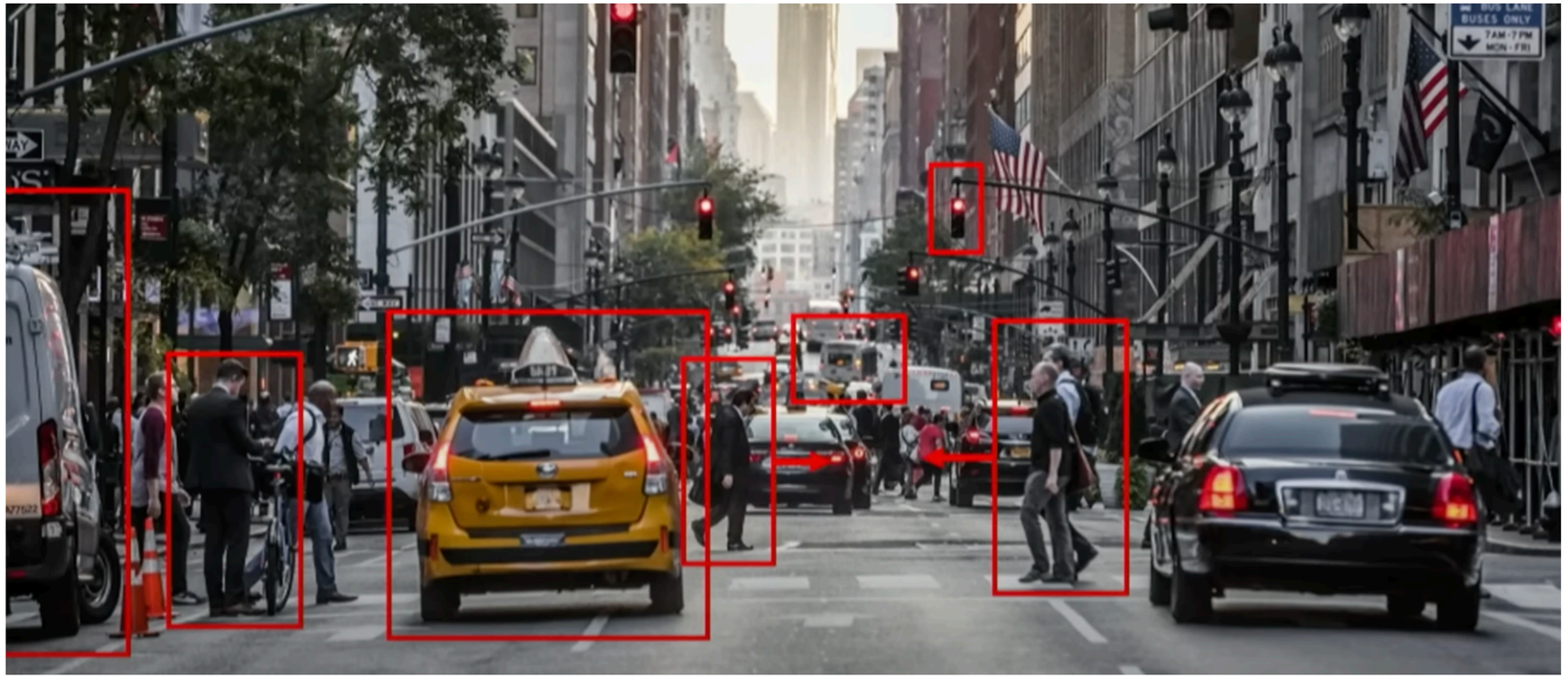
Introduction to Data Science

Lesson 10
Deep Learning for Computer Vision

Marija Stankova Medarovska, PhD
marija.s.medarovska@uacs.edu.mk

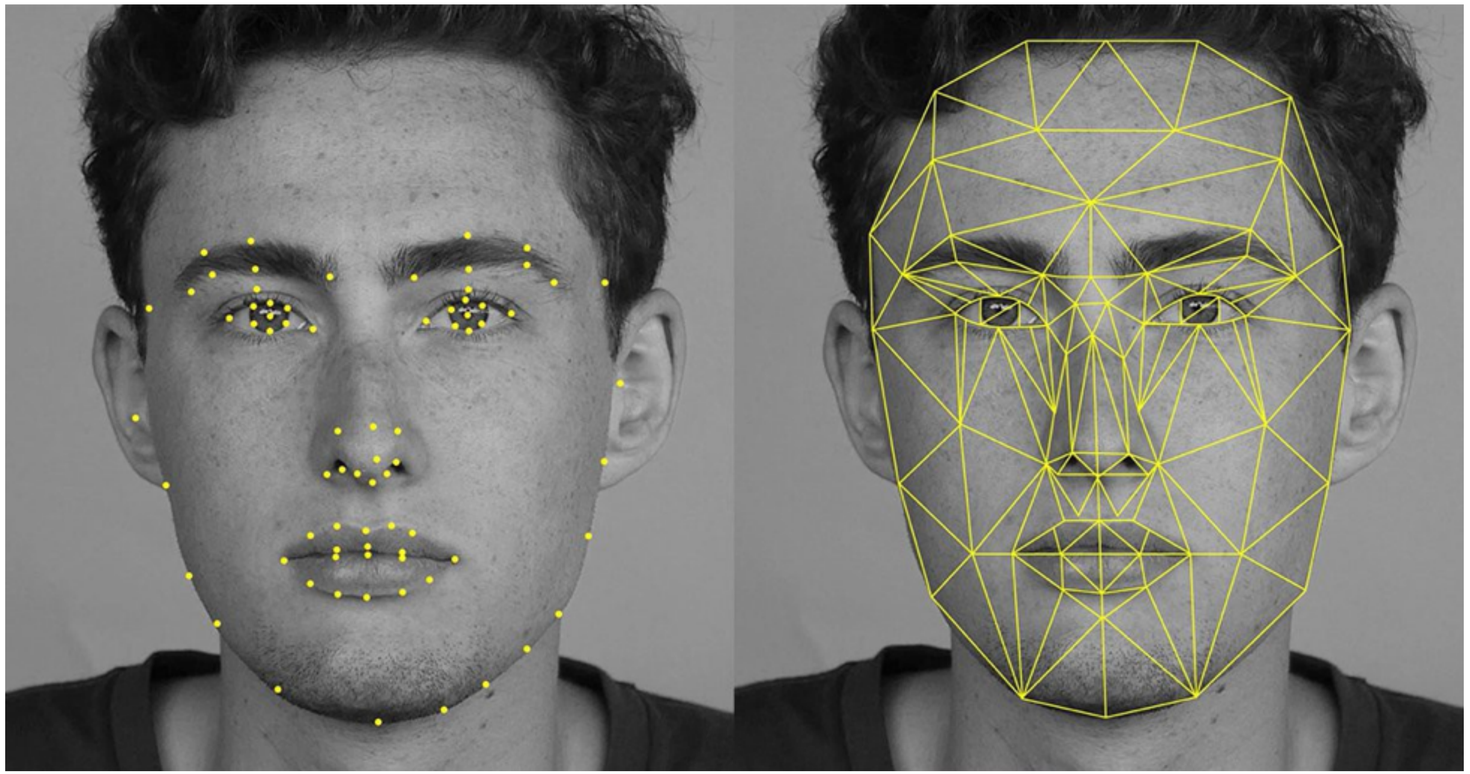
Deep Learning for Computer Vision

Computer Vision



To discover from images what is present in the world, where things are, what actions are taking place, to predict and anticipate events in the world.

Applications: Facial Detection and Recognition



Example: Deep learning can be used for facial detection and recognition technologies, in order to locate faces and match them to known identities.

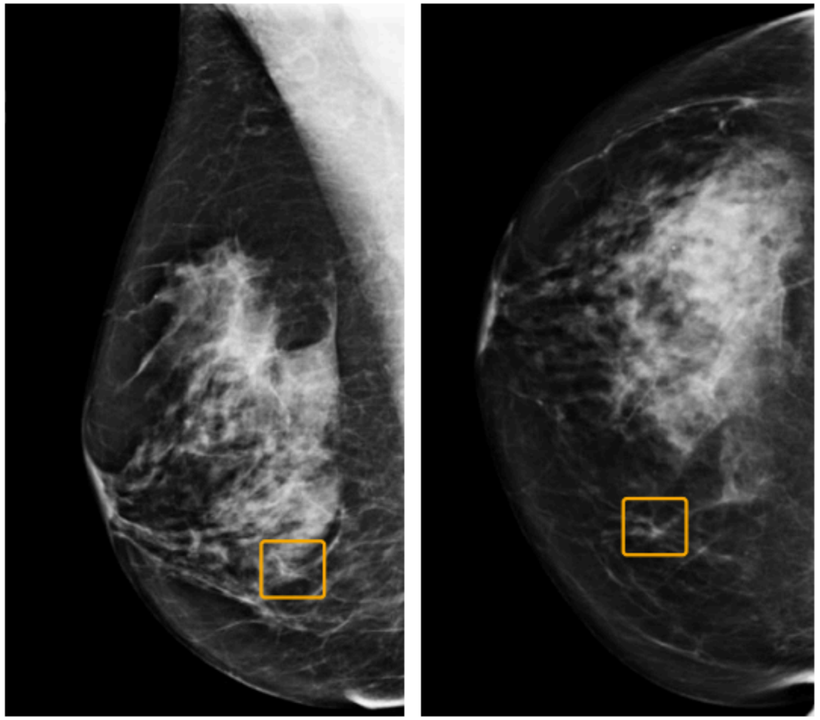
Applications: Self Driving Vehicles

Example: Self driving is among the most demanding of vision tasks: not only must the algorithm detect, localize, track, and recognize objects, but it has to do it in real time with near-perfect accuracy.

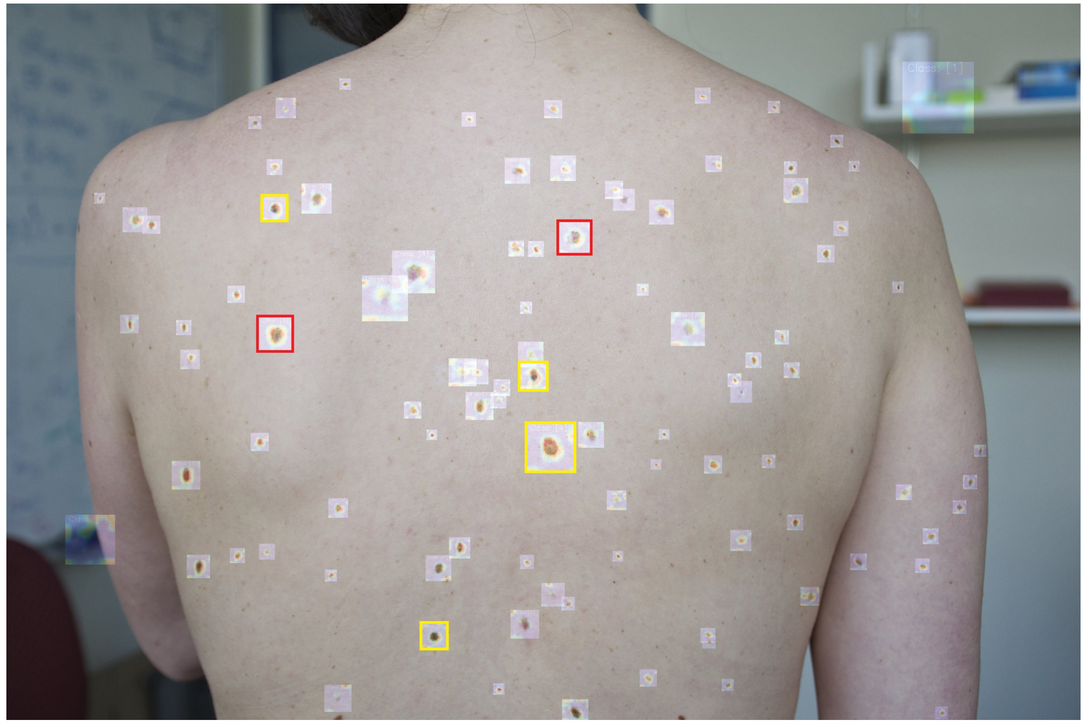
[1]https://www.youtube.com/watch?v=TUDiG7PcLBs&ab_channel=Tesla

Applications: Healthcare

Breast cancer



Skin cancer

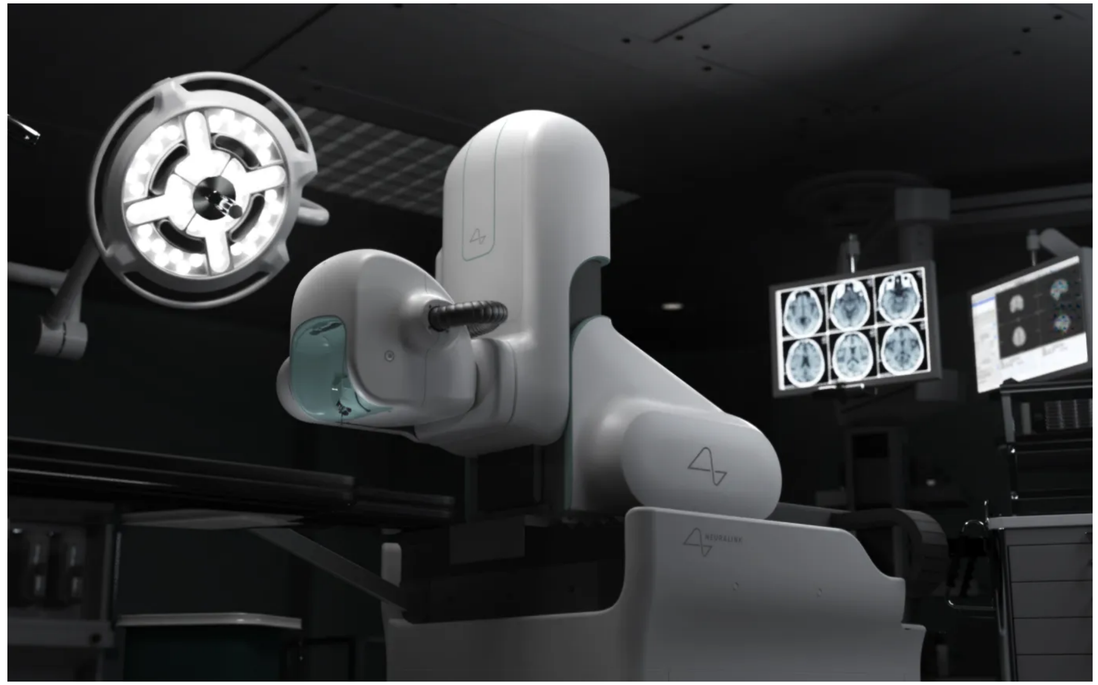


Example: In healthcare, deep learning models are trained to analyze various types of medical images such as X-rays, CT scans, MRI scans, ultrasound images etc.

