Computer Vision: Image Classification

What is computer vision?

- The part of artificial intelligence that deals with visual data
  - images
  - videos
  - other similar data: X-ray images, images with depth component RGBD images, ultra sound images
- One can see these images as grids of values called **pixels**
  - A "black and white" image, is a grid of values showing the shades from black (usually 0.0) to white (usually 1.0)
  - A color image is usually three grids of red, blue and green values
  - A depth image, like from a Kinect sensor adds a fourth value for depth

Why it is important?

- We can gather a lot of data with cameras!
  - Cameras are cheap!
  - They operate from a distance
    - It is much easier to take a picture of a volcano than to go and insert a temperature sensor
  - Eyesight is the human sense with the largest bandwidth

Computer vision problems

- Some of the problems are the same as the ones we had seen before, only adapted to visual data:
  - Classification of images, X-rays, videos etc.
    - kitten or bunny?
    - COVID or not COVID?
    - X-rated or not?
  - Regression of images, videos etc.
- Other problems are specific to computer vision
  - **Object detection** in images
  - **Object tracking** in videos
  - **Image segmentation**

The problem: Image classification

- Let us say I show you a picture

- Now you have to solve a multiple choice question:
  - [ ] Kitten
  - [x] Bunny
  - [ ] Puppy

What makes this a classification problem?

- **Inputs**:
  - A picture
- **Outputs**: a discrete set of choices (mutually exclusive)
  - yes / no
  - kitten / bunny / puppy
  - COVID / not-COVID
- This is exactly the same as before, the only difference is the input

Traditional approach: engineered features

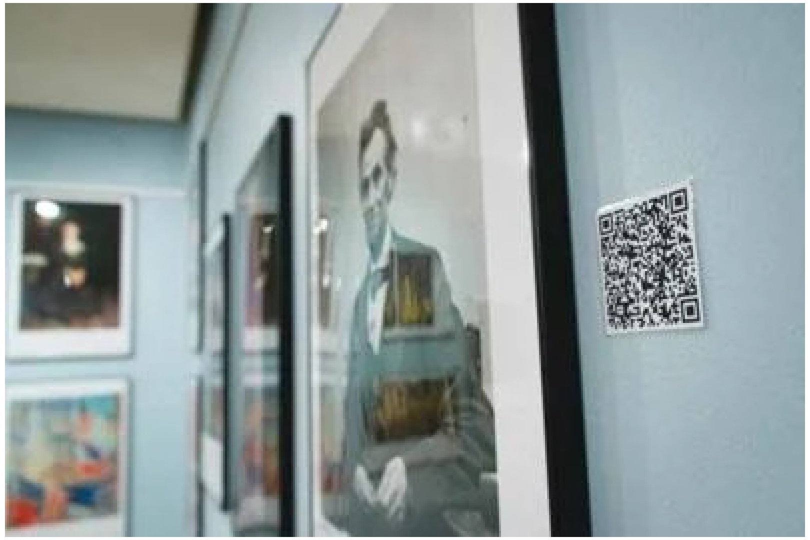
- For a long time, image classification relied on **engineered features**
- Identify some features of object classes that allow us to differentiate them from the alternatives:
    - Bunny: long ears, two large front teeth
    - Kitten: cat eye, whiskers
- Describe these mathematically (eg. whiskers are nearly-horizontal lines etc.)
- Write code to detect them.
- Perform the classification based on whether we find the features or not.
- A lot of effort, slow progress! We have to do it for each new class.

Engineered features today

- A well designed algorithm for engineered features can be very fast!
- Features in the natural world are usually not designed to be easy to capture!

Engineered features today

- Sometimes we can design the features such that they are easy to capture!



