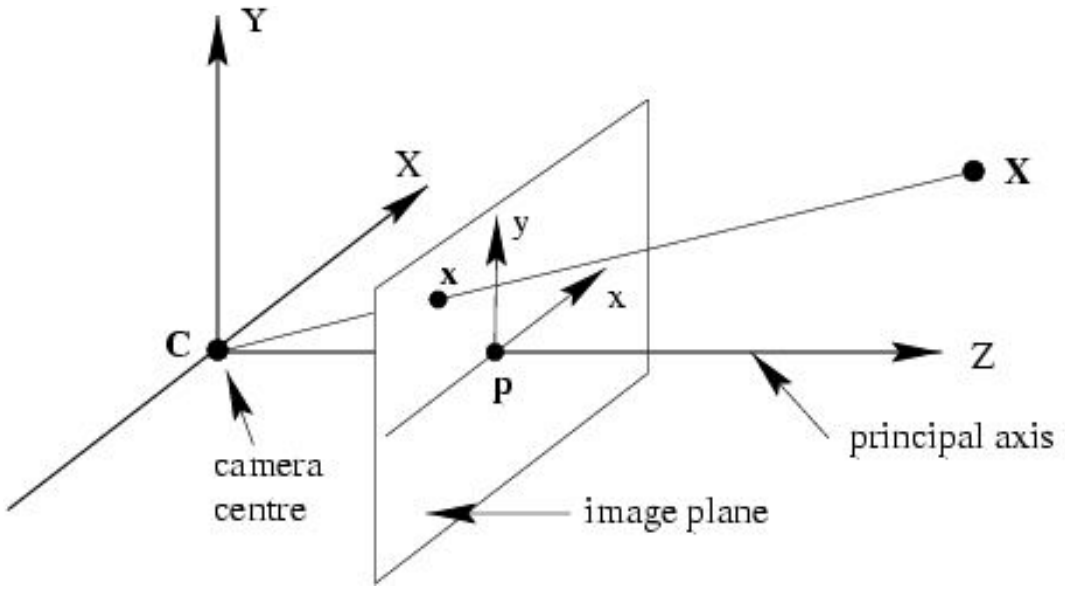
LOW LEVEL

Mainly pre-processing: image enhancement, image filtering and simple transformations.

The pinhole camera model



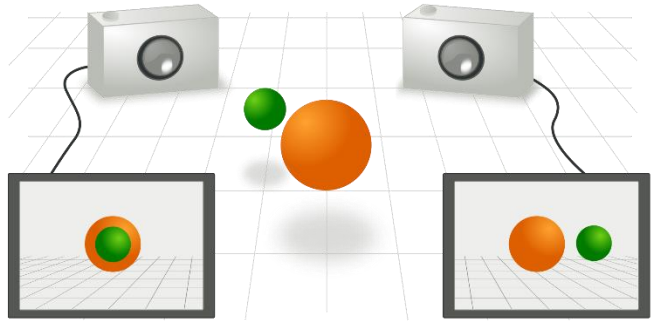
https://en.wikipedia.org/wiki/Camera_obscura

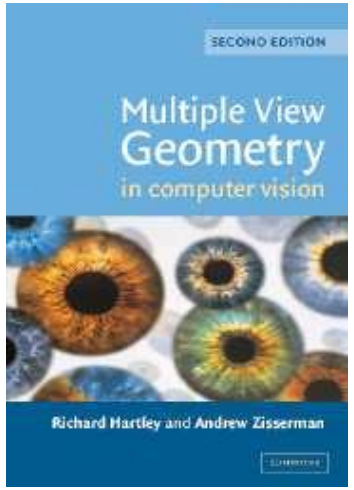


https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html

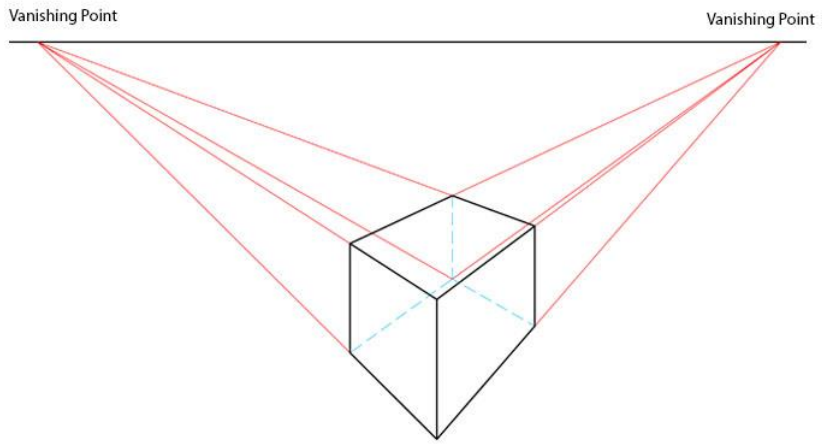
Applications of the pinhole camera model

- Perspective drawing
- Stereo (epipolar geometry) and 3D reconstruction
- Pose estimation
- Image stitching (panoramas)

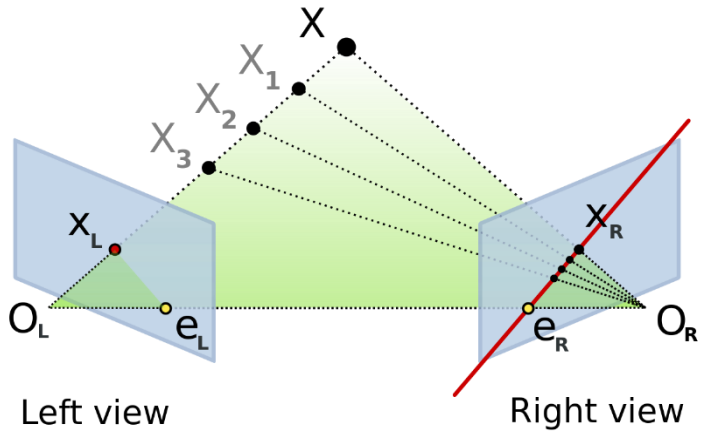




http://www.robots.ox.ac.uk/~vgg/hzbook/



https://en.wikipedia.org/wiki/Perspective_(graphical)

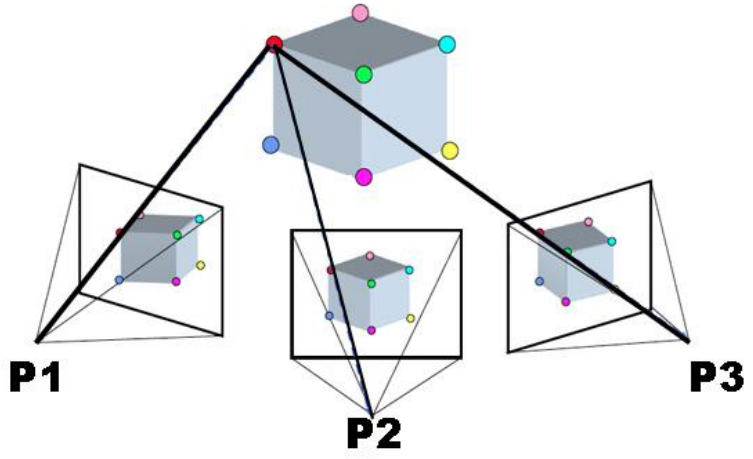


https://en.wikipedia.org/wiki/Epipolar_geometry

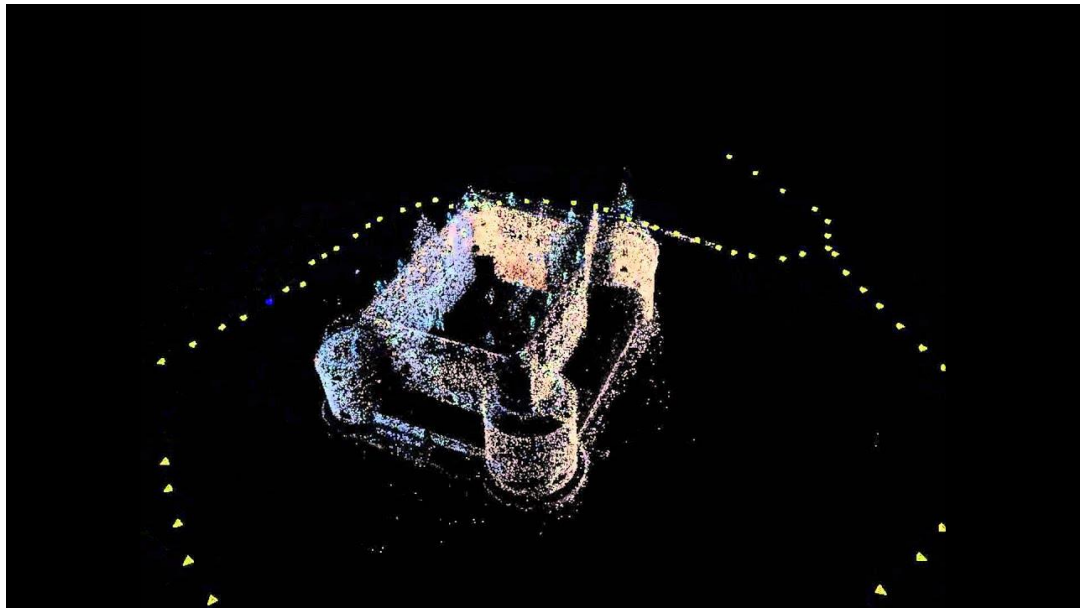
https://en.wikipedia.org/wiki/3D_reconstruction

3D scene reconstruction

- Reconstruct a 3D scene from multiple images taken from different viewpoints
- Related topics: Structure-from-motion, Stereo, Camera calibration, Camera pose estimation



https://es.wikipedia.org/wiki/Structure_from_motion



https://www.youtube.com/watch?v=i7ierVkXYa8

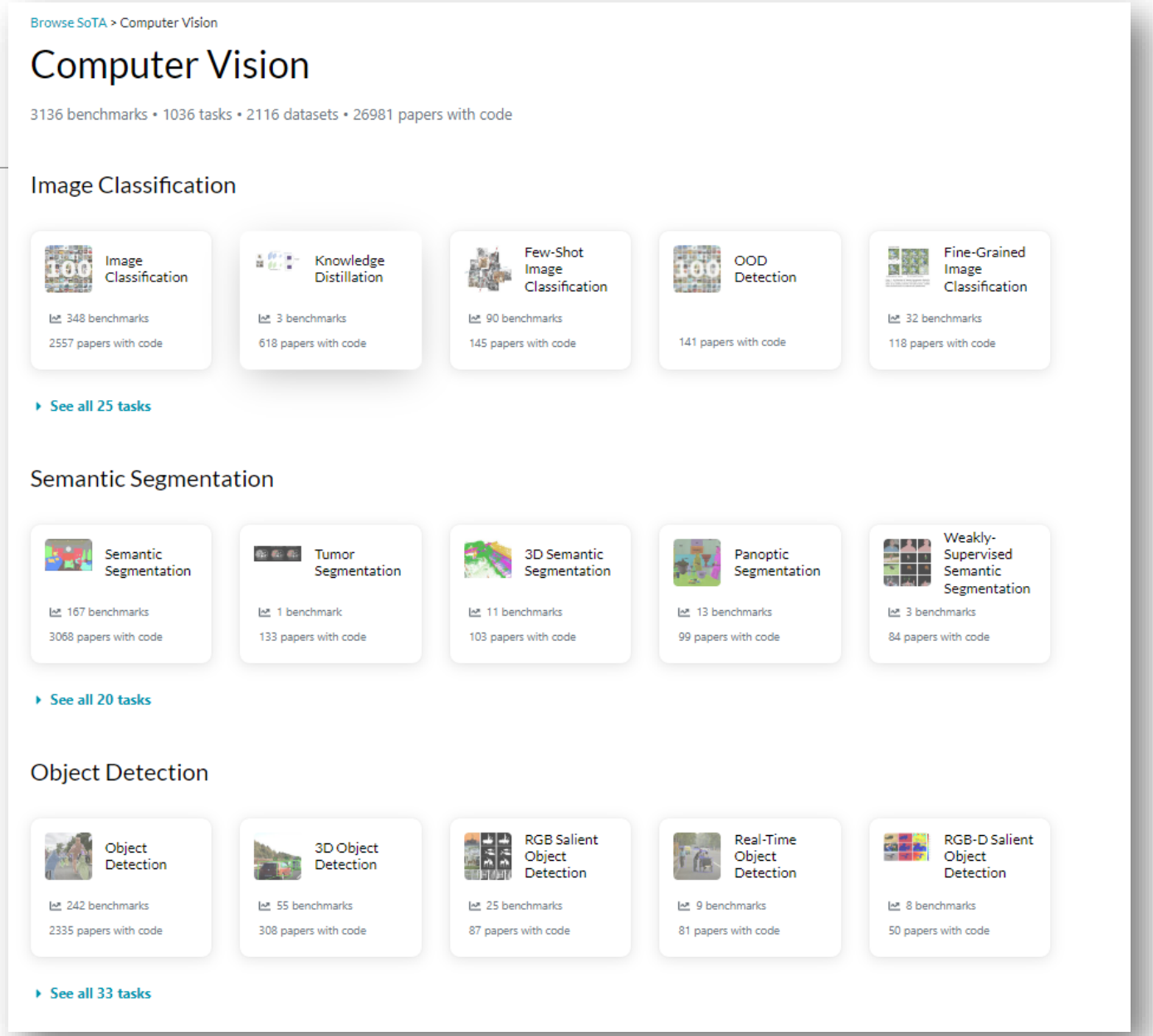
Object tracking

- Track moving objects in time series of images
- Related topics/synonyms: Motion estimation, Video tracking, Image registration, Optical flow

https://blog.nanonets.com/optical-flow/

More examples

- Visit Papers With Code
- https://paperswithcode.com/area/computer-vision



Towards image understanding

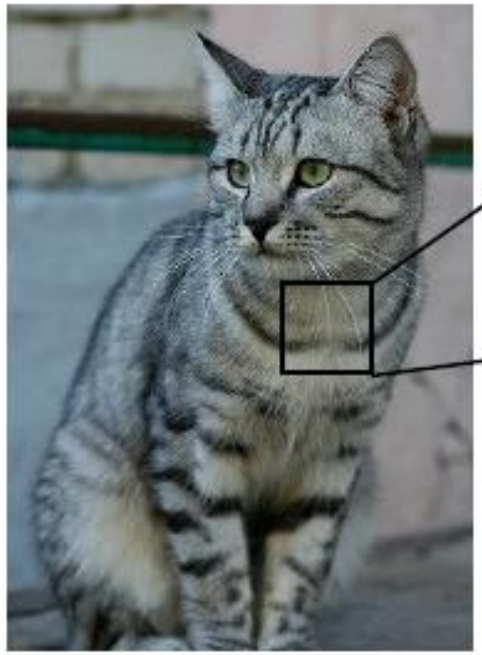
- **Example:** Image classification
- Given an image, tell which in a number of classes it belongs to
  - What type of object?
  - Quality control: OK or needs manual inspection?
  - View direction: frontal, side, top, bottom?

(assume given set of discrete labels)
{dog, cat, truck, plane, ...}

→ cat

Main problem – semantic gap

- Computers don’t see the way humans do




```
[[105 112 108 111 104  99 106  99  96 103 112 119 104  97  93  87]
 [ 91  98 102 106 104  79  98 103  99 105 123 136 110 105  94  85]
 [ 76  85  90 105 128 105  87  96  95  99 115 112 106 103  99  85]
 [ 99  81  81  93 120 131 127 100  95  98 102  99  96  93 101  94]
 [106  91  61  64  69  91  88  85 101 107 109  90  75  84  96  95]
 [114 108  85  55  55  69  64  54  64  87 112 129  98  74  84  91]
 [133 137 147 103  65  81  80  65  52  54  74  84 102  93  85  82]
 [128 137 144 140 109  95  86  70  62  65  63  63  60  73  86 101]
 [125 133 148 137 119 121 117  94  65  73  80  65  54  64  72  98]
 [127 125 131 147 133 127 126 131 111  96  89  75  61  64  72  84]
 [115 114 109 123 150 148 131 118 113 109 100  92  74  65  72  78]
 [ 89  93  90  97 108 147 131 116 113 114 113 109 106  95  77  80]
 [ 63  77  86  81  77  79 102 123 117 115 117 125 125 130 115  87]
 [ 62  65  82  89  70  71  80 101 124 126 119 101 107 114 131 110]
 [ 63  65  75  88  89  71  62  81 120 138 135 105  81  98 110 118]
 [ 87  65  71  87 106  95  69  45  76 130 126 107  92  94 105 112]
 [118  97  82  86 117 123 116  66  41  51  95  93  89  95 102 107]
 [164 146 112  80  82 120 124 104  76  48  45  66  88 101 102 109]
 [157 170 157 120  93  86 114 132 112  97  89  55  70  82  99  94]
 [138 128 134 161 139 100 109 118 121 134 114  87  65  53  69  86]
 [120 112  96 117 150 144 120 115 104 107 102  93  87  81  72  70]
 [123 107  96  86  83 112 153 149 122 103 104  75  88 107 112  99]
 [122 121 102  88  82  86  94 117 145 148 153 102  58  78  92 107]
 [122 164 148 103  71  56  78  83  93 103 119 139 102  61  69  84]]
```

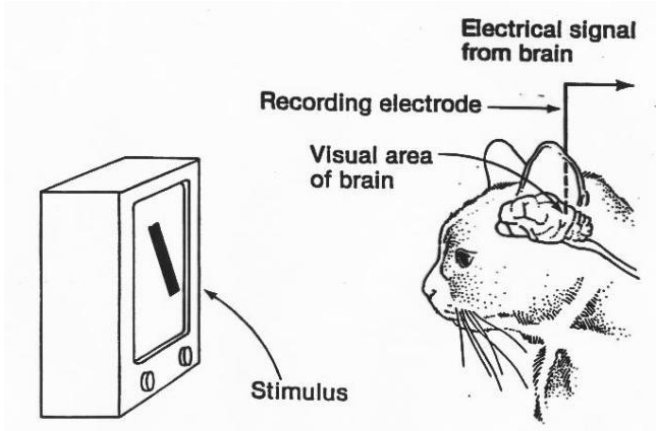
What the computer sees

An image is just a big grid of numbers between [0, 255]:

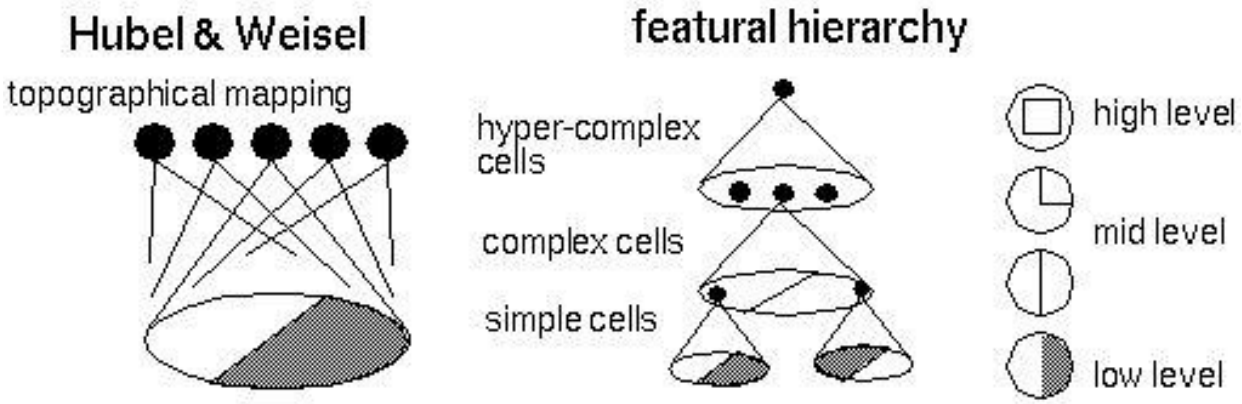
e.g. 800 x 600 x 3
(3 channels RGB)

Inspiration from neuroscience

- Famous experiment by Hubel and Weisel
- Feature hierarchy: Neurons along the visual pathway extract increasingly complex information from the pattern of light cast on the retina to construct and understand an image.
- Example: Edges → Corners → Boxes → Object parts → Whole objects

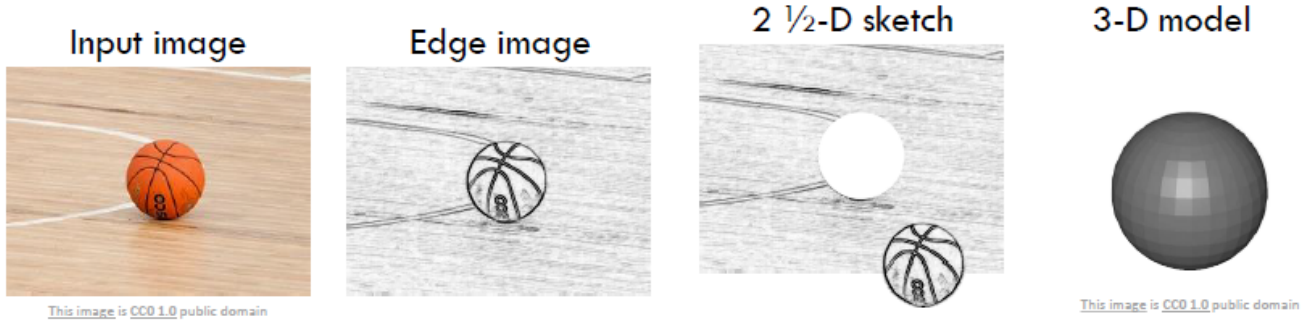


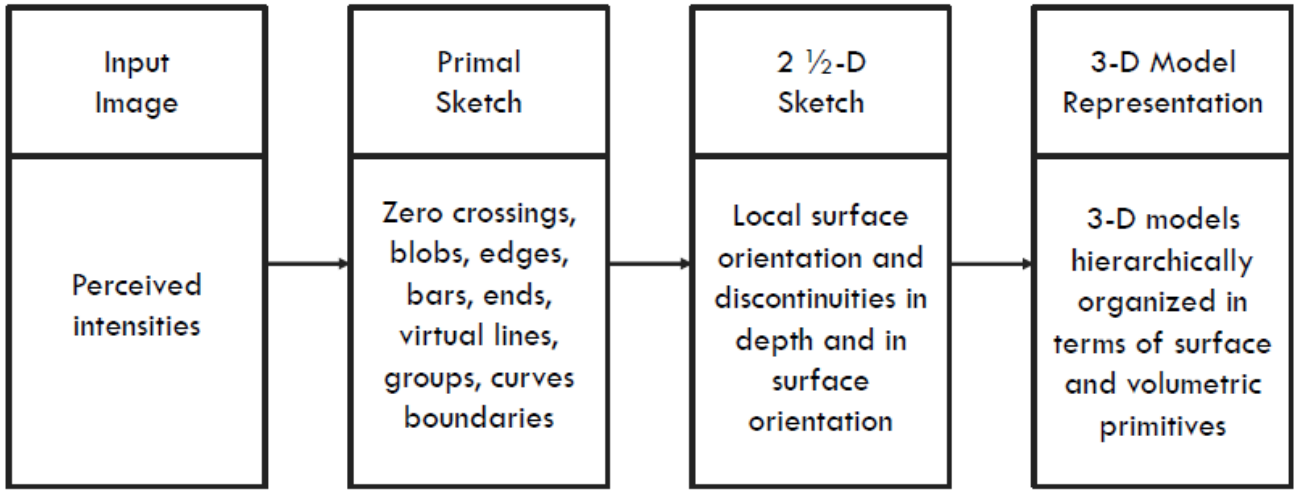
https://goodpsychology.wordpress.com/2013/03/13/235/
https://www.youtube.com/watch?v=Cw5PKV9Rj3o



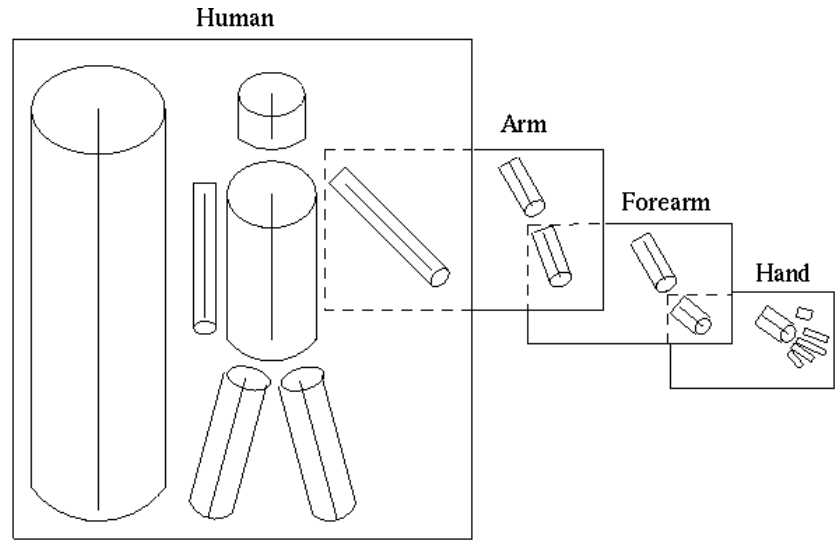
http://cns-alumni.bu.edu/~slehar/webstuff/pcave/hubel.html
https://knowingneurons.com/2014/10/29/hubel-and-wiesel-the-neural-basis-of-visual-perception/

Early computer vision models





edges, *bars*, *ends* and *blobs* were represented by a 5-tuple: (*type*, *position*, *orientation*, *scale*, *contrast*).