[1] https://news.mit.edu/2021/artificial-intelligence-tool-can-help-detect-melanoma-0402
[2] https://www.nature.com/articles/s41586-019-1799-6

Applications: Robotics

Neuralink's surgery robot



Tesla's humanoid robot (Optimus)

Example: In robotics, deep learning models provide robots with the ability to perceive and understand their environment through visual information.

[1] https://www.tesla.com/AI
[2] https://techcrunch.com/2020/08/28/take-a-closer-look-at-elon-musks-neuralink-surgical-robot/

Challenges in Computer Vision

Viewpoint variation

Pose variability

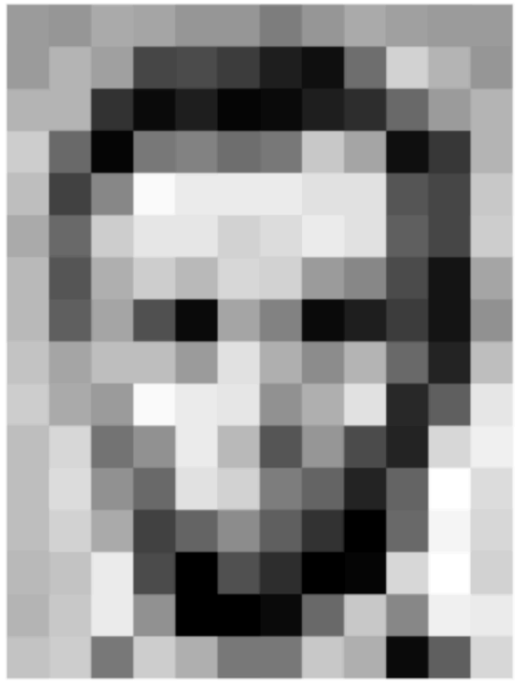
Illumination variation

Intra-class variation

Occlusion

Background clutter

Processing Images

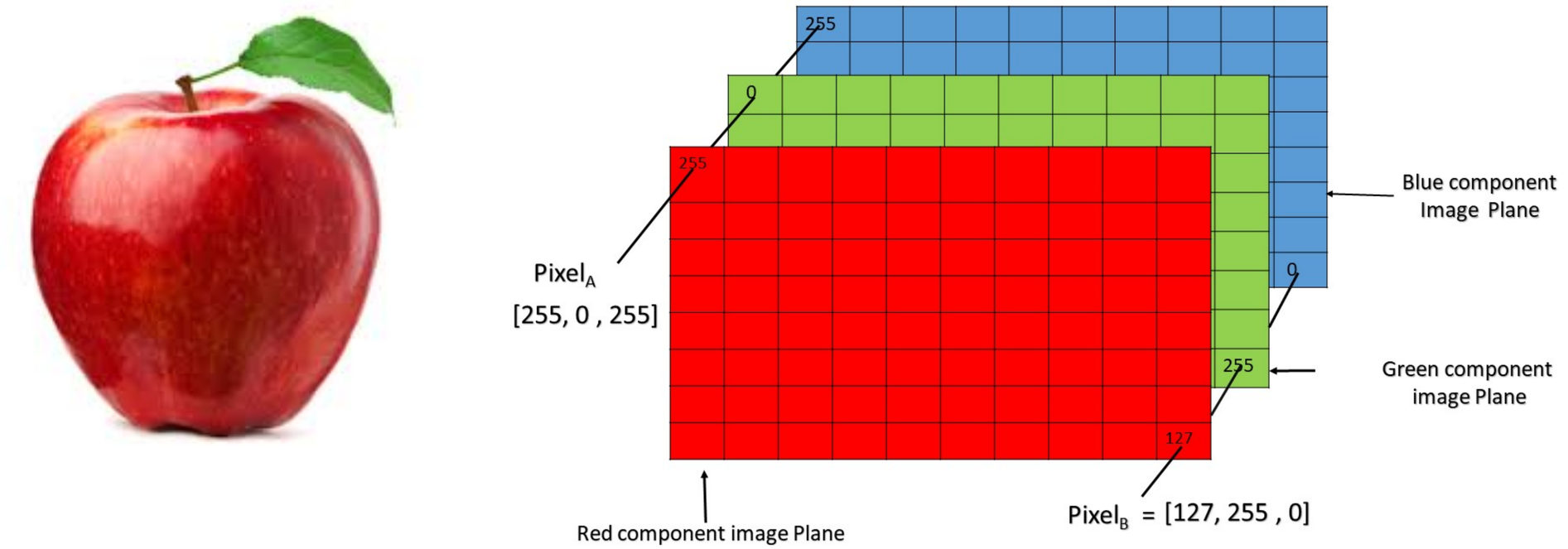


what the computer sees

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
196	206	123	207	177	121	123	200	175	13	96	218

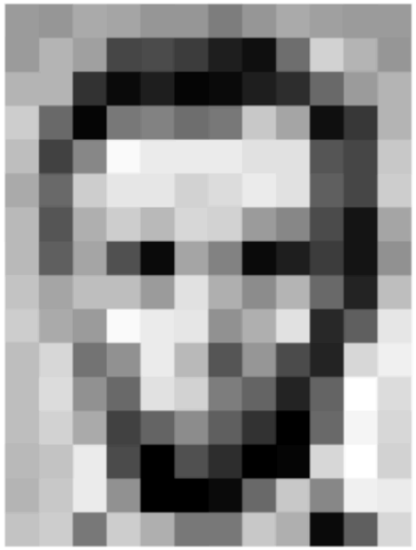
A greyscale image can be seen as a matrix of numbers [0,255], where 0 corresponds to black, indicating no intensity or absence of light and 255 corresponds to white, representing maximum intensity or full brightness.

Processing Images

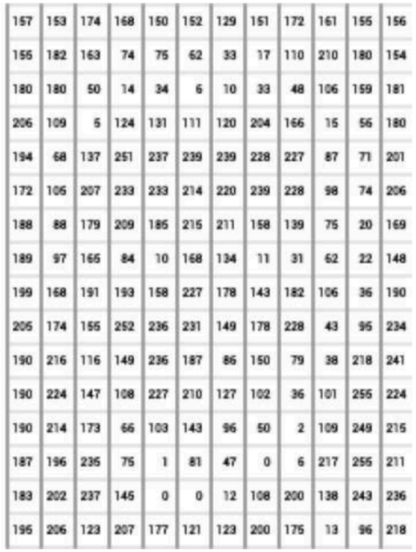


Colored images can be represented using RGB representation, where each pixel in an image is composed of three color channels: red, green, and blue. Each color component ranges from 0 to 255, where 0 indicates no intensity (absence of the color) and 255 indicates maximum intensity (full presence of the color). By combining different intensities of red, green and blue, a wide range of colors can be represented, forming a three-dimensional array (height × width × channels).

Example Computer Vision Task



Input Image



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

Pixel Representation

classification

Lincoln	0.8
Washington	0.1
Jefferson	0.05
Obama	0.05

Example: Image classification task, where the goal is to output the probability of belonging to a particular class.

High Level Feature Detection

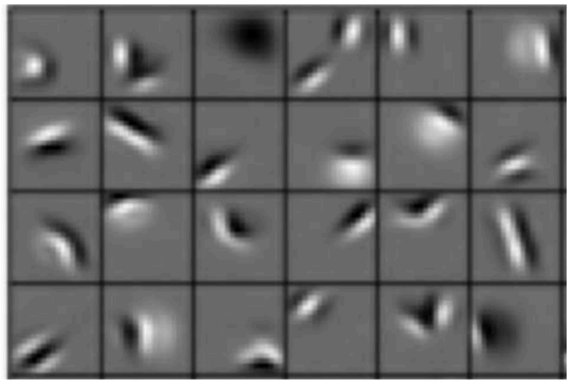
Nose,
Eyes,
Mouth

Wheels,
License Plate,
Headlights

Door,
Windows,
Steps

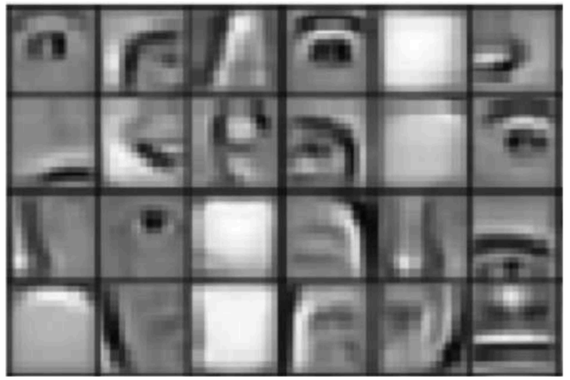
Deep Learning for Feature Extraction

Low level features



Edges, dark spots

Mid level features



Eyes, ears, nose

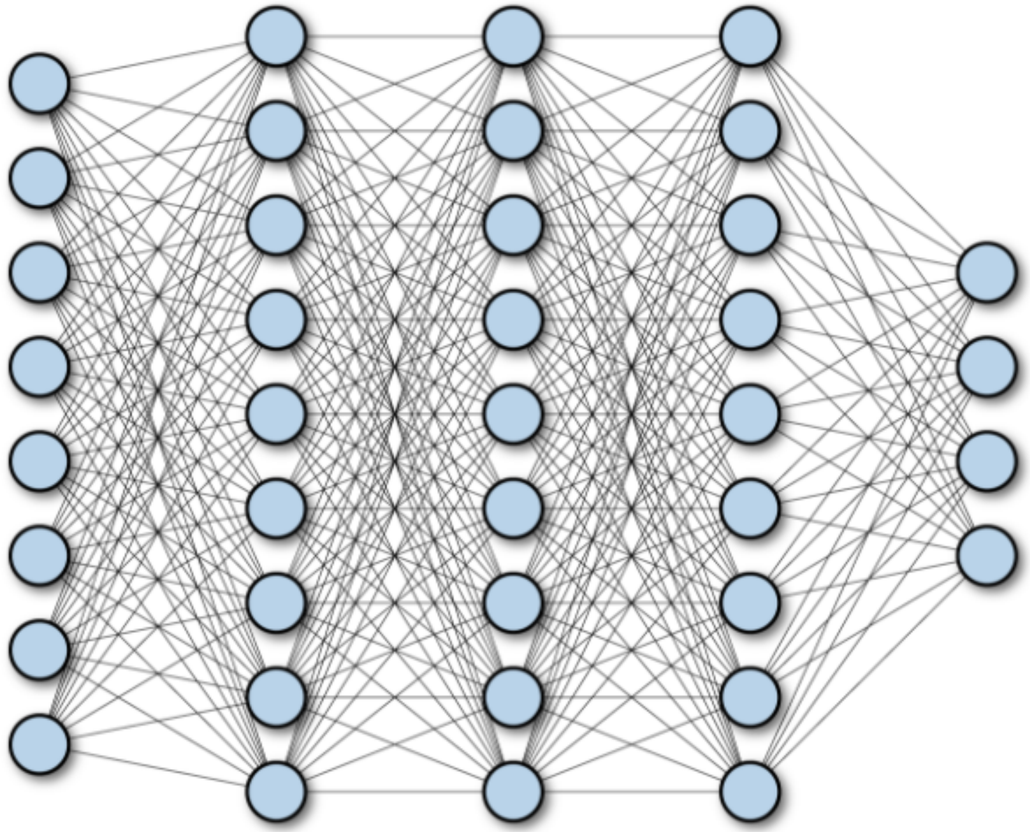
High level features

Facial structure

Example: A neural network trained to recognize complex objects in images may form internal layers that detect useful features: edges, corners, ellipses, eyes, faces.

Fully Connected Neural Network

A fully connected neural network (FCNN) is a type of neural network architecture where each neuron in one layer is connected to every neuron in the next layer.



