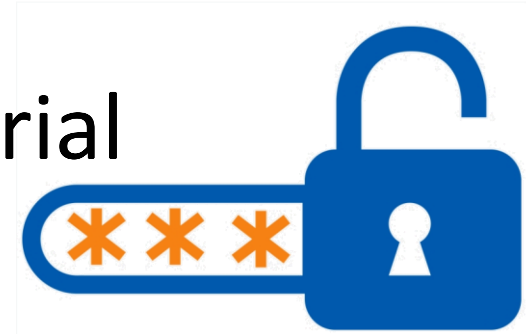
Lebanese University
Faculty of Science
M2 Computer Science
Cybersecurity

Image Classification using CNN

Neural Networks Tutorial



1. Experiment Goal
2. Dataset
3. CNN Architecture
4. Experimental Environment
5. Experiment Road Map
6. Experiment

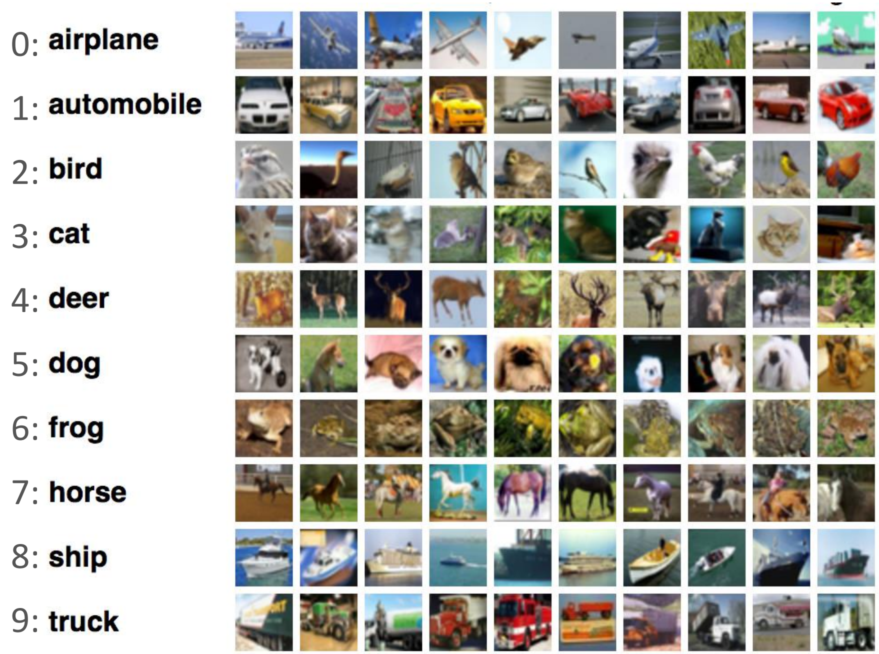
Experiment Goal

- In this lab, you will discover how to develop a CNN model for object photo classification.
- This experiment recognizes image categories based on the CIFAR10 dataset.

Problem Definition
↓
Data Preparation
↓
Architecture Design
↓
Training CNN algorithm
↓
Testing Trained Model
↓
Model Deployment

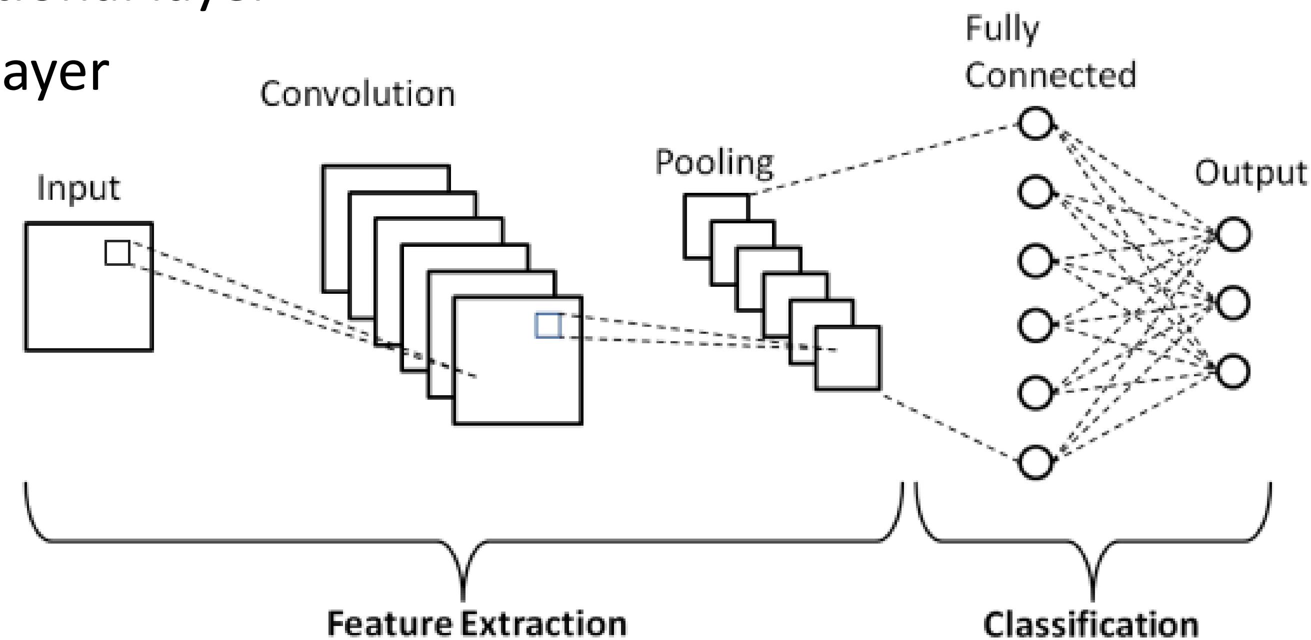
CIFAR-10 dataset

- Canadian Institute For Advanced Research
- 60 000 images
  - 10 classes: airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks
  - each 6,000x low-res color images (32x32)
- various kinds of CNNs tend to be the best at recognizing the images in CIFAR-10
- CIFAR-10 is a labeled subset of the 80 million tiny images dataset.
  - students were paid to label all of the images