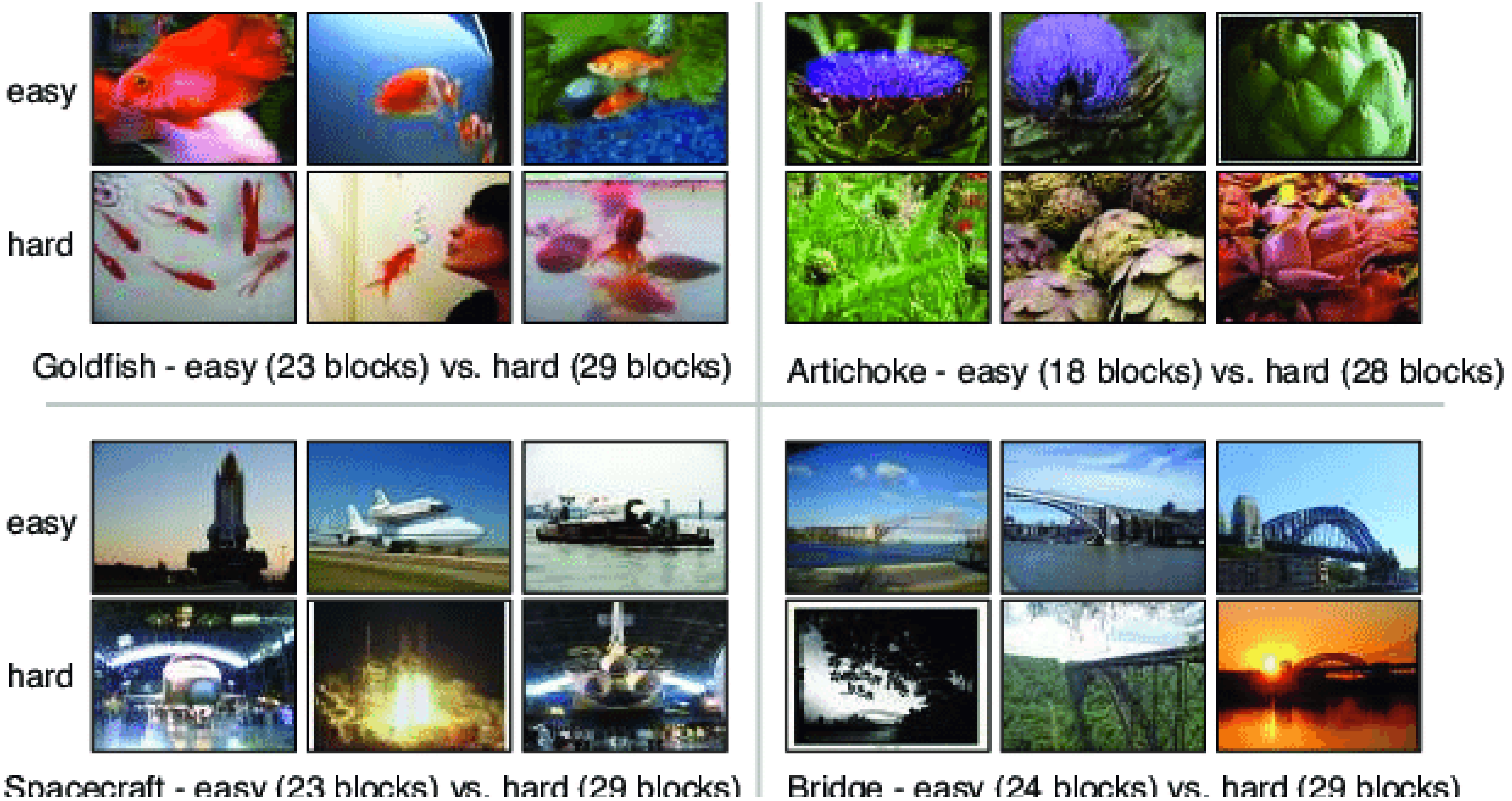
- Examples include
    - Barcodes, QR-codes
    - ARUCO codes used in robotics
- They **might** play a role in the self-driving transition

Learning an image classifier

- Training data:
  - collection of images + labels
  - an easy way to do it is to create directories for "kitten", "bunny", "puppy" and collect images in those directories
  - we separate **training data** and **test data**
- We train our model on the training data
  - Usually, we try to achieve a model that works well on the training data...
  - But not by memorizing all the images and their labels!
- What we want, is a model that works well on the test data
  - As well as in other future situations, e.g. future bunnies and kittens.