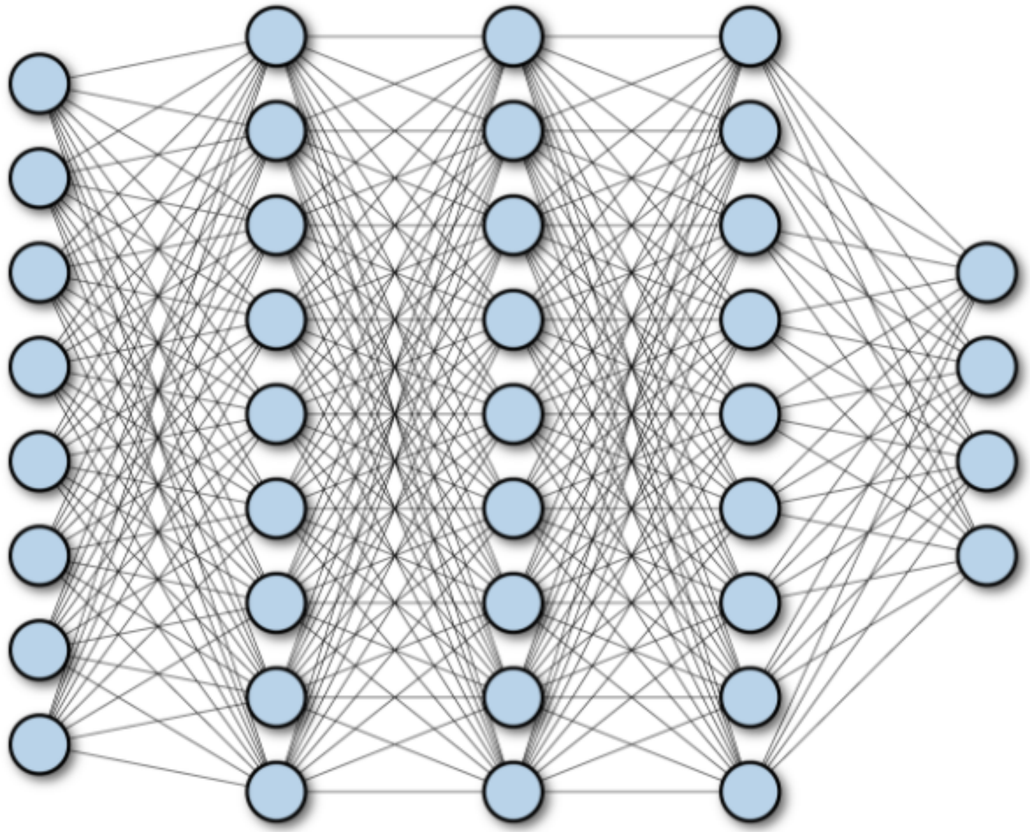
Issues with applying FCNN to computer vision problems:

No spatial information: An image cannot be thought of as a simple vector of input pixel values, primarily because adjacency of pixels really matters. If we were to construct a network with fully connected layers and an image as input, we would get the same result whether we trained with unperturbed images or with images all of whose pixels had been randomly permuted

Fully Connected Neural Network

Too many parameters: Furthermore, suppose there are *n* input features and n units in the first hidden layer, to which the pixels provide input. If the input and the first hidden layer are fully connected, that means n^2 weights. For a typical megapixel RGB image (n = 1000 x 1000 x 3), that's 9 trillion weights.

Such a vast parameter space would require correspondingly vast numbers of training images and a huge computational budget to run the training algorithm.

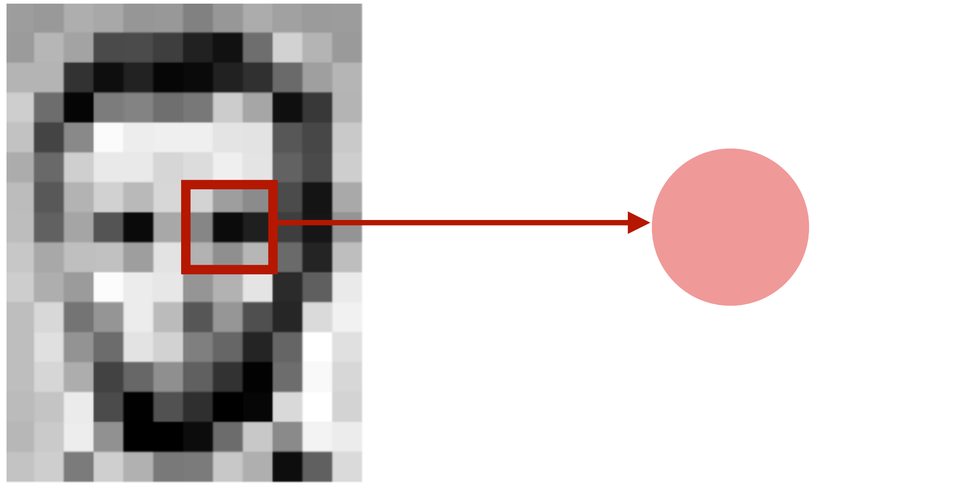


Make Use of Spatial Structure

These considerations suggest that we should construct the first hidden layer so that each hidden unit receives input from only a small, local region of the image.

- respects adjacency, at least locally
- it cuts down the number of weights: if each local region has $l \ll n$ pixels, then there will be $ln \ll n^2$ weights in all

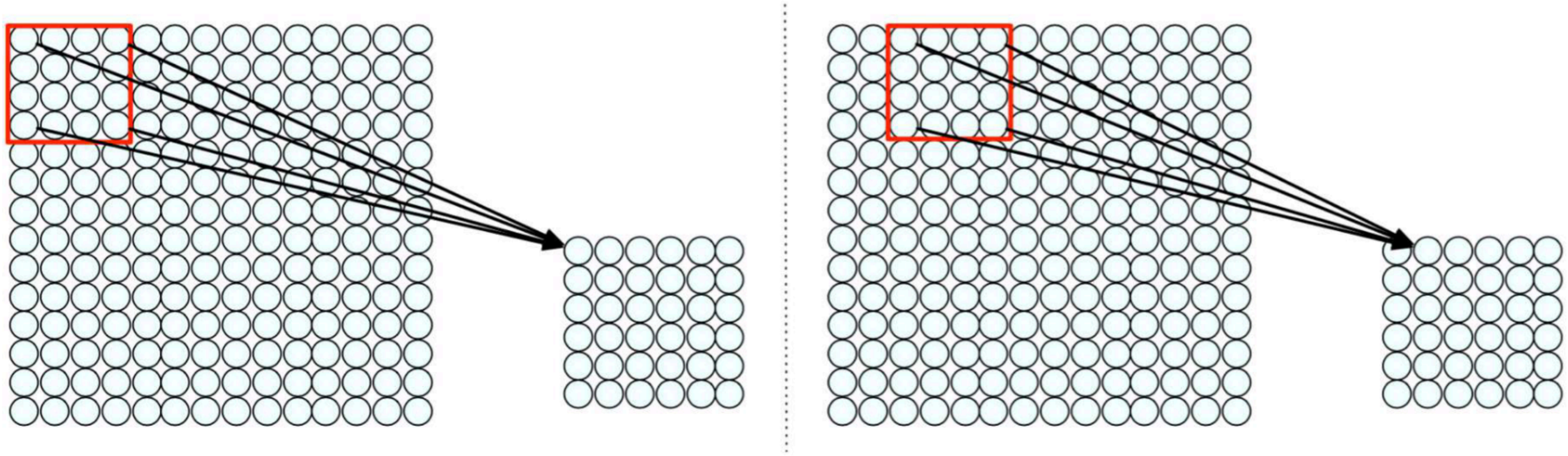
Thus, by injecting some prior knowledge we can develop models that have far fewer parameters and can learn much more quickly.



Input image (2D) Neuron in hidden layer

Idea is to connect the pixels that are part of a patch in the input layer to a single neuron in the subsequent layer. The neuron can only see this region's values. Example of a 3x3 patch (9 pixels/ inputs) connected to one hidden layer neuron.

Make Use of Spatial Structure



Connect patch in input layer to a single neuron in subsequent layer.

Convolution Operation

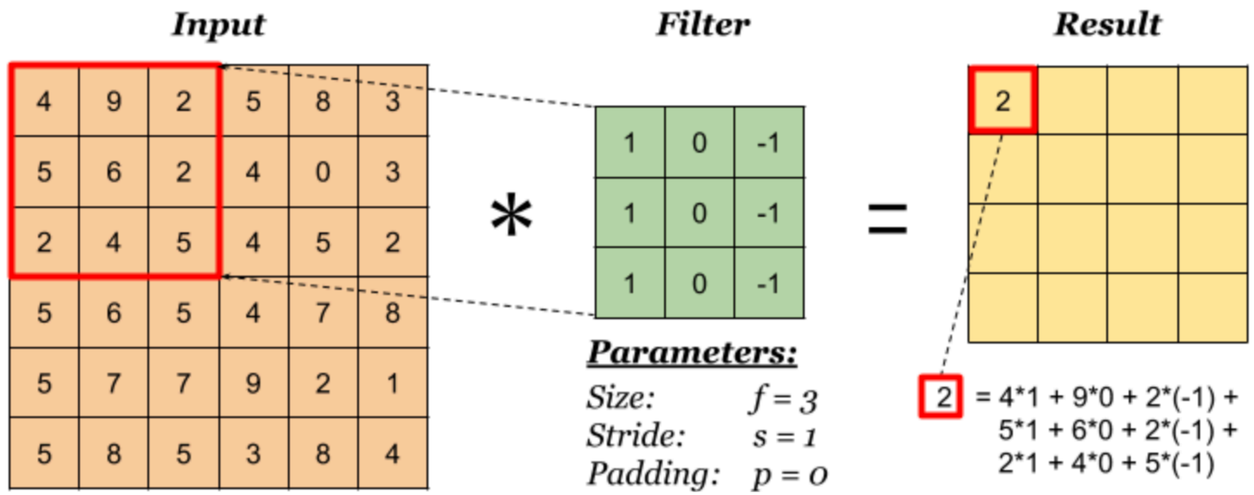
Convolution Operation

A pattern of weights that is replicated across multiple local regions is called a filter (or kernel) and the process of applying the filter to the pixels of the image (or to spatially organized units in a subsequent layer) is called convolution.

- overlay the filter to the input, perform element wise multiplication, and add the result

The output dimension is calculated with the following formula ($\lfloor \rfloor$ is floor operation):

$$n^{[l]} = \lfloor \frac{n^{[l-1]} + 2p^{[l-1]} - f^{[l]}}{s^{[l]}} + 1 \rfloor$$



Input dimension: 6 x 6

feature map

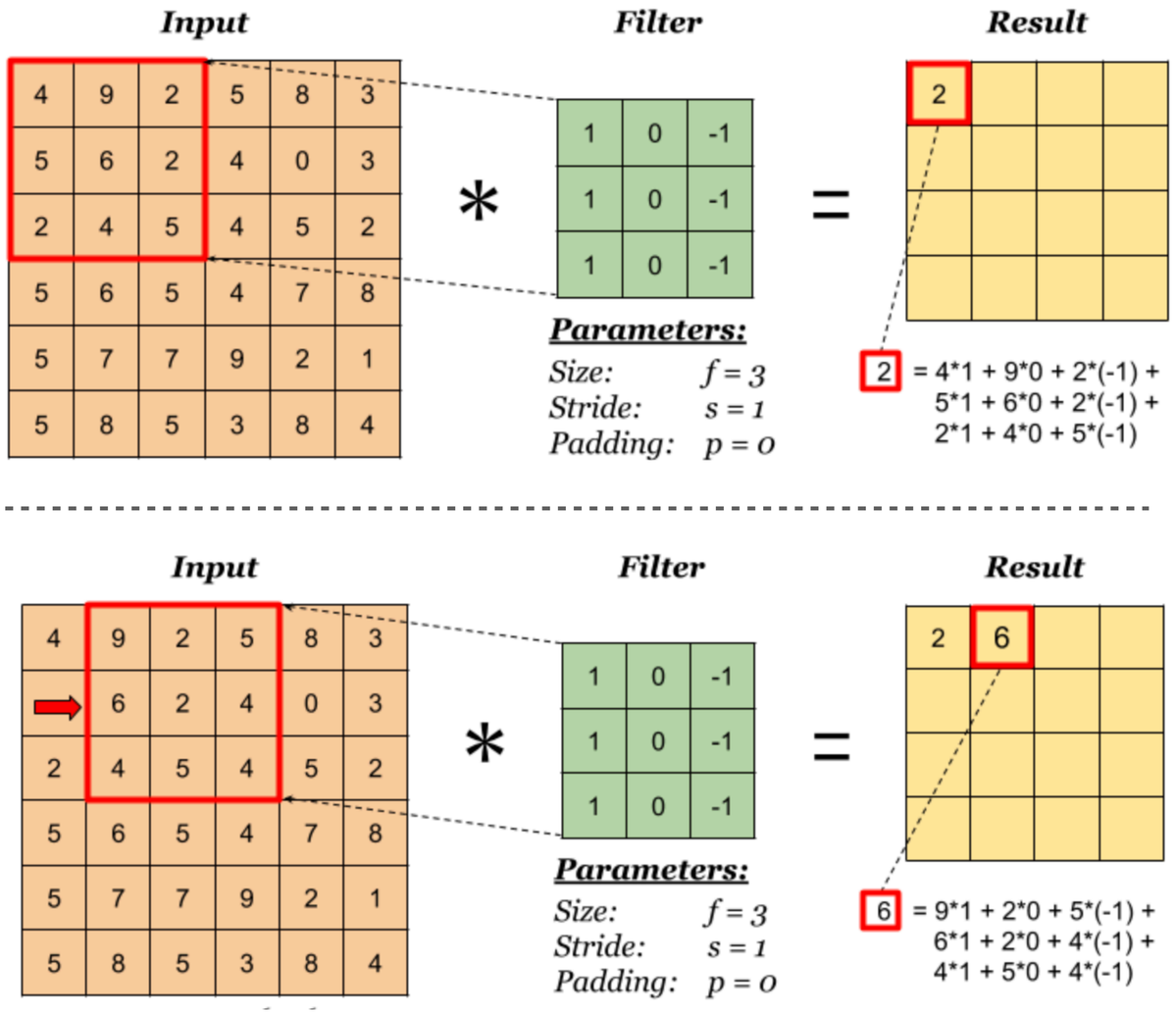
Convolution Operation

At each next step, the filter is sliding sequentially over a different patch of the input image.