CNN Architecture

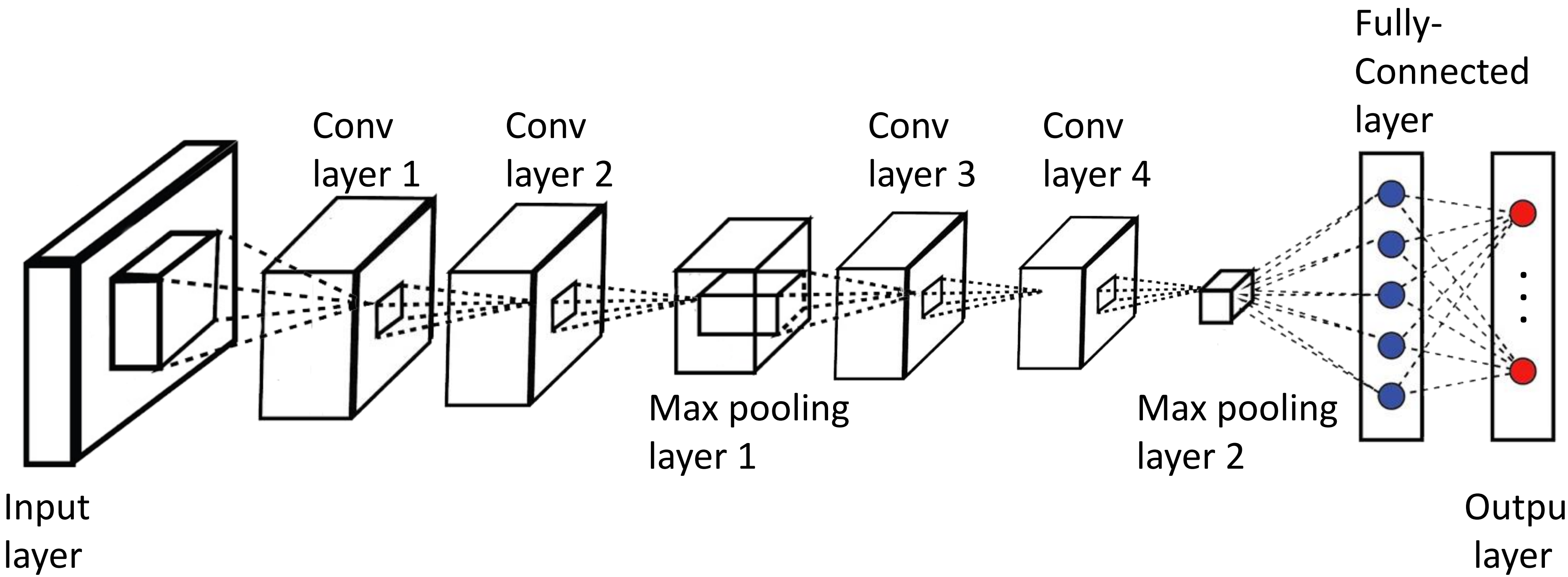
- Simple CNN Architecture Example
  - 1 Convolutional layer
  - 1 Pooling layer
  - 1 FC layer



CNN Architecture

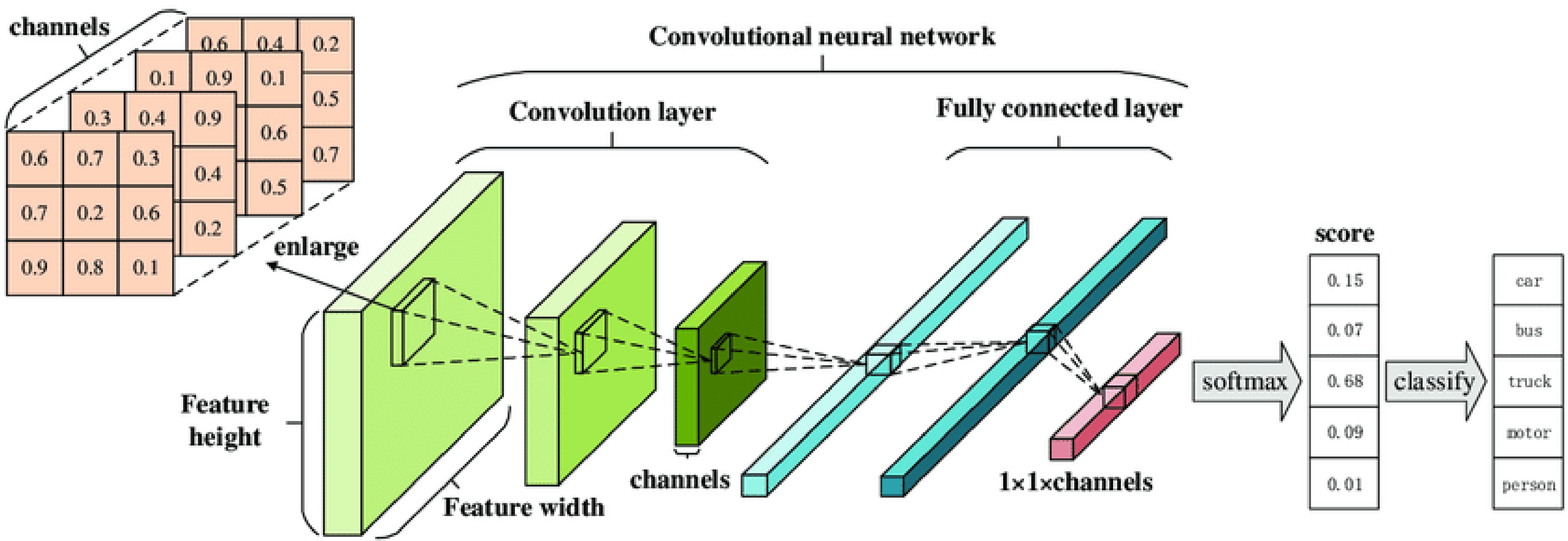
- 10 layers NN
- input and output layers
- 8 hidden layers deep CNN
  - 4 Convolutional layers
  - 2 Pooling layers
  - 1 Fully Connected layer