David Marr, 1970s

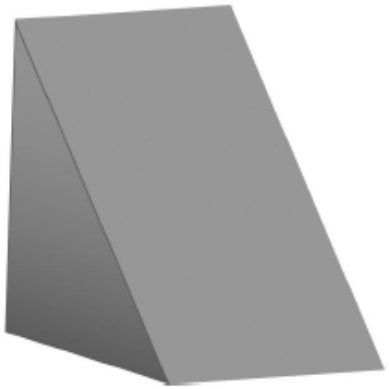
All steps are hard-coded

http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/GOMES1/marr.html

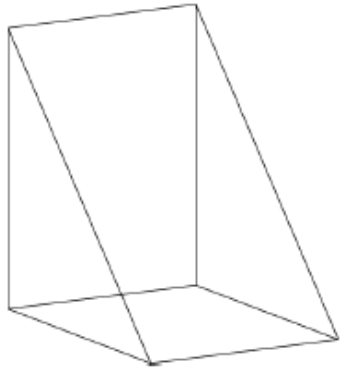
Early computer vision models

Blocks World
Lawrence Roberts
Massachusetts Institute of Technology, 1963

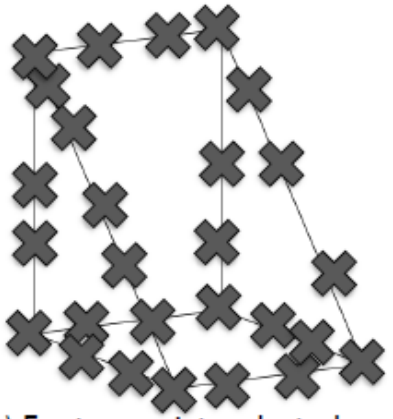
Extracting 3D information about solid objects from 2D photographs of line drawings.



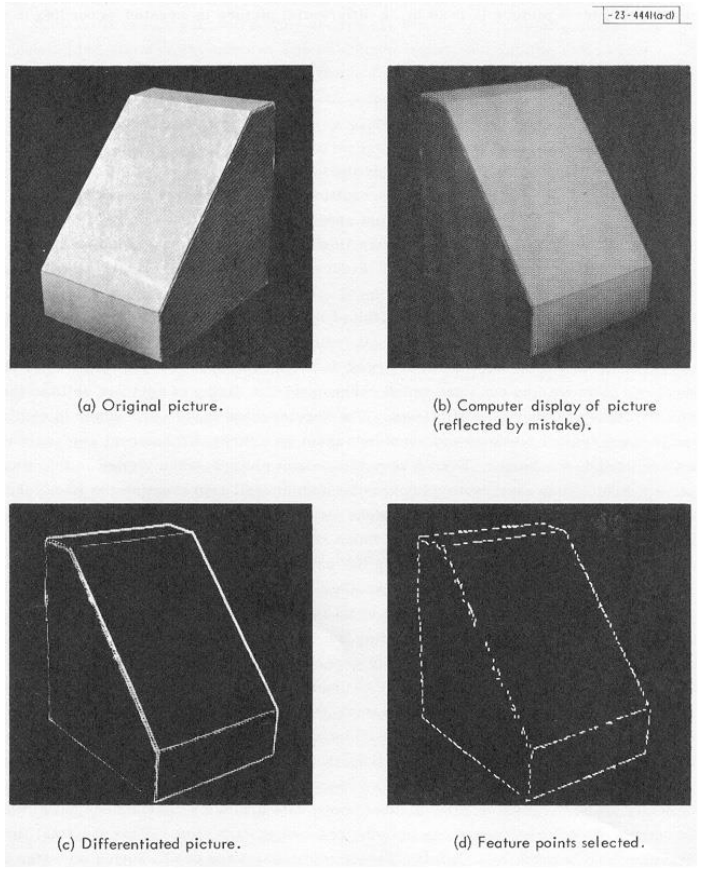
(a) Original picture



(b) Differentiated picture



(c) Feature points selected



Limited application in the real world

Challenges

Viewpoint variation

Scale variation

Deformation

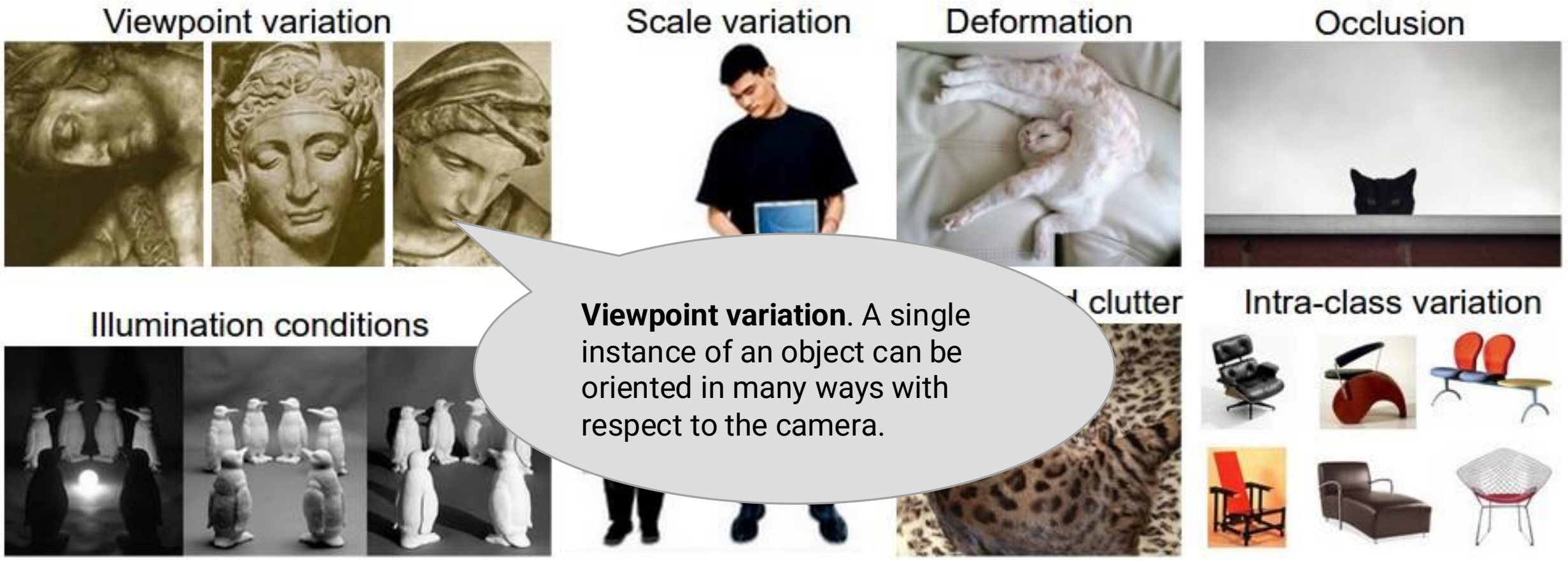
Occlusion

Illumination conditions

Background clutter

Intra-class variation

Challenges



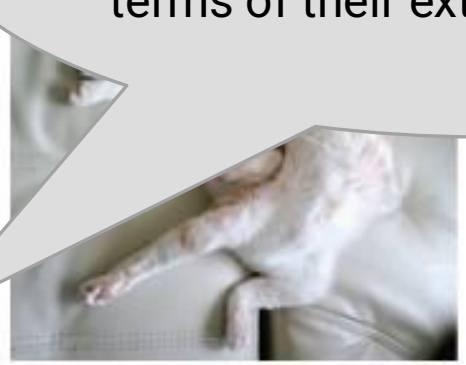
Viewpoint variation. A single instance of an object can be oriented in many ways with respect to the camera.

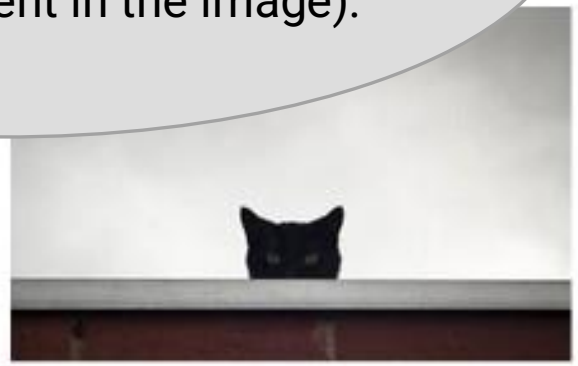
Challenges

Scale variation. Visual classes often exhibit variation in their size (size in the real world, not only in terms of their extent in the image).

Viewpoint variation

Scale variation



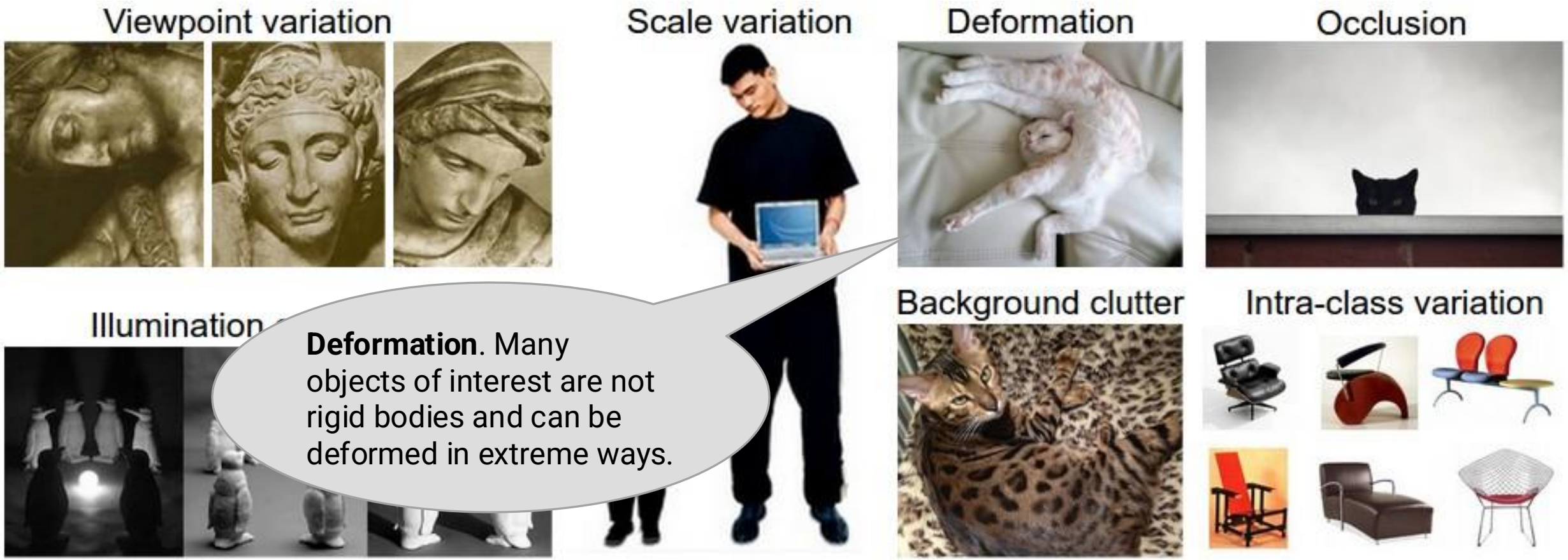


Illumination conditions

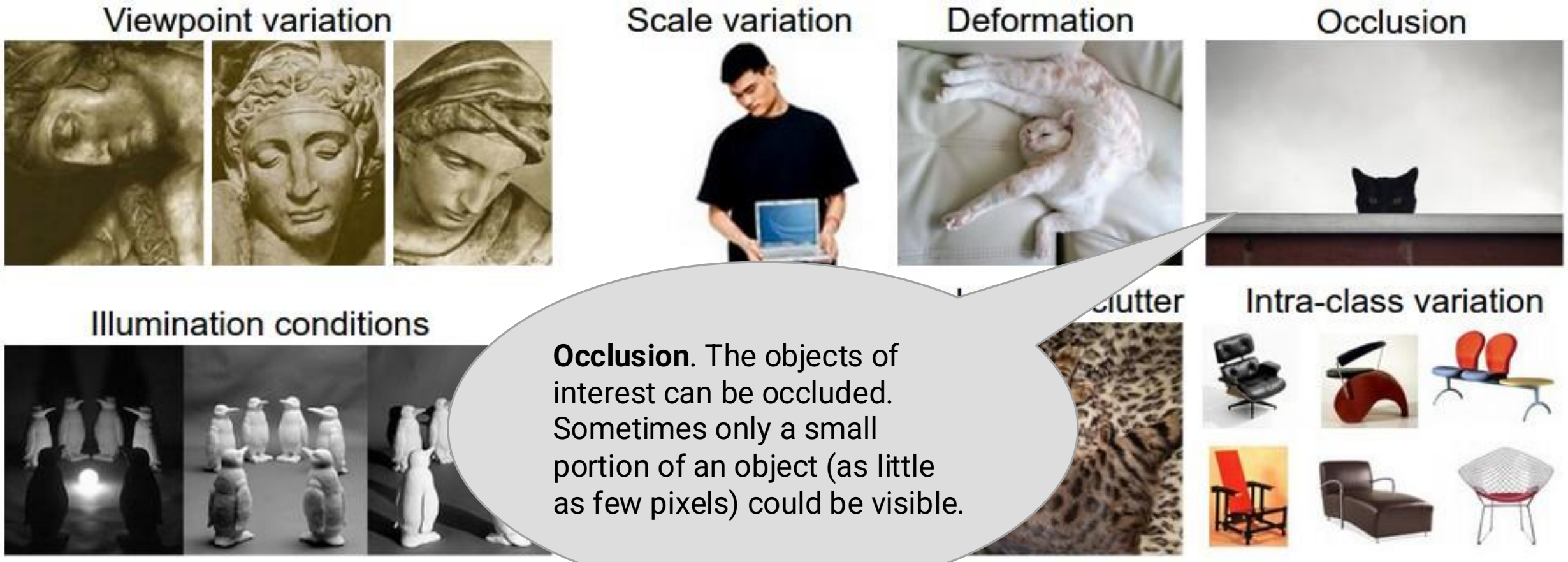
Background clutter

Intra-class variation

Challenges

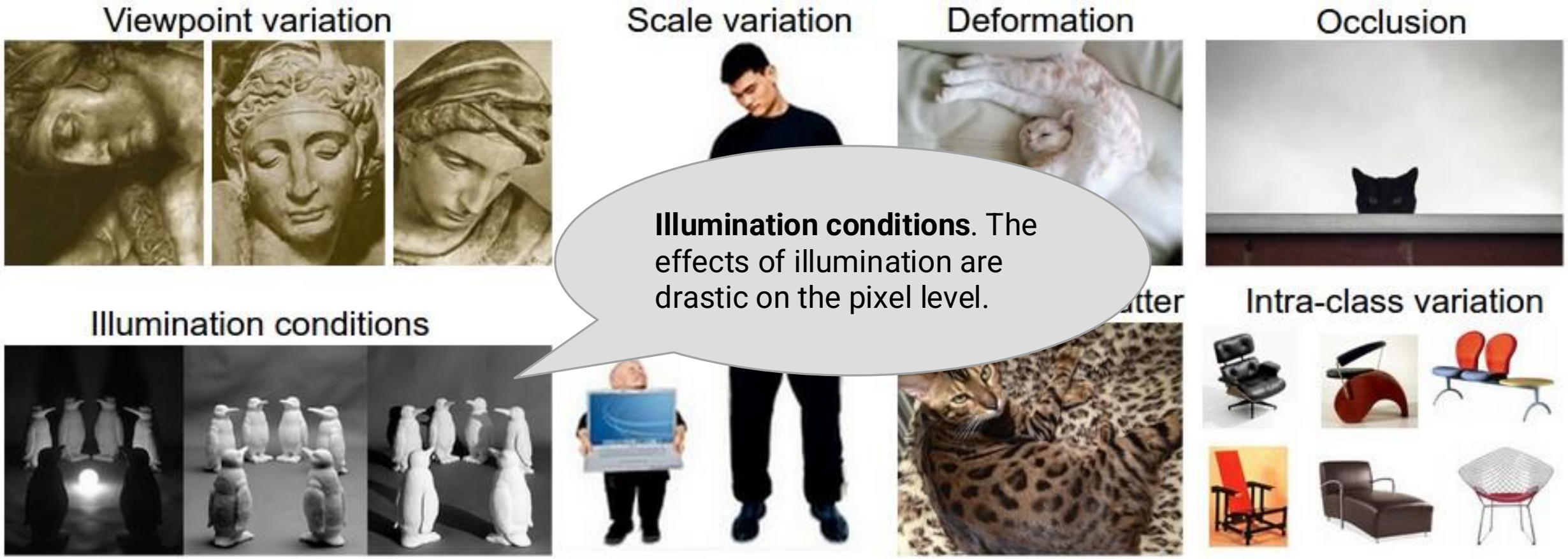

Viewpoint variation
Scale variation
Deformation
Occlusion
Illumination
Background clutter
Intra-class variation
Deformation. Many objects of interest are not rigid bodies and can be deformed in extreme ways.

Challenges

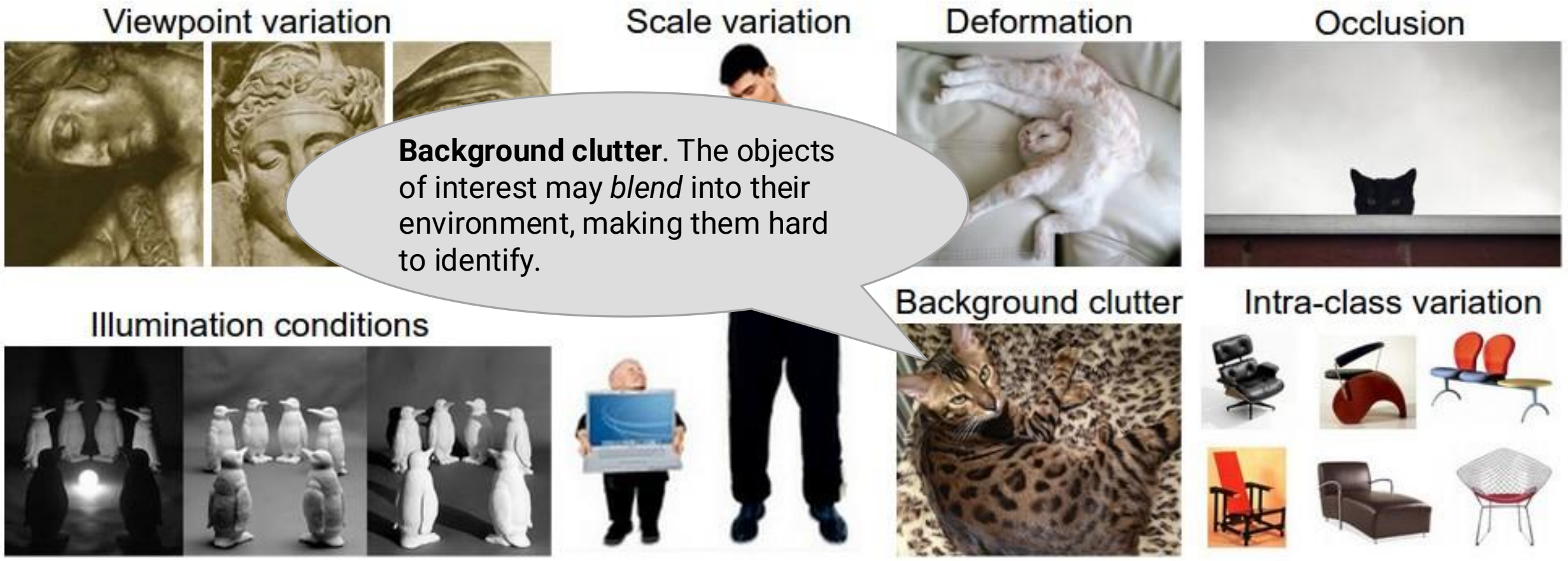


Occlusion. The objects of interest can be occluded. Sometimes only a small portion of an object (as little as few pixels) could be visible.

Challenges

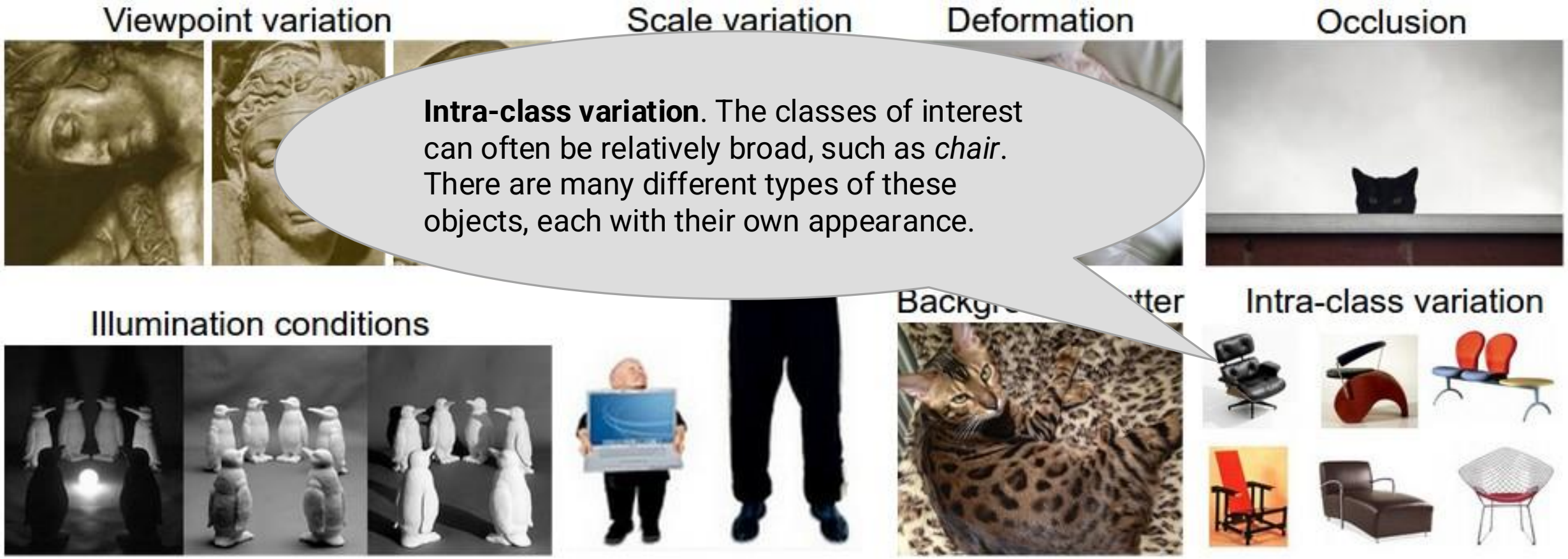

Viewpoint variation
Scale variation
Deformation
Occlusion
Illumination conditions
Intra-class variation
Illumination conditions. The effects of illumination are drastic on the pixel level.

Challenges



Background clutter. The objects of interest may *blend* into their environment, making them hard to identify.