In traditional image processing techniques, filters were handcrafted based on domain knowledge. With deep learning, they are weights to be learned directly from the data.

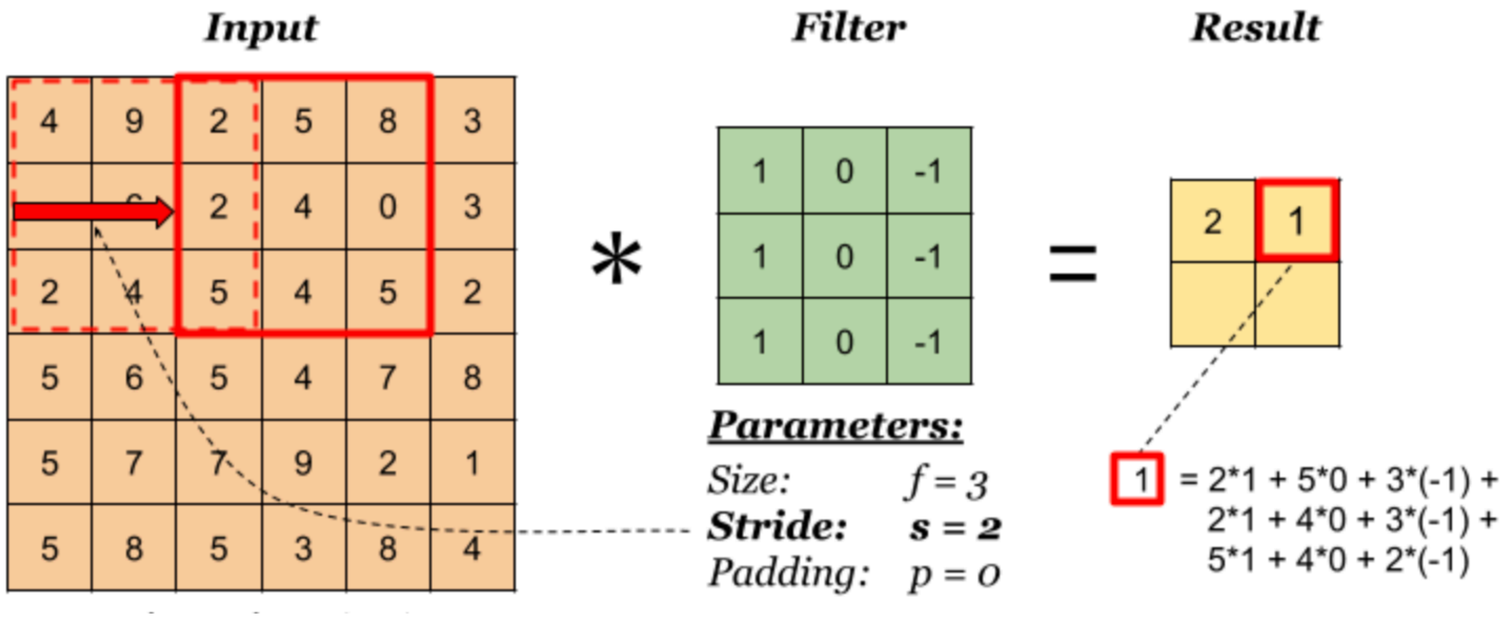
Why convolutions:

- Parameter sharing: a feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.
- Sparsity of connections: in each layer, each output value depends only on small number of inputs



Stride

Stride governs how many cells the filter is moved in the input to calculate the next cell in the result.



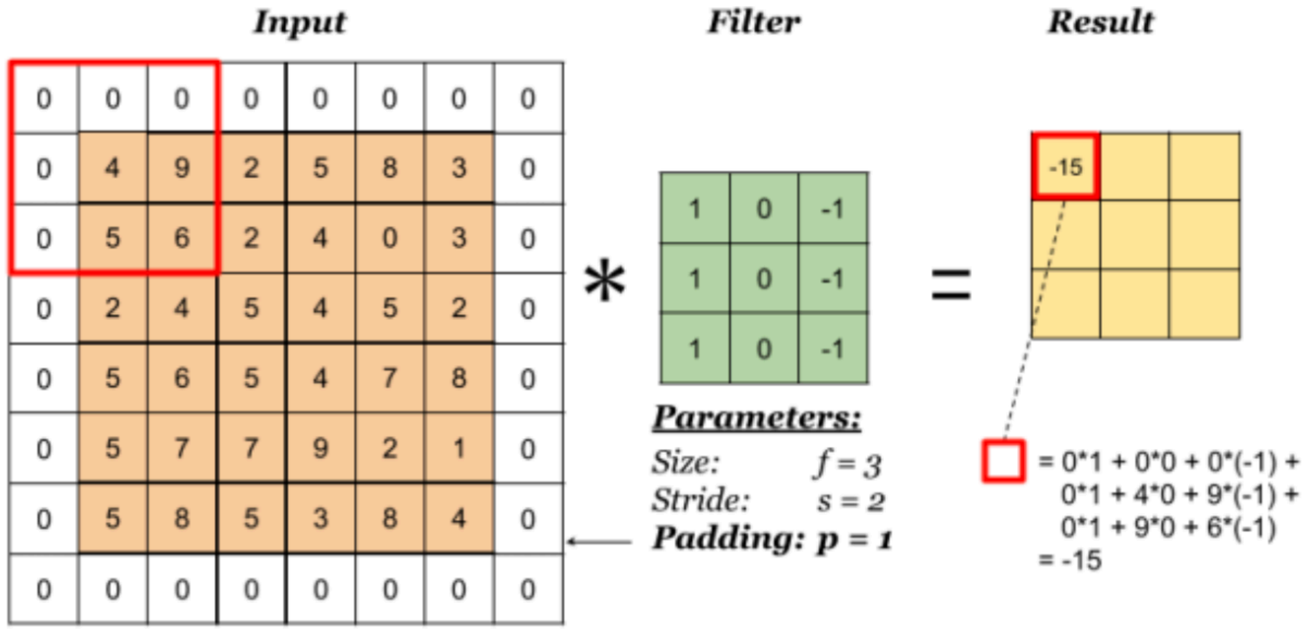
Example: Filter with stride s = 2

Padding

With padding we add additional pixels around the input data before applying a convolution operation.The purpose is to control the spatial dimensions of the output.

- same means that the padding is adjusted so that the input and output have the same spatial dimensions
- valid means no padding is added to the input

Without padding, very few values at the next layer would be affected by pixels as the edges of an image.

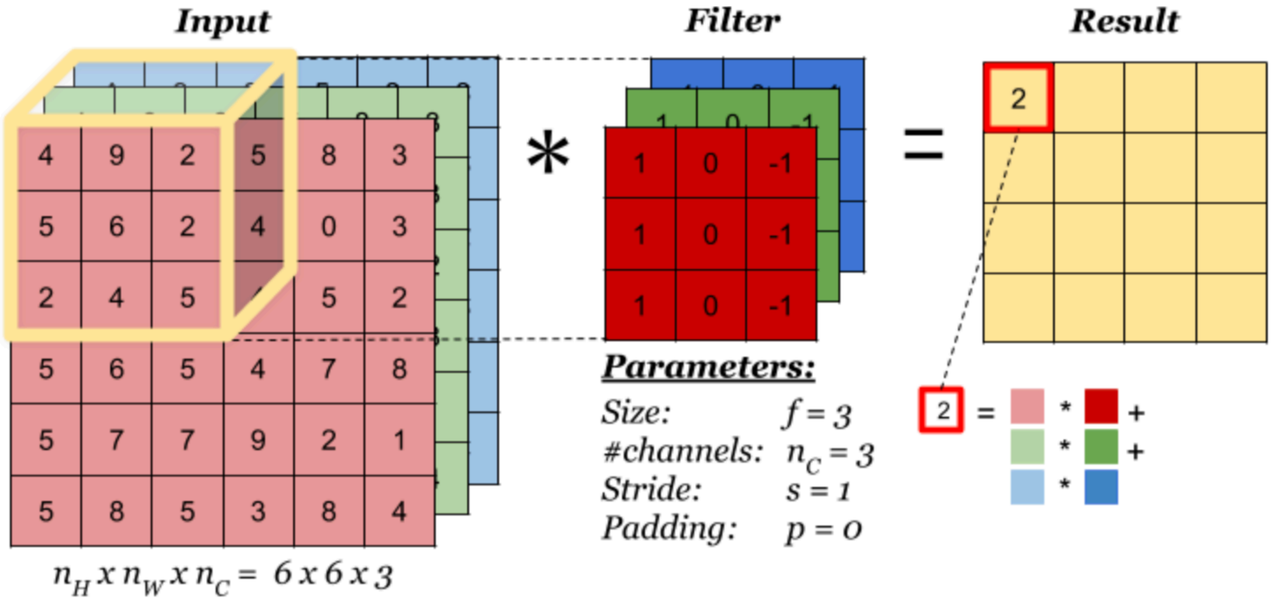


Example: Padding of size one - a single layer of pixels is added around the input image, by default set to 0.

Convolution Operation on Volume

When the input has more than one channels (e.g. an RGB image has 3 chanels, red, green and blue), the filter should have matching number of channels.

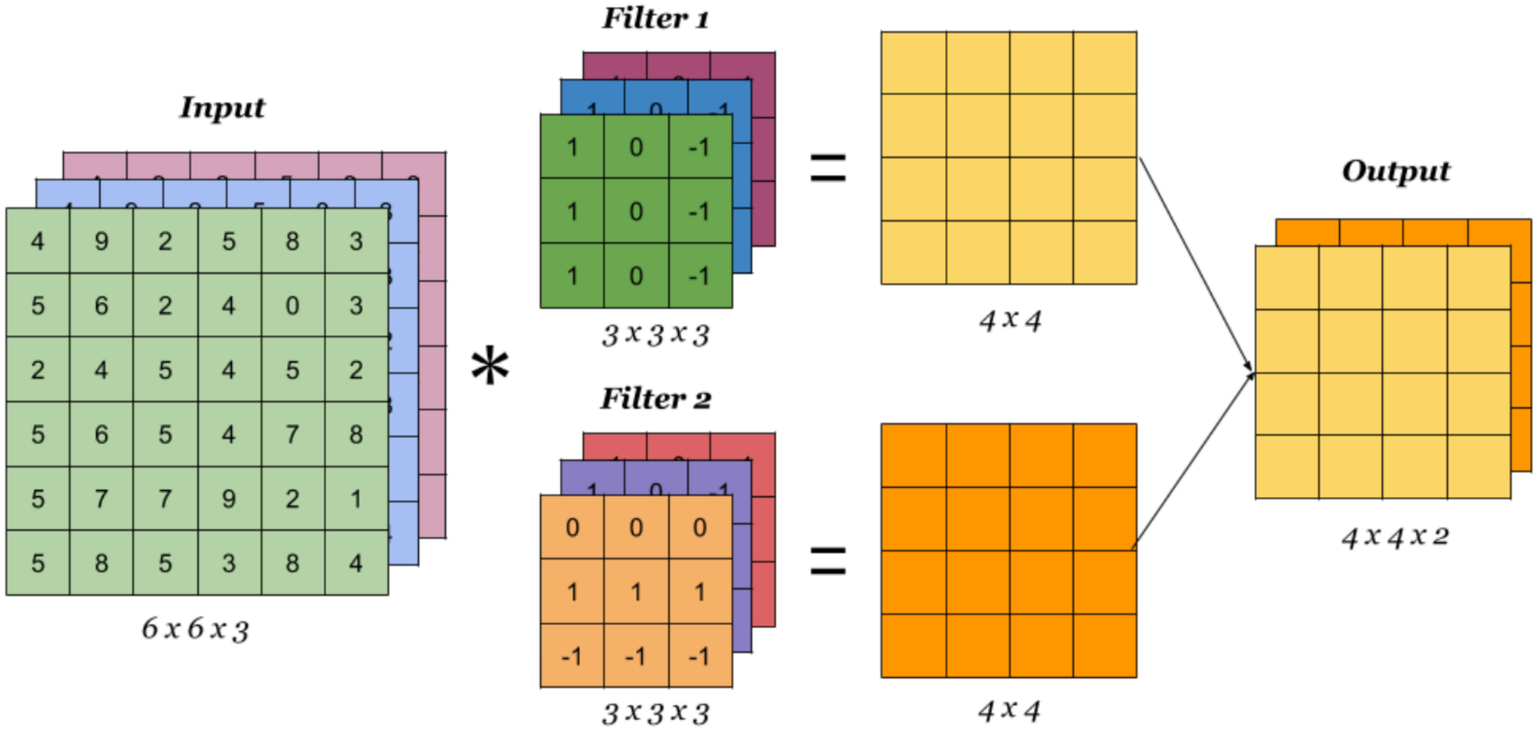
In order to calculate one output cell, a convolution is performed on each matching channel, then the result is added together.