CNN Architecture



CNN Terms

- channels, height, width



Convolution

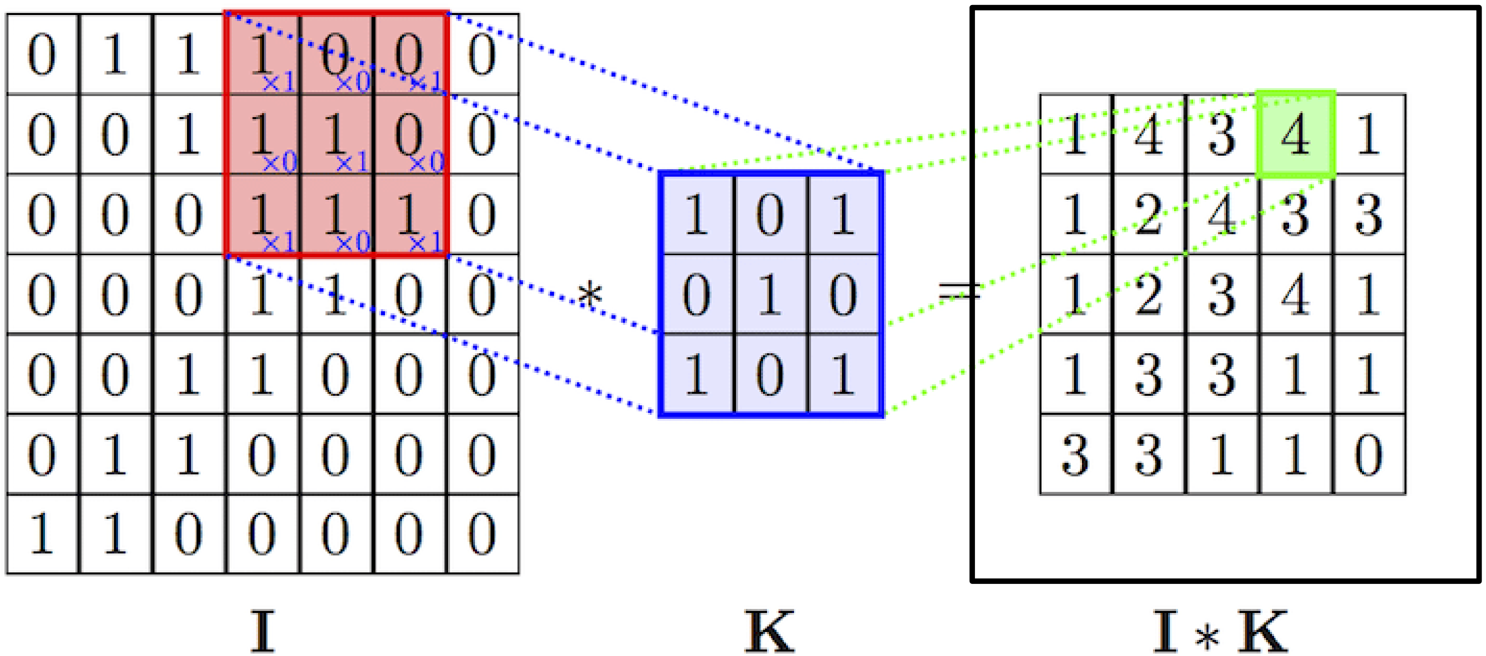
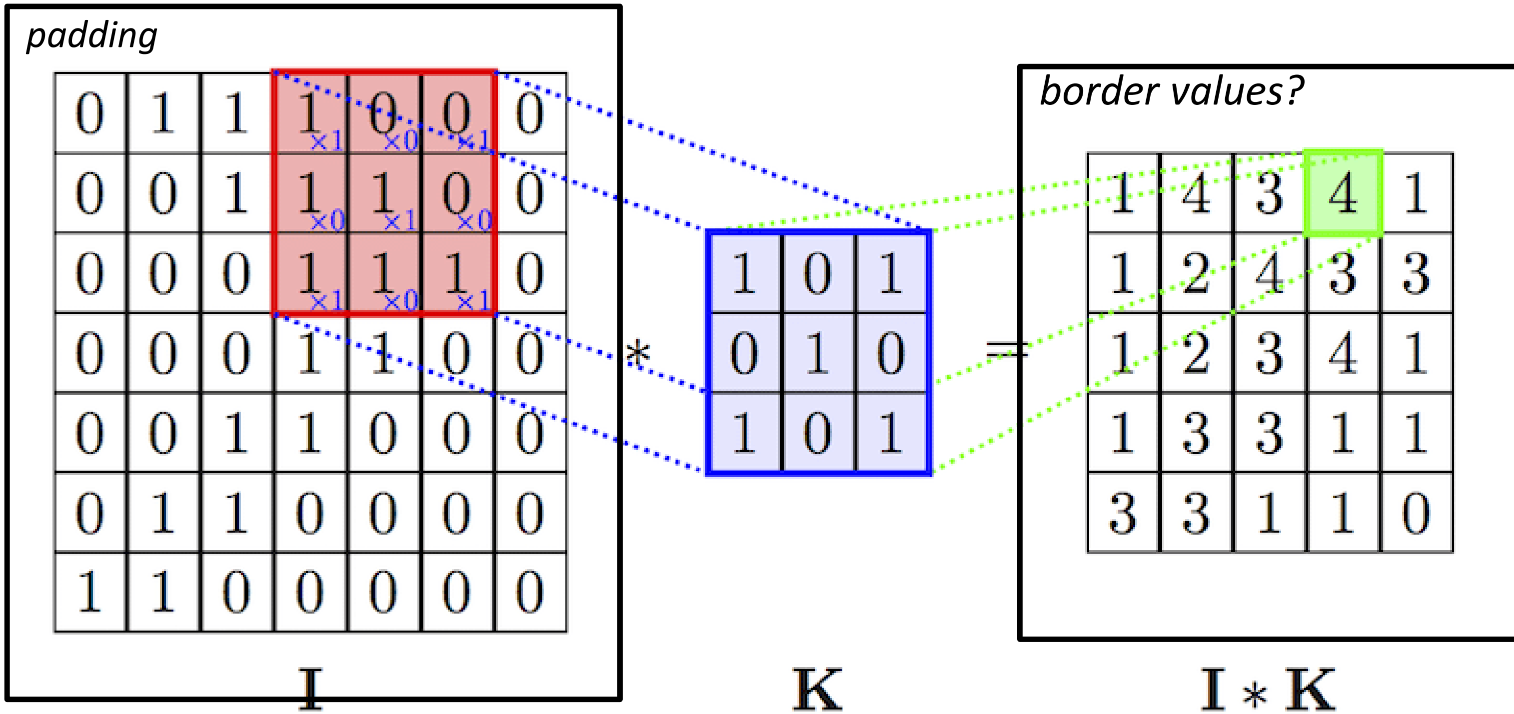


image input

kernel

Padding



We do padding to get output dim = input dim

which values? zero | mirror | repetition|... optional

Why Convolution?

- kernel to detect horizontal borders
- zero padding

$$\begin{array}{|c|c|c|c|c|}\hline 0&0&0&0&0\\\hline 1&1&1&1&1\\\hline 0&1&0&0&1\\\hline 0&1&1&0&1\\\hline 1&0&0&1&0\\\hline\end{array} \otimes \begin{array}{|c|c|c|}\hline 0&0&0\\\hline 1&1&1\\\hline 0&0&0\\\hline\end{array} = \begin{array}{|c|c|c|c|c|}\hline 0&0&0&0&0\\\hline 2&3&3&3&2\\\hline 1&1&1&1&1\\\hline 1&2&2&2&1\\\hline 1&1&1&1&2\\\hline\end{array}$$