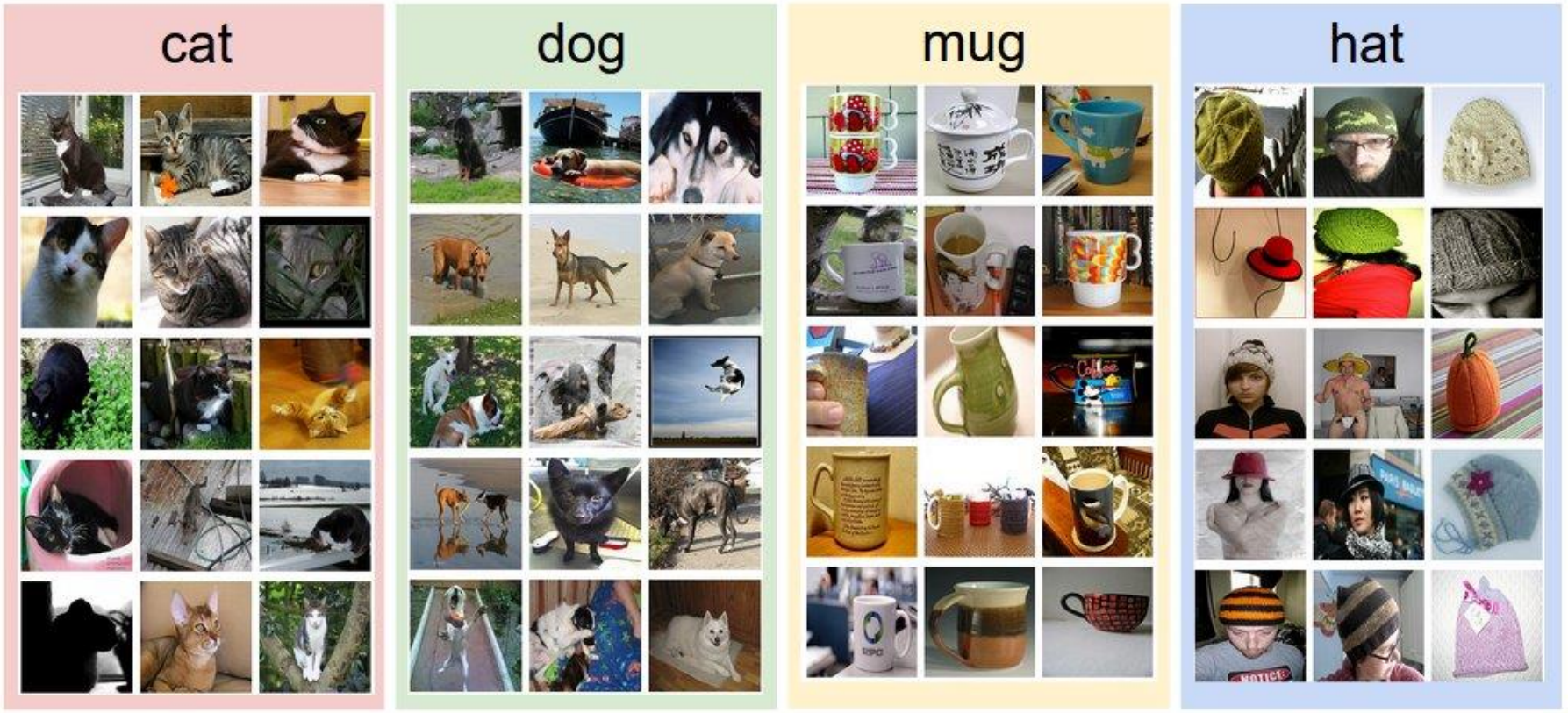
Challenges



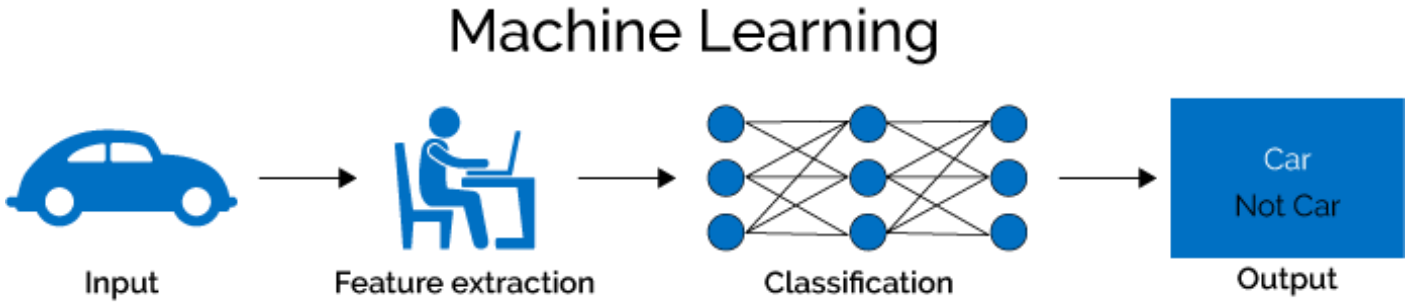
Intra-class variation. The classes of interest can often be relatively broad, such as *chair*. There are many different types of these objects, each with their own appearance.

Data-driven approach

- No obvious way to hard-code the algorithm for recognizing a cat, a dog, or anything else
- Our approach: Learn from a **training dataset** of labelled images.



Basic idea



Data-driven approach using traditional machine learning

- Extract simple features like edges and blobs from the input image
- Feed those features into a machine learning model
- Train the machine model to solve the task at hand

Note: Under the hood, the machine learning model learns to combine the simple features into more complex features (like in the brain).

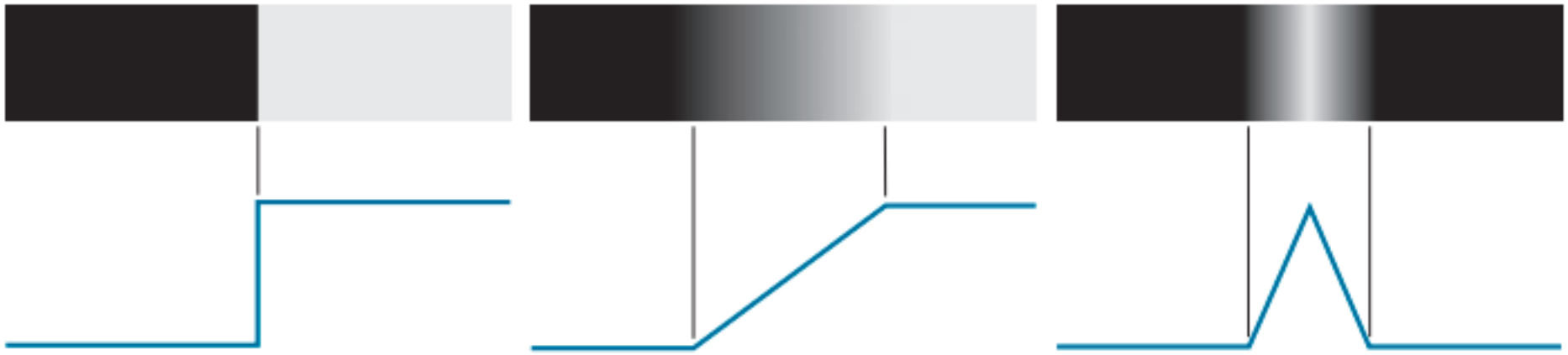
Image features

Edges = rapid changes in image intensity, color or texture

Source: https://en.wikipedia.org/wiki/Edge_detection

Image features

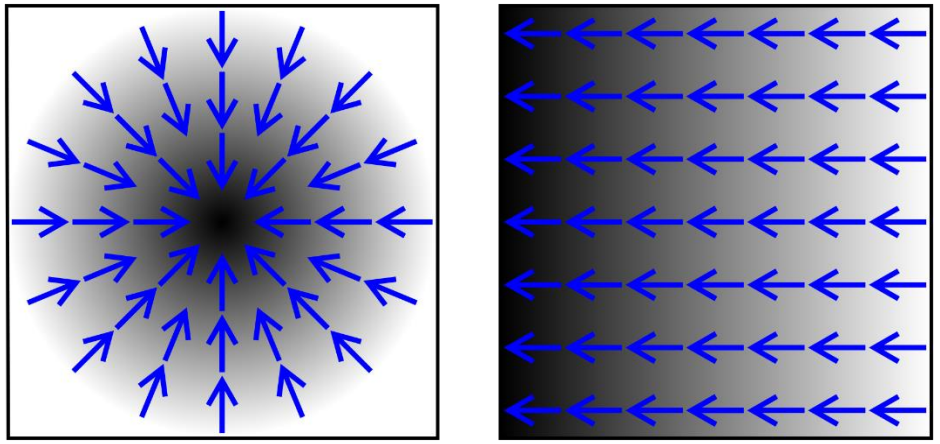
Edges = rapid changes in image intensity, color or texture



Edges can have different intensity profiles.
How do we detect them?
(Answer: convolution)

Image features

Edges = rapid changes in image intensity, color or texture

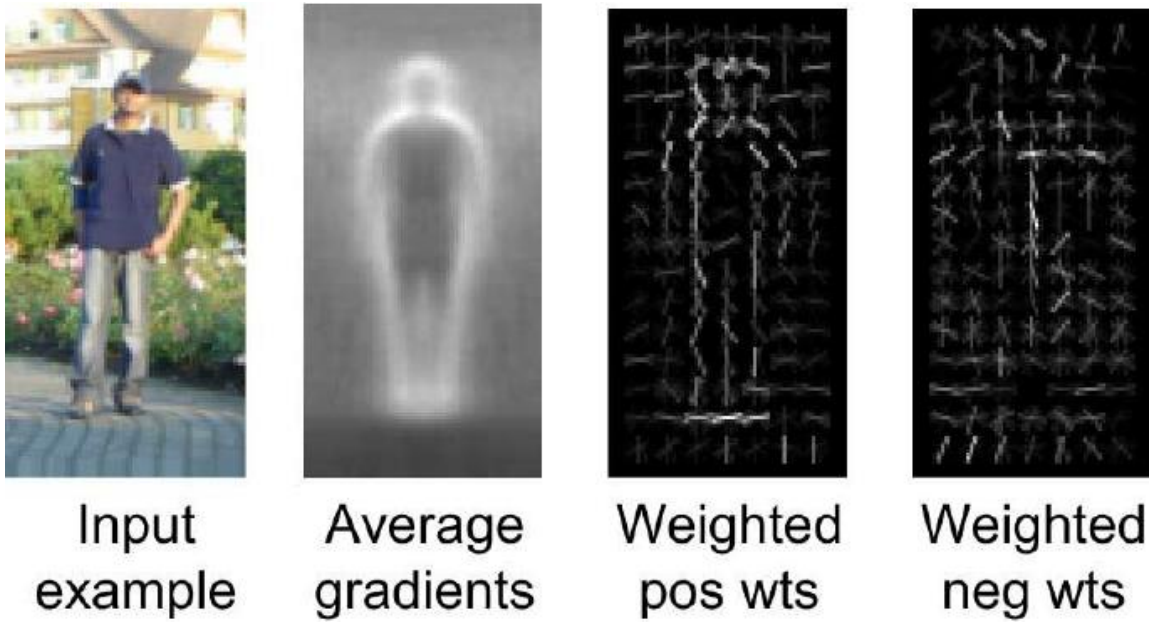


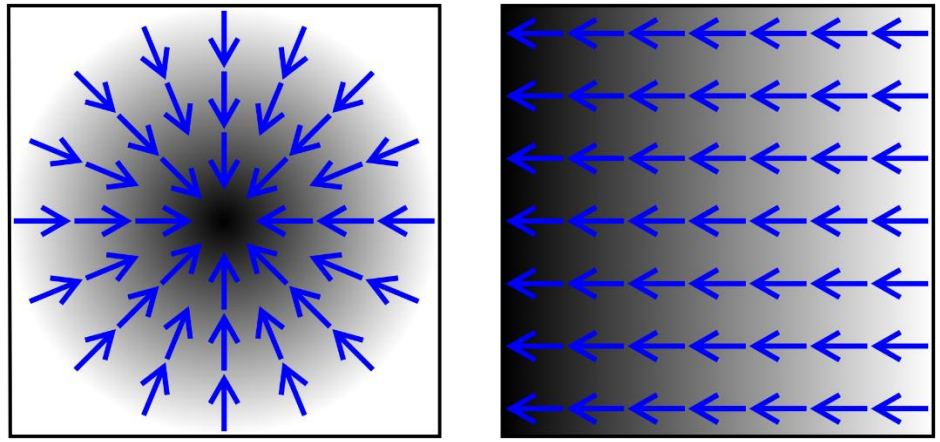
https://en.wikipedia.org/wiki/Image_gradient

Note: We can use image gradients to determine the direction of edges.

Image features

Histograms of oriented gradients (HoG)



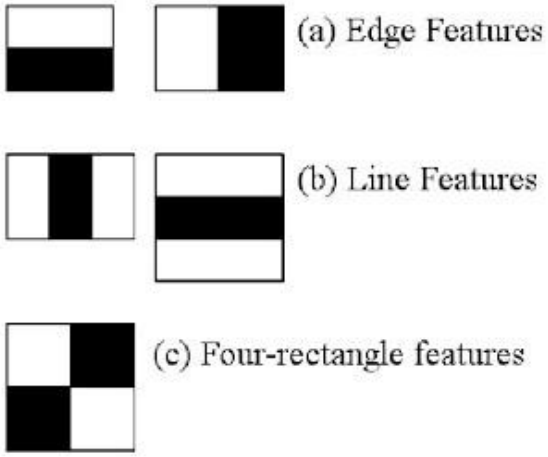


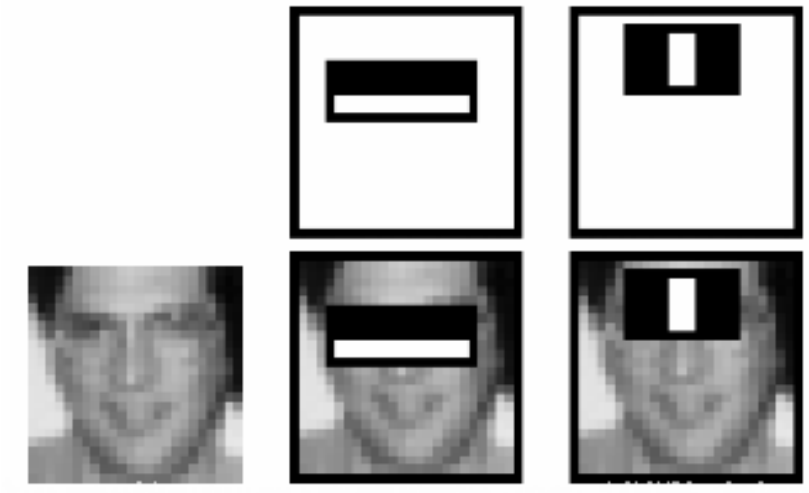
https://en.wikipedia.org/wiki/Image_gradient

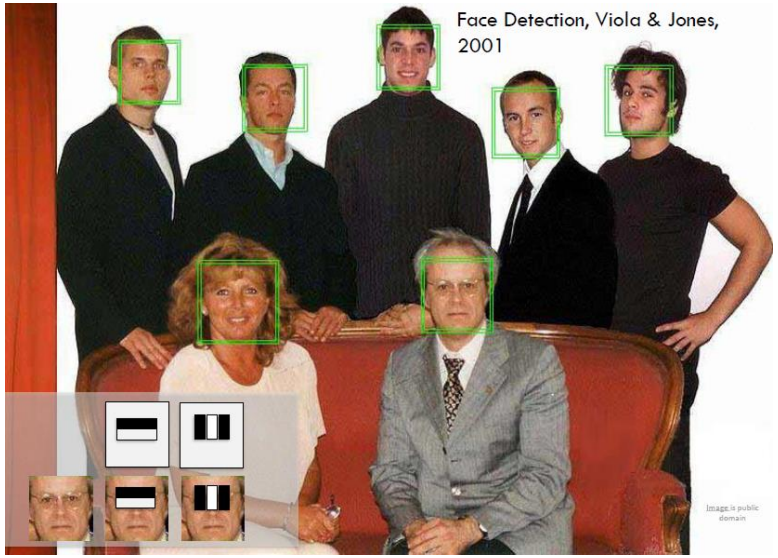
http://vision.stanford.edu/teaching/cs231b_spring1213/papers/CVPR05_DalalTriggs.pdf

Image features

More advanced structures (filters)



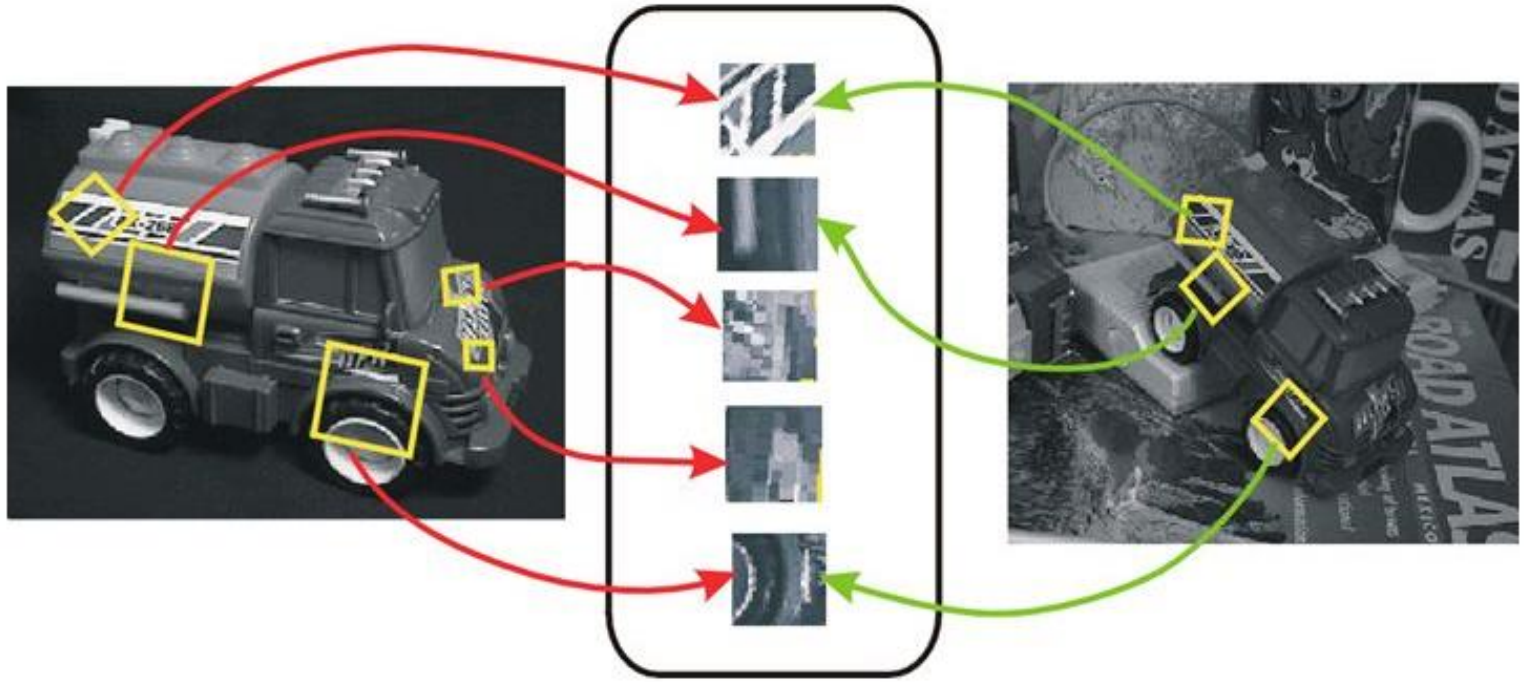




https://docs.opencv.org/3.4.1/d7/d8b/tutorial_py_face_detection.html

Image features

Interest points and descriptors (e.g., SIFT)



https://medium.com/software-incubator/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40

https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf

Image features

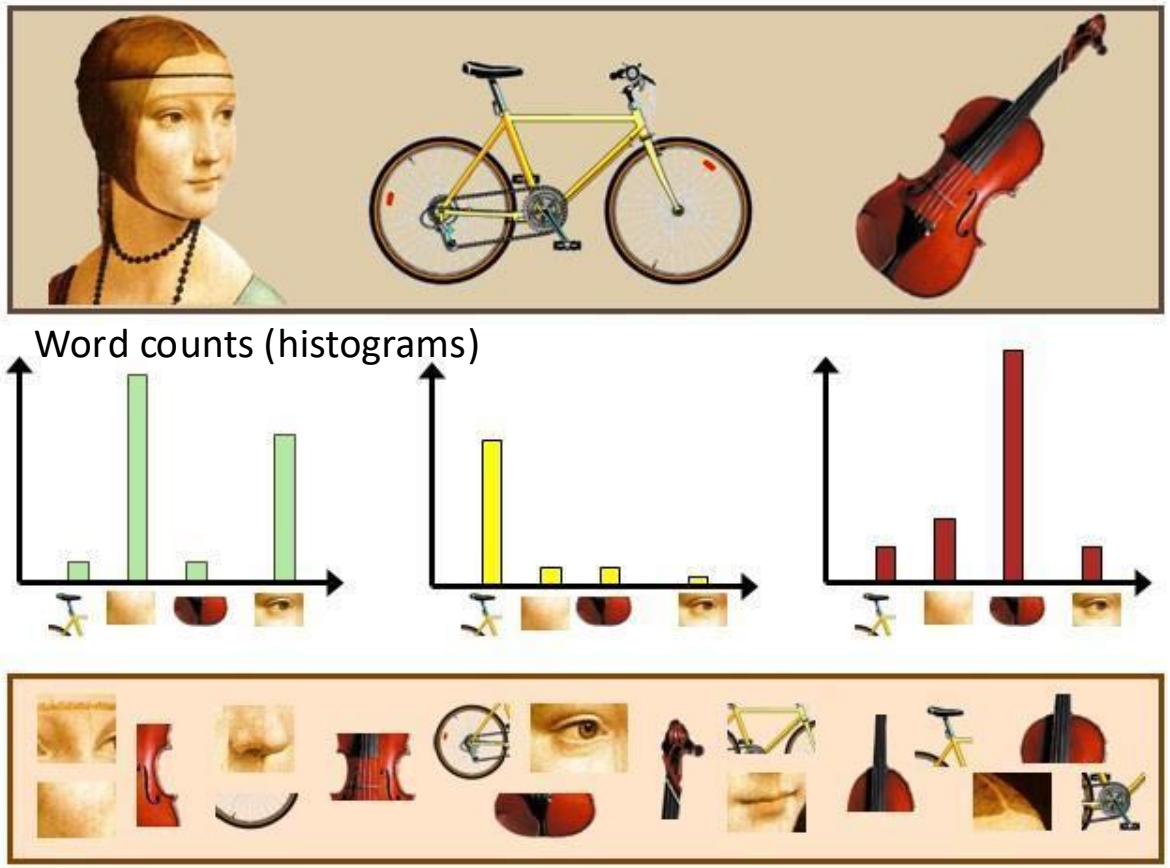
Cluster similar pixels (e.g., similar color) into so-called super pixels

Normalized Cut (Shi & Malik, 1997)

Object detection and recognition

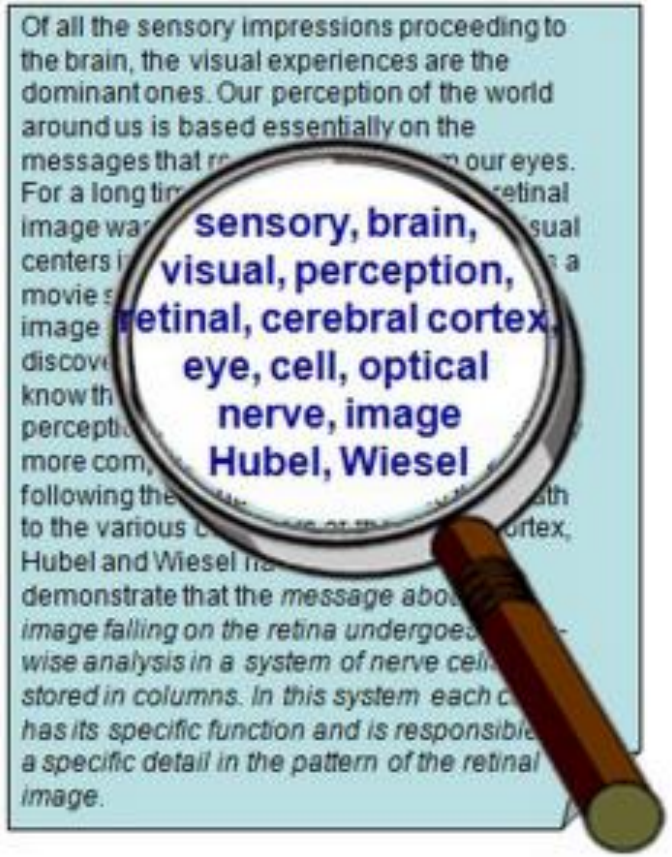
- Combine information from image features
- Two examples:
  - Bag of words
  - Part-based model

Bag of words



Word counts (histograms)

Bag of visual words (**= image features**)

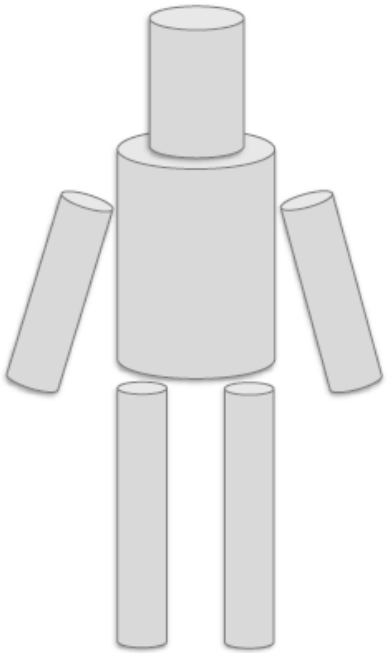


https://prateekvjoshi.com/2014/08/17/image-classification-using-bag-of-words-model/

Part-based models

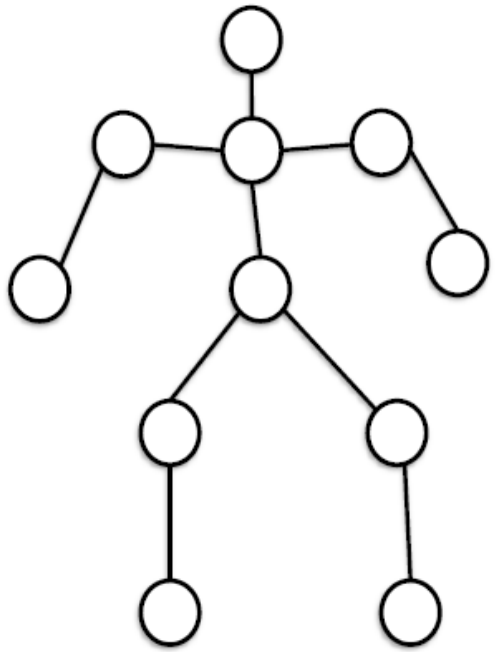
- Generalized Cylinder

Brooks & Binford, 1979



- Pictorial Structure

Fischler and Elschlager, 1973

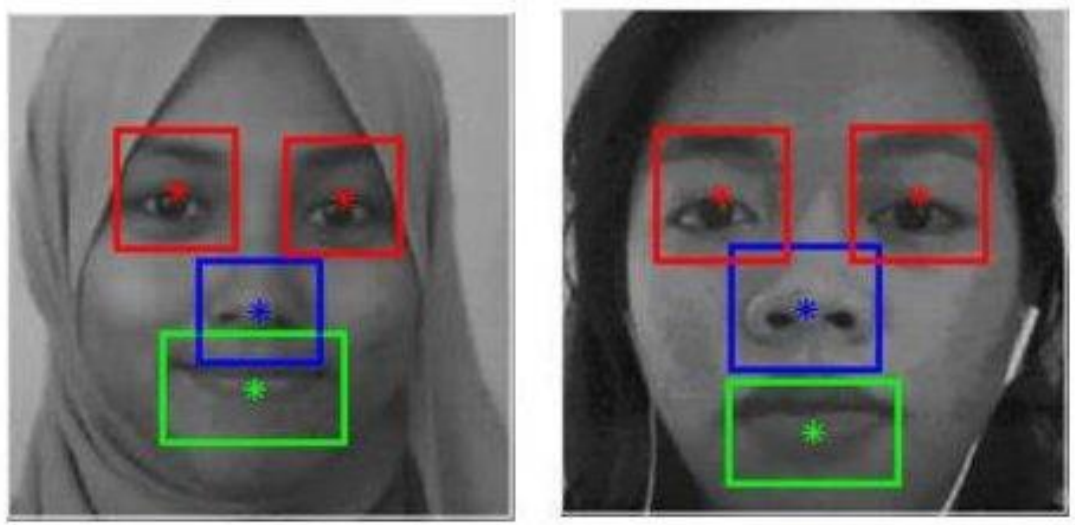


Recall how the brain works:
Look for simple features like edges and blobs, then combine into more complex features like corners.

