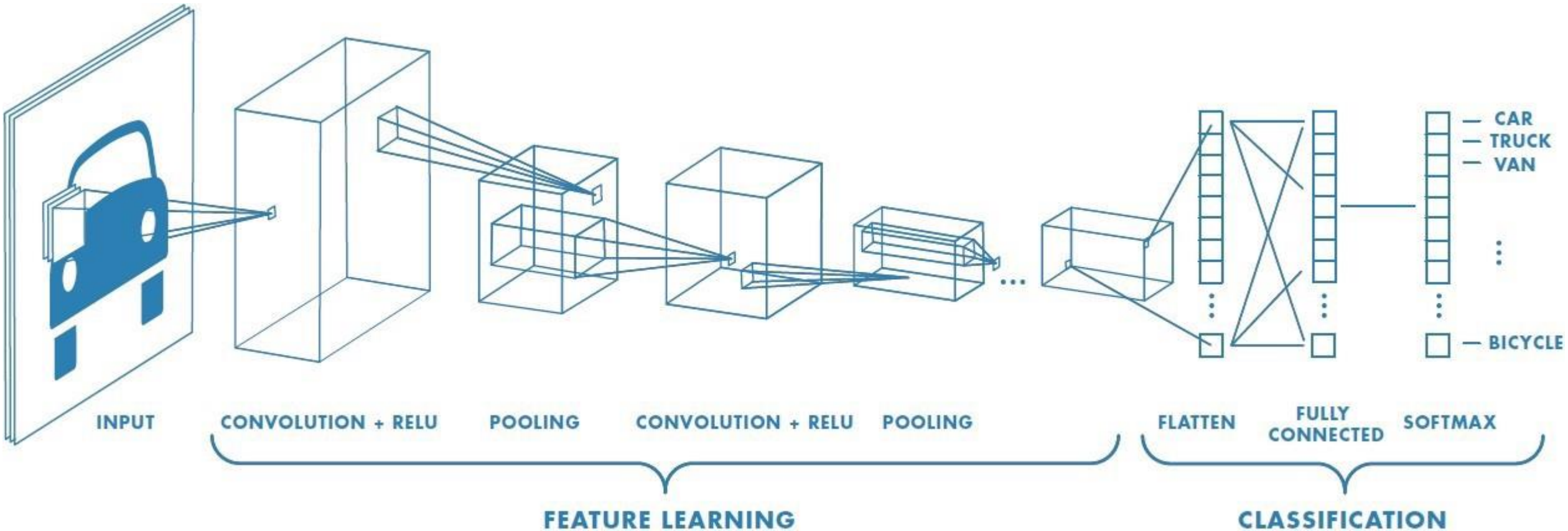
Classification Model

Convolutional Neural network

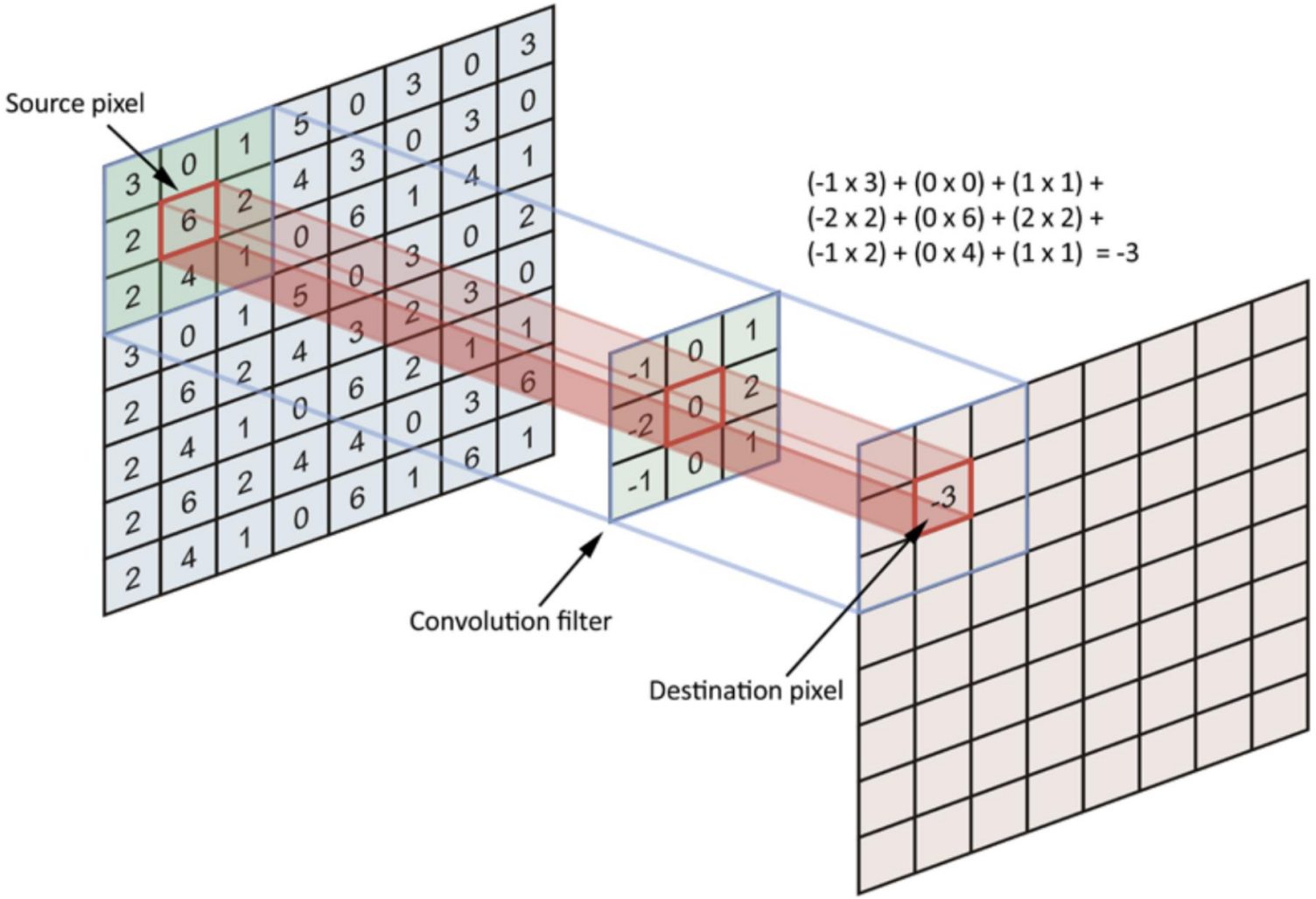
Convolutional Neural network(CNN) terminology