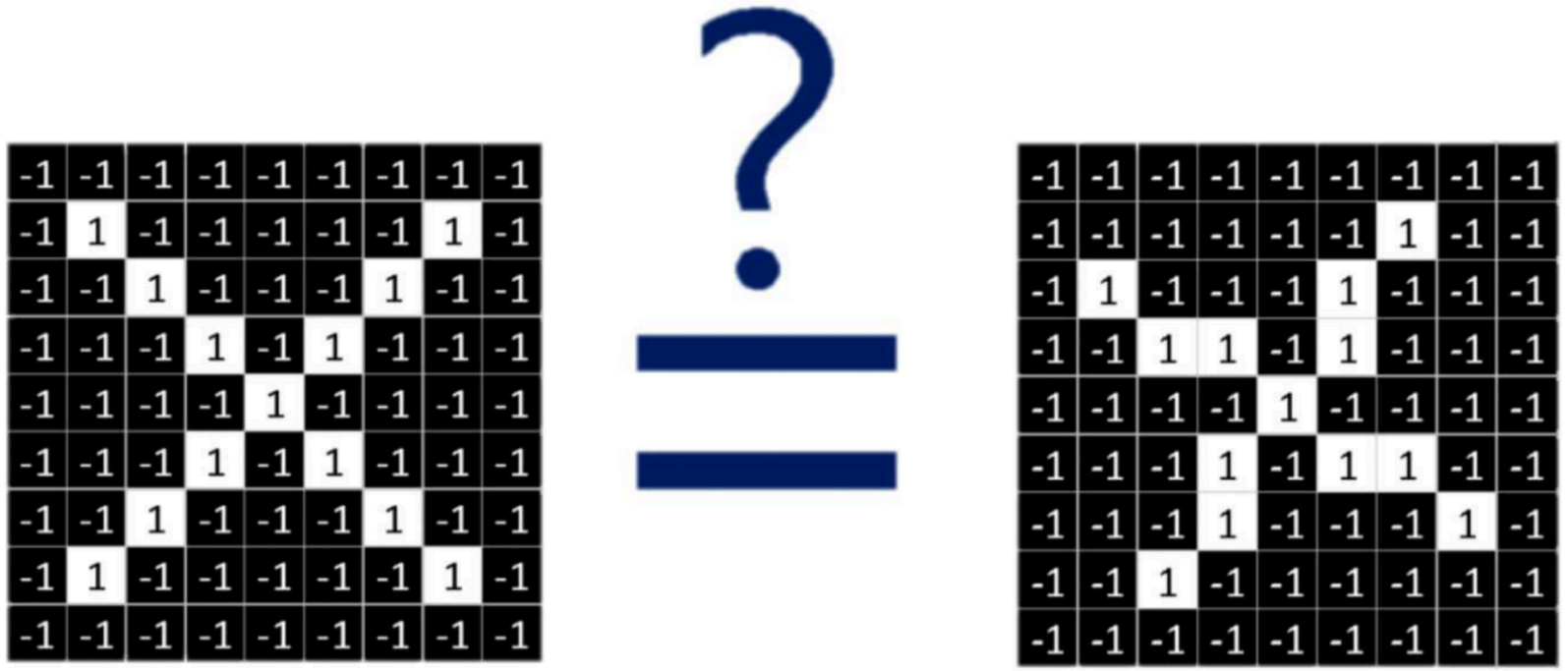
Example: Padding of size one - a single layer of pixels is added around the input image, by default set to 0.

Convolution Operation with Multiple Filters

Multiple filters can be used in a convolution layer to detect multiple features. The output of the layer then will have the same number of channels as the number of filters in the layer.

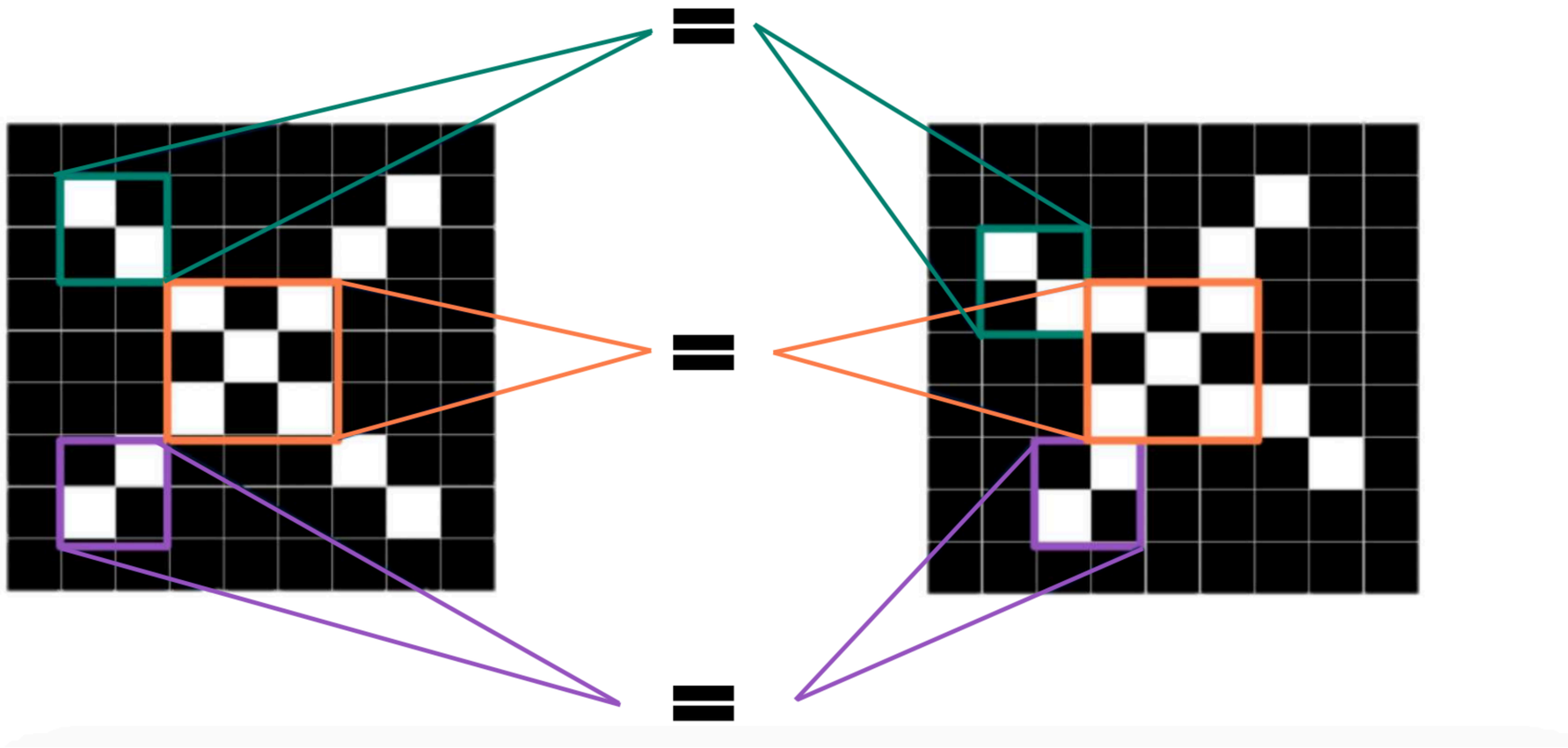


Example of Feature Extraction with Convolution



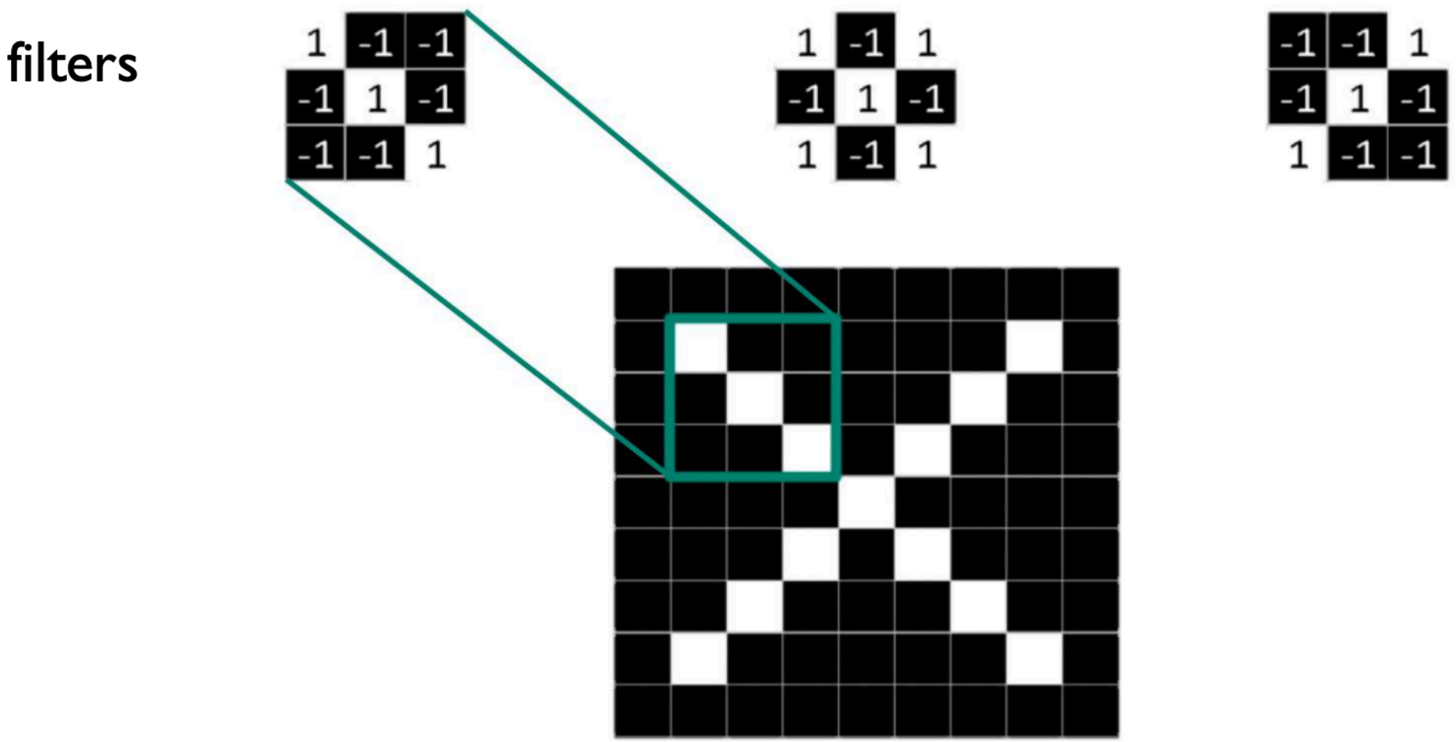
Example: The input image is represented as matrix of pixel values, for simplicity only two values are used -1 for black and 1 for white color. We want to be able to classify the X as the X even if it's shifted, shrunk, rotated, deformed.

Example of Feature Extraction with Convolution



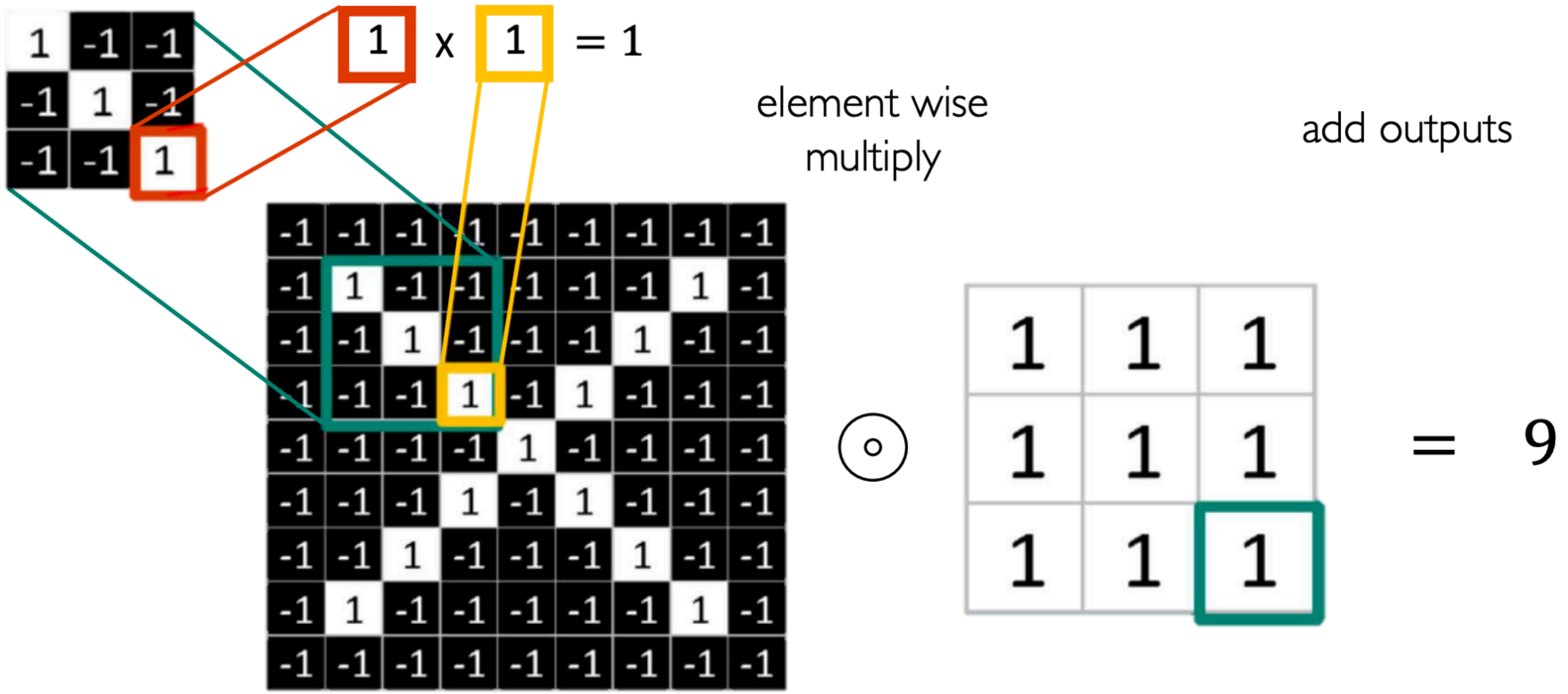
Example: Detect whether important features can be found approximately on the same spot for each image