Part-based models can be thought of as an implementation of this. Basic idea is to combine “image features” into object parts.

Object parts are then connected based on tree-like data structures (or graphs).

Part-based models



Model training:

- Learn from examples how individual parts look (e.g., eyes, nose, etc.)
- Learn from examples how parts are spatially organized (e.g., one eye on each side of the nose)

Model inference:

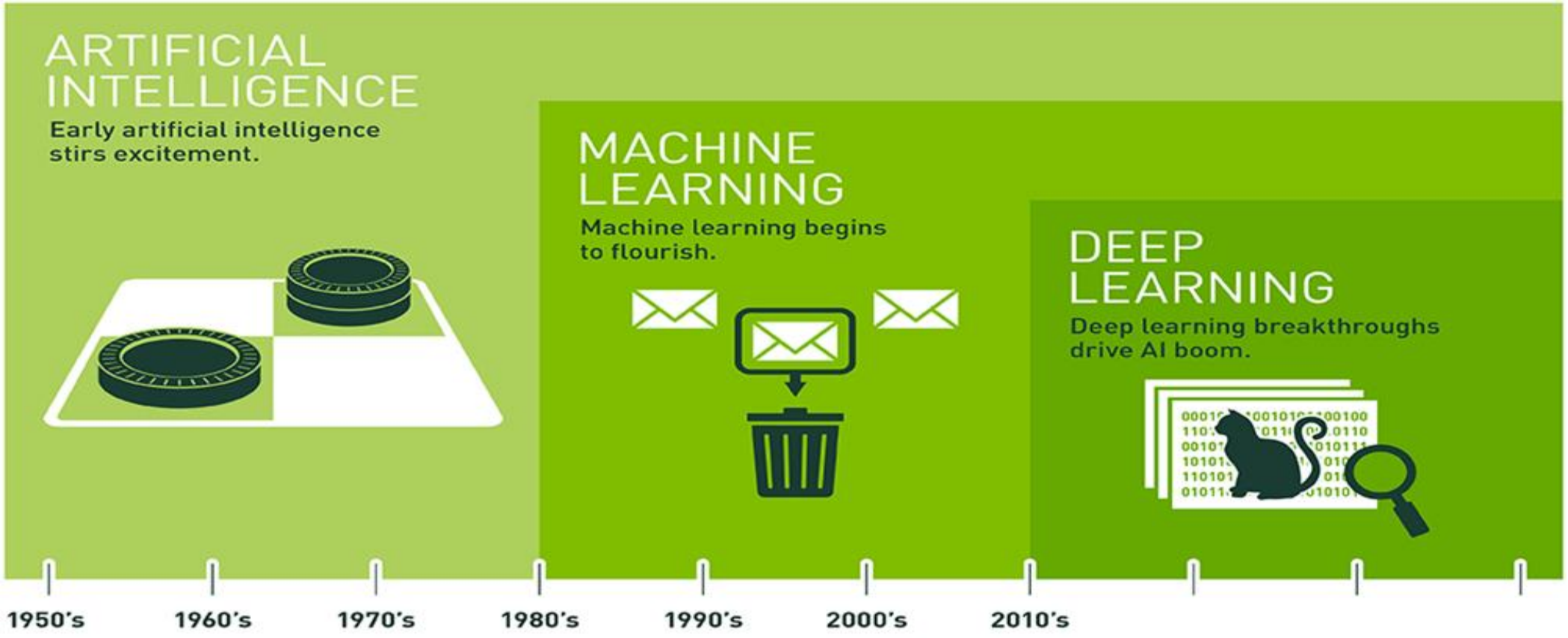
- Look for candidate parts
- Is there a spatial configuration of the candidates that is consistent with the object you are looking for?

http://rogerioferis.com/VisualRecognitionAndSearch2014/material/papers/DPMCVPR98.pdf

Deep Learning

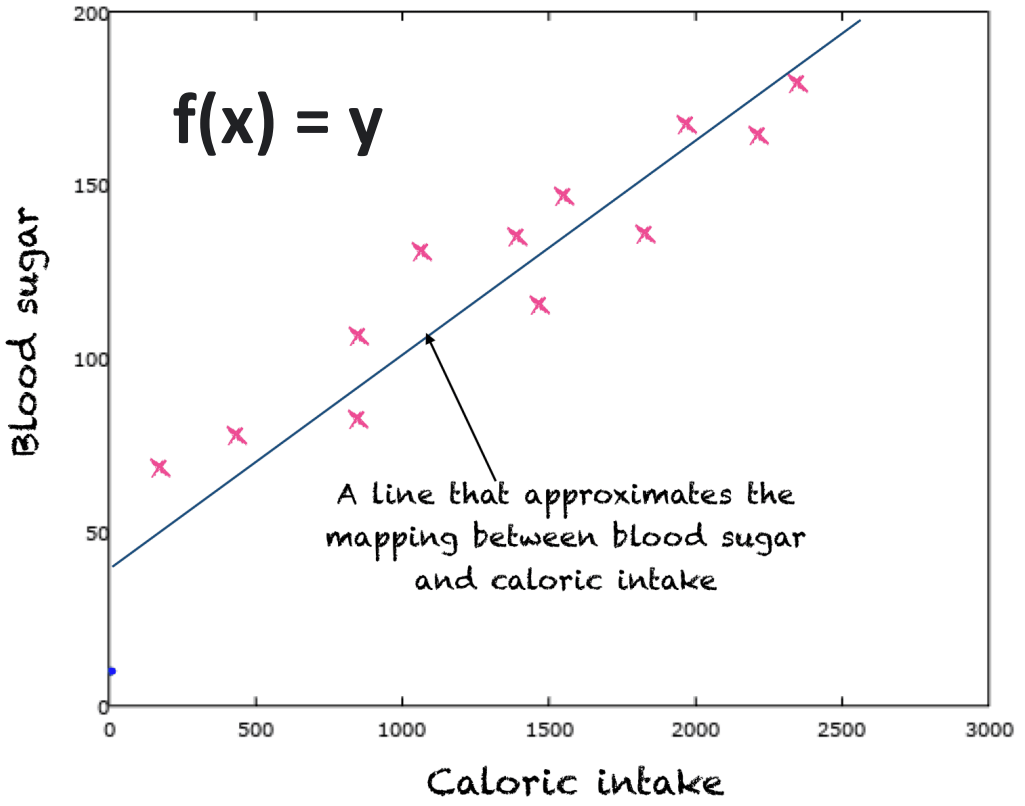
INTRODUCTION

Definitions



Machine Learning in a nutshell

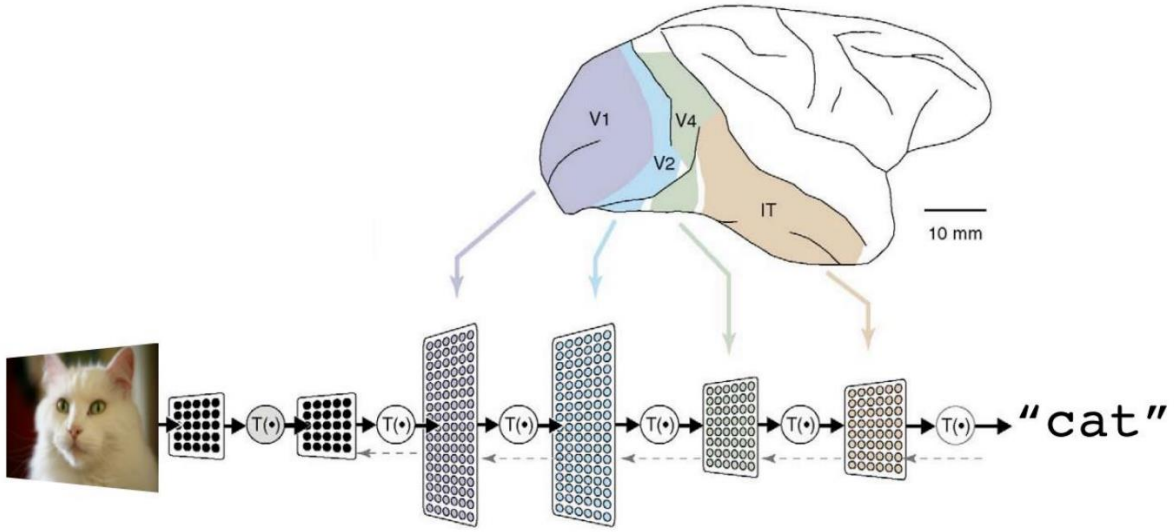
- Functions describe the world – everything you can think of is described by functions!
- A function is just a system of inputs (**x**) and outputs (**y**).
- If you know the function **f**, you can always calculate the correct output **y**, given any input **x**.
- Say we know some of the inputs (**x**) and outputs (**y**), but we don't know the function used to produce them.
- Is there a way to "reverse engineer" that function?
- If we could construct such a function, we could use it to calculate a y-value, given an x-value that is not in our original dataset.
- What we need is a function approximation or more generally, a function approximator.
- Fundamentally, machine learning models are function approximators.
- Neural networks are universal function approximators (i.e., they can approximate any function).



https://machinelearningmastery.com/a-gentle-introduction-to-approximation

What is deep learning?

- Neural networks are machine learning algorithms inspired by the structure and function of the brain.
- They have been around since the 1970s, but the deep learning revolution started around 10 years ago mainly due to **Big data** (huge amounts of *labelled* data!) **+ GPUs** (Graphics Processing Units).



A deep neural network consists of a hierarchy of layers, whereby each layer transforms the input data into more abstract representations (e.g. edge -> nose -> face). The output layer combines those features to make predictions.

Convolutional Neural Networks

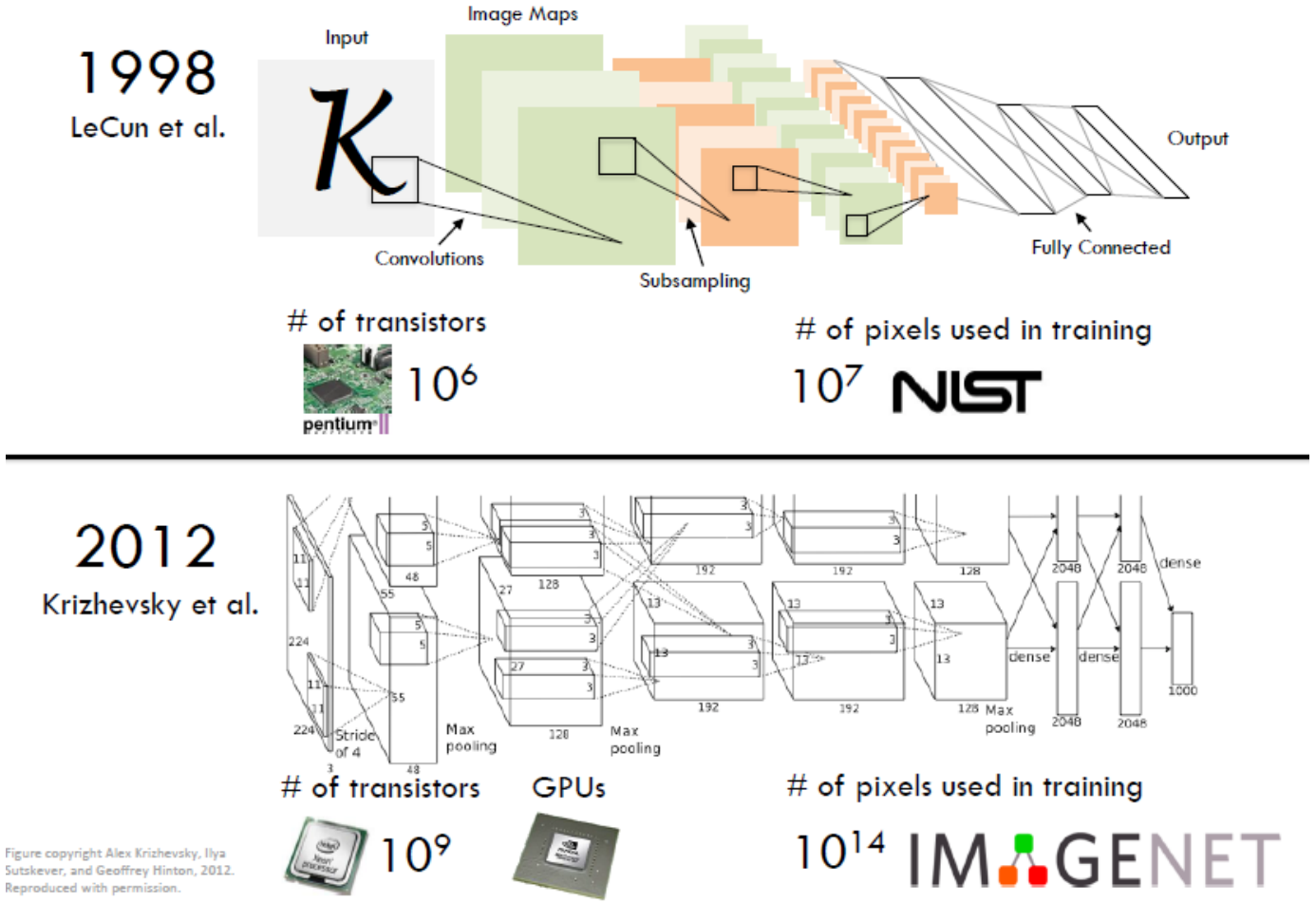
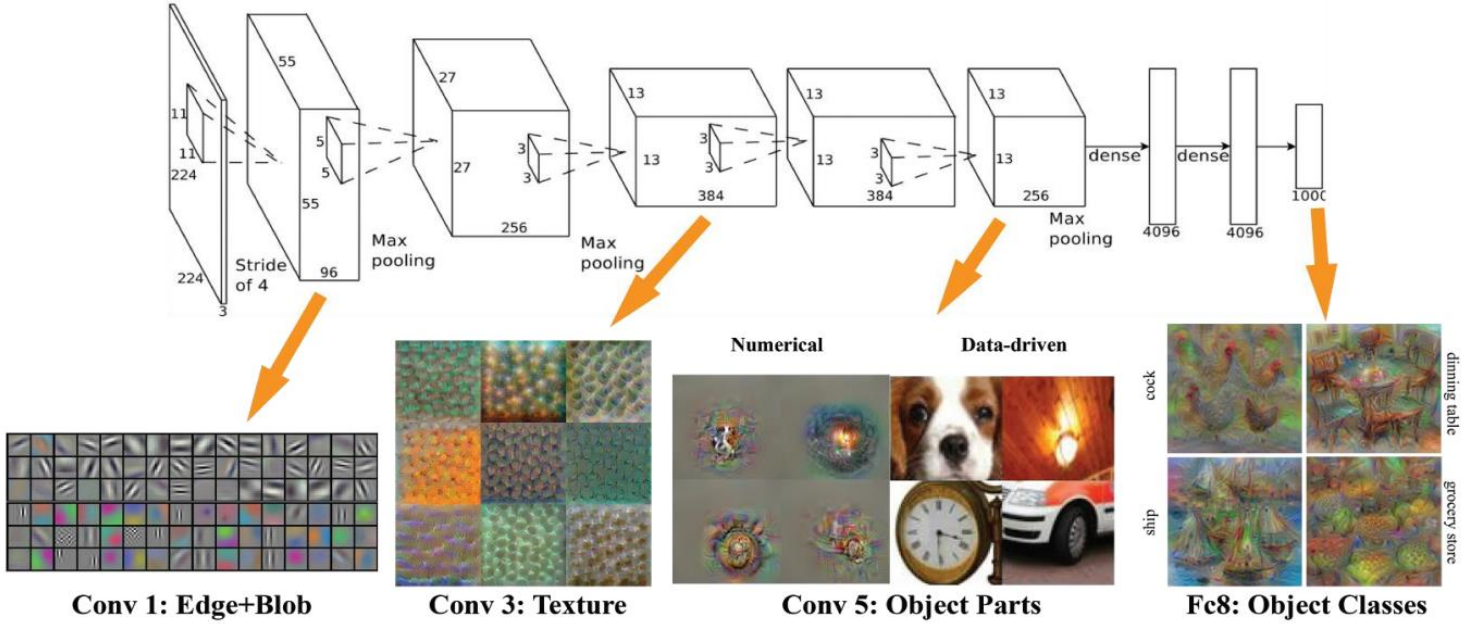


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf

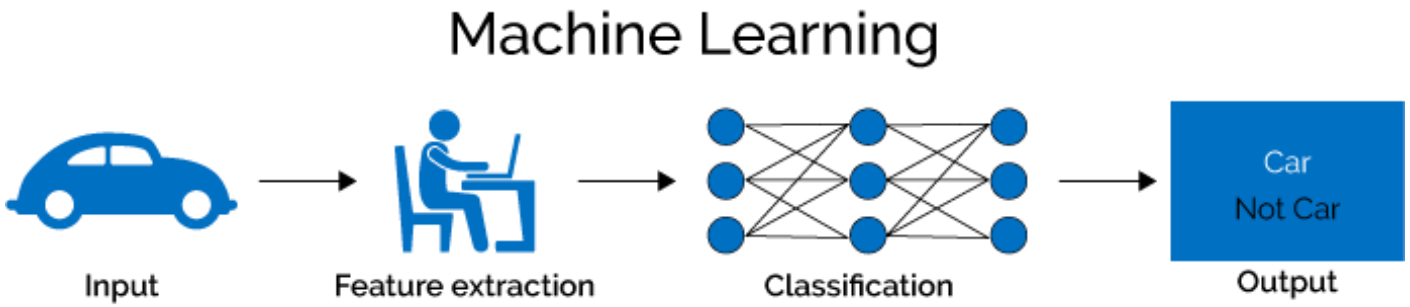
The Big Bang of Deep Learning

- AlexNet is the name of a specific neural network architecture that won the ImageNet Large Scale Visual Recognition Challenge in 2012.
- The network achieved a top-5 error of 15.3%, more than 10.8 percentage points ahead of the runner up.



http://www.image-net.org/papers/imagenet_cvpr09.pdf

Why is deep learning so powerful?



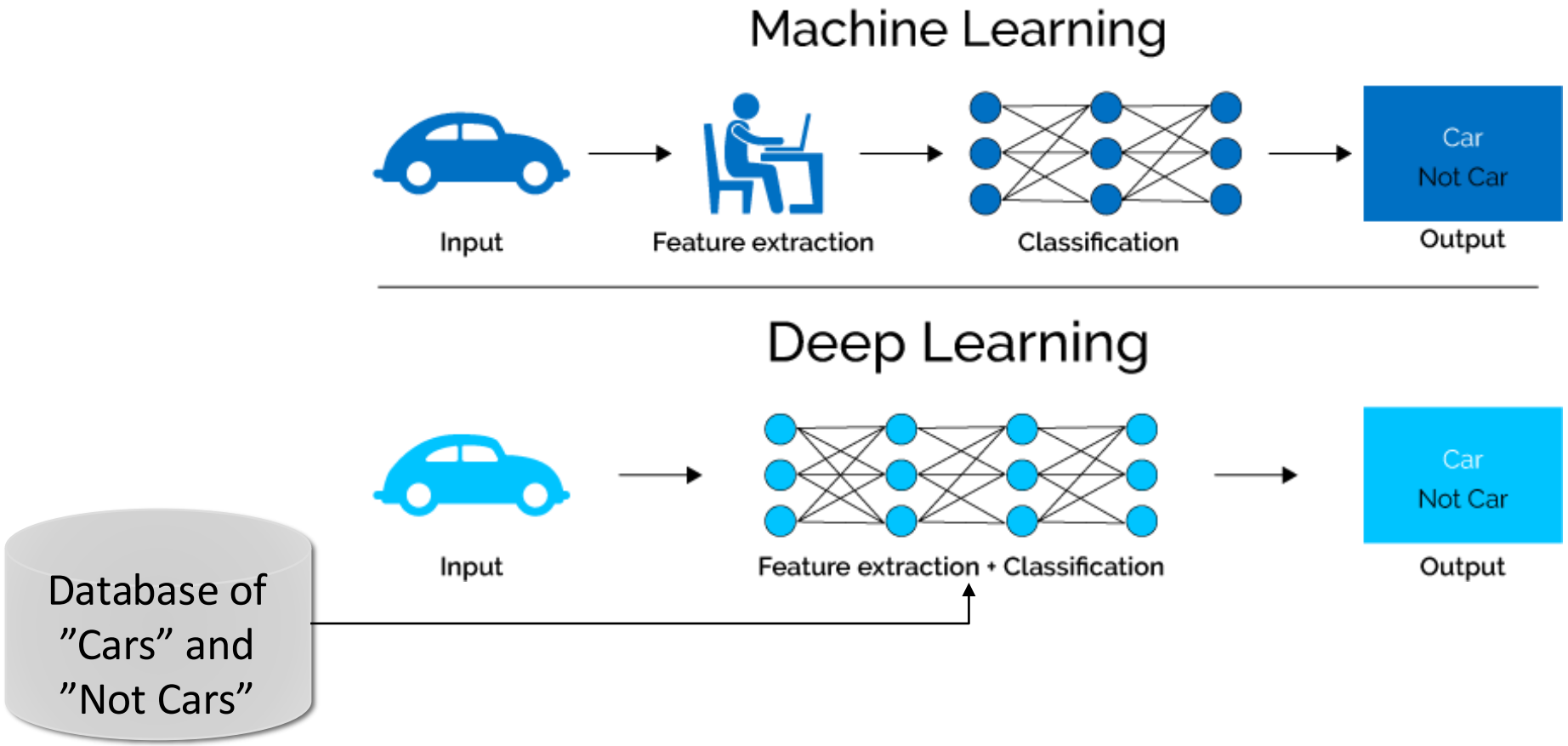
Expert-written computer program

Traditional Approach

- Requires domain experts
- Time consuming
- Error prone
- Not scalable to new problems

Why is deep learning so powerful?

Solve complex problems without domain knowledge – you just need data!