- change kernel to detect another feature

Exercise

h

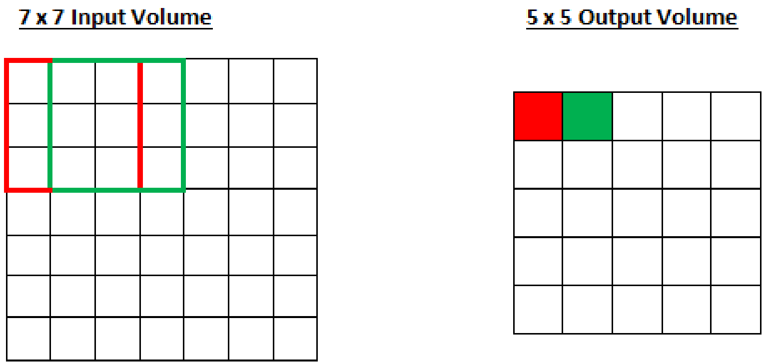
2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

f

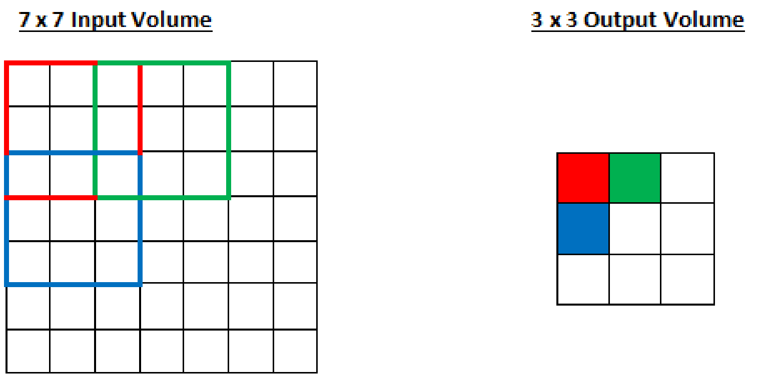
1	-1	-1
1	2	-1
1	1	1

Stride

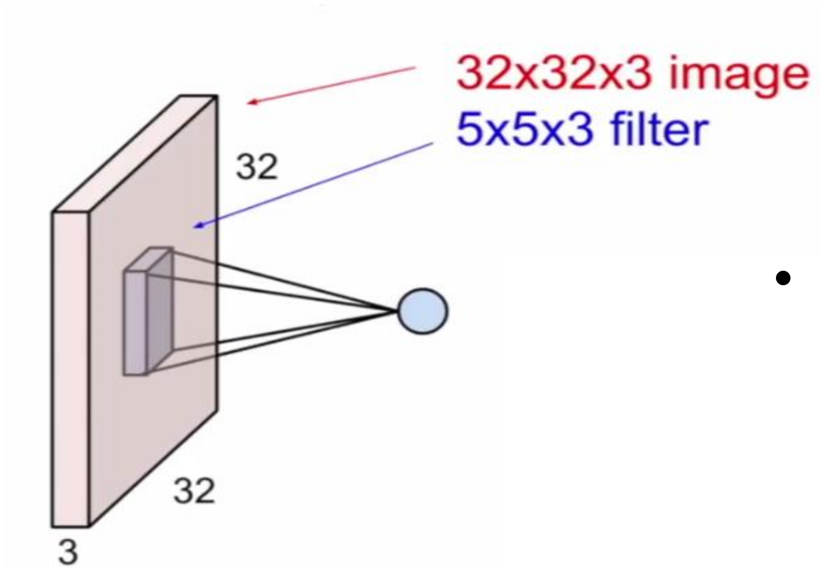
- stride = 1
  - slide the kernel to visit all values one column by one column, one row by one row



- stride = 2
  - each 2 values
  - col yes col no
  - row yes row no



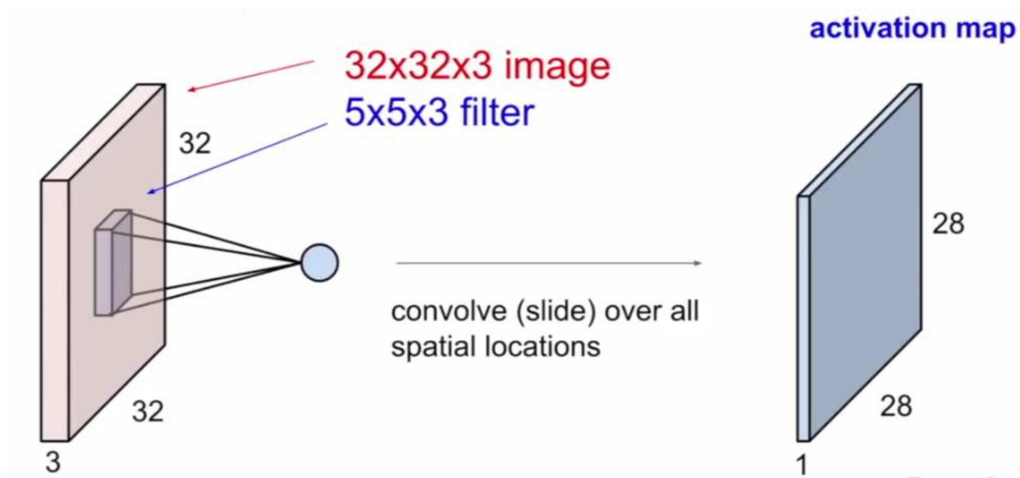
Convolutional Layer



- images = set of pixels
  - matrix (NxNx3)
  - height x width x depth
  - 3 color channels (RGB)
  - → depth = 3
- Convolutional Layer
  - → use set of **learnable** filters
  - detect presence of specific features or patterns
  - matrix (MxMx3)
  - same depth as image

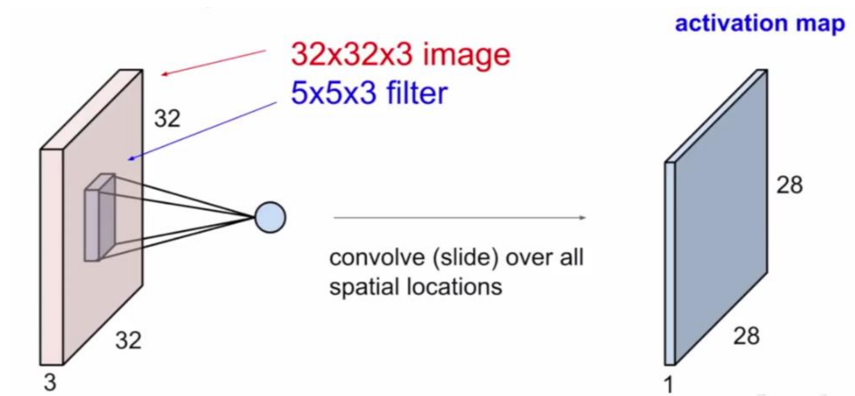
Use of filters to get Activation Maps

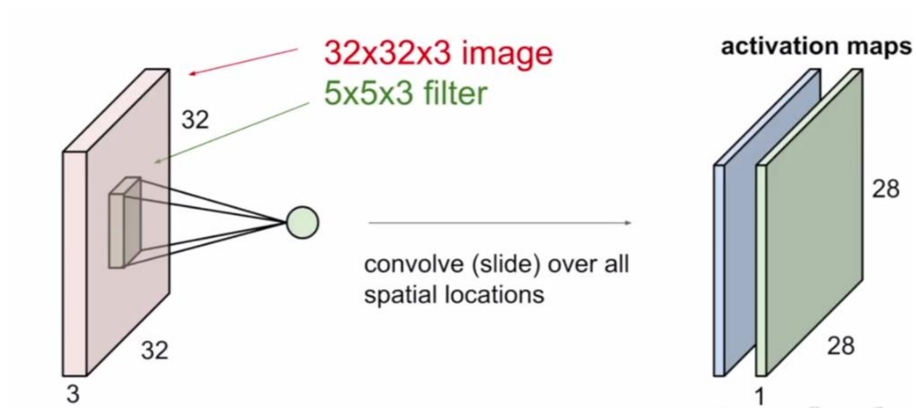
- filter → convolved (slided) across width & height of image
  - dot product computed
  - →activation map