Example of Feature Extraction with Convolution



Example: Filters to detect important features for X such as diagonal lines or crossing at the centre

Example of Feature Extraction with Convolution

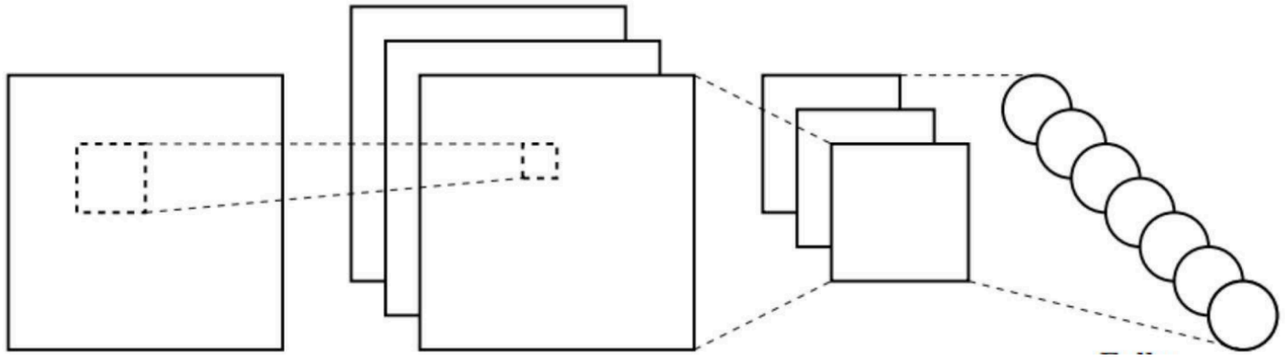


Example: Use convolution to overlay the filter to the input, perform element wise multiplication, and add the result

Convolution Neural Networks (CNNs)

Convolutional Neural Networks

A convolutional neural network (CNN) is a neural network that contains spatially local connections, at least in the early layers and has patterns of weights (filters) that are replicated across the units in each layer.



Example: Overview of a Convolutional Neural Network (CNN)

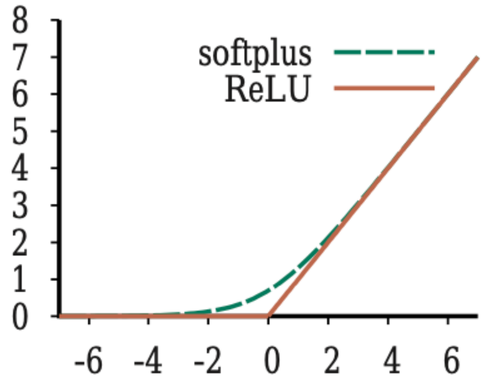
Types of layers in a convolutional network:

- Convolution
- Pooling
- Fully connected layer

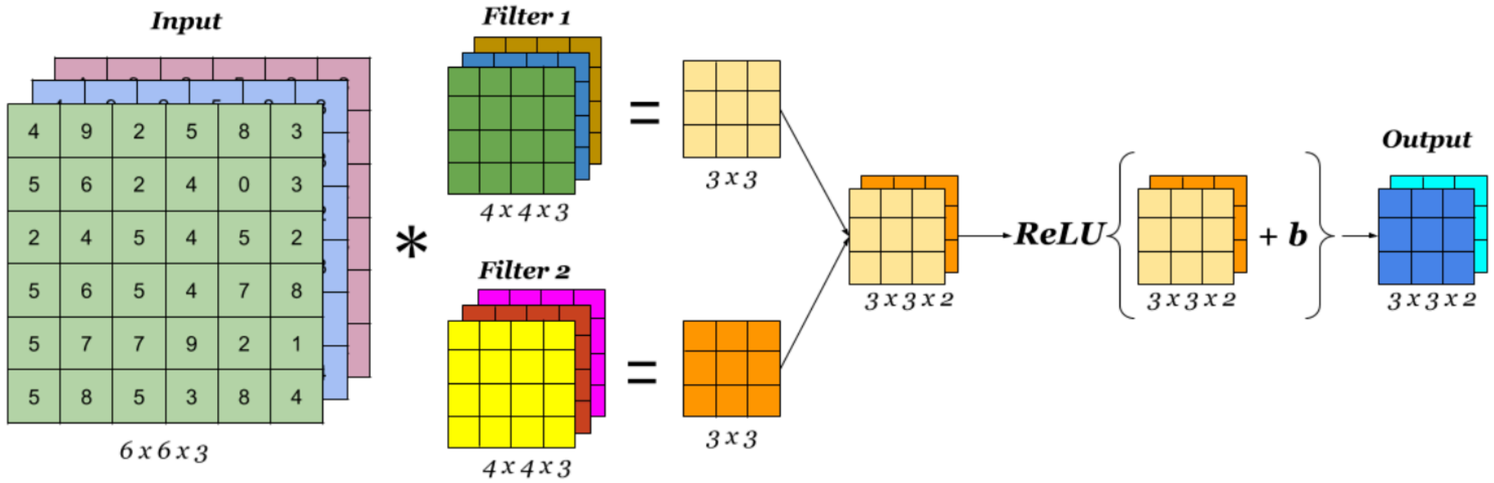
One Convolution Layer

The convolutional layers are used for feature extraction. To make up a convolution layer, a bias (ϵ R) is added and an activation function such as ReLU (rectified linear unit), that introduces non-linearity, is applied.

The receptive field of a neuron is the portion of the sensory input that can affect that neuron's activation.



$$\mathrm{ReLU}(x) = \max(0, x)$$



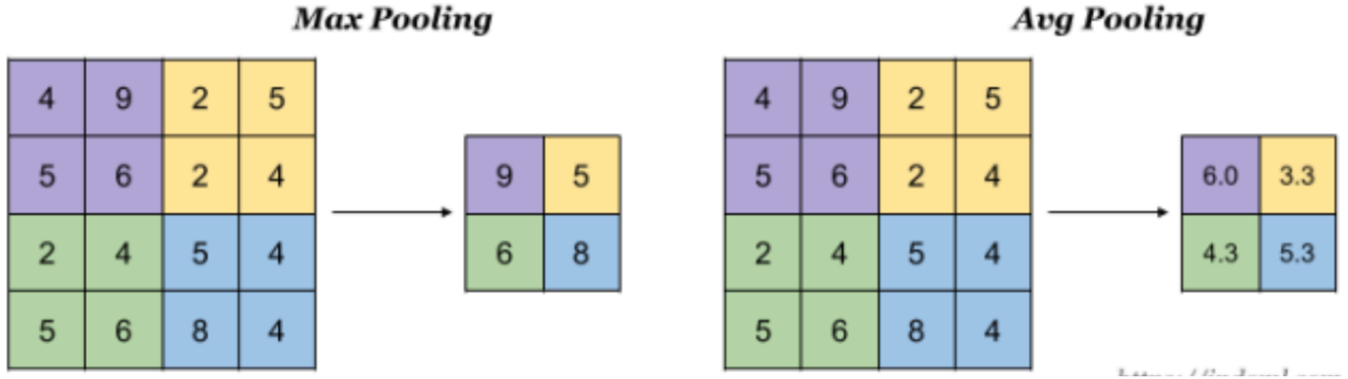
Example: One Convolution Layer

Pooling Layer

A pooling layer in a neural network summarizes a set of adjacent units from the preceding layer with a single value. Pooling works just like a convolution layer, but the operation that is applied is fixed rather than learned (it doesn't have parameters).

Common forms of pooling:

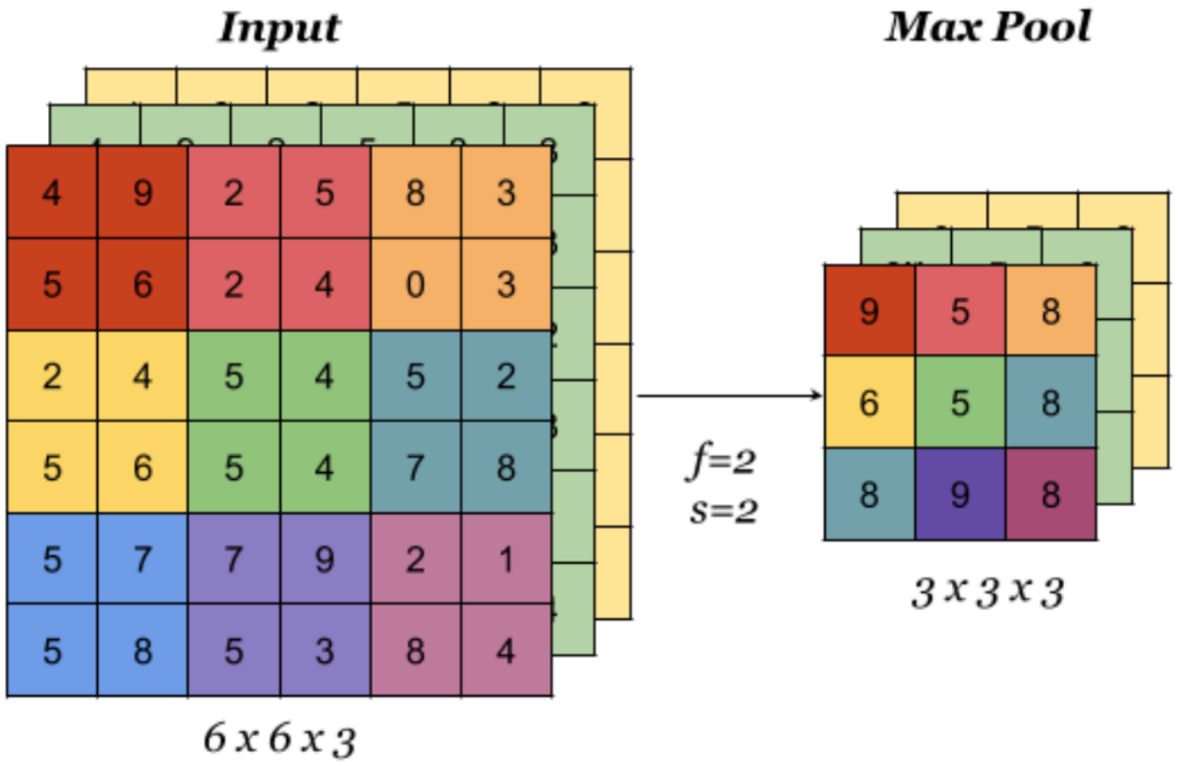
- Average-pooling computes the average value of its inputs
- Max-pooling computes the maximum value of its inputs (more widely used)



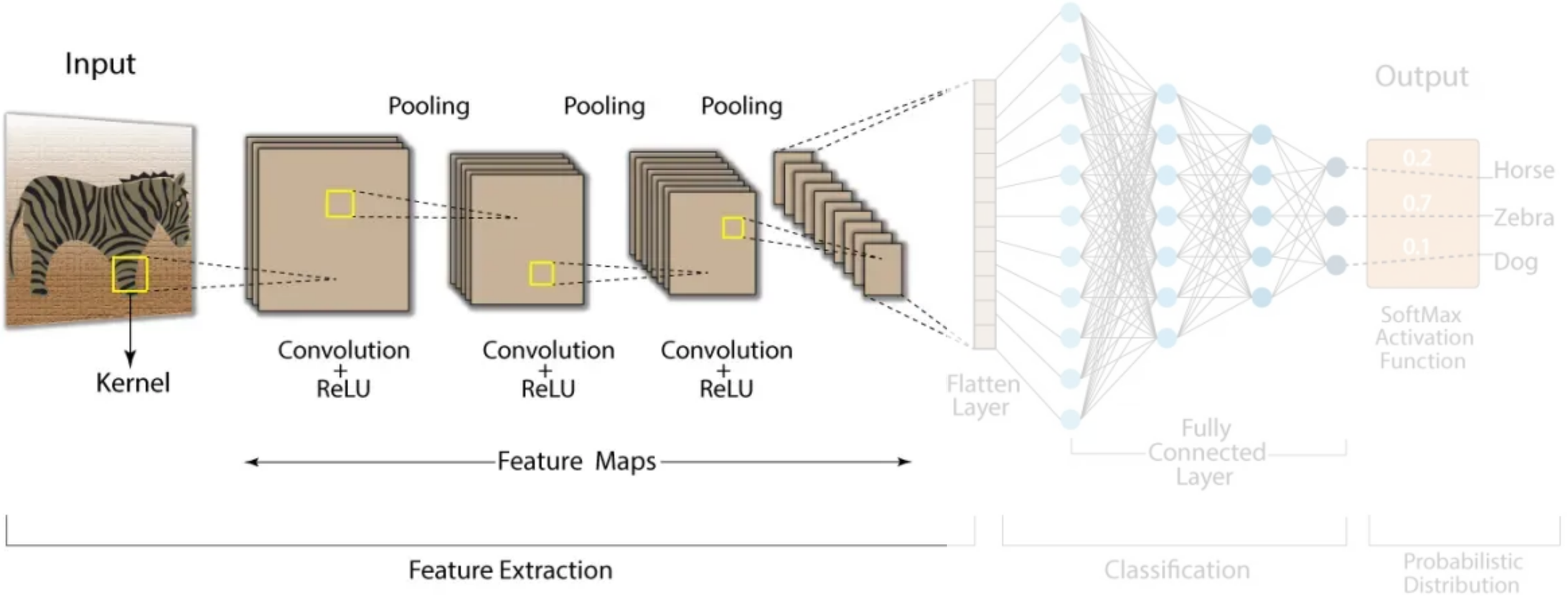
Example: Pooling with 2x2 filters and stride 2
The goal of the pooling layer is to downsample (reduce) the dimensionality of the input. This also reduces the number of weights required in subsequent layers, leading to lower computational cost and possibly faster learning.