SOFTWARE 2.0

Andrej Karpathy
Director of AI at Tesla

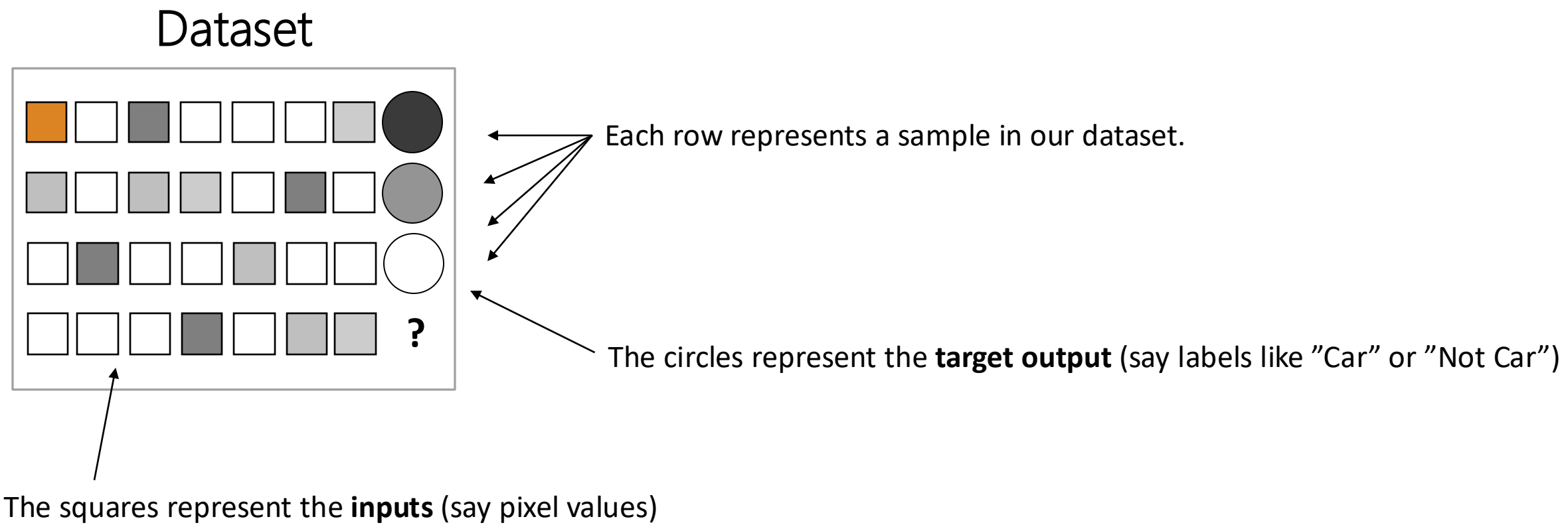
Medium
Nov 11, 2017
Software 2.0

Neural networks *are not just another classifier, they represent the beginning of a fundamental shift in how we write software. They are Software 2.0.*

The "classical stack" of ***Software 1.0*** *is what we're all familiar with […] It consists of explicit instructions to the computer written by a programmer.*

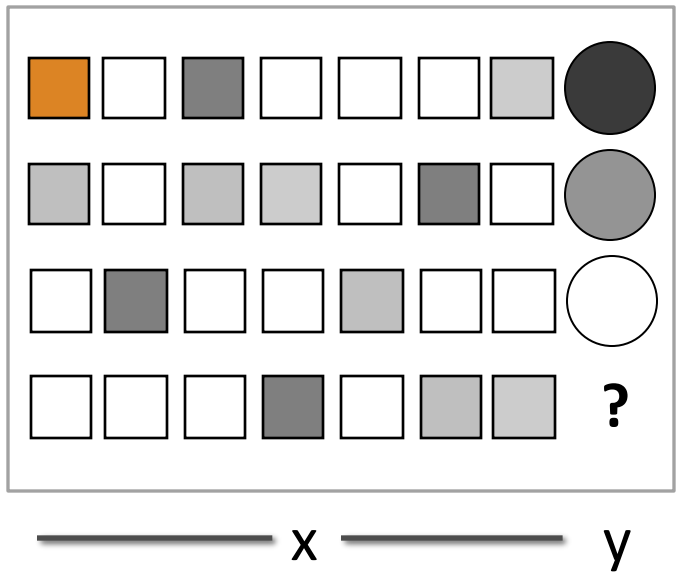
In contrast, ***Software 2.0*** *is written in neural network weights. No human is involved in writing this code […] Instead, we specify some constraints on the behavior of a desirable program (e.g., a dataset of input output pairs of examples) and use the computational resources at our disposal to search the program space for a program that satisfies the constraints.*

Learning principle



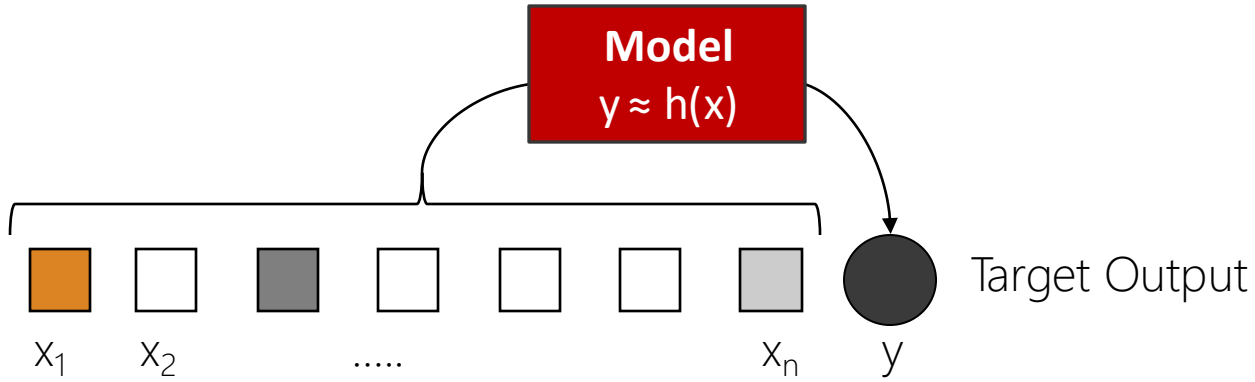
Learning principle

Dataset

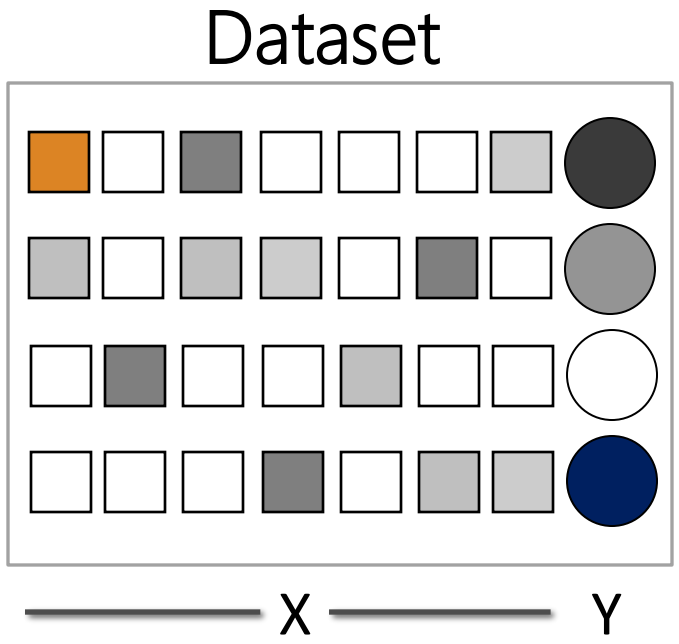


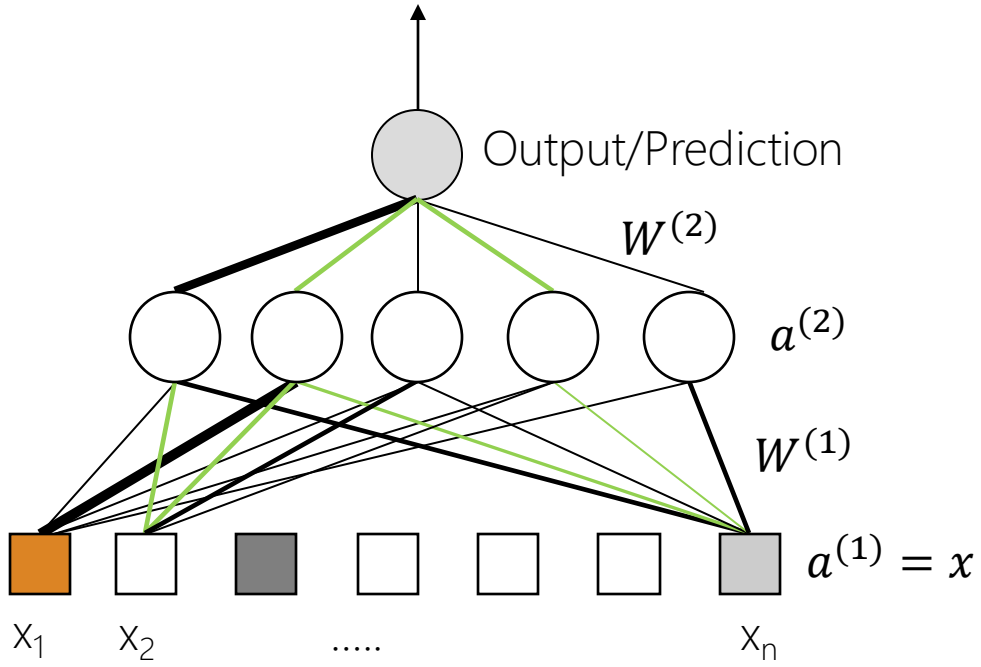
Goal:

- Learn a **model** that is good at predicting **y** from **x** across the entire training dataset
- A good/useful model generalizes to unseen data



Learning principle



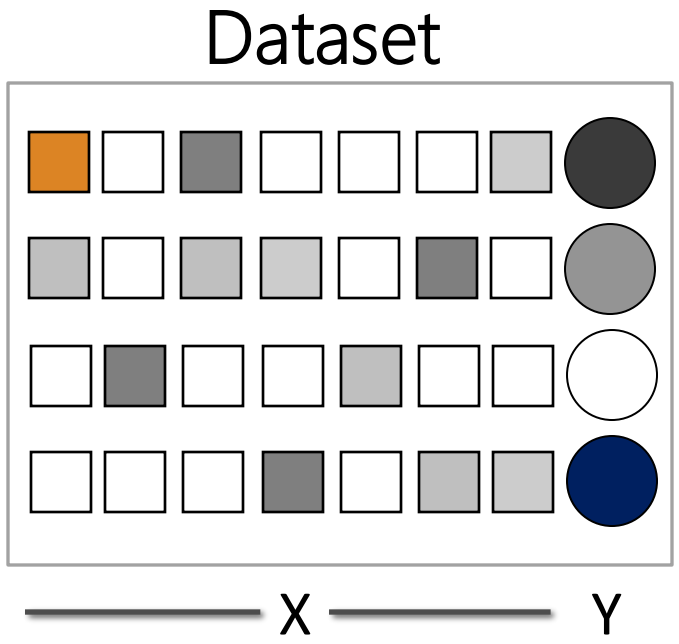


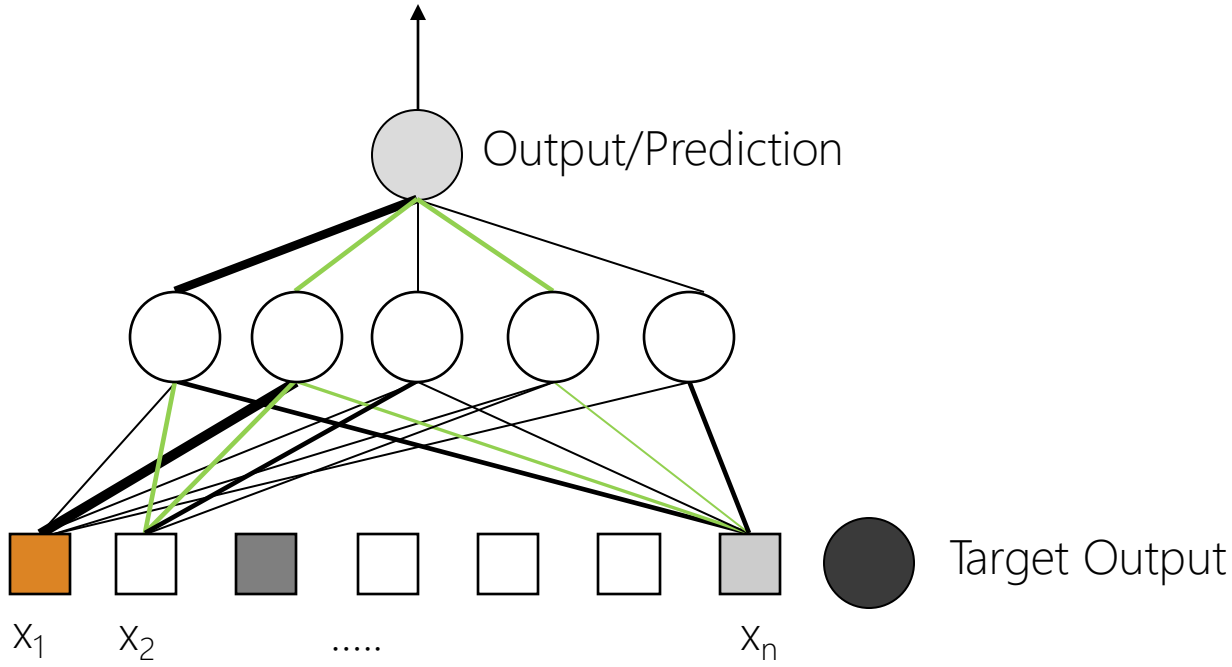
Neural network:

- White circles are called **neurons**.
- Output/Prediction

$$a^{(1)} = x$$
$$z^{(j+1)} = W^{(j)}a^{(j)} + b^{(j)}$$
$$a^{(j+1)} = \sigma\left(z^{(j+1)}\right)$$

Learning principle

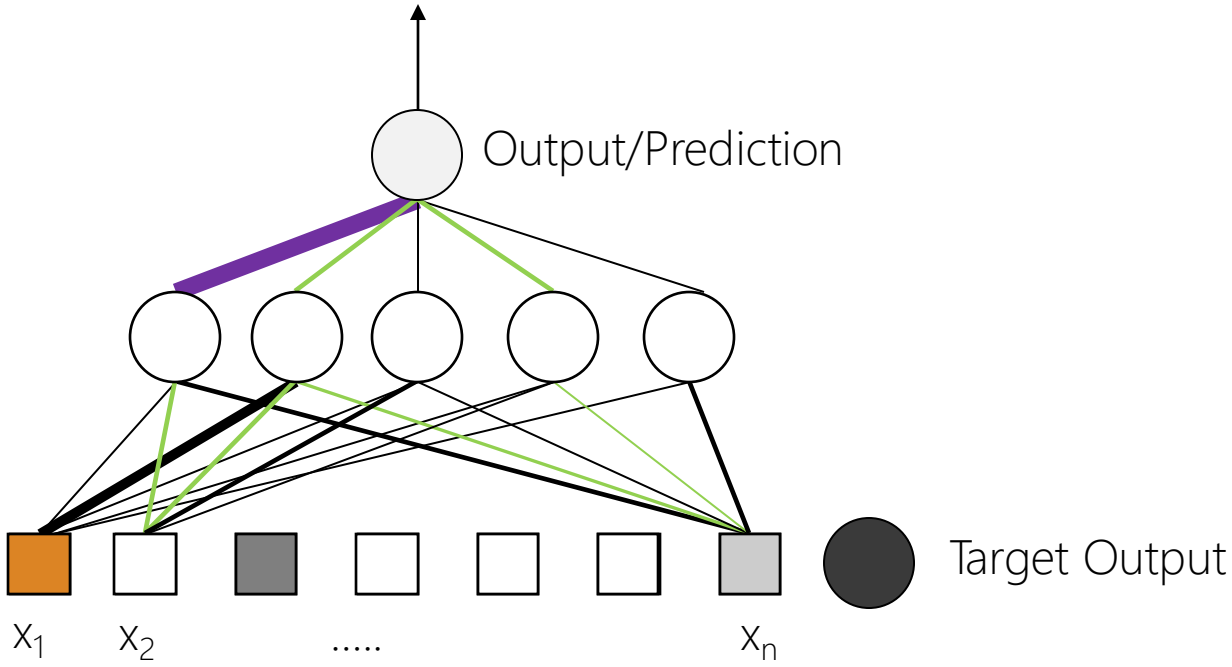


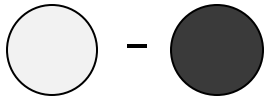


Error: ○ − ● = 5

Learning principle

Adjusting the weights either increases or decreases the error.

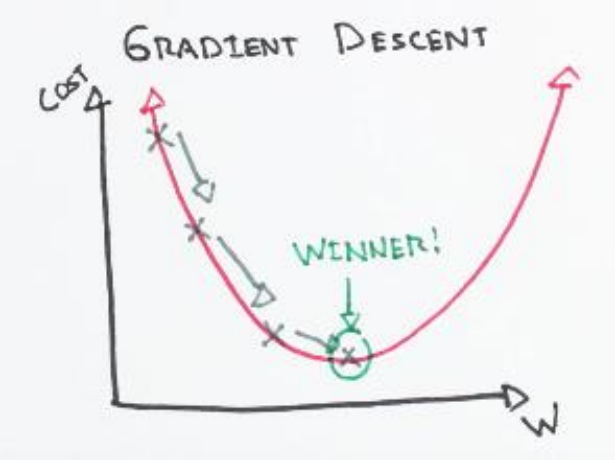


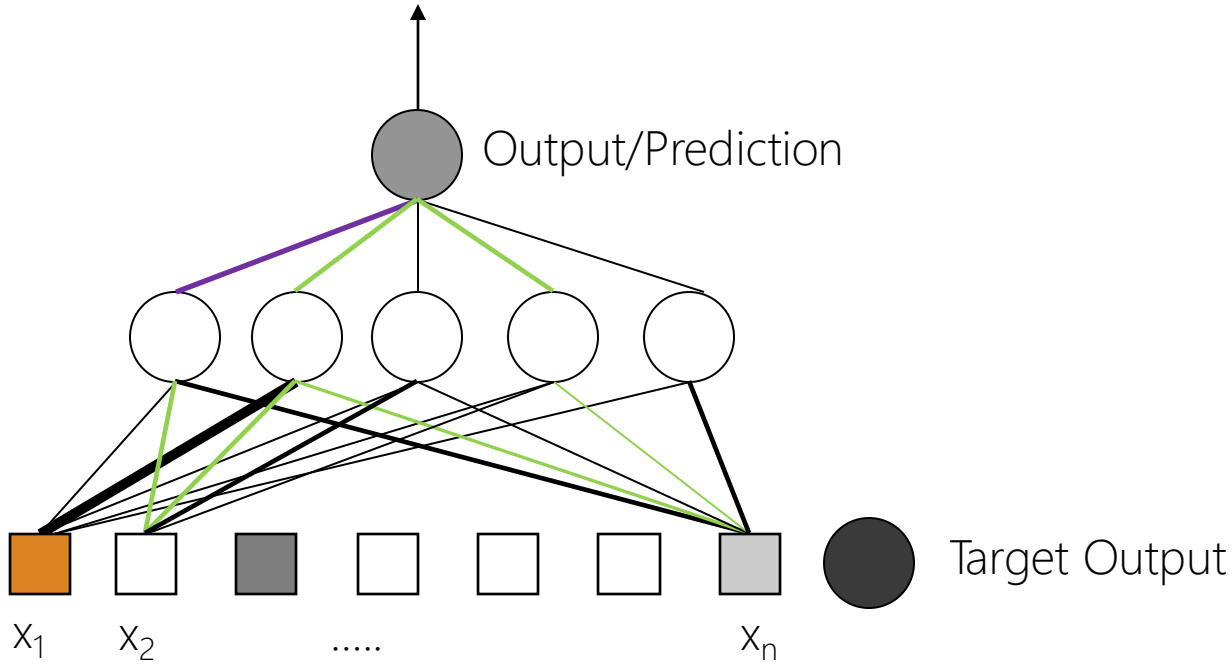
Error:  = 15

Learning principle

Training is the process of adjustning the weights such that the error is small for all samples in your dataset.

Algorithm: Gradient descent

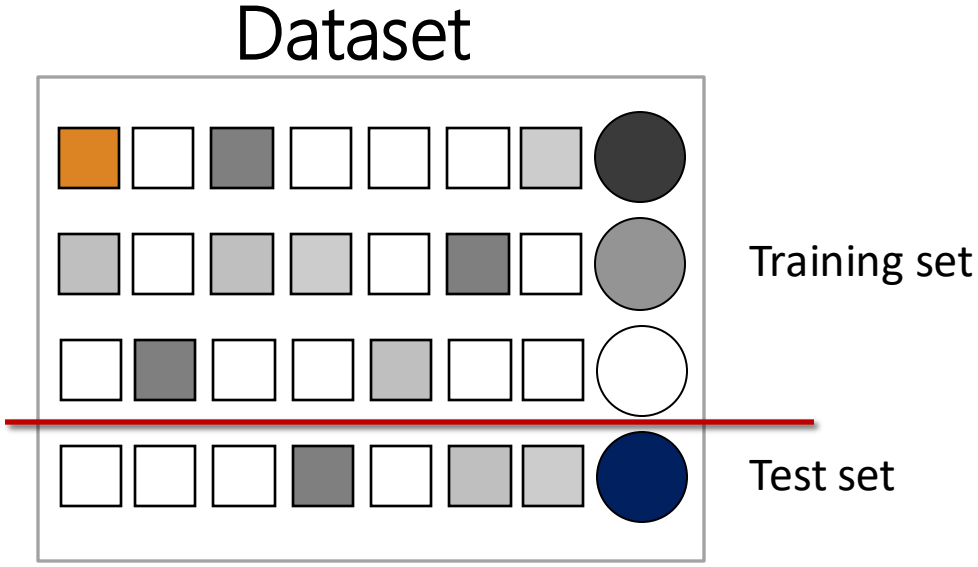


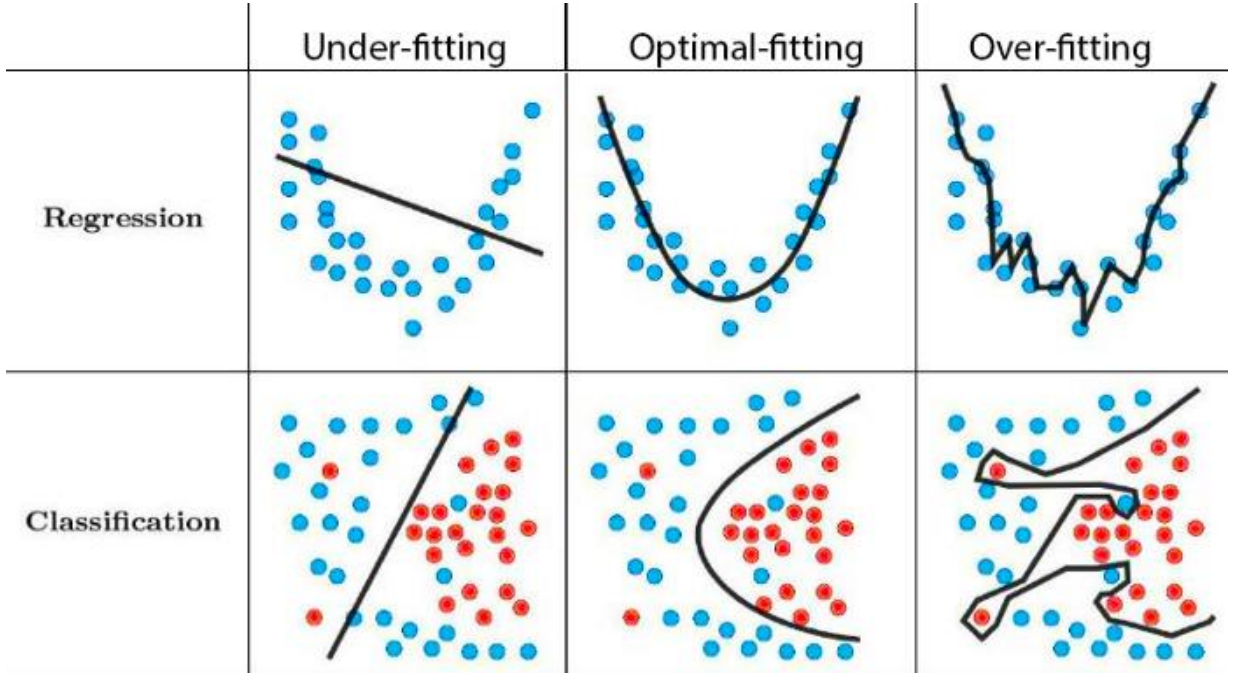


Error: ● − ● = 2.5

Evaluating model performance

- Neural networks are incredibly good at memorizing the training data, in which case our model most likely wont generalize to unseen data. This is called **overfitting**.
- To evaluate model performance (i.e., how well does it generalize to unseen data) and to check for overfitting, it is standard practise to train and test your model on separate subsets of the data.

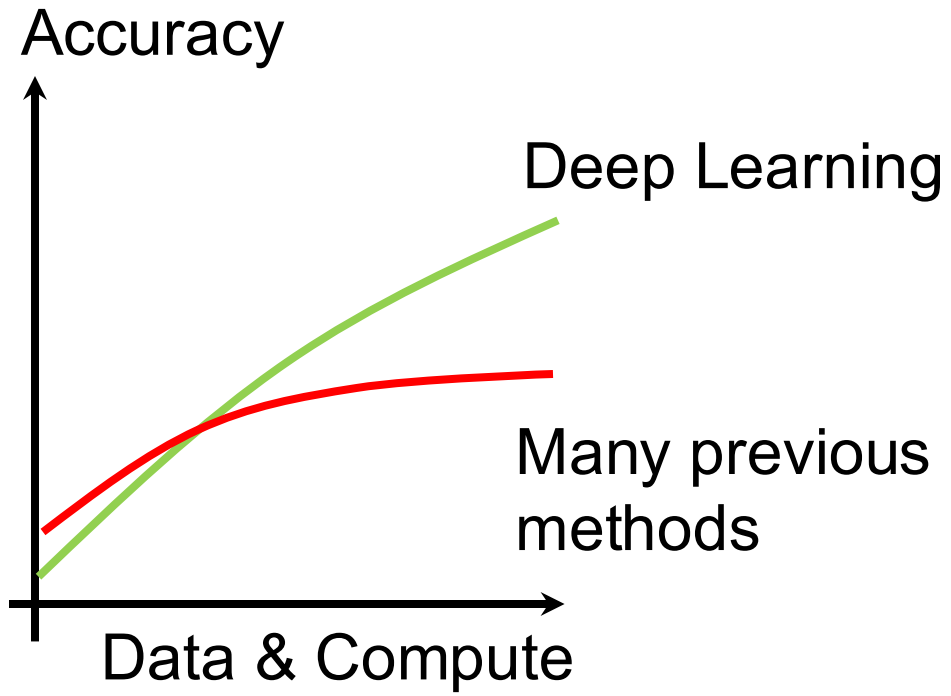




HOW DOES IT WORK?

Feature extraction using convolution

- The traditional approach of using hard-coded features + machine learning doesn't scale well.
- What separates deep learning from other machine learning techniques is that it is able to extract learnable features.
- How?