Convolutional Neural network(CNN) terminology

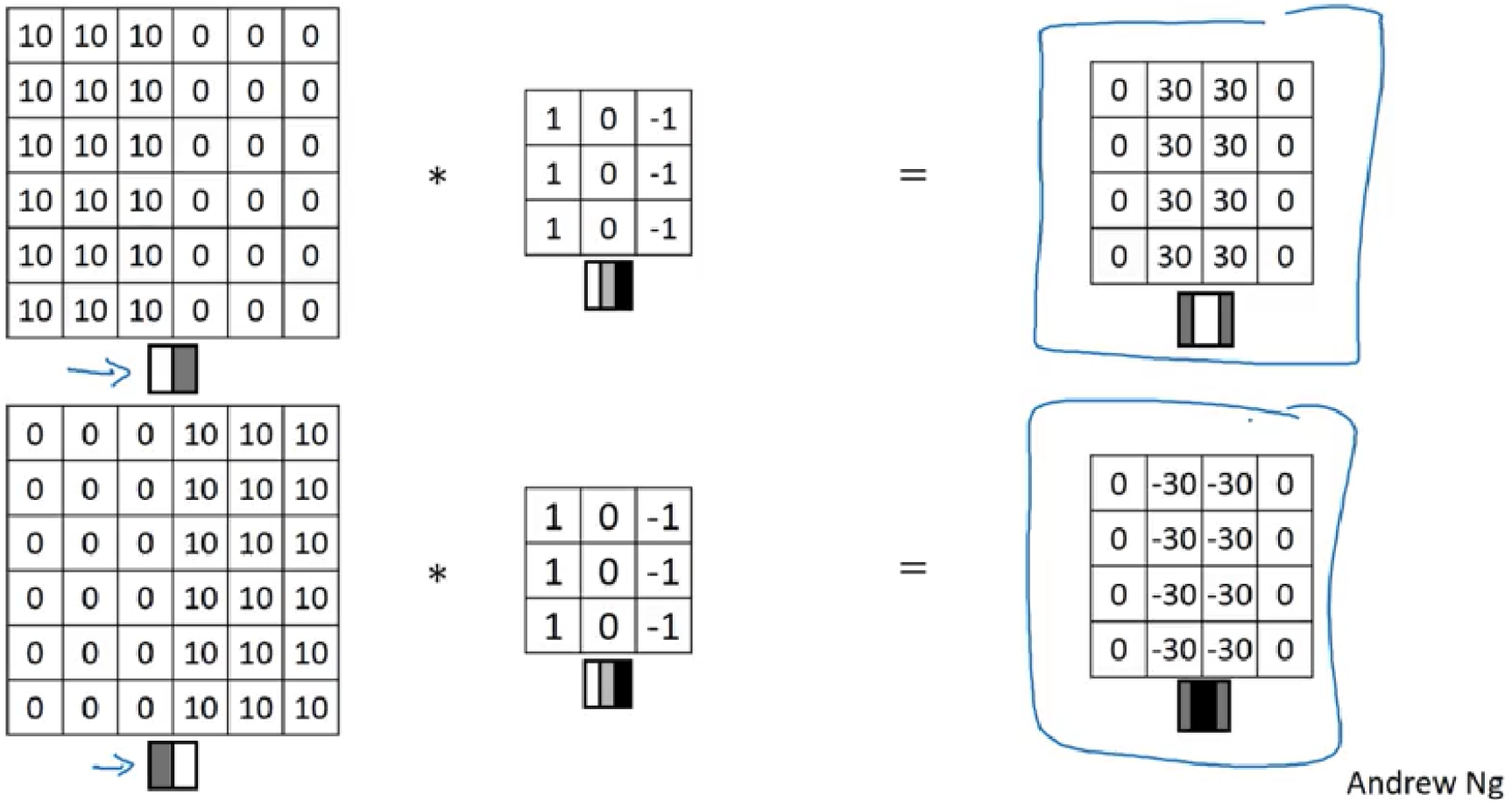
- **Convolution**
- **Convolutional mask (kernel)**
- **Edge detection mask**
- **Padding**
- **Convolution Stride**
- **Convolutions over volume**
- **Pooling**

CNN terminology: convolution



- **Convolution**
  - **Operation:**
    - **Dot Product or**
    - **Weighted sum**
  - **Task**
    - **Local pattern detection**
      - **Search for a particular local pattern in input**

CNN terminology: convolution



Andrew Ng

- **Ex.**
  - **Vertical Local pattern detection**
  - **Sign of convolution results**
    - **According to convolutional mask**
  - **Adaptive mask**
    - **Learning from data or domain problem**

