CHƯƠNG 4
MẠNG NEURAL TÍCH CHẬP (P2)

Khoa Khoa học và Kỹ thuật thông tin
Bộ môn Khoa học dữ liệu

Nội dung

1. Các lớp (layers) trong mạng tích chập.
2. Kiến trúc 1 mạng tích chập.
3. Implement LeNET-5.

CÁC LỚP (LAYER) QUAN TRỌNG TRONG MẠNG TÍCH CHẬP

Các loại lớp (layer) trong mạng tích chập

- Lớp tích chập.
  + Ký hiệu: CONV.
- Lớp Pooling.
  + Ký hiệu: POOLING.
- Lớp Fully Connected.
  + Ký hiệu: FC.

CONV Layers

Lớp tích chập - CONV

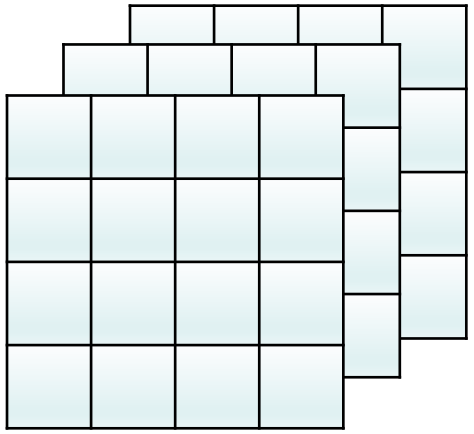
- Đây là lớp cốt lõi trong mạng tích chập.
- Bản chất của các lớp CONV là các bộ lọc (filters) dùng để trích xuất các thông tin có giá trị.
- Các bộ lọc thường có kích thước nhỏ và là số lẻ, có độ sâu cùng với độ sâu của dữ liệu đầu vào.
- Các lớp tích chập có học tham số (parameters).

Các ký hiệu

- Filter size: $f^{[l]}$
- Padding: $p^{[l]}$
- Stride: $s^{[l]}$
- Number of filter: $n_c^{[l]}$

Kiến trúc lớp CONV

Input: $n_H^{[l-1]} \times n_W^{[l-1]} \times n_c^{[l-1]}$



Output: $n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

Each filter f: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$

Activations: $a^{[l]} \rightarrow n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

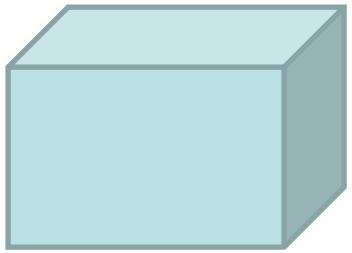
Weights: $(\boldsymbol{f^{[l]} \times f^{[l]} \times n_c^{[l-1]}}) \times \boldsymbol{n_c^{[l]}}$

Bias: $n_c^{[l]}$

$$n_W^{[l]} = \left\lfloor \frac{n_W^{[l]} + 2 * p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

$$A^{[l]} \rightarrow m \times n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$$

Ví dụ

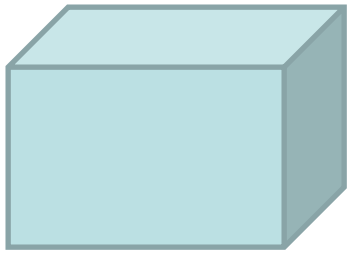


39x39x3

$n_H^{[0]} = n_W^{[0]} = 39$

$n_c^{[0]} = 3$

$f^{[1]} = 3$
$p^{[1]} = 0$
$s^{[1]} = 1$
$n_c^{[1]} = 10$

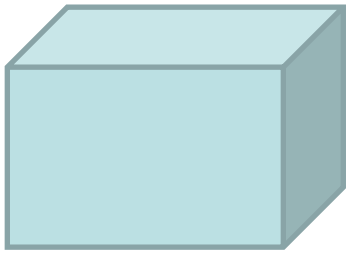


$n_H^{[1]} = n_W^{[1]} = \frac{n_W^{[1]} + 2*p^{[1]} - f^{[1]}}{s^{[1]}} + 1 = \frac{39+2*0-3}{1} + 1 = 37$

$n_c^{[1]} = 10$

37x37x10

$f^{[2]} = 5$
$p^{[2]} = 0$
$s^{[2]} = 2$
$n_c^{[2]} = 20$

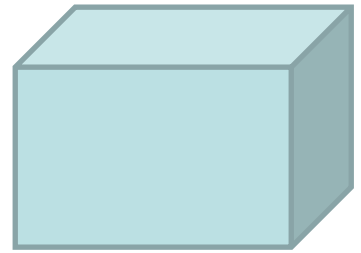


$n_H^{[2]} = n_W^{[2]} = \frac{n_W^{[2]} + 2*p^{[2]} - f^{[2]}}{s^{[2]}} + 1 = \frac{37+2*0-5}{2} + 1 = 17$