Use of filters to get Activation Maps

- ≠ filters
- →detect ≠ features
- → ≠ activation maps

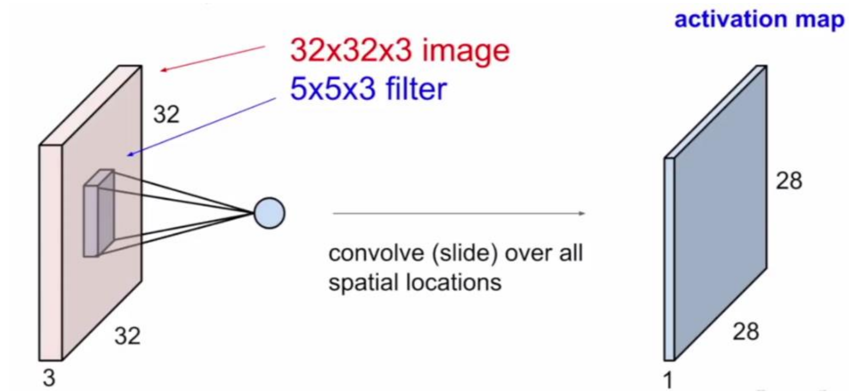


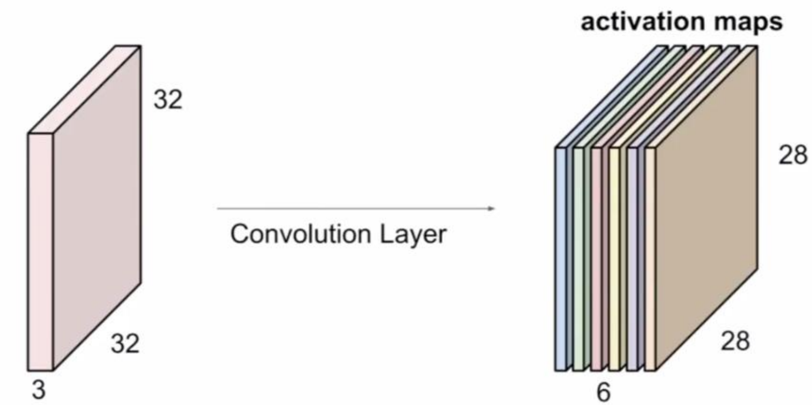


Use of filters to get Activation Maps

- ≠ filters
- →detect ≠ features
- → ≠ activation maps

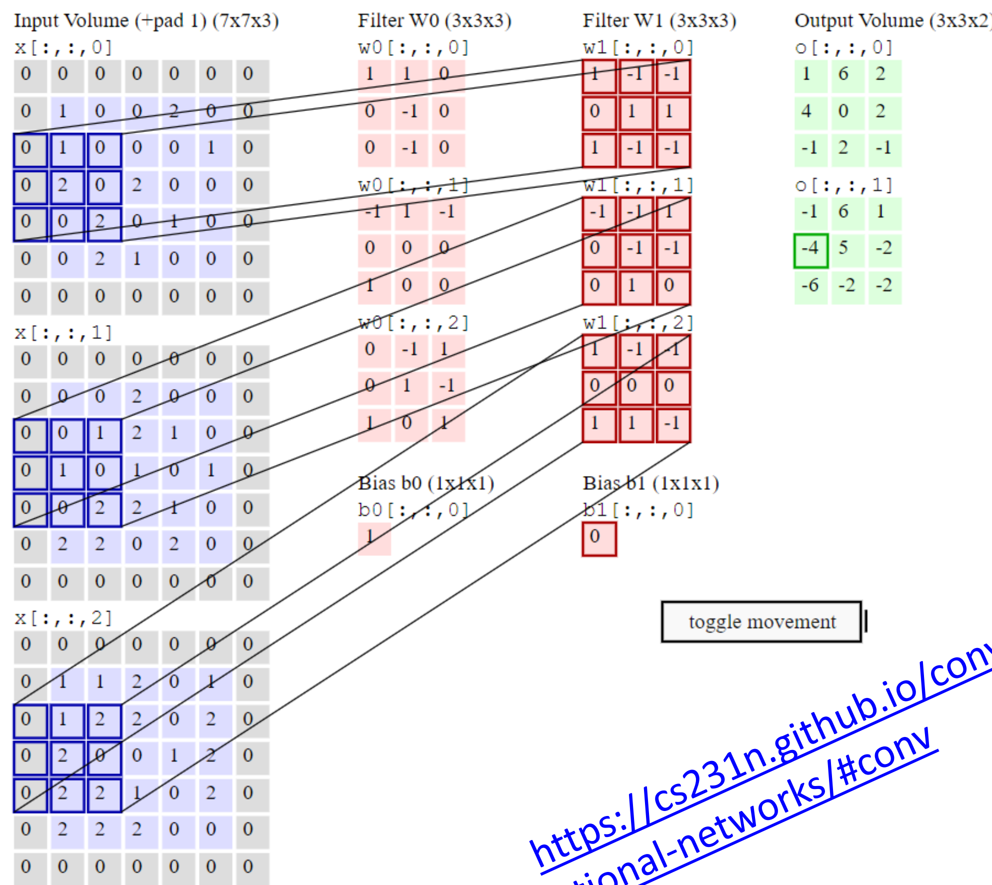
- stack up AMs
- → get new image
- size 28x28x6