Convolution padding

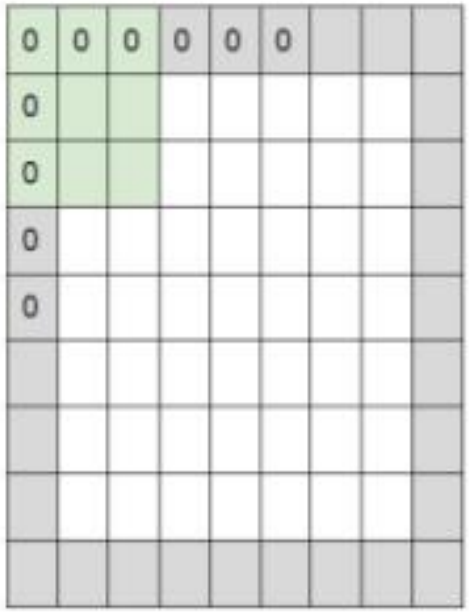
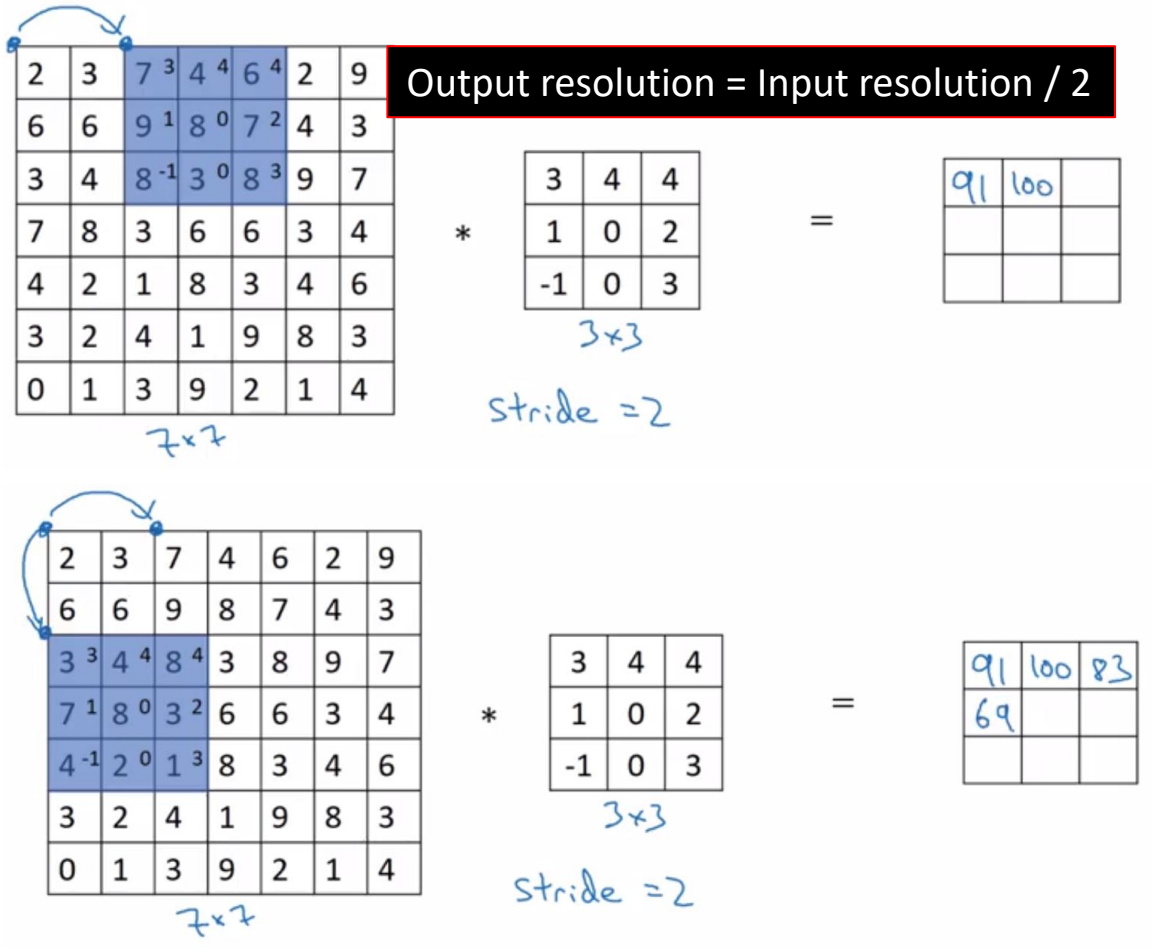


Fig Credit: A Karpathy, CS231n

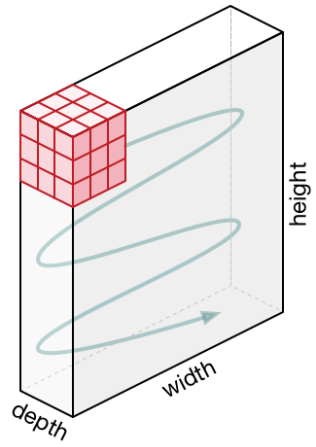
- **Padding**
  - **Boundary extending**
    - **In order to maintain convolution results the same dimension as input**
  - **Padding size**
    - **Depend on mask size**
  - **Padding value**
    - **Zero (mostly used)**
    - **Reflected border**
    - **Circular index**