Pooling Layer

When done on input with multiple channels, pooling reduces the height and width of the representation but keeps the number of chanels unchanged.

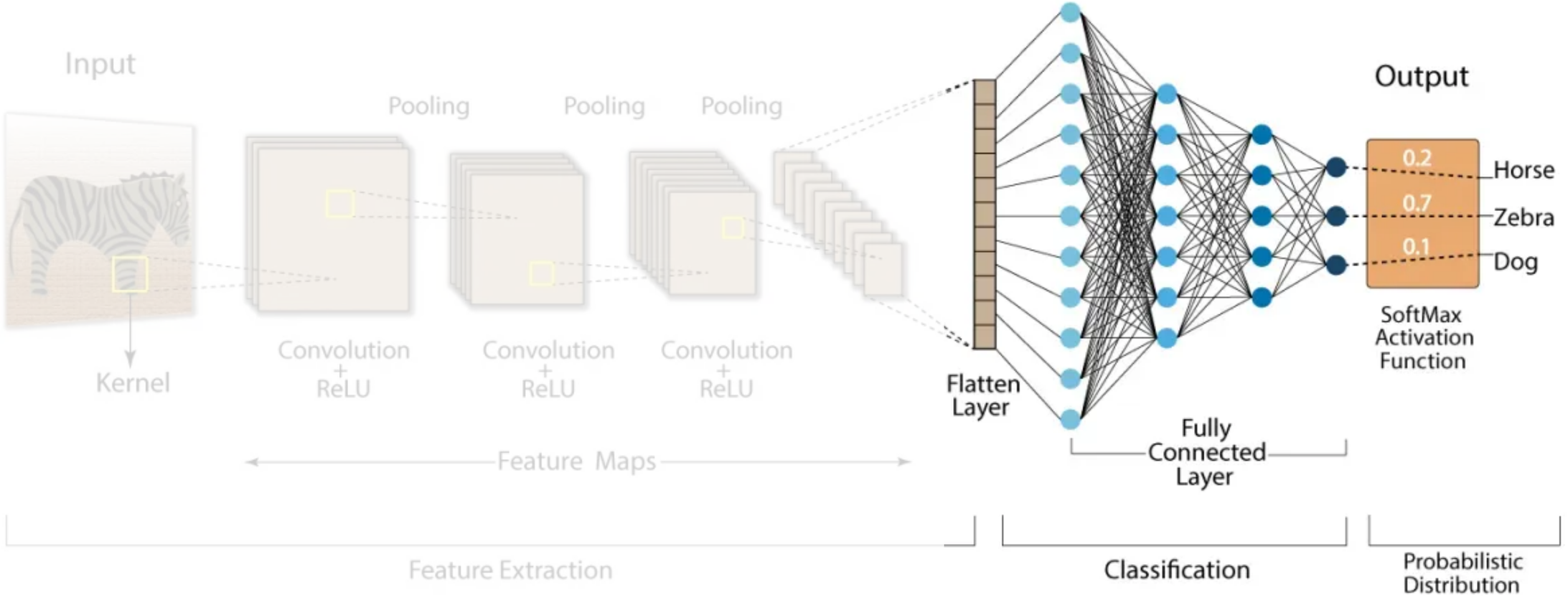


CNN for Classification: Feature Learning



1. Learn features in input image through convolution
2. Introduce non-linearity through activation function (real-world data are non-linear)
3. Reduce dimensionality using pooling layers

CNN for Classification: Class output

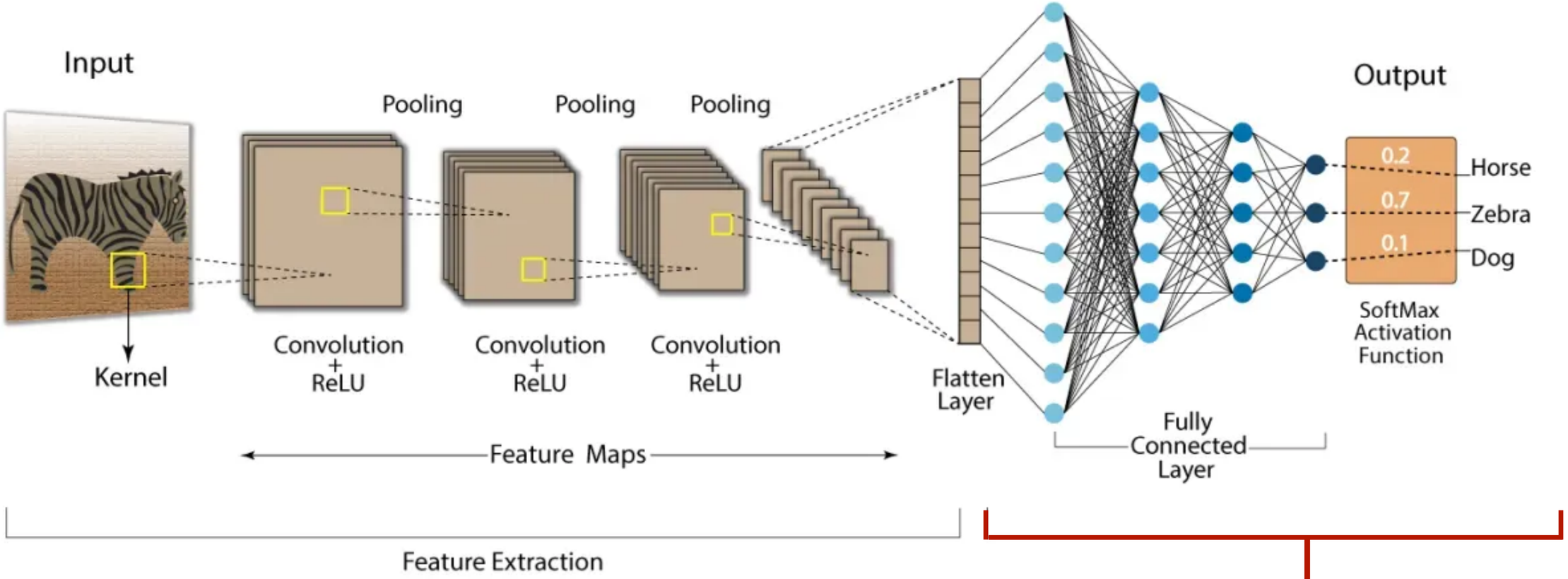


1. The convolutional and pooling layers output high-level features of input
2. Fully connected layer uses these features for classifying the input image
3. Express output as probability of image belonging to a particular class

$$\text{softmax}(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

convert output scores from a model into probabilities that represent the likelihood of each class

CNN for Many Applications



Classification

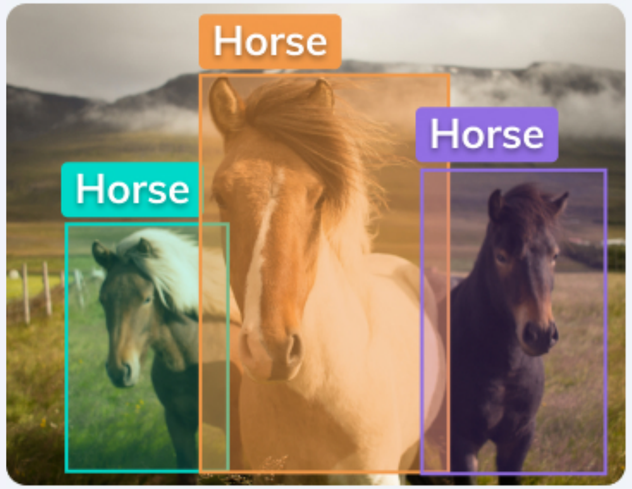
Regression

Object Detection

Segmentation

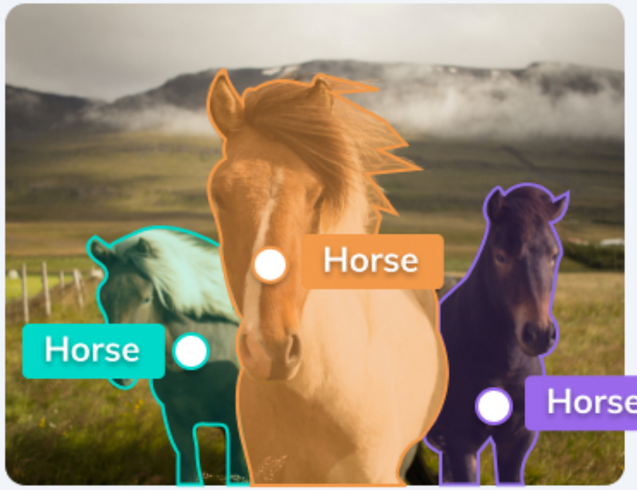
...

CNN for Many Applications



Object Detection

Semantics Segmentation



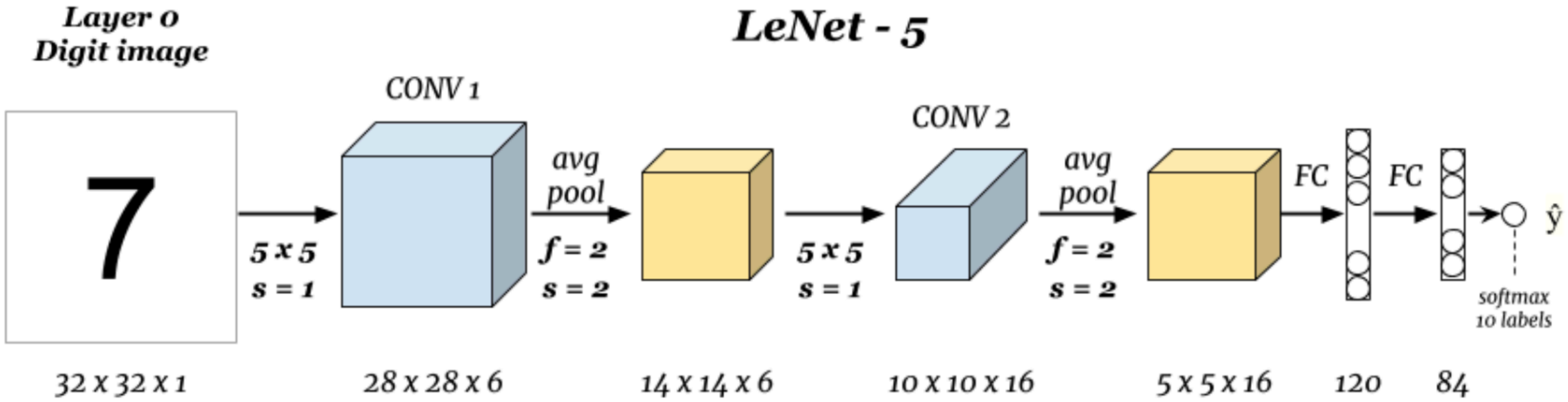
Instance Segmentation

The feature extraction part can be used in combination with fully connected layer and output for different computer vision tasks such s classification, regression, object detection, segmentation etc.

Example Architectures

LeNet – 5

LeNet-5 (1998) is a pioneering convolutional neural network architecture that was one of the first successful CNN architectures applied to handwritten digit recognition tasks.