Multi-Kernel Convolution Calculation

- Input image 5x5x3
- Zero padding = 1
- **Two** convolution kernels 3x3x3
- Stride = 2
- Feature map 3X3X2



https://cs231n.github.io/convolutional-networks/#conv

Activation Map Size

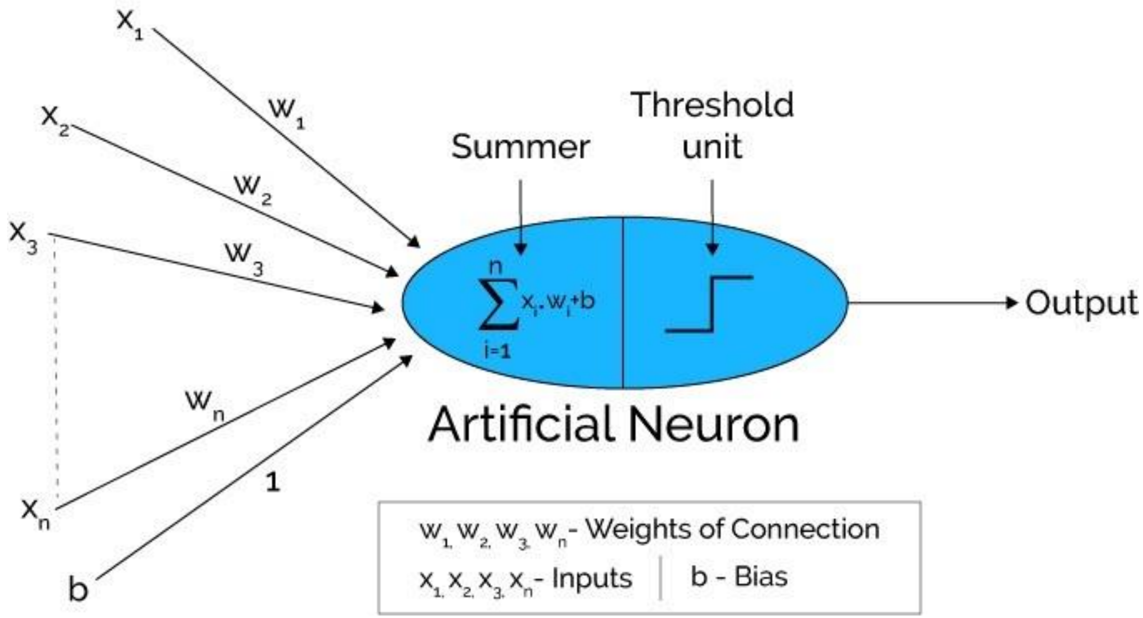
- (N + 2P — F)/ S + 1
  - N = dimension of image
  - P = padding
  - F = dimension of filter
  - S = stride

- $N = 5$
- $P = 1$
- $S = 2$
- $F = 3$

- $\mathrm{AMSIZE} = (5 + 2 \times 1 - 3)/2 + 1 = 3$

Input image 5x5x3
Zero padding = 1
Two convolution kernels 3x3x3
Stride = 2
Feature map 3X3X2

Activation Function

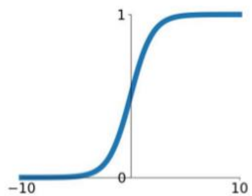


$$Y = \text{Activation}(\Sigma(weight * input) + bias)$$

Activation Functions

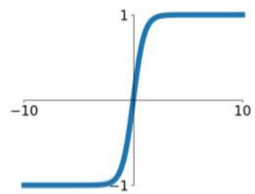
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



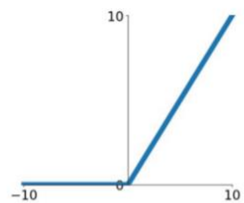
tanh

$\tanh(x)$



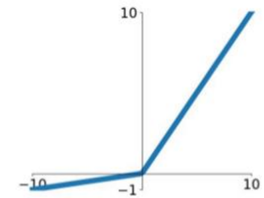
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

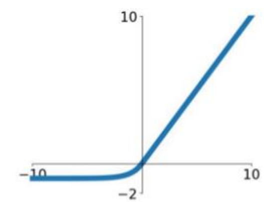


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

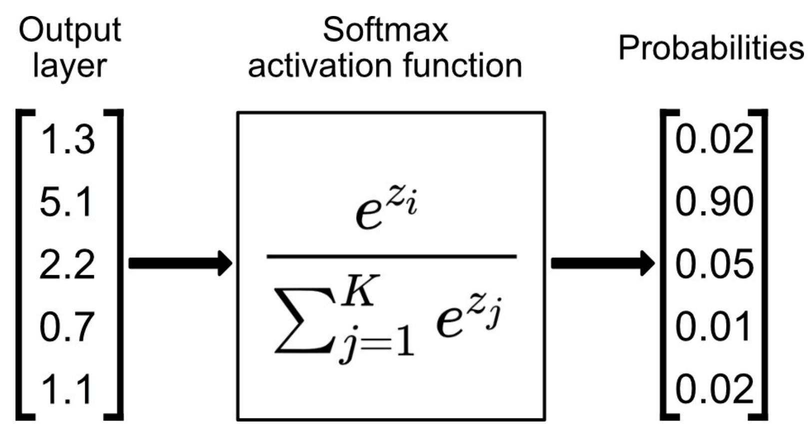
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



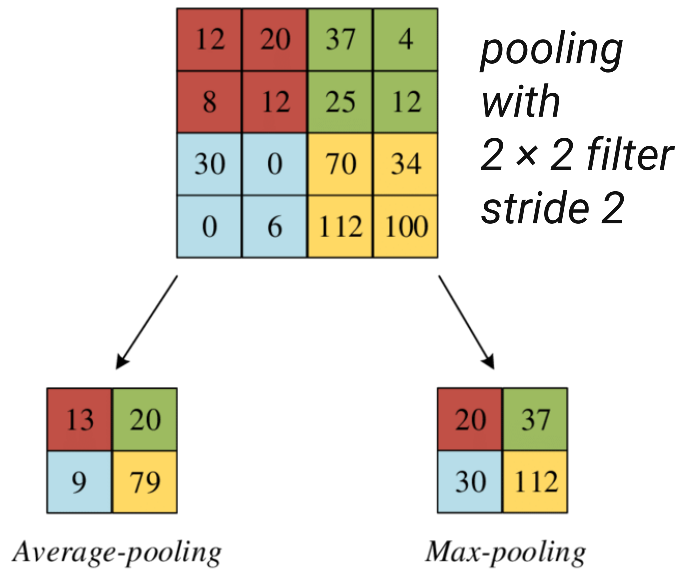
Choosing the right AF

- rule of thumb
- in convolution layer
  - if you really don’t know what AF to use,
  - → then simply use RELU
    - as it is a general AF and is used in most cases these days
- in output layer
  - if your output is for binary classification
    - → then, sigmoid function is very natural choice for output layer
  - for general classification
    - use softmax



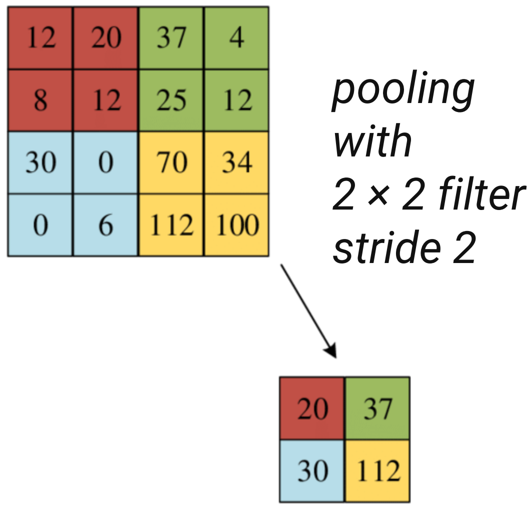
Pooling Layer

- between convolution layers
  - reduce number of params and computation
  - control overfitting by progressively reducing the spatial size of the network
- 2 operations
  - average pooling
  - maximum pooling