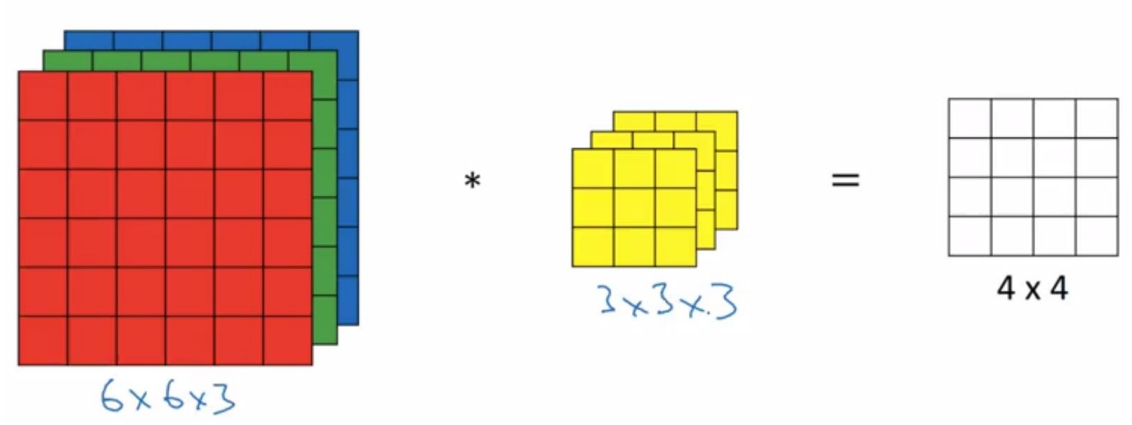
Convolution Stride



- **Stride**
  - **Convolution resolution selection**
    - **Stride = 1**
      - **Convolution on every input position**
        - **Reserve output resolution = input resolution**
    - **Stride > 1**
      - **Convolution skipping**
        - **Output resolution < Input resolution**

Convolution over volume

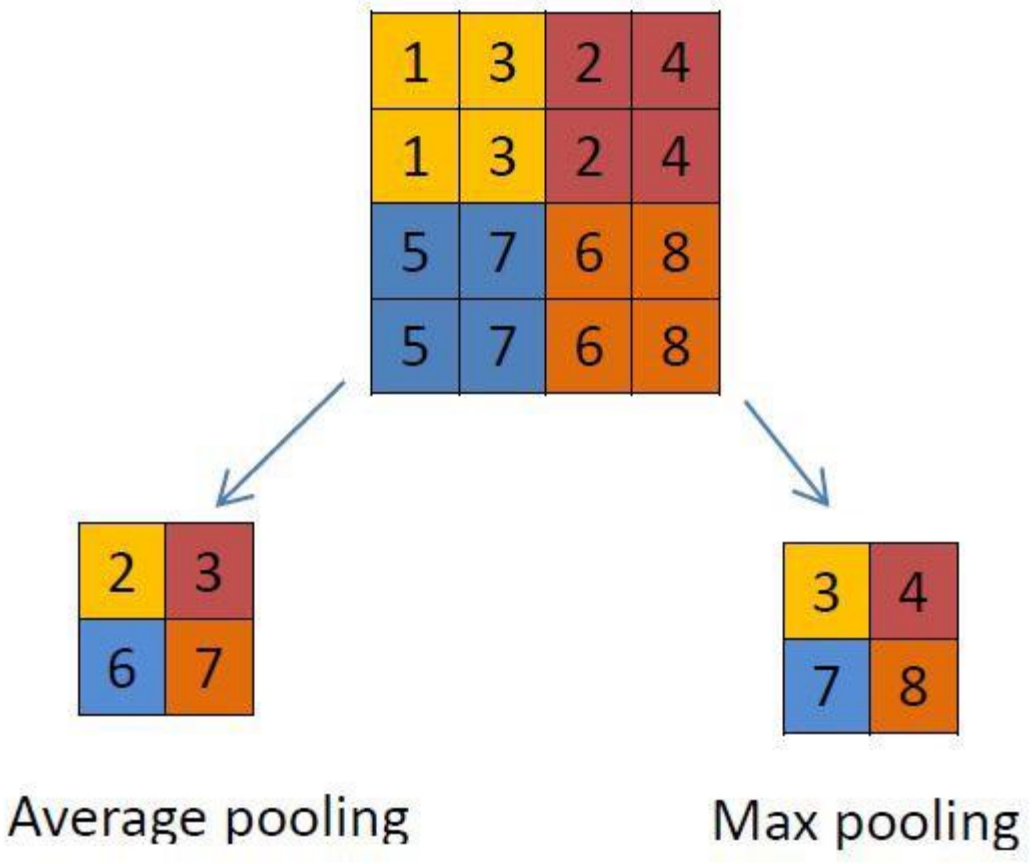




- 3 input planes / 3 convolutional masks / 1 output
- **Output Results (with no zero padding example) = 4x4**
  - = Red plane (1) * mask (1) + Green plane (2) * mask (2) + Blue plane (3) * mask (3)