Short history: the Imagenet competition

- An annual contest run from 2010-2017 where software programs competed to correctly classify images:
  - 1.2 million images
  - 1000 classes
- 2010, 2011: winners were feature engineered programs, about 30% error rate, slow progress
  - expectation was that problem will be maybe solved by the end of the century


easy
hard
Goldfish - easy (23 blocks) vs. hard (29 blocks)
Artichoke - easy (18 blocks) vs. hard (28 blocks)
easy
hard
Spacecraft - easy (23 blocks) vs. hard (29 blocks)
Bridge - easy (24 blocks) vs. hard (29 blocks)

Imagenet

- 2012: winner was AlexNet, a team from the University of Toronto
  - Convolutional Neural Network (CNN)
  - 15% error rate
  - Leveraged GPUs
- 2015: a model called ResNet (Residual Network)
  - 3.57% error rate, better than human
- 2017: competition terminated, as problem was considered solved

How it works: convolutional neural networks

- The models that won the ImageNet competition were **convolutional neural networks**
- A convolution is a mathematical operation that depends on a small patch of an image (**this is a simplification!**)
  - It is described by a small matrix of numbers such as this (this convolution detects a vertical line):

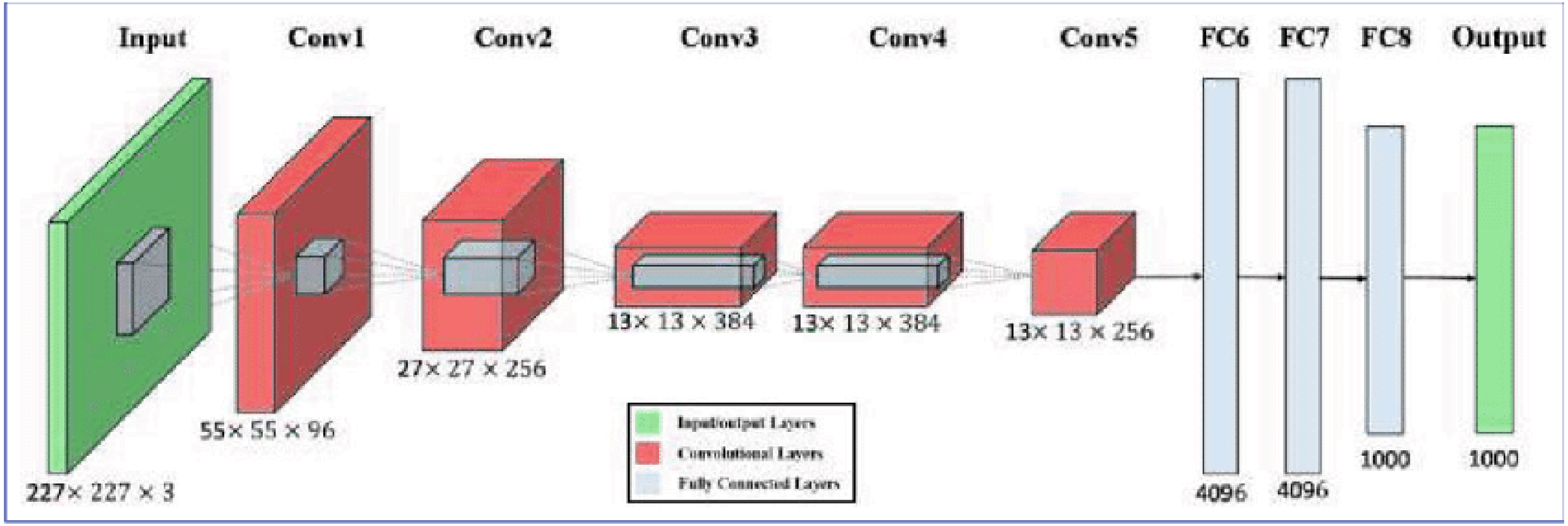
```
0, -1, 1
0, -1, 1
0, -1, 1
```

- In the 1980s we used to come up with these numbers ourselfs
  - The idea is to learn them from the training data

Deep neural networks

- A convolution can only capture a small feature in a small patch in the image
- Idea: stack them on top of each other
  - Convolution on top of convolutions
  - Features assembled from features
- This stacking means a neural network with multiple layers:
  - We call this a deep neural network
  - AlexNet, the 2012 winner had 8 layers
  - ResNet the 2015 winner has 152 layers (turns out making things this deep is usually not worth it)

AlexNet architecture



- It is a remarkably simple architecture, for something that solves a problem we couldn't think we can solve until 2100!