Max-pooling

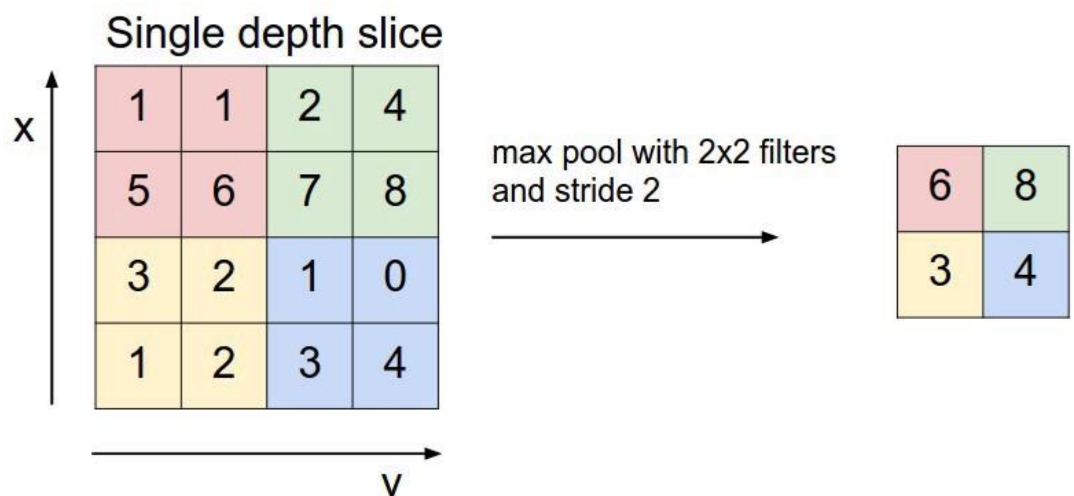
- take out only the maximum from a pool
  - filters sliding
  - at every stride
    - take maximum param
    - drop rest
  - → network down-sampled
- no alteration in depth



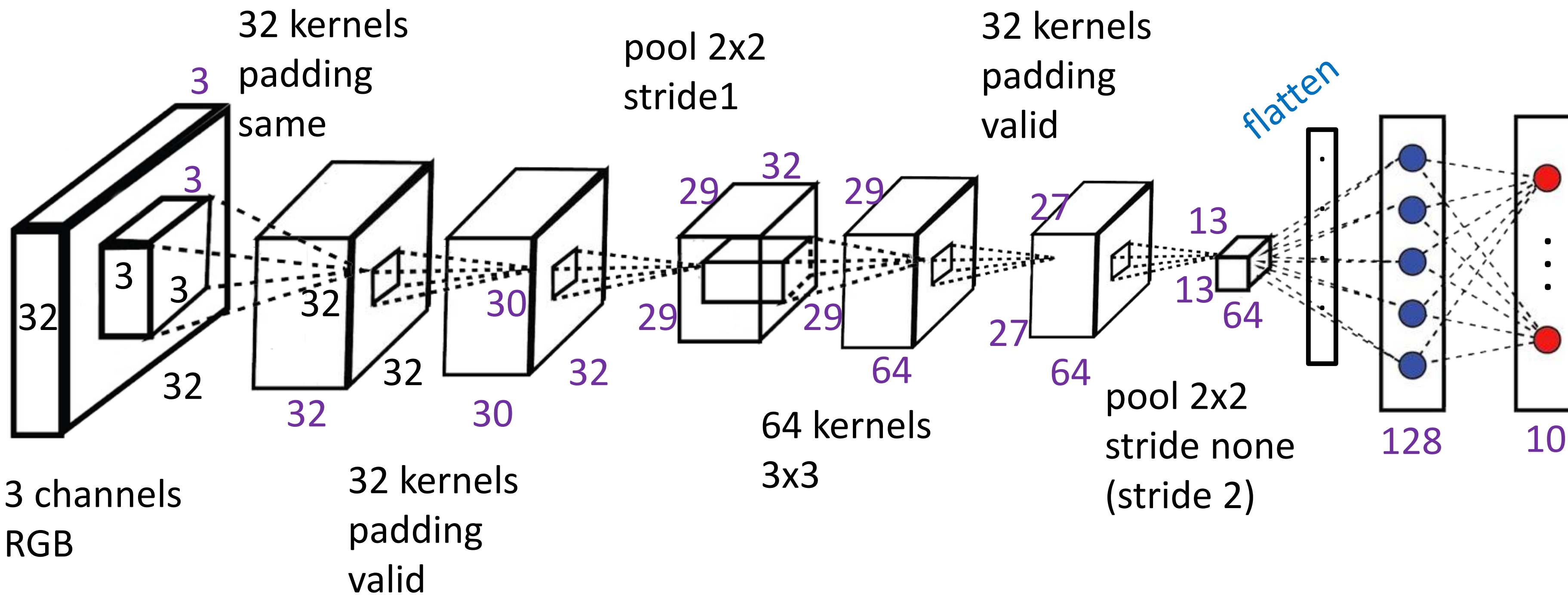
Max-pooling

Output of Pooling Layer size

- $(N — F)/ S + 1$
  - N = dimension of input to pooling layer
  - F = dimension of filter
  - S = stride



CNN Architecture Overview



Experimental Environment

- We will use colab (check Google_Colab.pdf)
  - no need to worry install anything
  - everything is ready on the cloud
- The NumPy and TensorFlow frameworks are used.
  - The NumPy arrays are used as the image objects.
  - The TensorFlow framework is mainly used to create deep learning algorithms and build a convolutional neural network (CNN).