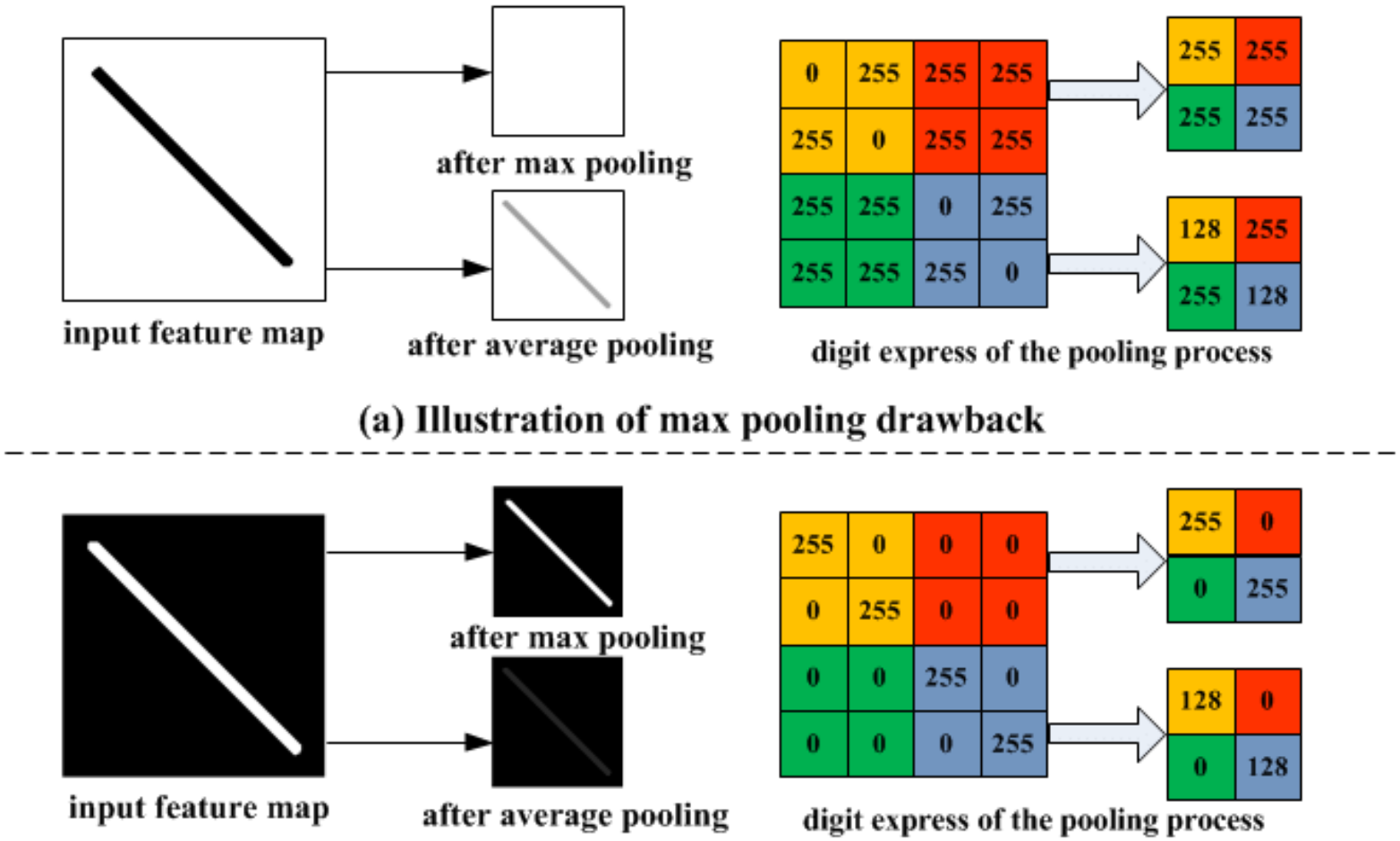
- **Sum all detected local pattern from all Red / Green / Blue planes**
  - Return a single output of all detected patterns in input planes

Convolution Pooling

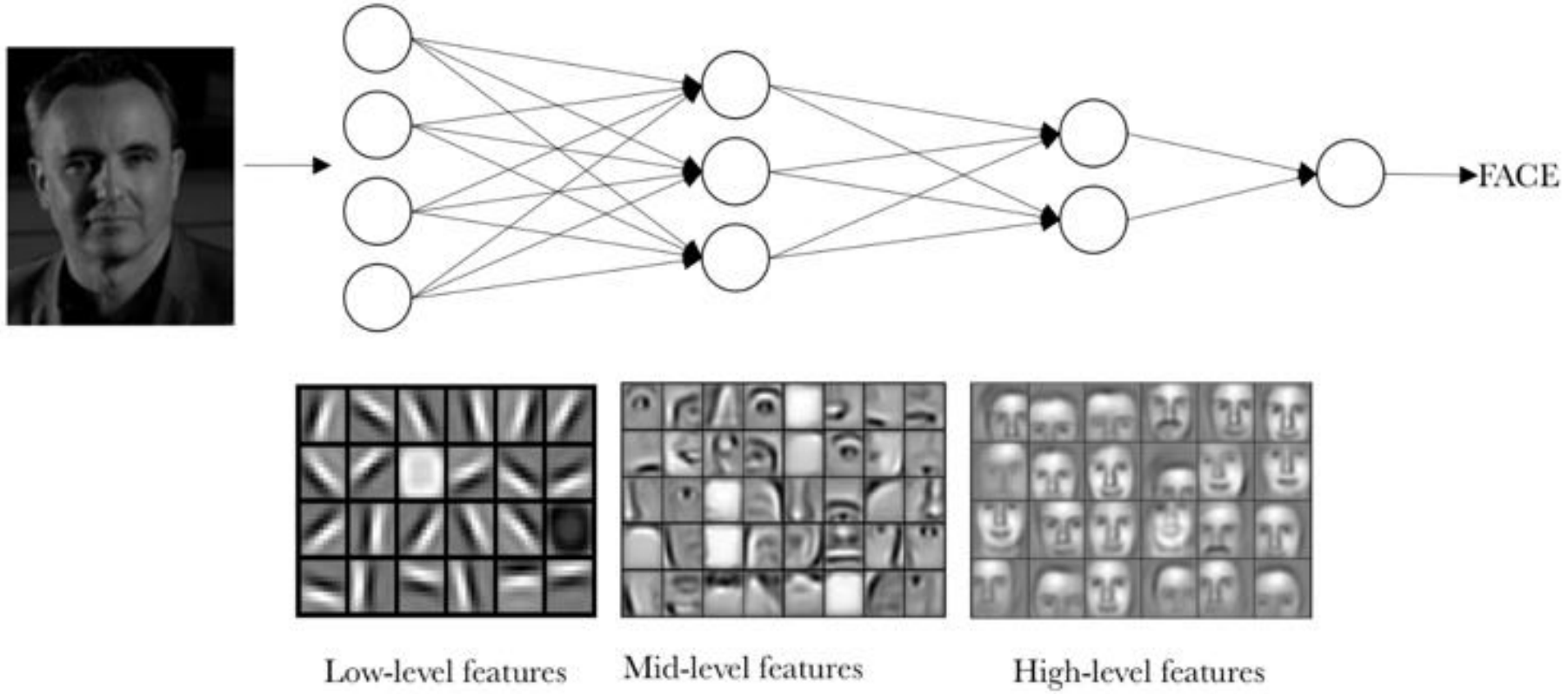


- Reduce the dimensionality and the number of parameters and computation in the network.
  - This shortens the training time and controls overfitting.
- Pooling functions
  - Max pooling
  - Average pooling