Feature extraction using convolution

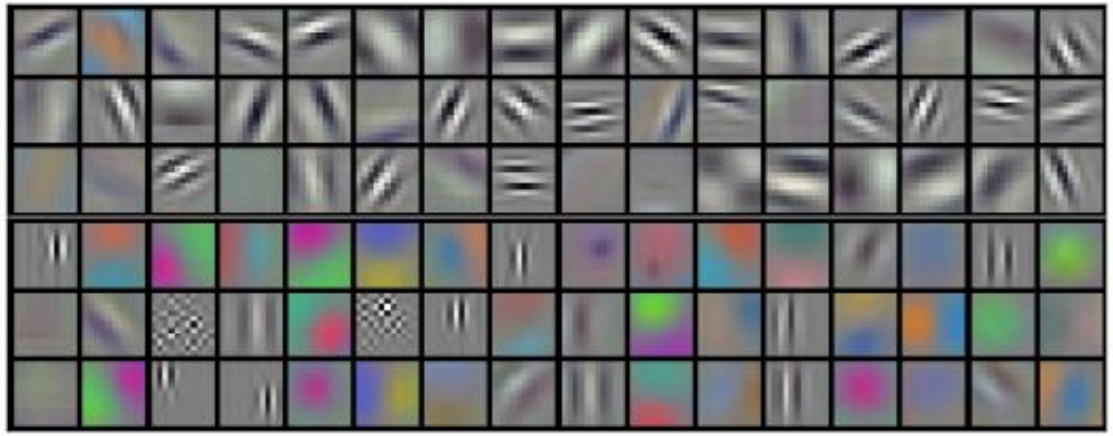
Input image

Convolution Kernel

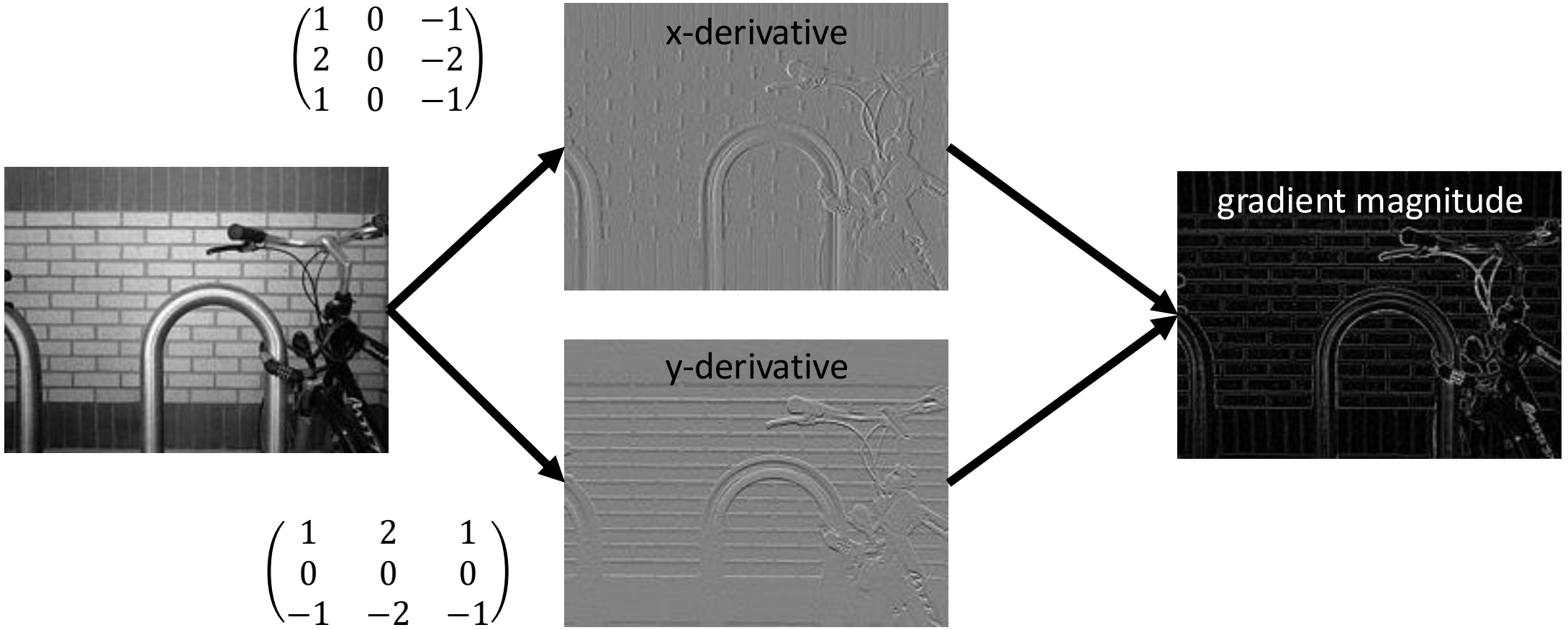
$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Feature map

Convolution kernels learned by a neural network.



Feature extraction using convolution

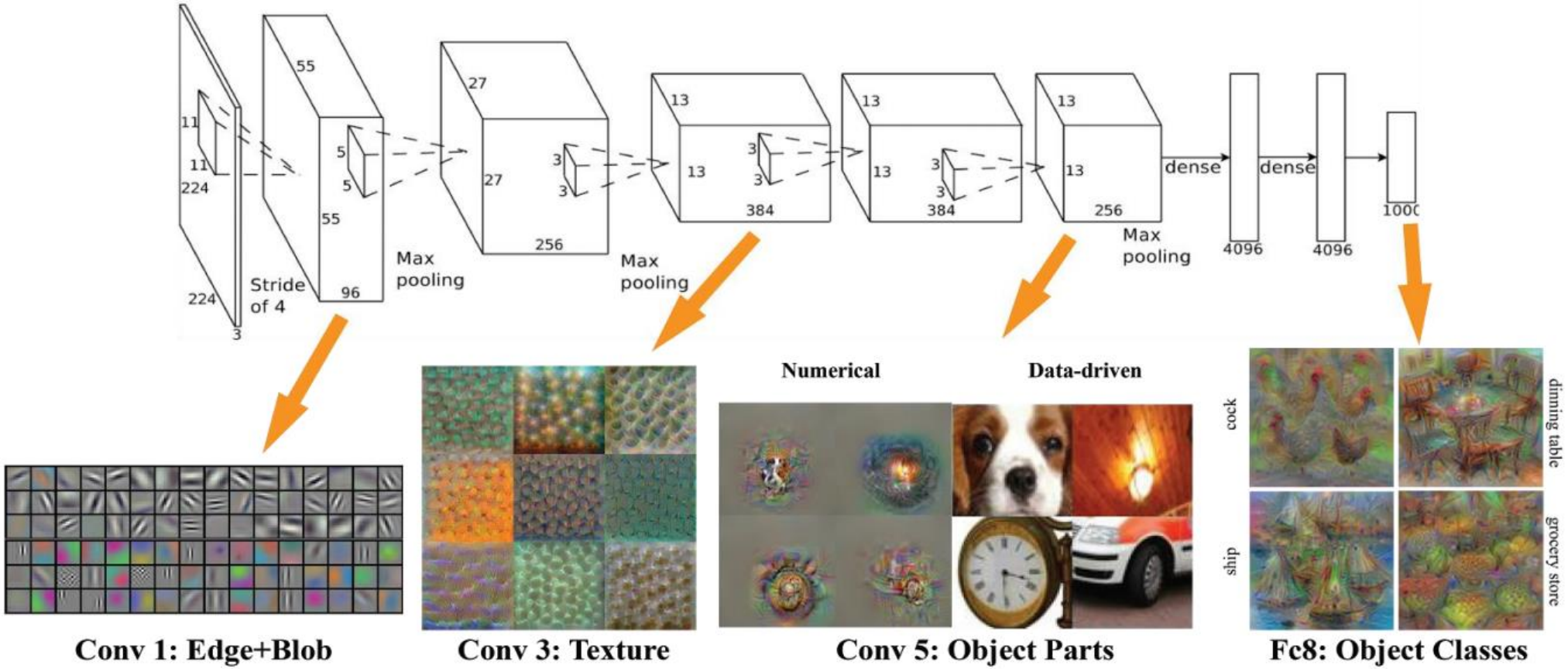


Sobel filters act as edge detectors.
(formally they estimate the derivative with some built-in noise reduction – we will use them in Lab 1)

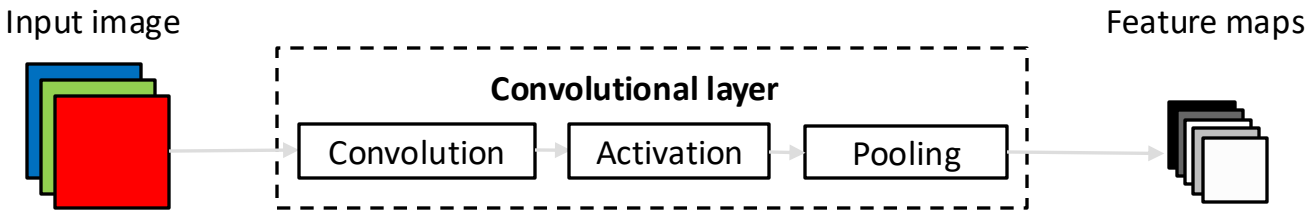
https://en.wikipedia.org/wiki/Sobel_operator

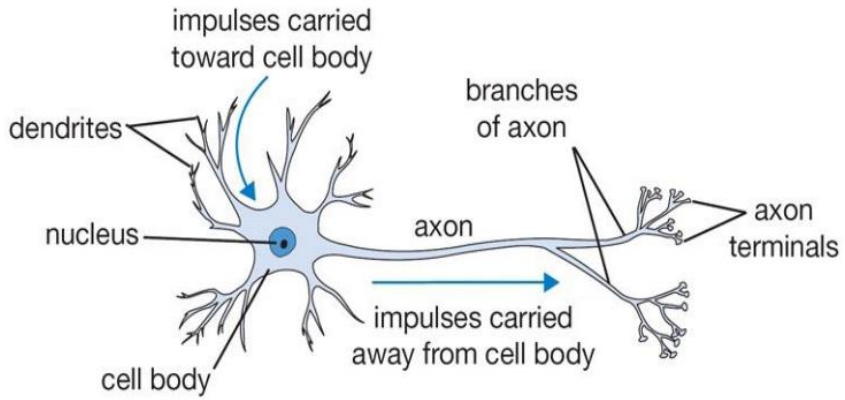
Features of features

- Deep = features of features of …

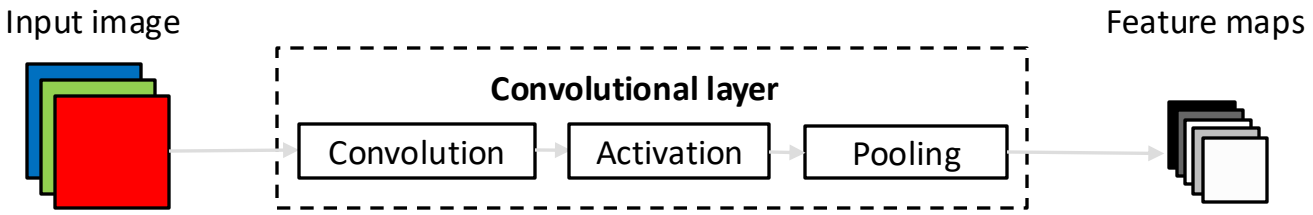


Convolutional Neural Networks

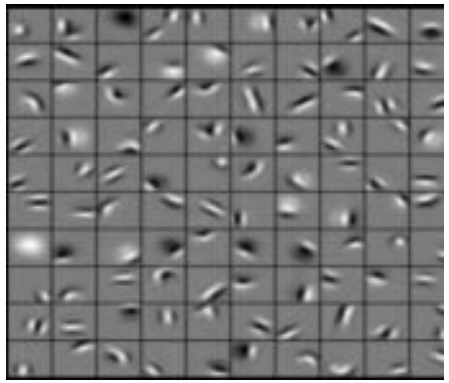




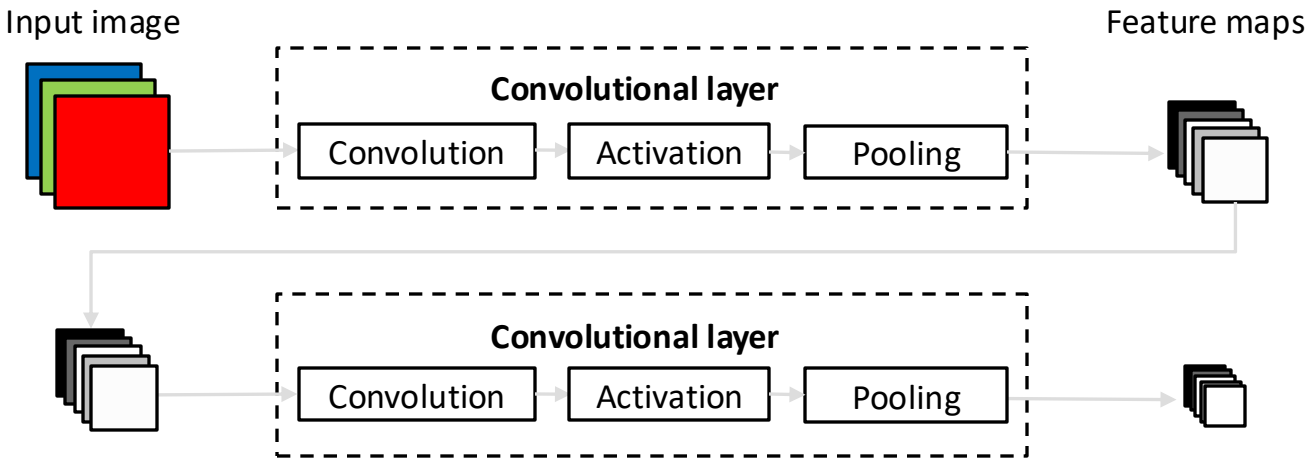
Convolutional Neural Networks



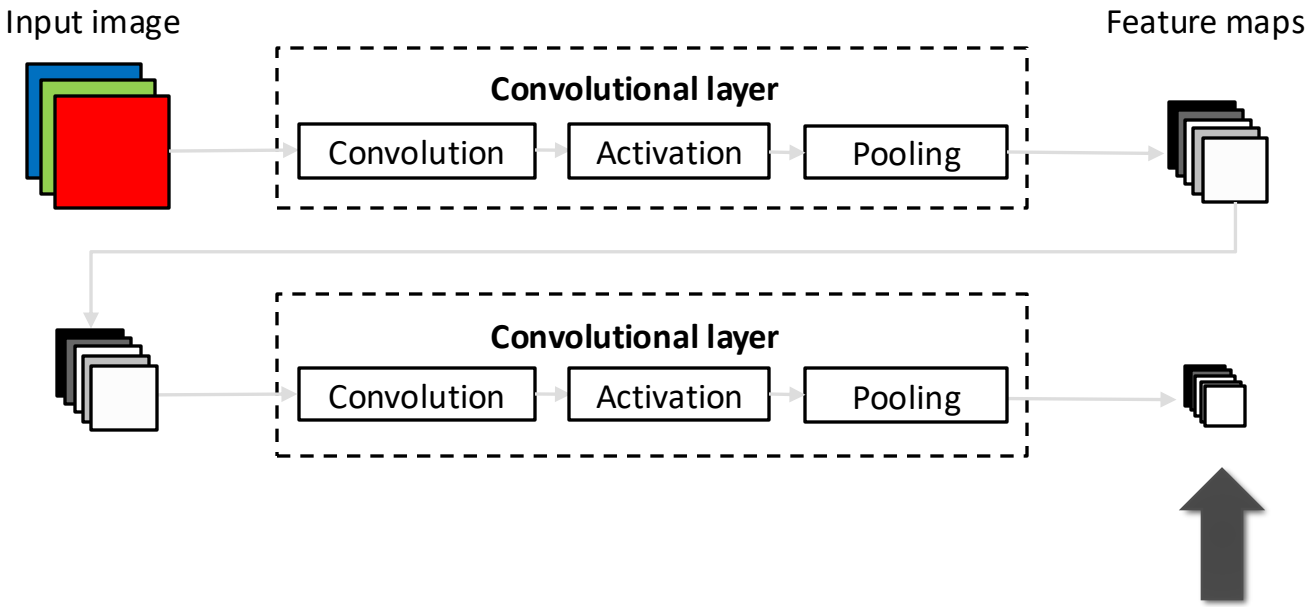
1st layer feature maps:
Tells the network where to find simple features like edges and blobs.



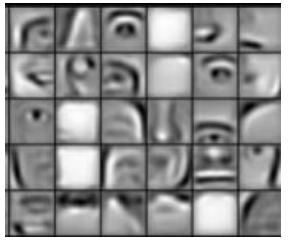
Convolutional Neural Networks



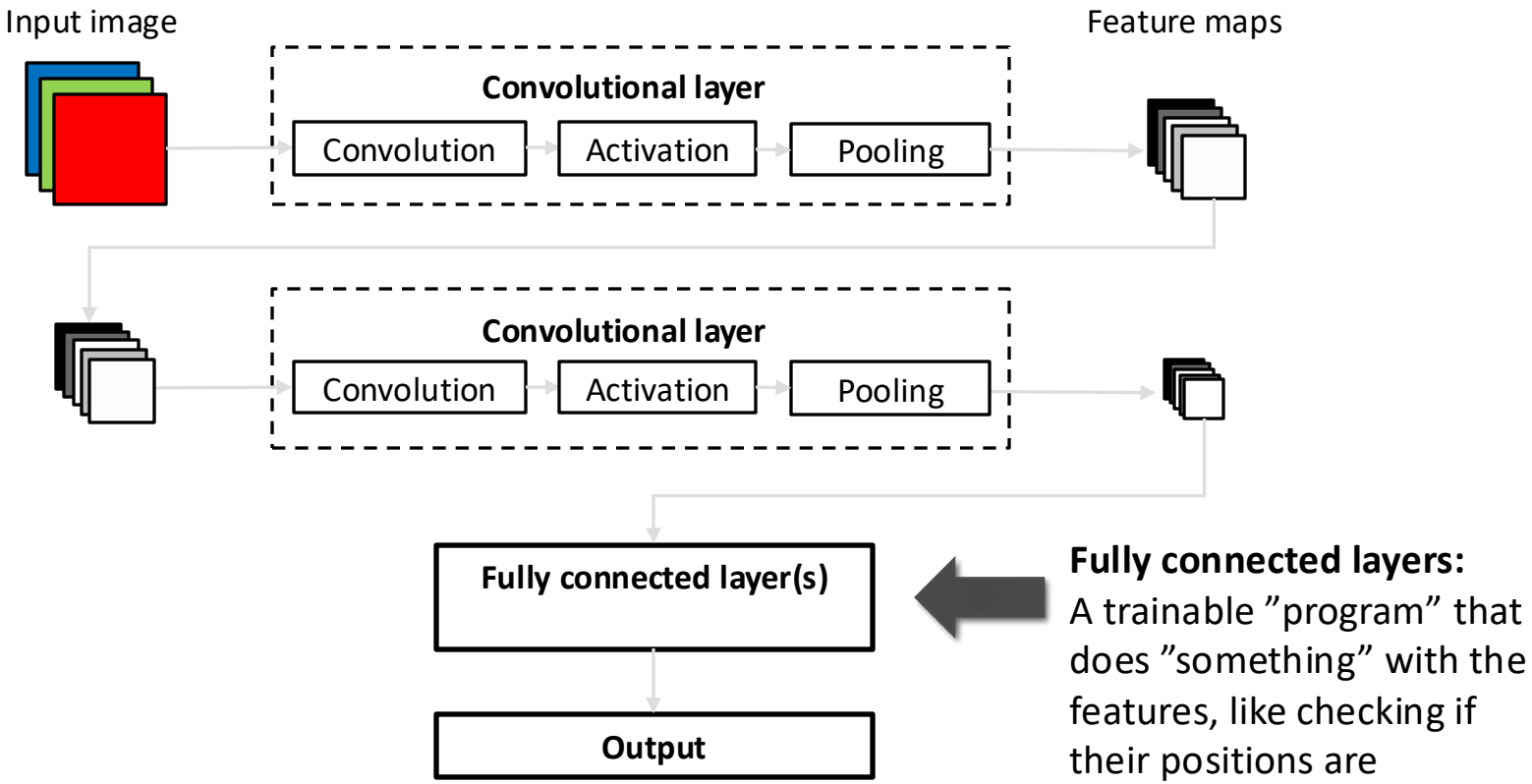
Convolutional Neural Networks



2nd layer feature map:
Tells the network where to find more complex features like eyes, nose, etc.

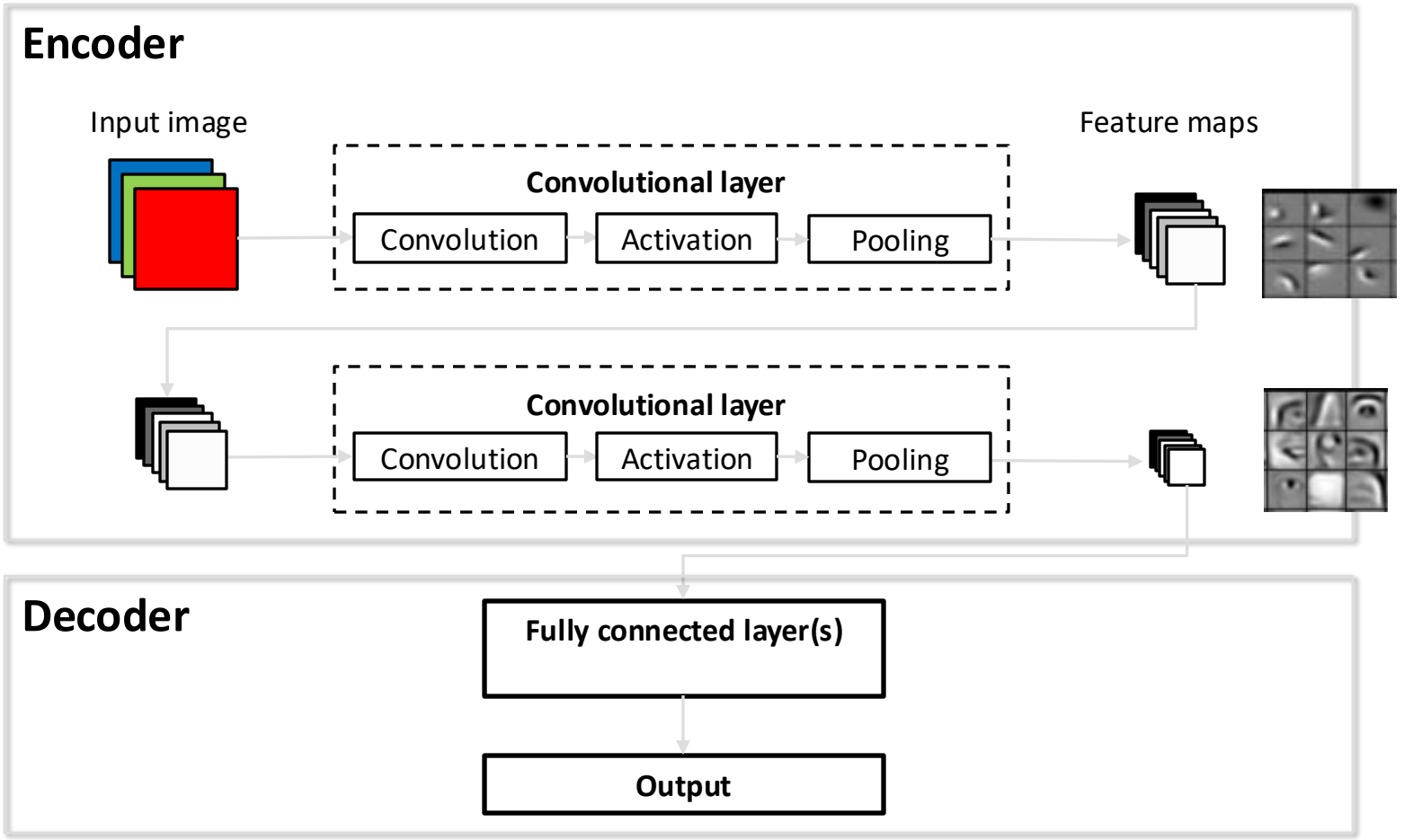


Convolutional Neural Networks



Fully connected layers:
A trainable ”program” that does ”something” with the features, like checking if their positions are consistent with a face.

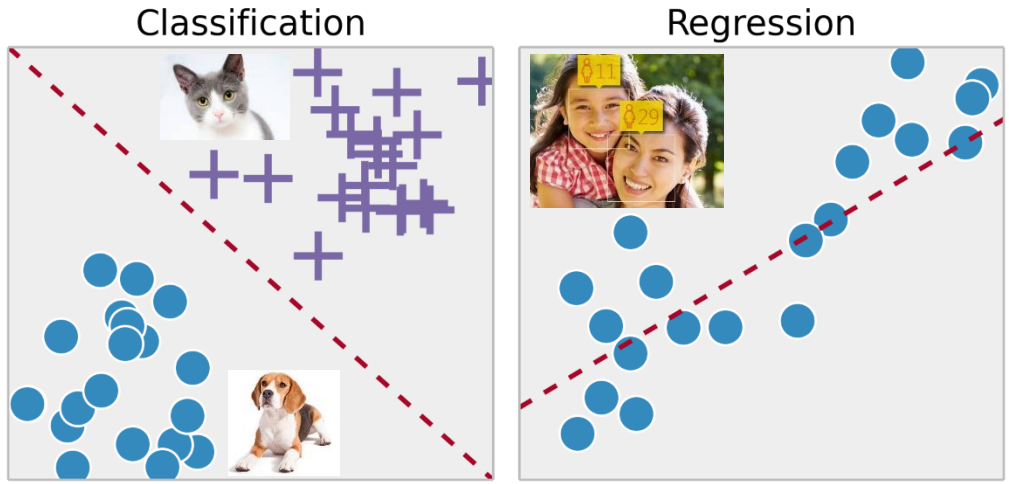
Encoder/decoder perspective



Encoder/decoder perspective

Classifier
Uses features to distinguish between two or more classes.