Image Recognition Implementation

1. Module Importing
2. Training Data Preparation
3. Model Creation
4. Model Training
5. Model Testing

The Preparation of the Training Data

- A very important step
  1. Data preparation and labelling
  2. Data standarization
  3. Data cleaning
  4. Data pre-processing

- In this experiment everything is ready inside Keras

Module Importing

Tensorflow:

- open-source library for a number of various tasks in machine learning
- provides both high-level and low-level APIs
- most popular in-depth learning framework

Keras:

- Keras is a neural network library inside **TensorFlow**
- provides only high-level APIs

numpy:

- scientific computing package based on python
- process numerical calculation

Module Importing

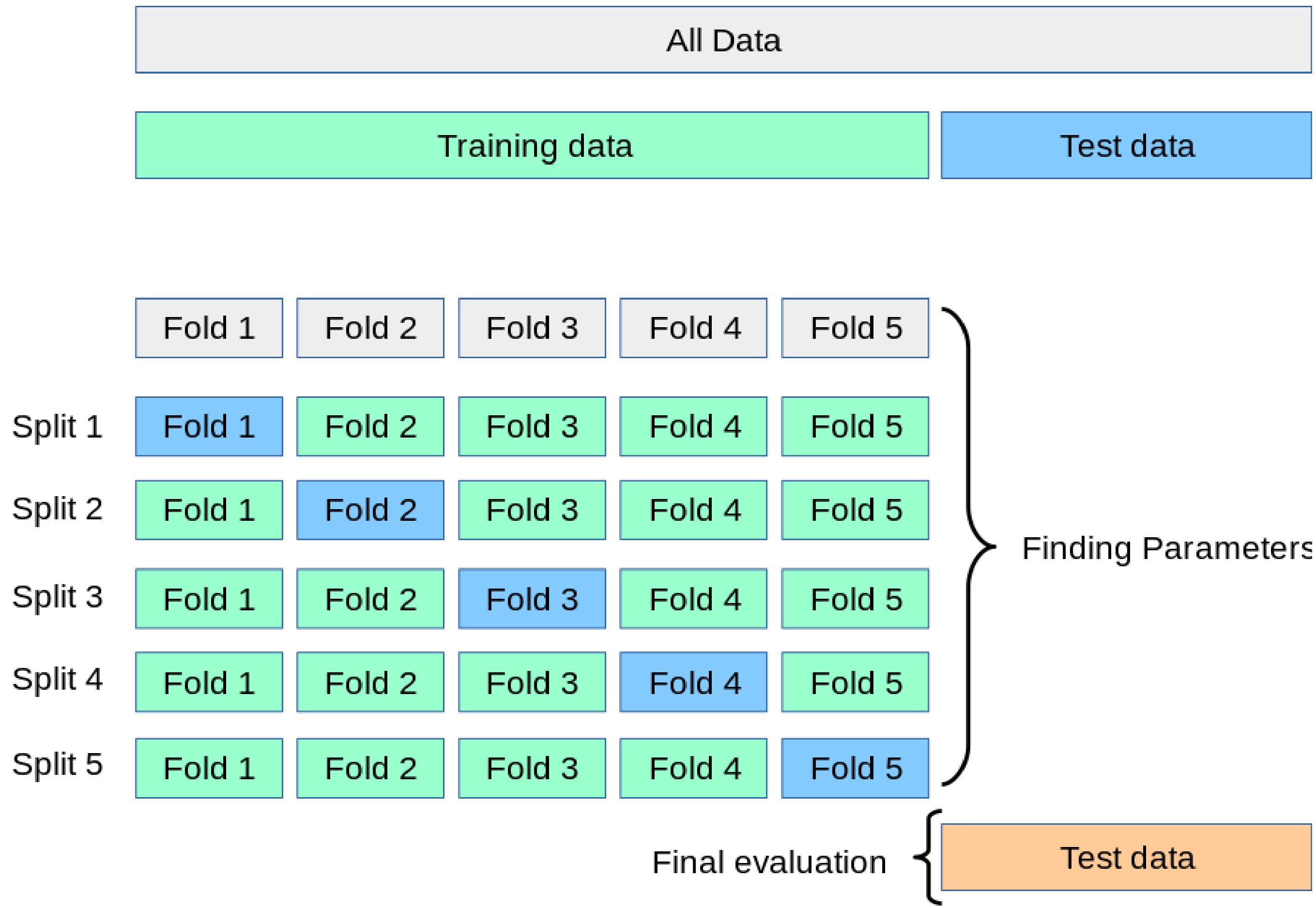
Matplotlib:

- is a comprehensive library for creating static, animated, and interactive visualizations in Python.

pyplot:

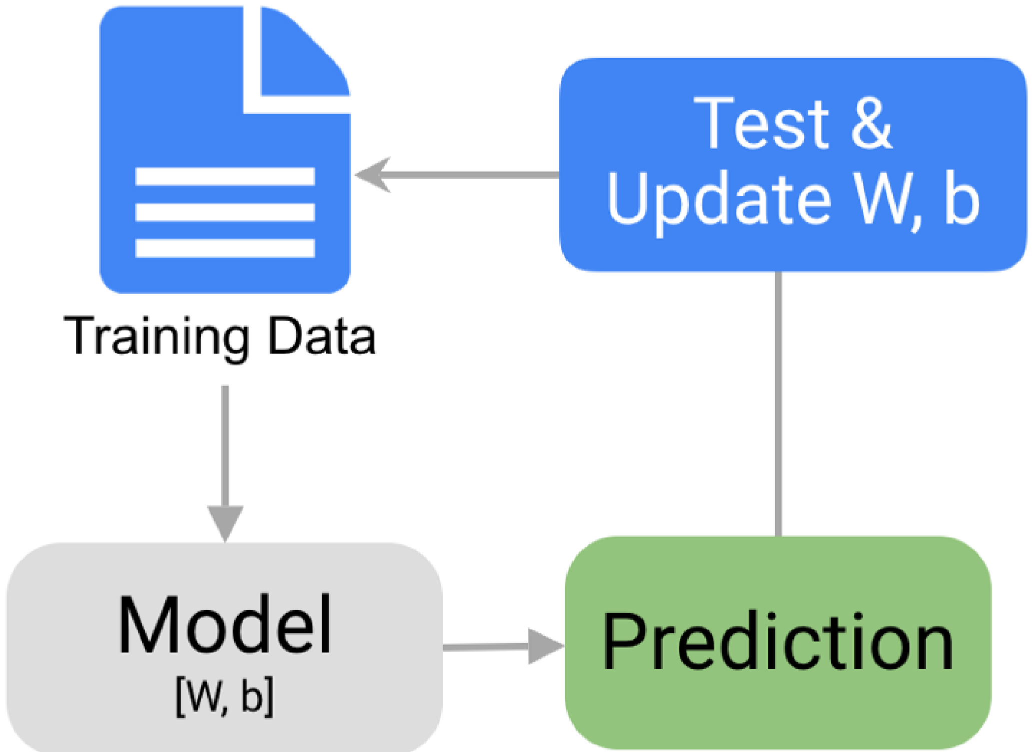
- inside matplotlib
- collection of command style functions that make matplotlib work like MATLAB.
- Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

k-fold cross validation



Hyper Parameter

- A model hyperparameter is a configuration that is external to the model and whose value cannot be estimated from data.
  - network structure → HP
    - nb of layers
  - variables which determine how the network is trained → HP
    - learning rate
  - size & number of kernels → HP
  - bias → not HP
    - just param that the model has to set
    - nb bias == nb kernels
  - stride → HP
  - padding → HP
  - AF → HP
    - choice of activation function is also HP


Test &
Update W, b
Training Data
Model
[W, b]
Prediction