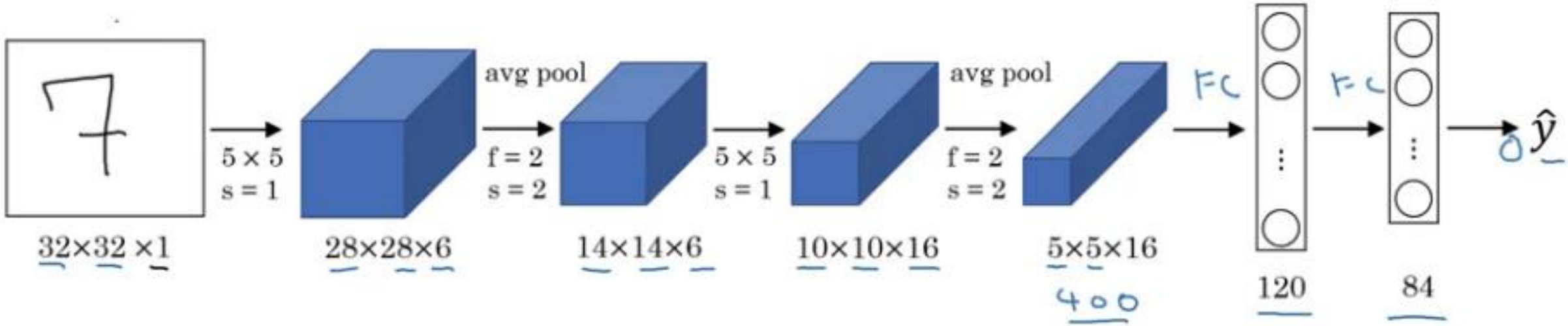
Convolution Pooling



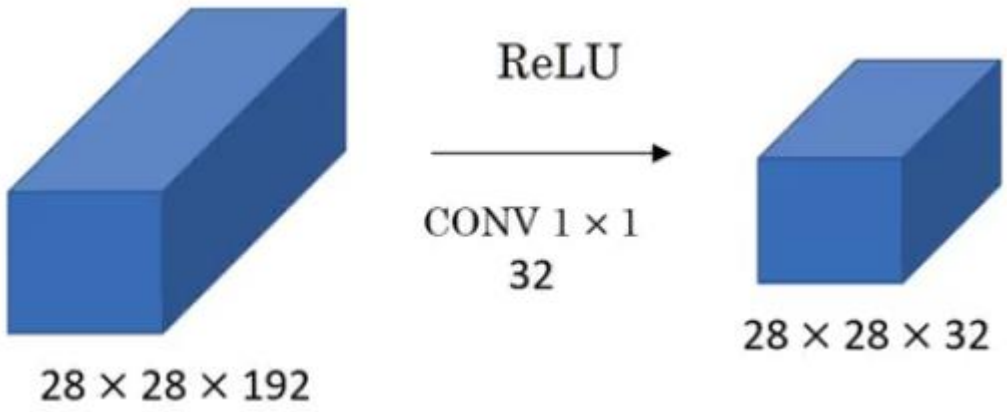
Multi-level feature extraction



Example of CNN structure (Lenet)



CNN structure (resnet)

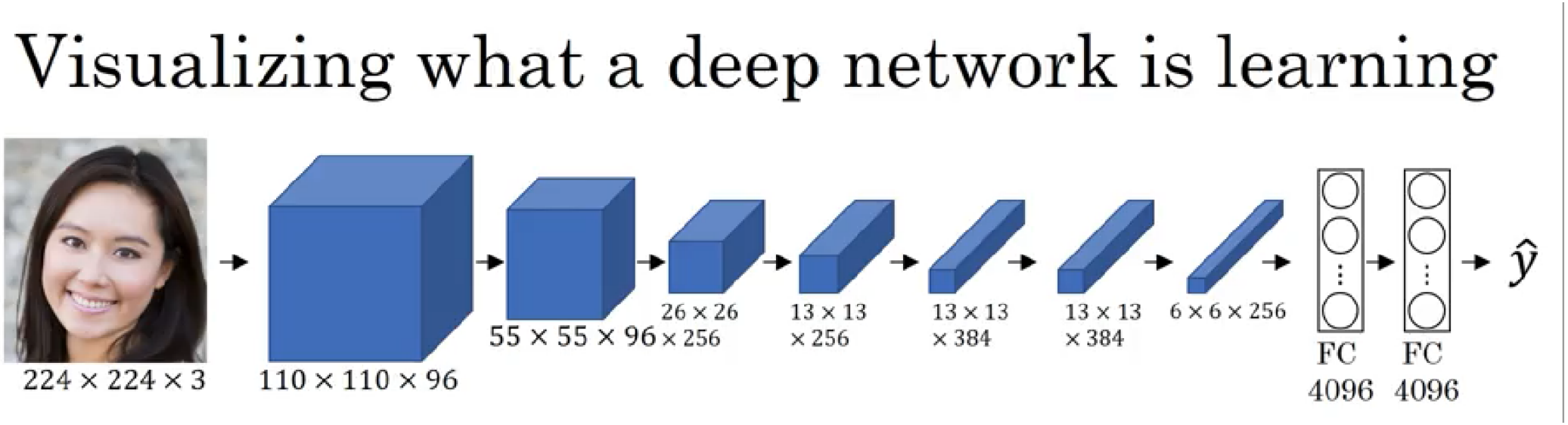


The basic idea of using 1 X 1 convolution is to reduce the number of channels from the image. A couple of points to keep in mind:

We generally use a pooling layer to shrink the height and width of the image

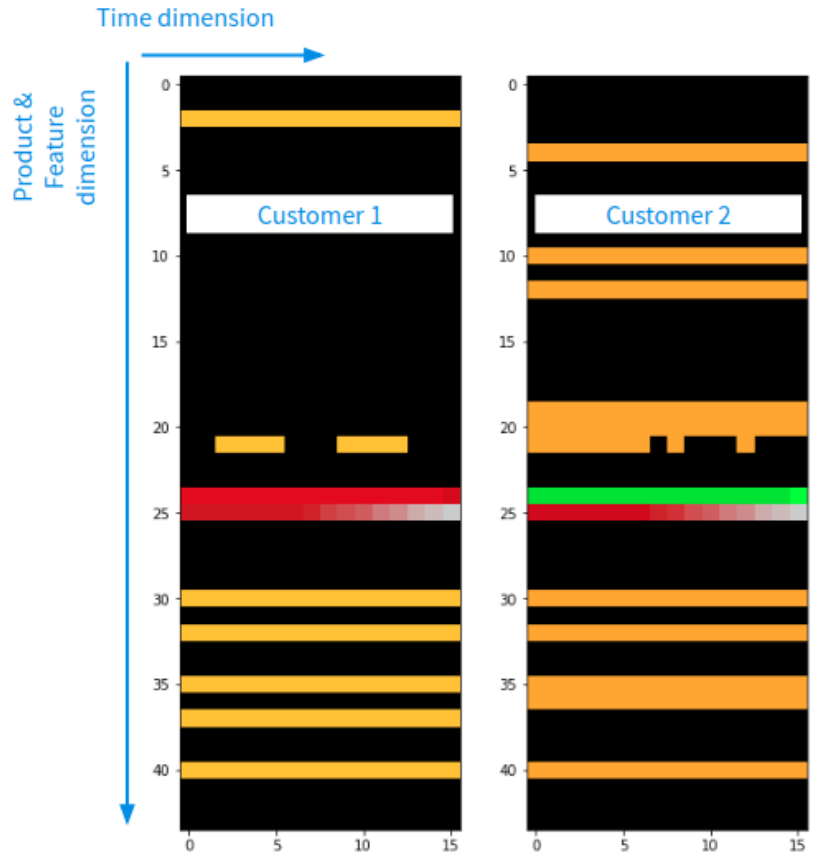
To reduce the number of channels from an image, we convolve it using a 1 X 1 filter (hence reducing the computation cost as well)

Activity: CNN for images



1. ระบุ Convolutional layer Structure (Filter size, padding, stride, maxpooling)
2. คำนวณจำนวนพารามิเตอร์ทั้งหมดในโครงสร้าง (convolutional + fully connected layers)

CNN for time series data for bank product prediction



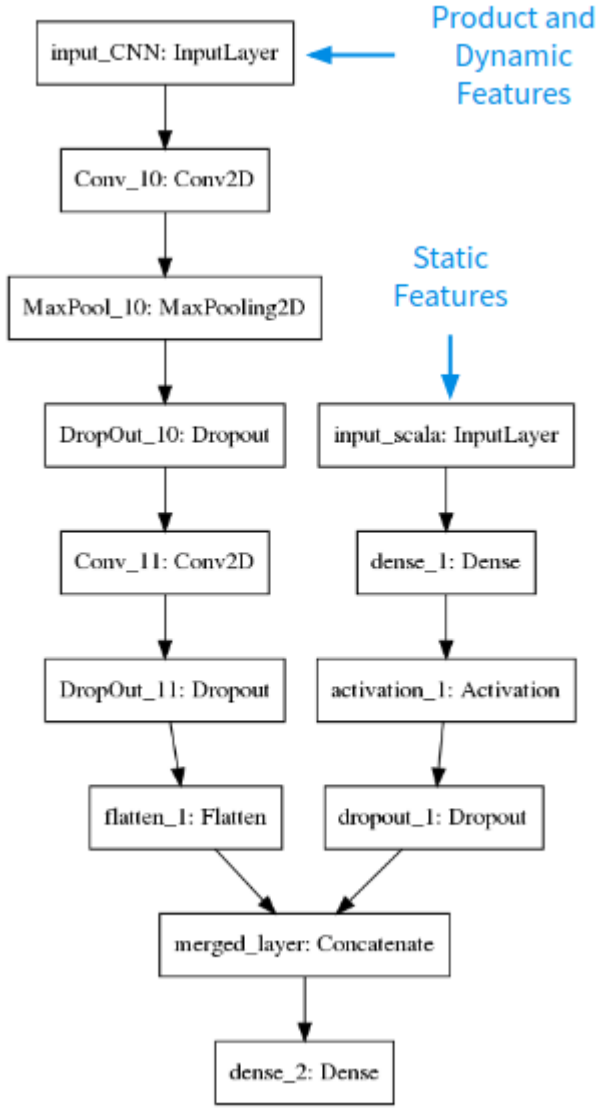
Some features like income and relationship with the bank changed overtime

ended up with some 45 by 17 pixels

n_features = 45
timesteps = 17

Training shape: [samples, timesteps, n_features]