Output:
Discrete labels
(”Dog” or ”Cat”)



Regressor
Uses features to predict some functional relationship.

Output:
Real numbers
(”Age” of person in image)

Decoder

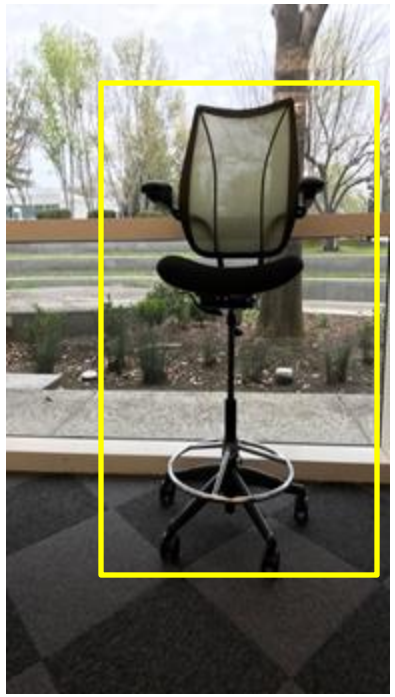
Fully connected layer(s)

Output

Computer vision tasks

Image Classification

Image Classification + Localization



Object Detection

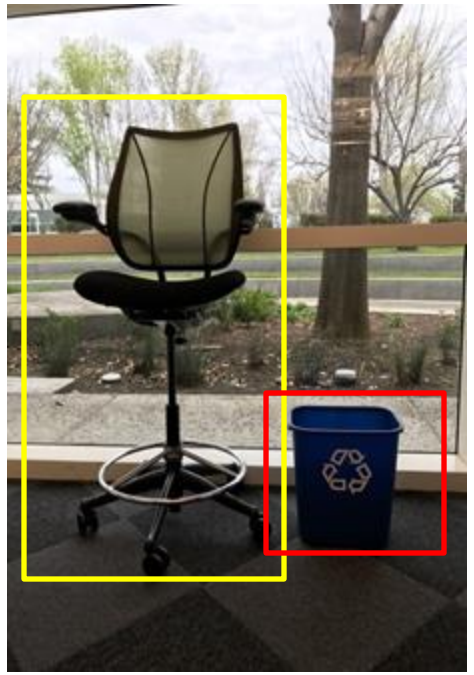


Image Segmentation

Advanced computer vision tasks

Colorize gray-scale images

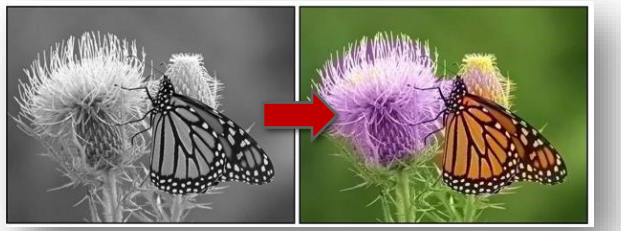
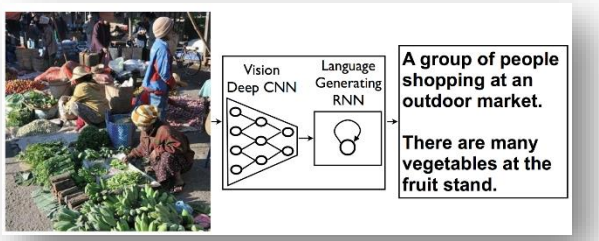
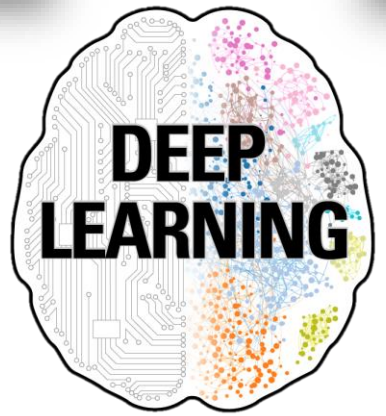


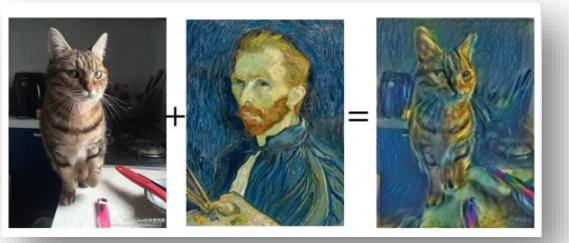
Image captioning



Turn horses into zebras

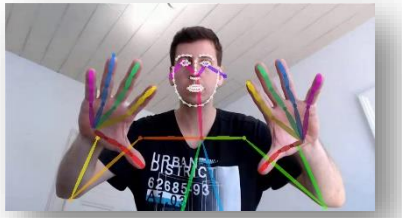


Turn images into Van Gogh paintings



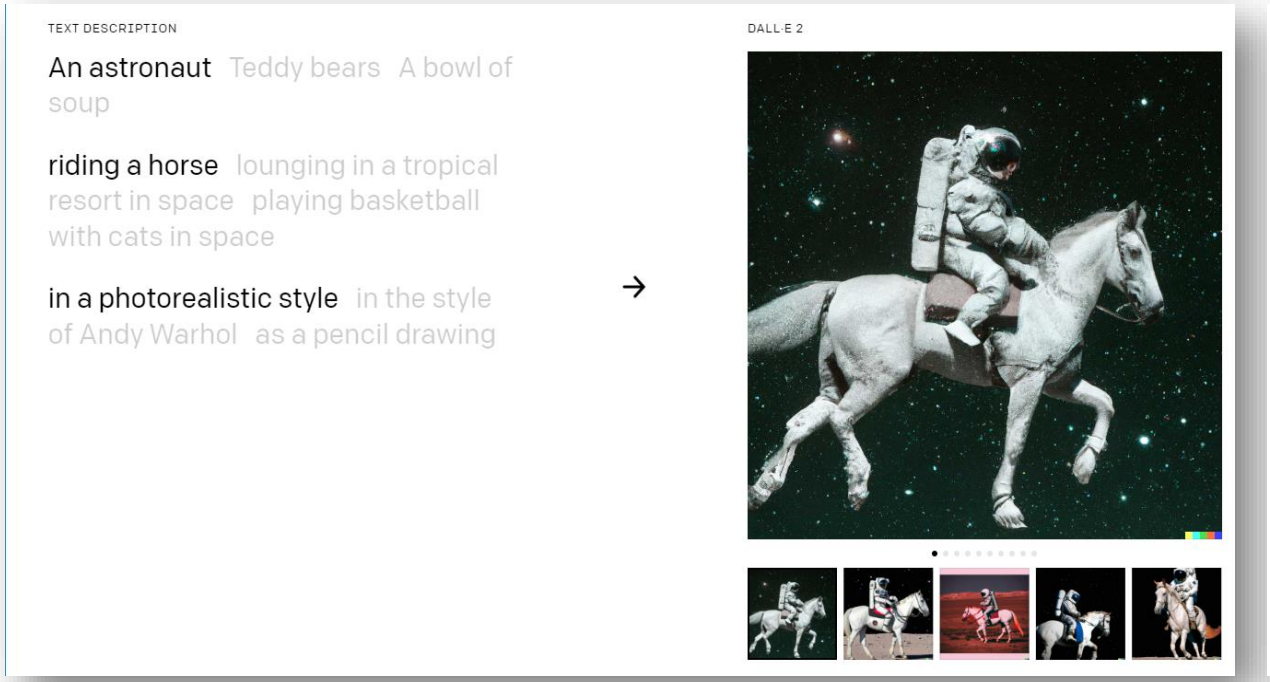
”Dream” images of fake celebrities

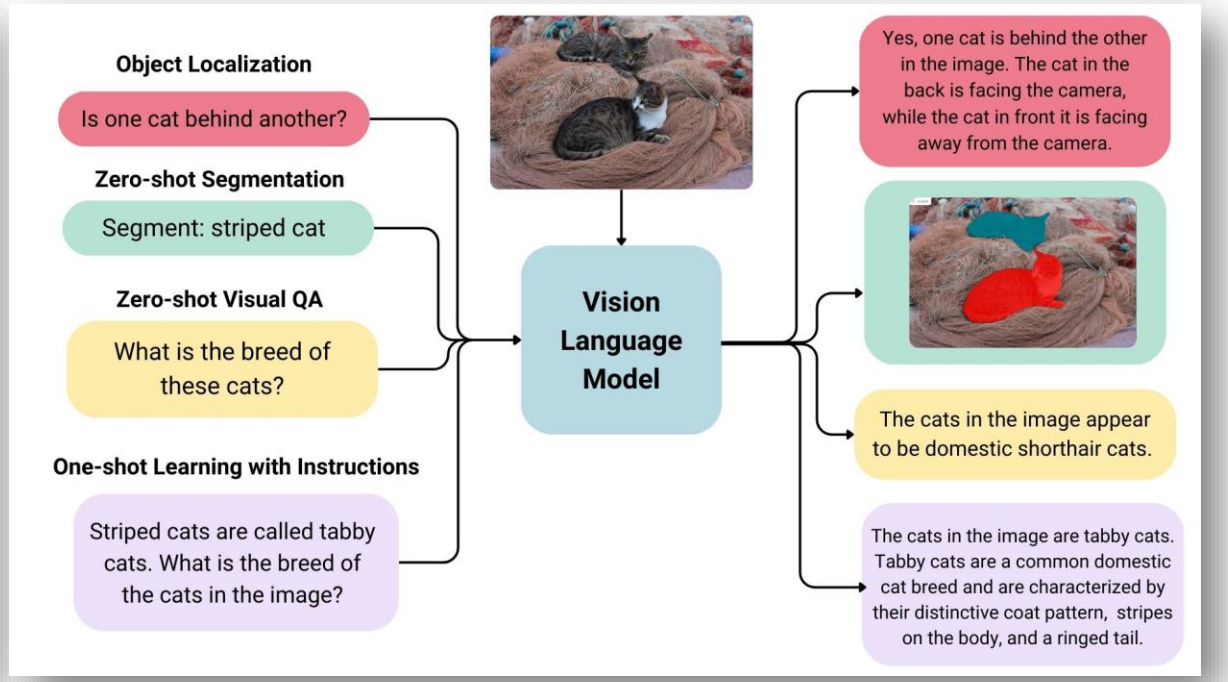
Detect human body pose



Trend: Visual Language Models

- VLMs are models that can learn simultaneously from images and texts to tackle many tasks, from image generation (”Generative AI”) to visual question answering and image captioning.



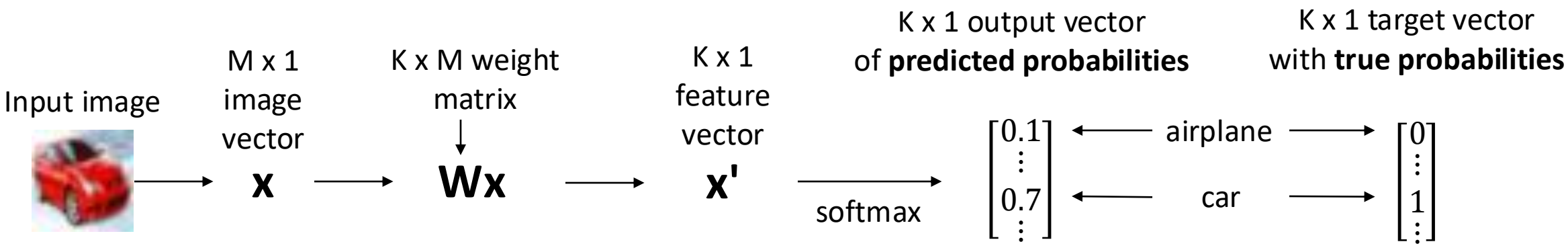


Ways to inspire ideas for your project

- Pick a framework and go through some tutorials: PyTorch, Keras, …
  - PyTorch tutorials: https://pytorch.org/tutorials/index.html
  - Keras examples: https://keras.io/examples/
  - Keras developer guide: https://keras.io/guides/
- Brainstorm useful X → Y mappings
- Experiment! Try something out, e.g., reproduce results from public GitHub repos.
- Papers with Code: https://paperswithcode.com/area/computer-vision

Lecture plan

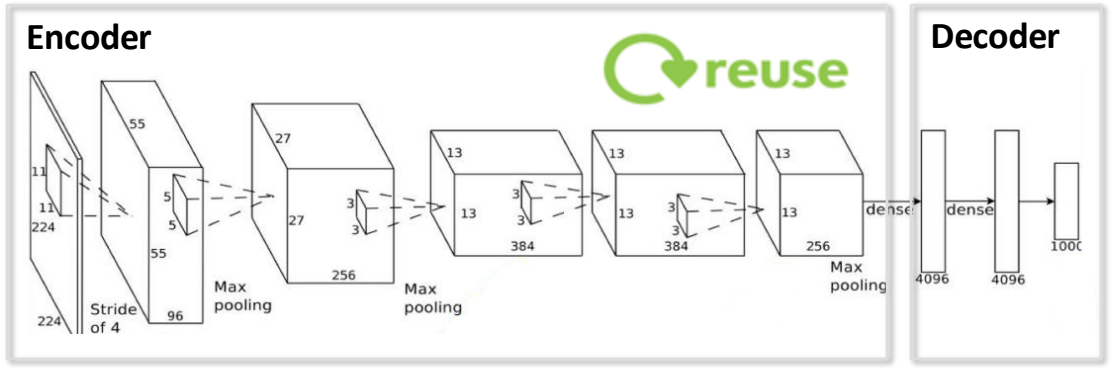
Week 2: Machine Learning fundamentals



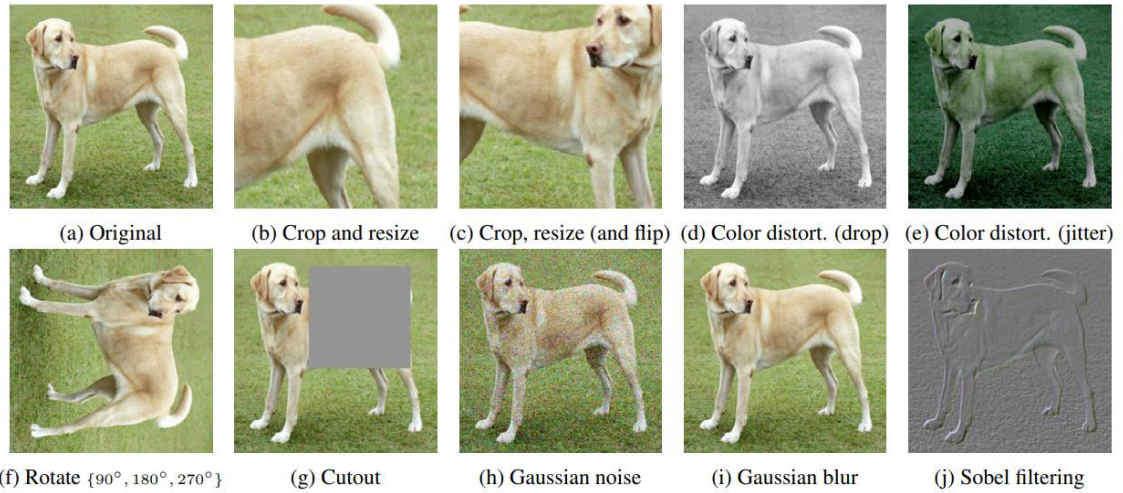
Week 2: Practical info about the course project

- Task is to recognize three types of Russian tanks
  - BMD4, BMP2, and T72
- Dataset
  - Self-made from Google Image search
  - 500 images from each class
- Neural network architecture: ResNet50
- Training:
  - Transfer learning using ResNet50 backbone (encoder) pre-trained on ImageNet
  - Data augmentation
- Evaluation / analysis
  - Inspect loss curves
  - Confusion matrix
  - Qualitative assessment of mis-classified images
  - Heatmaps (GradCAM – covered in Lecture 10)

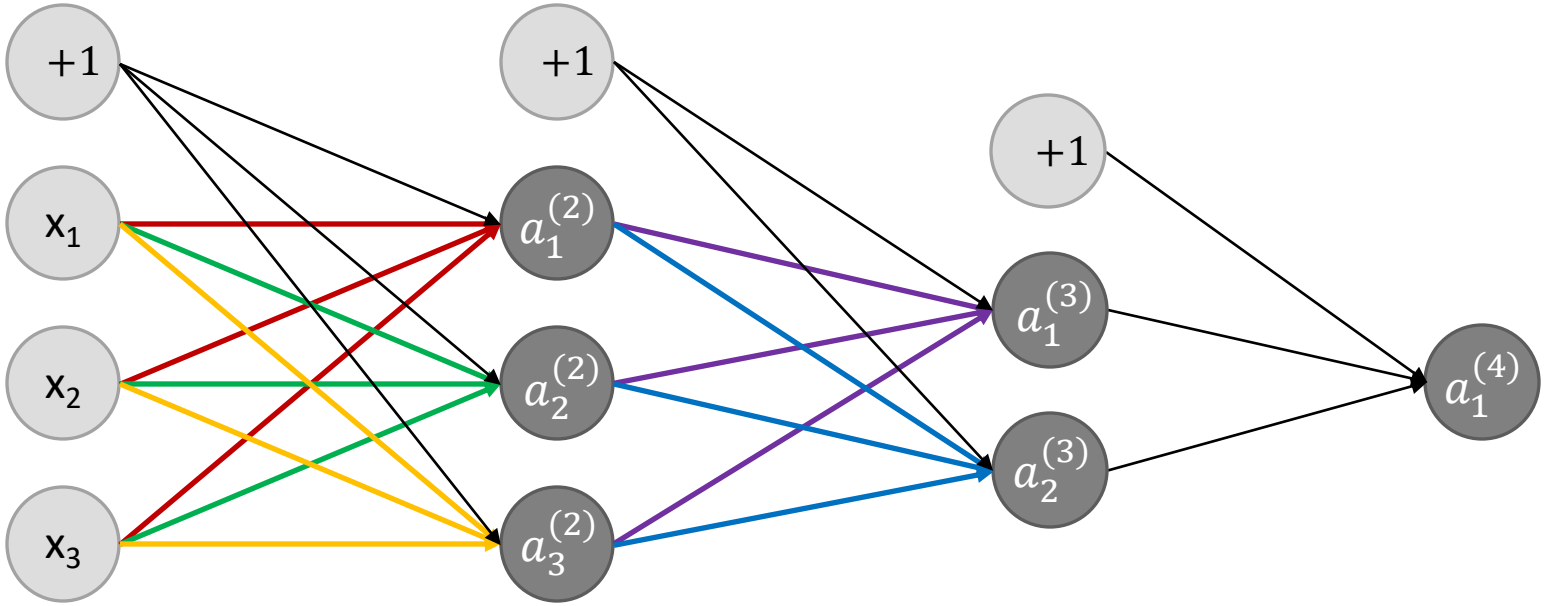
Transfer learning



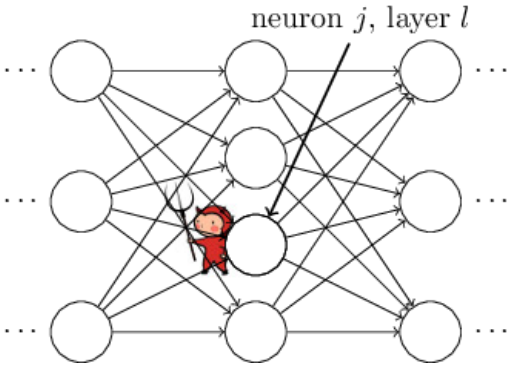
Data augmentation



Week 3: Traditional Neural Networks

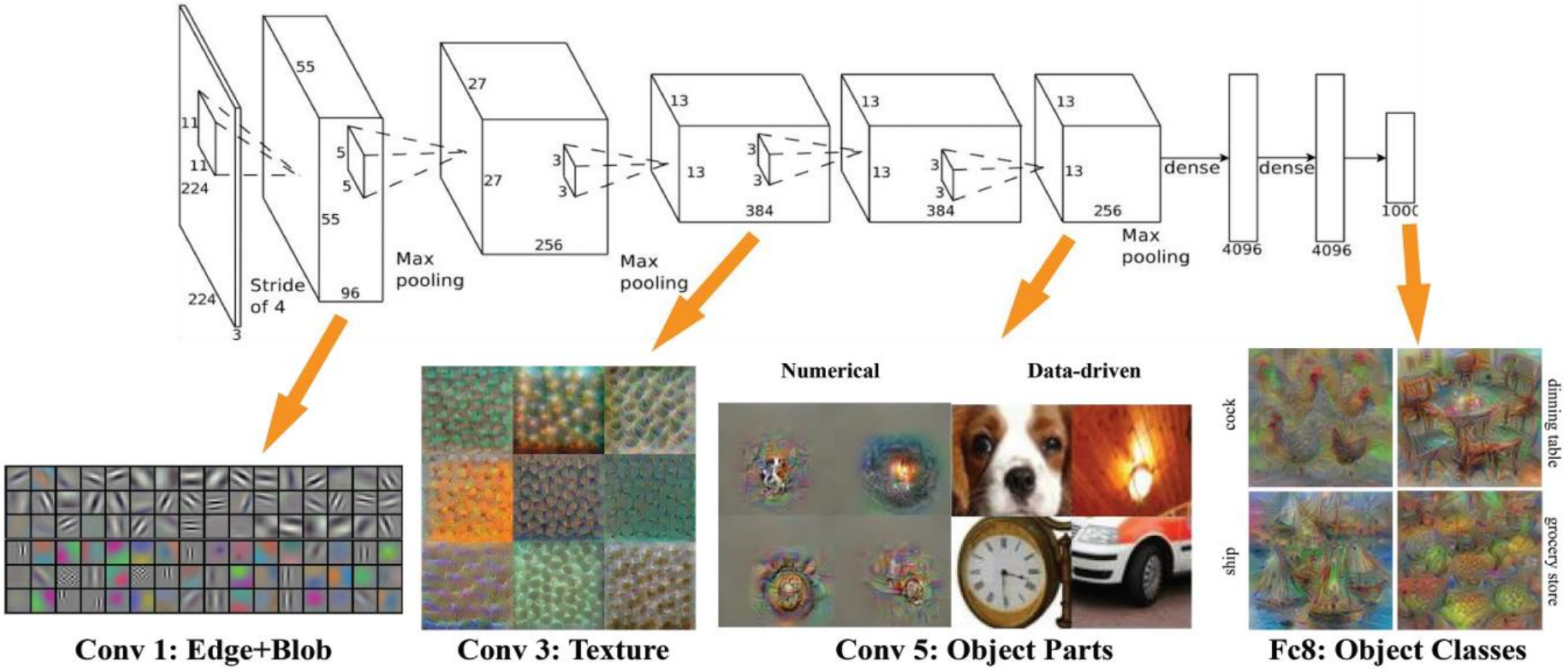


Backpropagation algorithm



$$a^{(1)} = x$$
$$z^{(j+1)} = W^{(j)}a^{(j)} + b^{(j)}$$
$$a^{(j+1)} = \sigma\left(z^{(j+1)}\right)$$

Week 4: Convolutional Neural Networks

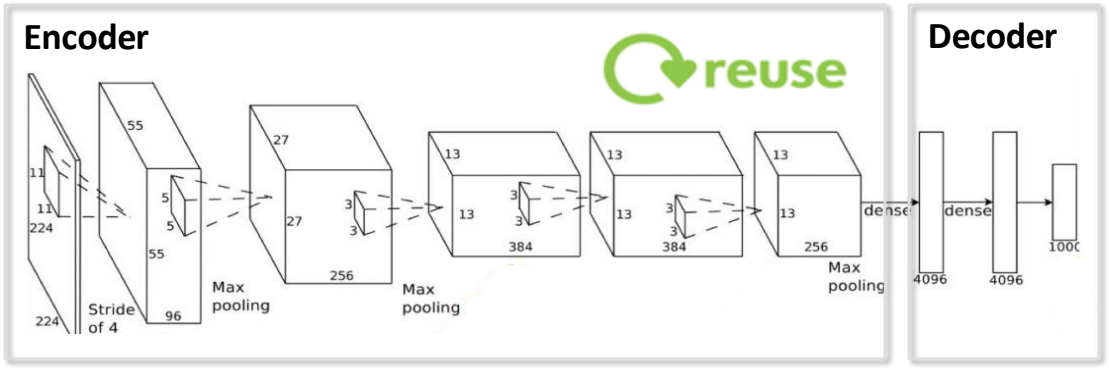


Week 4: How to write a good report

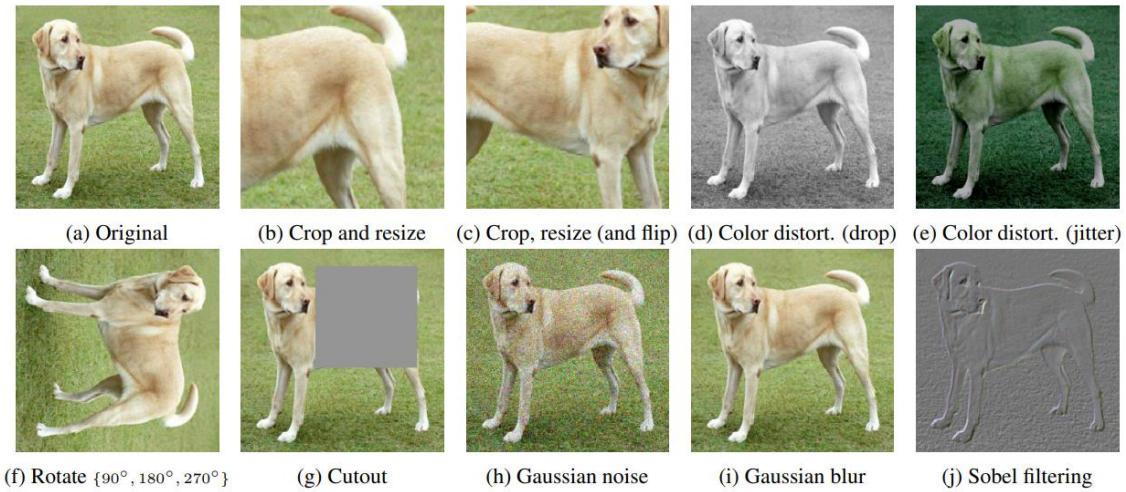
- Use this arXiv template
- Report should be structured like a research paper:
  - Abstract
  - Introduction (Includes motivation and objectives of your project. Be realistic and be specific!)
  - Related work (Summarize at least 2-3 key references. What makes your approach different?)
  - Methods (What did you do and how? Describe network architecture, data set, experiments, etc.)
  - Results (Objectively summarize your results and compare to related work or state-of-the-art)
  - Discussion (How well did you do? What worked? What didn't work?)
- Final report between 6-8 pages.