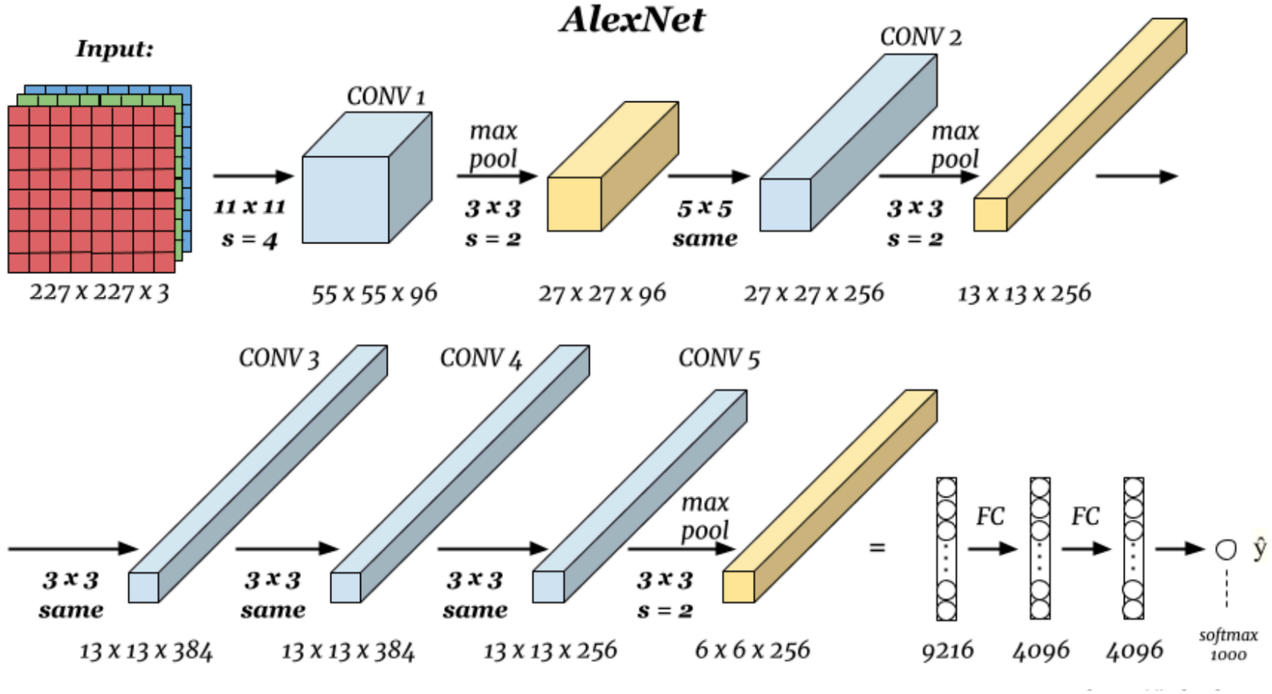


Number of parameters: ~ 60 thousands.

[1] LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), pp.2278-2324.

AlexNet

AlexNet (2012) achieved significant breakthroughs in image classification tasks, winning the ImageNet competition with a substantial margin. Improvements included, among other, the use of the ReLU activation function, dropout and efficient training on multiple GPUs in parallel.

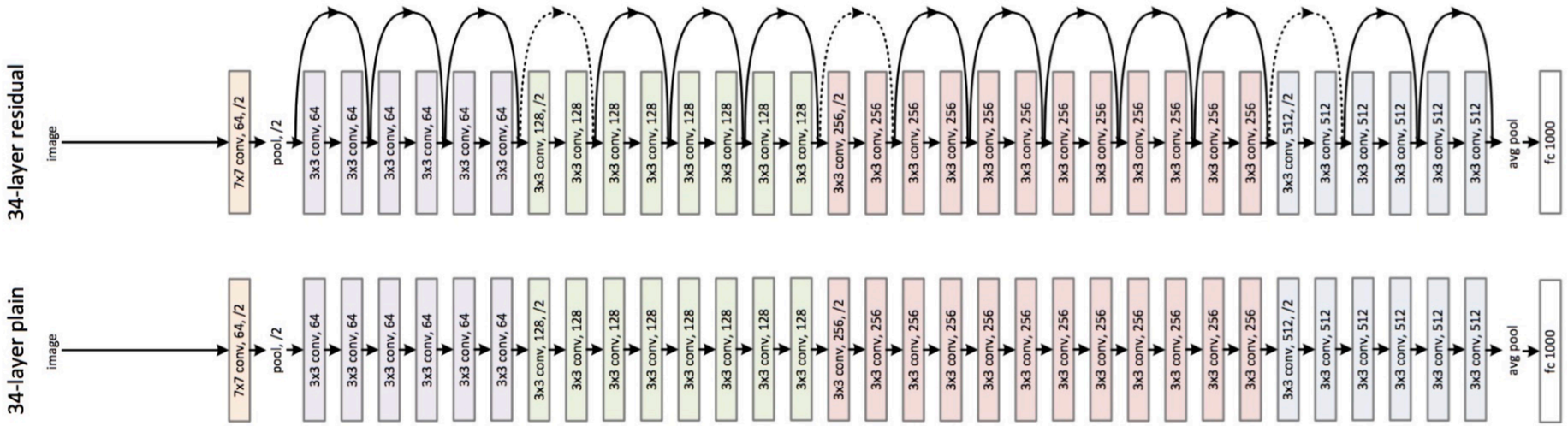


Number of parameters: ~ 60 millions.

[1] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*.

ResNet

ResNet (Residual Network) (2016) addresses the problem of vanishing gradients in very deep neural networks by introducing skip connections or residual connections, where output from one layer is fed to layer deeper in the network. This allowed for the training of much deeper networks than was previously possible.

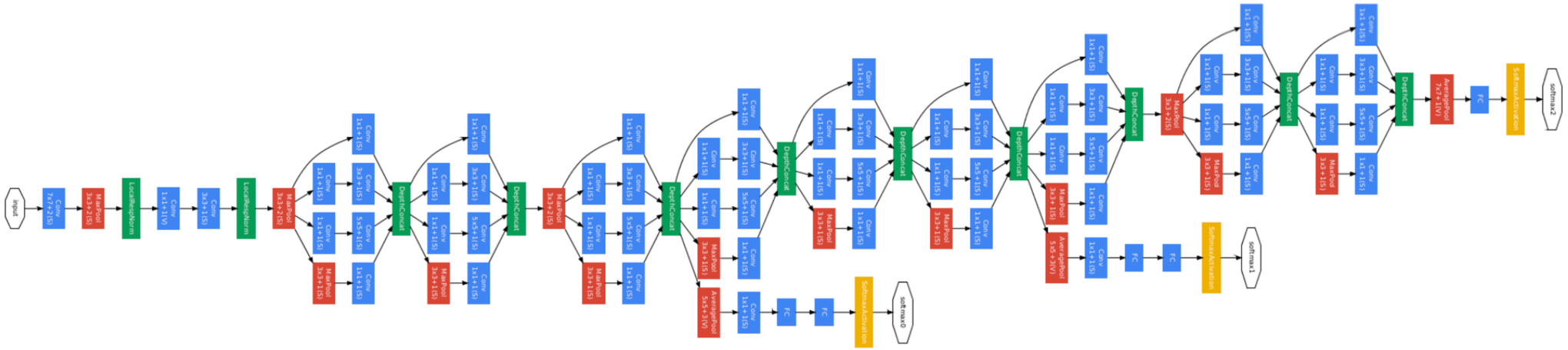


Comparison of 34 layers ResNet with plain network

[1] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

Inception (GoogLeNet)

The Inception (GoogLeNet) network (2015), is known for its use of inception modules, which capture features at multiple spatial scales using filters of different sizes within the same layer.



GoogleLeNet architecture

[1]Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

Benchmark Datasets

The performance of the deep learning algorithms for computer vision tasks are commonly evaluated on publicly available benchmark datasets such as ImageNet, COCO (Common Objects in Context), PASCAL VOC (Visual Object Classes), Cityscapes etc.

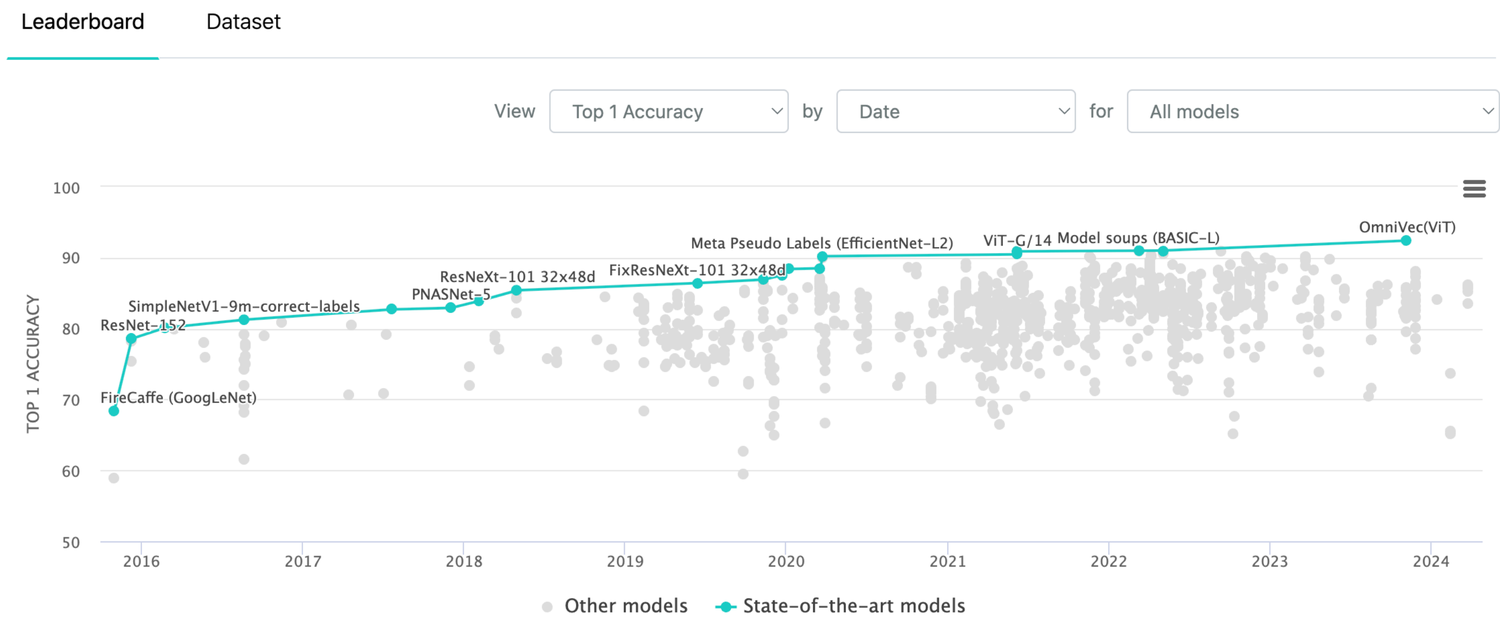


Image Classification on ImageNet (leaderboard)

[1https://paperswithcode.com/sota/image-classification-on-imagenet

Data Enhancement

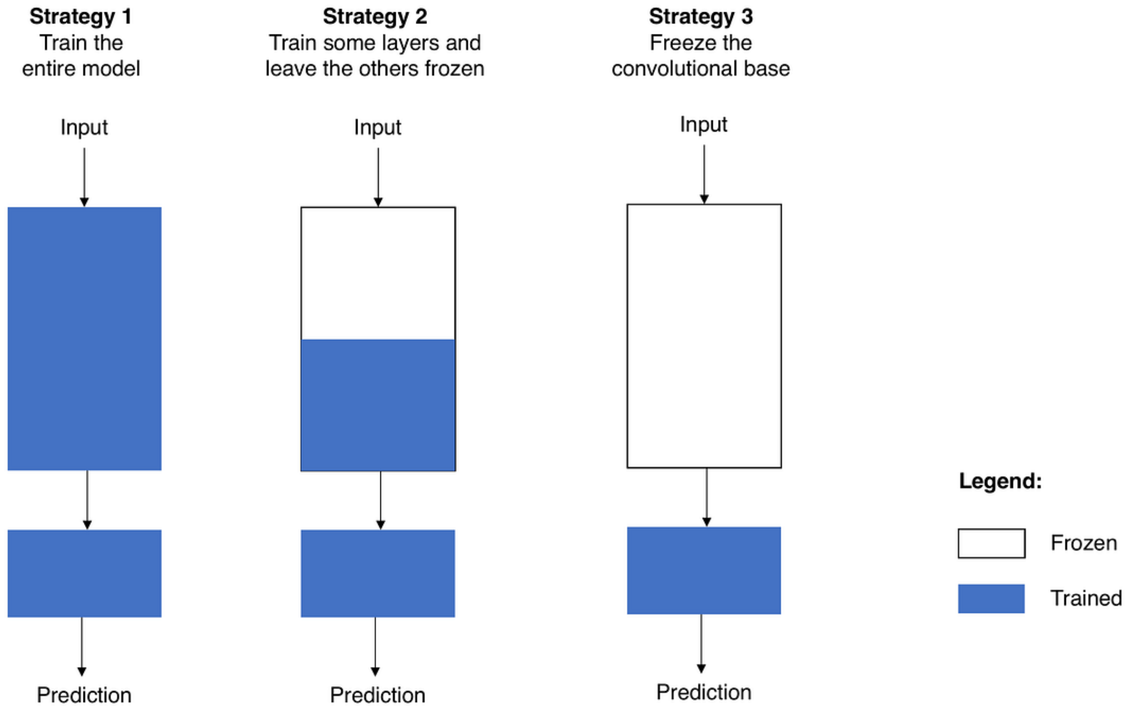
Transfer Learning

In transfer learning, experience with one learning task helps an agent learn better on another task.

For neural networks, learning consists of adjusting weights, so the most plausible approach for transfer learning is to copy over the weights learned for task A to a network that will be trained for task B. The weights are then updated by gradient descent in the usual way using data for task B.

Common approach with pretrained model:

- freeze the first few layers of the pretrained model that serve as feature detectors
- modify the parameters of the higher levels only



Example (left to right): a. Use the weights of the pretrained model only for initialising the weights, b. train some layers of the network and freeze the others, c. freeze the feature extraction part and only train the last fully connected and output layers. Freezing layers means that you prevent the weights of certain layers from being updated during training.

Data Augmentation

Data augmentation artificially increase the size of the training dataset by applying various transformations to the existing data.

These transformations include operations such as mirroring, cropping, colour shifting, rotations, etc.

It helps improve model generalization and robustness by exposing it to diverse variations of the input data.

Example: Transformations to the original image including flipping (mirroring), rotating, cropping, colour distortion, sharpening and random erasing.

Appendix