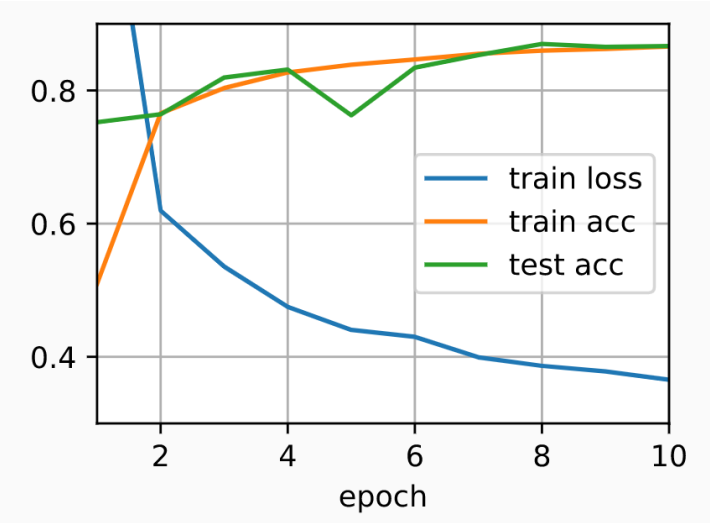
input_shape=(timesteps, n_features)



https://towardsdatascience.com/convolutional-neural-network-on-a-structured-bank-customer-data-358e6b8aa759

Dropout layer

https://d2l.ai/chapter_multilayer-perceptrons/dropout.html



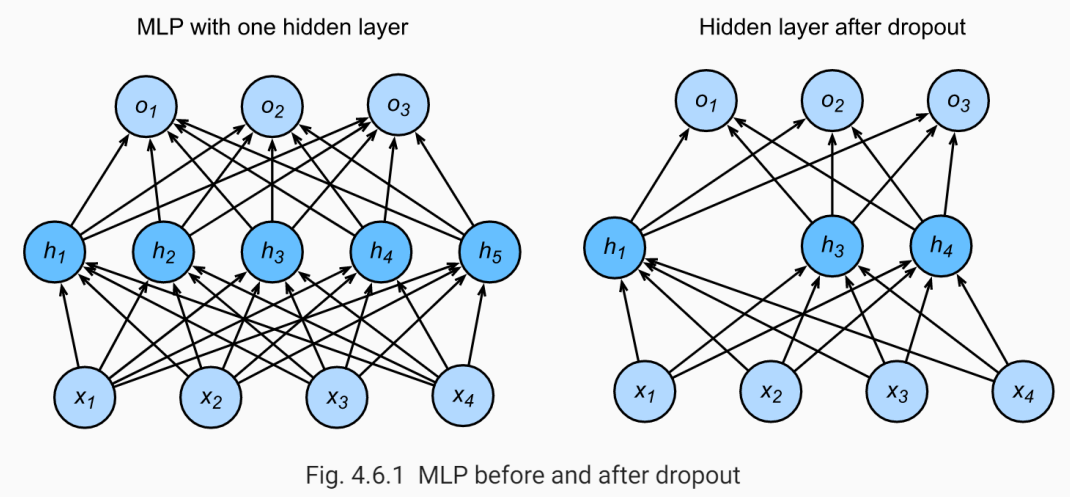
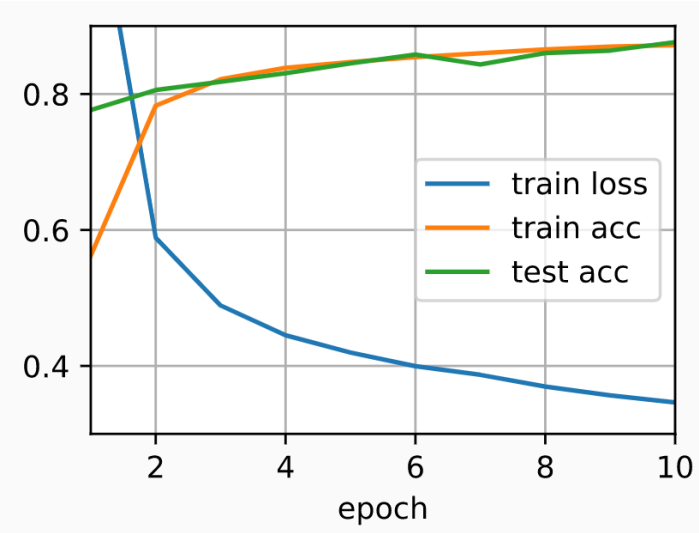


Fig. 4.6.1 MLP before and after dropout



Dropout drops out some nodes of the network to reduce overfitting and learn a fraction of the weights in the network in each training iteration.

dropout rate, *p* = 0.5, yields the maximum

https://wiseodd.github.io/techblog/2016/06/25/dropout/

network without Dropout may perform better at the training phase while Dropout network may perform worse. However, at the test time, Dropout network is not just performed better, but *consistenly better*.

conv2d_7 (Conv2D)	(None, 32, 6, 32)	40992
batch_normalization_8 (Batch	(None, 32, 6, 32)	128
activation_8 (Activation)	(None, 32, 6, 32)	0
conv2d_8 (Conv2D)	(None, 32, 6, 32)	40992
batch_normalization_9 (Batch	(None, 32, 6, 32)	128
activation_9 (Activation)	(None, 32, 6, 32)	0
max_pooling2d_2 (MaxPooling2	(None, 32, 3, 16)	0
flatten_2 (Flatten)	(None, 1536)	0
dense_2 (Dense)	(None, 128)	196736
batch_normalization_10 (Batc	(None, 128)	512
activation_10 (Activation)	(None, 128)	0
dropout_2 (Dropout)	(None, 128)	0
feature_dense (Dense)	(None, 100)	12900
batch_normalization_11 (Batc	(None, 100)	400
activation_11 (Activation)	(None, 100)	0
dense_3 (Dense)	(None, 2)	202
activation_12 (Activation)	(None, 2)	0

Feature extraction @Intermediate layer

```
intermediate_layer_model = Model(inputs=model.input,

outputs=model.get_layer('feature_dense').output)
```