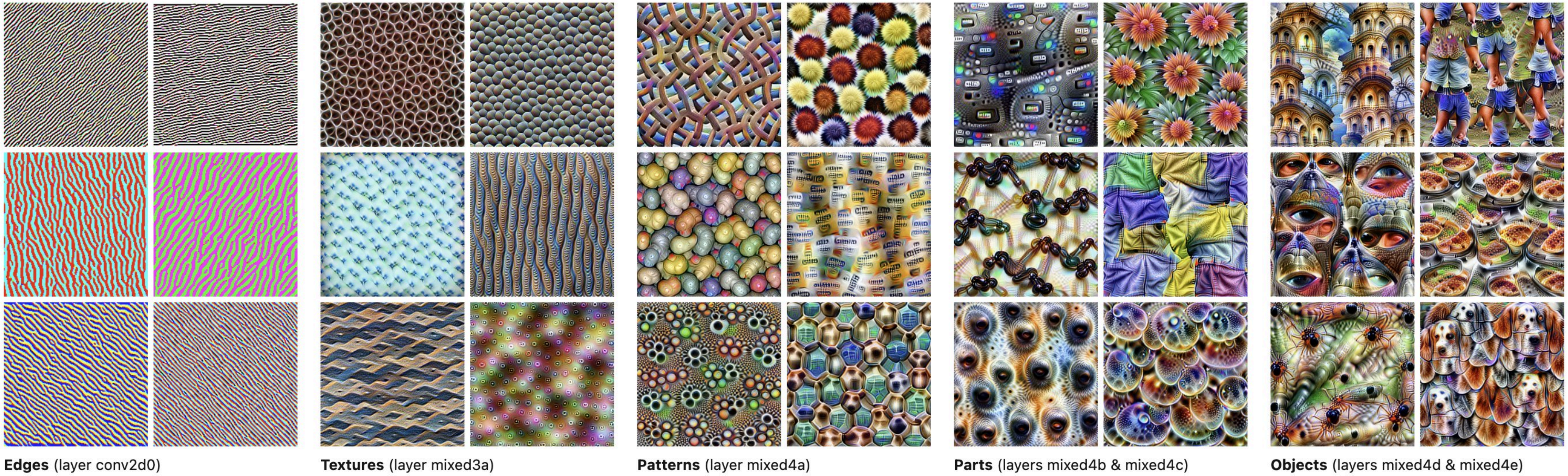
AlexNet code

```
class AlexNet(nn.Module):
    def __init__(self, num_classes=1000):
        super().__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=11, stride=4, padding=2),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),

            nn.Conv2d(64, 192, kernel_size=5, padding=2),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),

            nn.Conv2d(192, 384, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),

            nn.Conv2d(384, 256, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),

            nn.Conv2d(256, 256, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
        )

        self.classifier = nn.Sequential(
            nn.Dropout(),
            nn.Linear(256 * 6 * 6, 4096),
            nn.ReLU(inplace=True),

            nn.Dropout(),
            nn.Linear(4096, 4096),
            nn.ReLU(inplace=True),

            nn.Linear(4096, num_classes),
        )
```

Learning features

- The magic is not in the architecture, but in the training.
- The system learns a collection of features, starting from low level: horizontal lines, vertical lines, splotches of color etc.
- Then it learns to assemble them into higher level features: a horizontal and vertical line form a corner
- And even higher levels...
- The only rule is that it retains the features that help in the classification
    - The ones that lower the error rate (the **loss function**)

Low level and high level features in CNNs



From Chris Olah, Alexander Mordvintsev, Ludwin Schubert,

https://distill.pub/2017/feature-visualization/

Other techniques and more recent trends

- In the last several years there are other approaches that play in the image classification space
- Eg. techniques based on the same architecture as those used for large language models: **visual transformers**, **vision-language models** etc.
- Efforts in specific domains:
  - Face recognition
  - Improving energy efficiency and computation requirements

Applications

- Ask an LLM for applications

Pitfalls and dangers: Illegal surveillance

- The ability to perform facial recognition quickly and inexpensively changed one of our fundamental assumptions about public spaces
- There was an implicit assumption that in some scenarios you are an anonymous member of the crowd with the privacy implication that your comings and goings are private
- This is not the case any more: you are almost always in the sight of a security camera, and software is available that can recognize you.
- **Debate:** are the security benefits outweigh the privacy costs?

Try it out: cat, dog or monkey