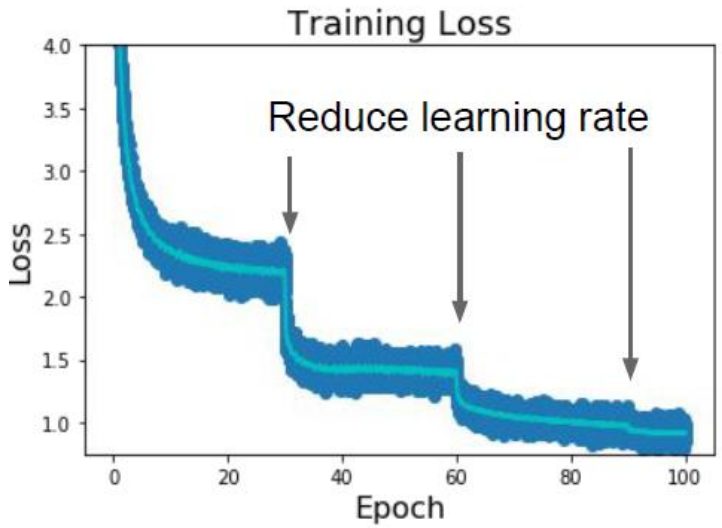
Week 5+6: Training ConvNets

Transfer learning

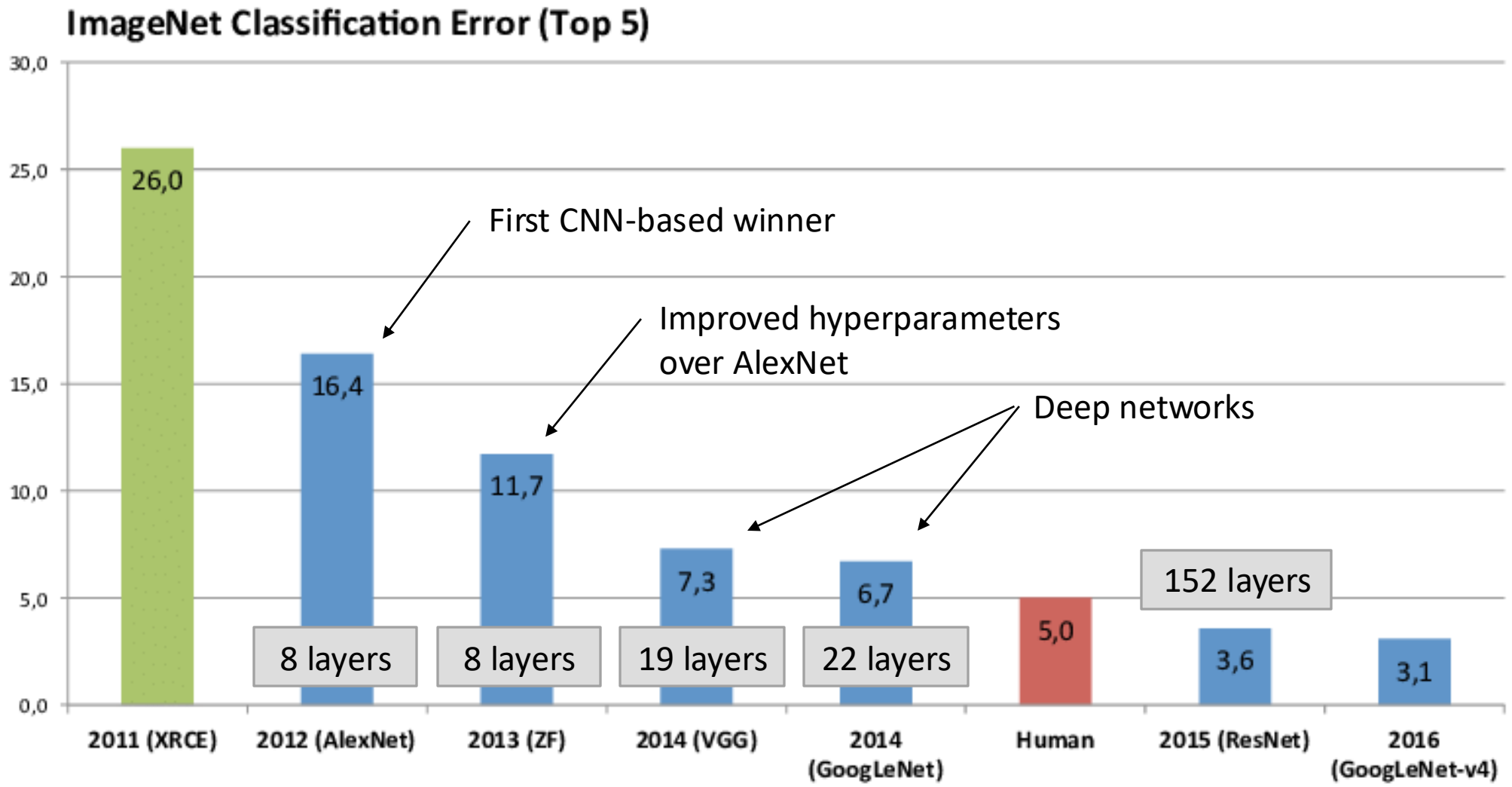


Data augmentation





Week 7: ConvNet architectures



Week 8: Object detection and segmentation

Classification

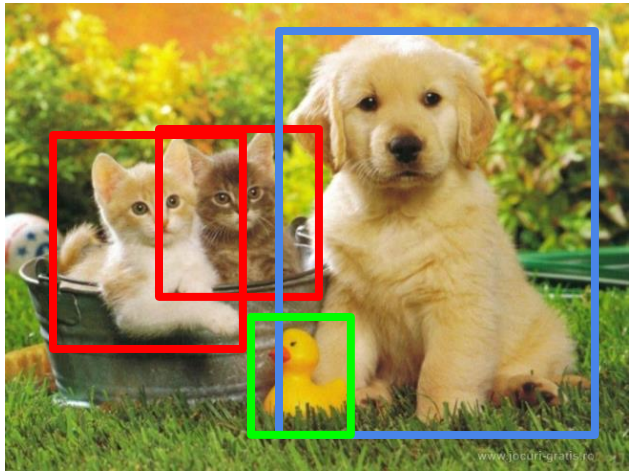
CAT

Classification
+ Localization

CAT

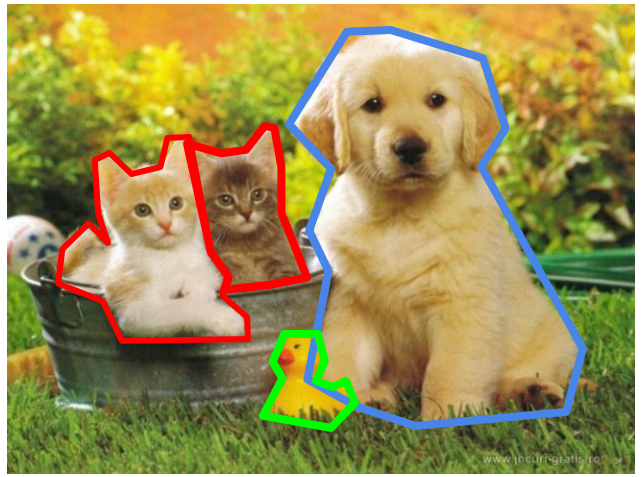
Single object

Object Detection



CAT, DOG, DUCK

Image
Segmentation

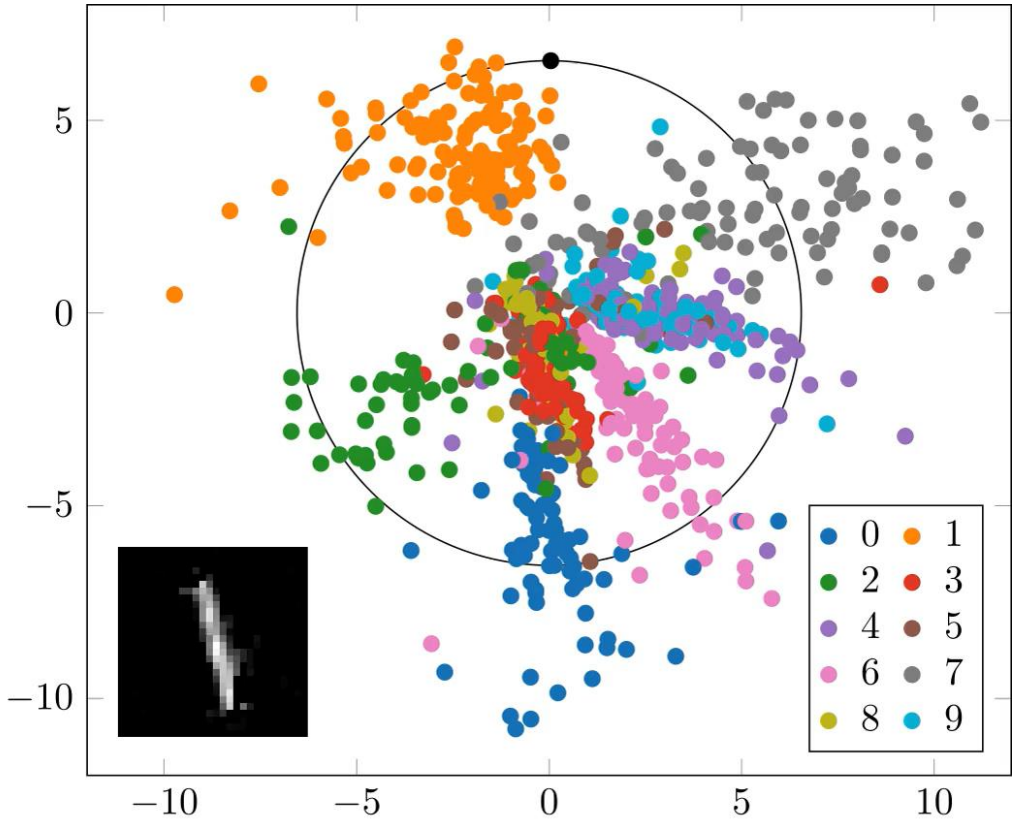


CAT, DOG, DUCK

Multiple objects

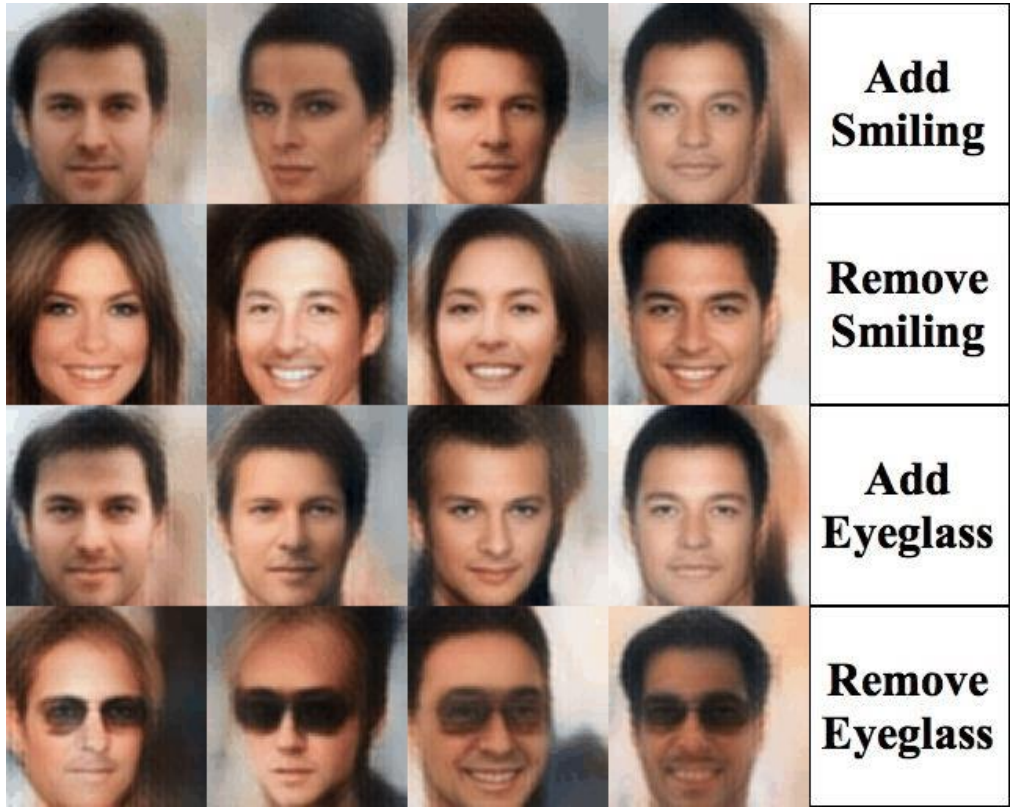
Week 9: Generative models

https://gertjanvandenburg.com/blog/autoencoder/



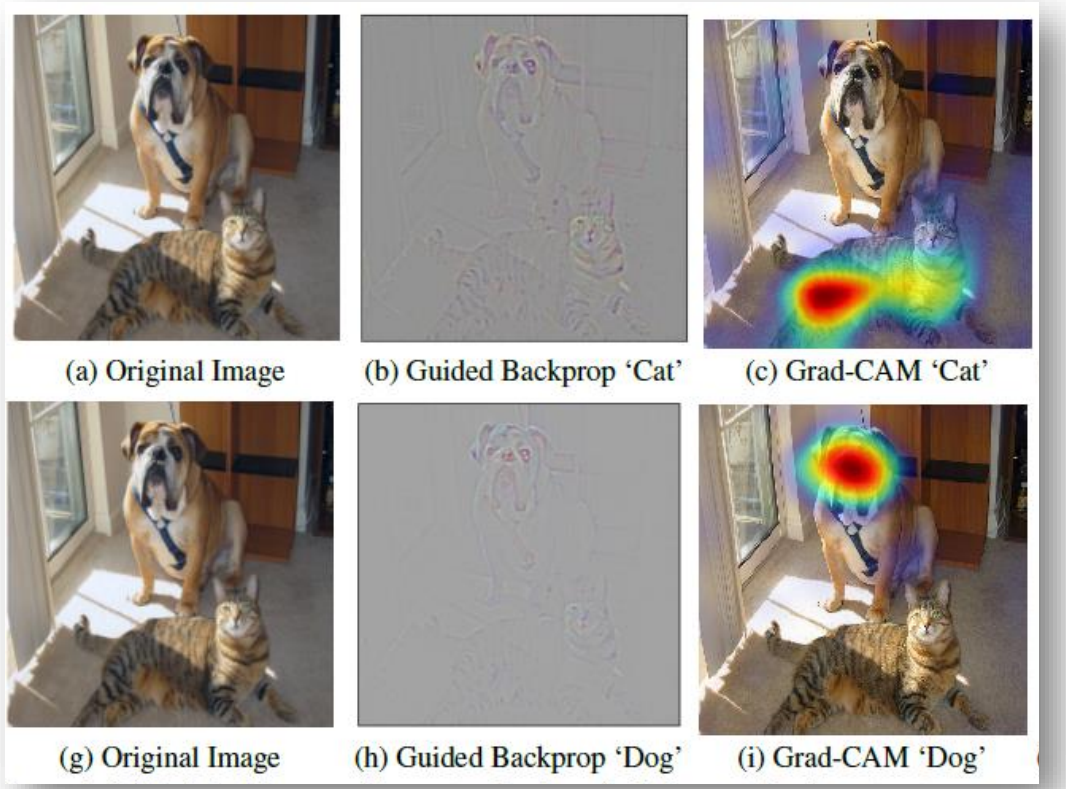
Regular Autoencoder on MNIST dataset

https://github.com/houxianxu/DFC-VAE



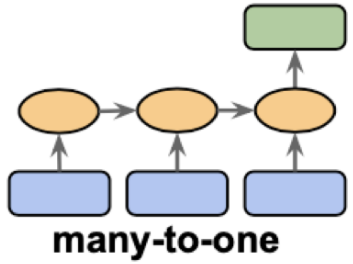
Variational Autoencoder on CelebA dataset

Week 10: Visualizing and understanding ConvNets

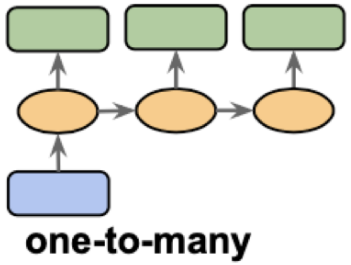


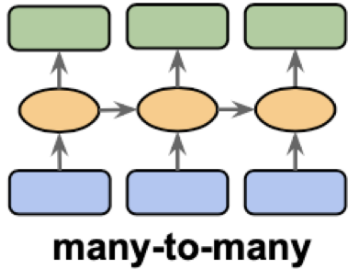
Week 11: Sequence models

Example
Input: Sentence
Output: Sentiment

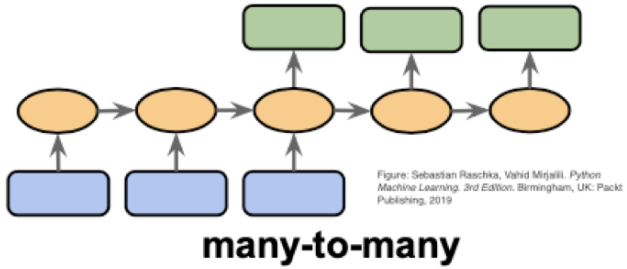


Example
Input: Image
Output: Caption



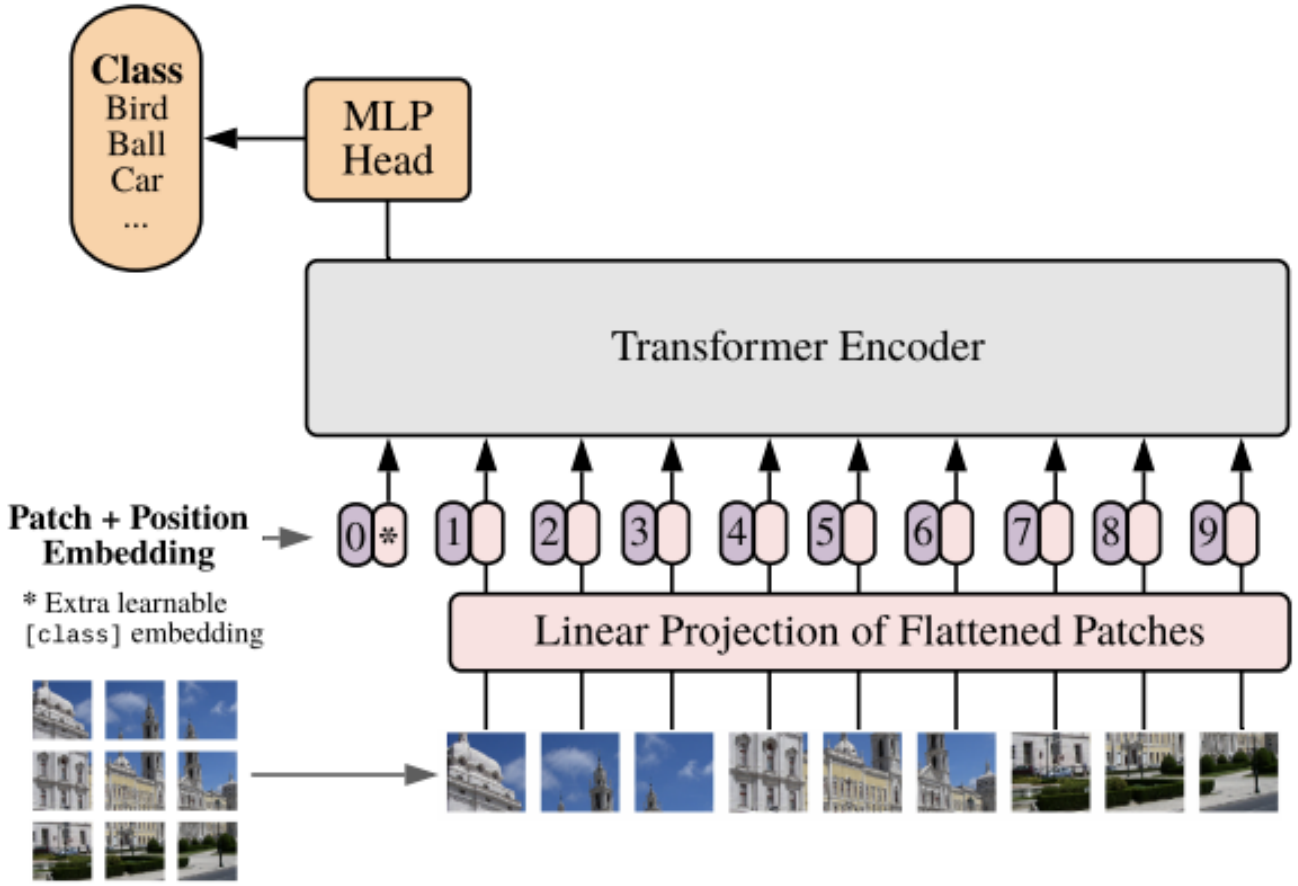


Example
Input: Video
Output: Human pose



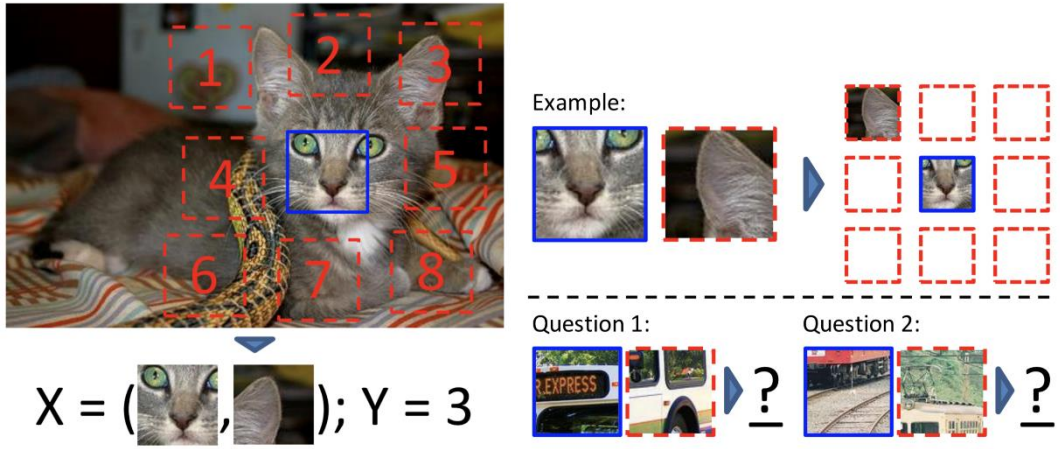
Example
Input: English sentence
Output: German translation

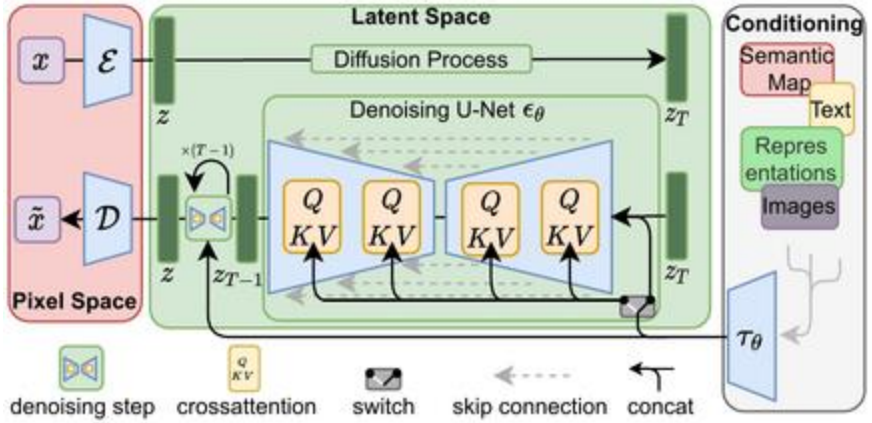
Week 12: Vision Transformers



Week 13: Advanced topics

- Self-supervised learning
- Diffusion models





Week 14

- Practical info about the exam
- Guest lecture