$n_c^{[2]} = 20$

17x17x20

$f^{[3]} = 5$
$p^{[3]} = 0$
$s^{[3]} = 2$
$n_c^{[3]} = 40$



$n_H^{[3]} = n_W^{[3]} = \frac{n_W^{[3]} + 2*p^{[3]} - f^{[3]}}{s^{[3]}} + 1 = \frac{17+2*0-5}{2} + 1 = 7$

$n_c^{[3]} = 40$

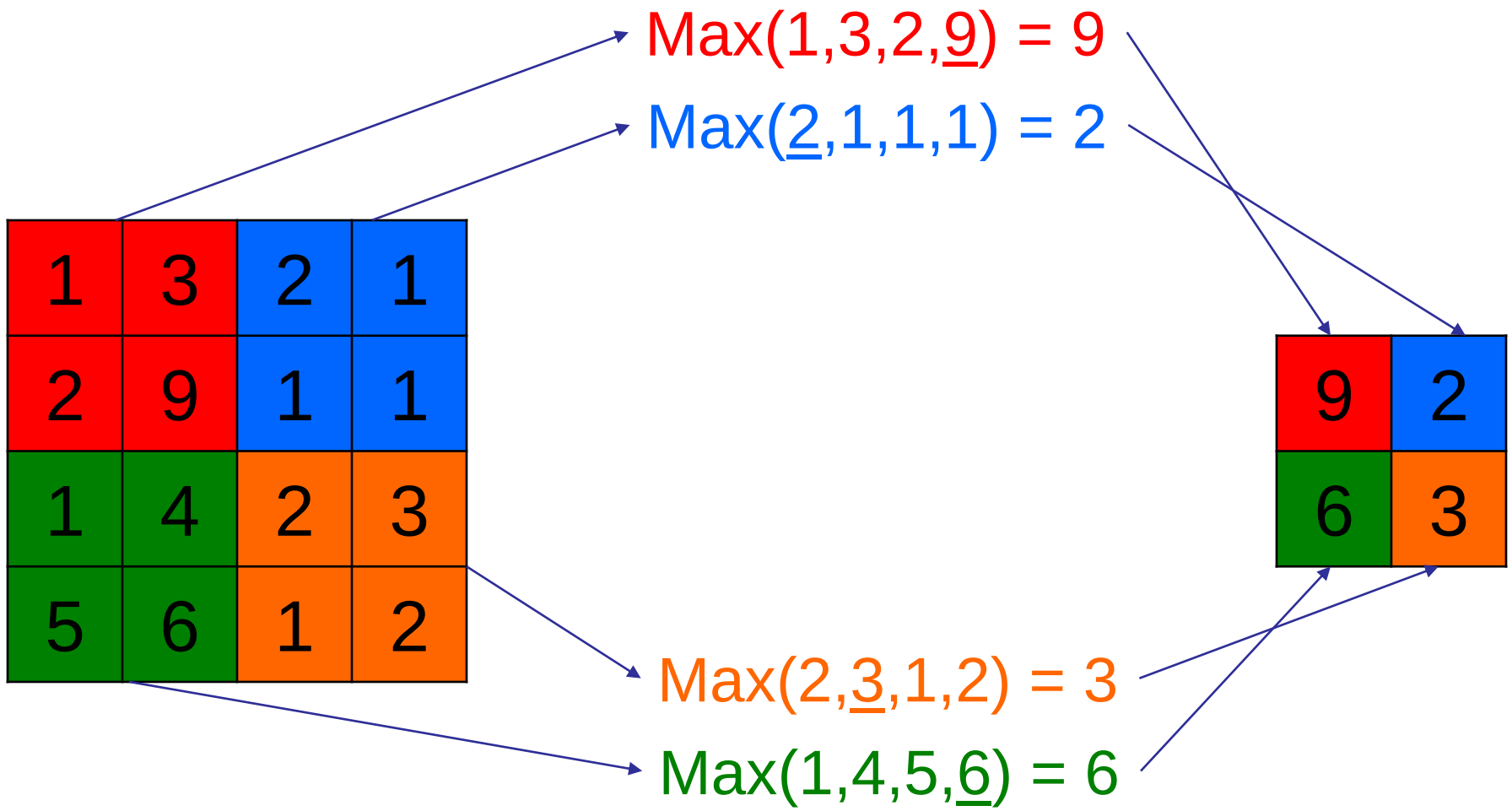
7x7x40

POOLING Layers

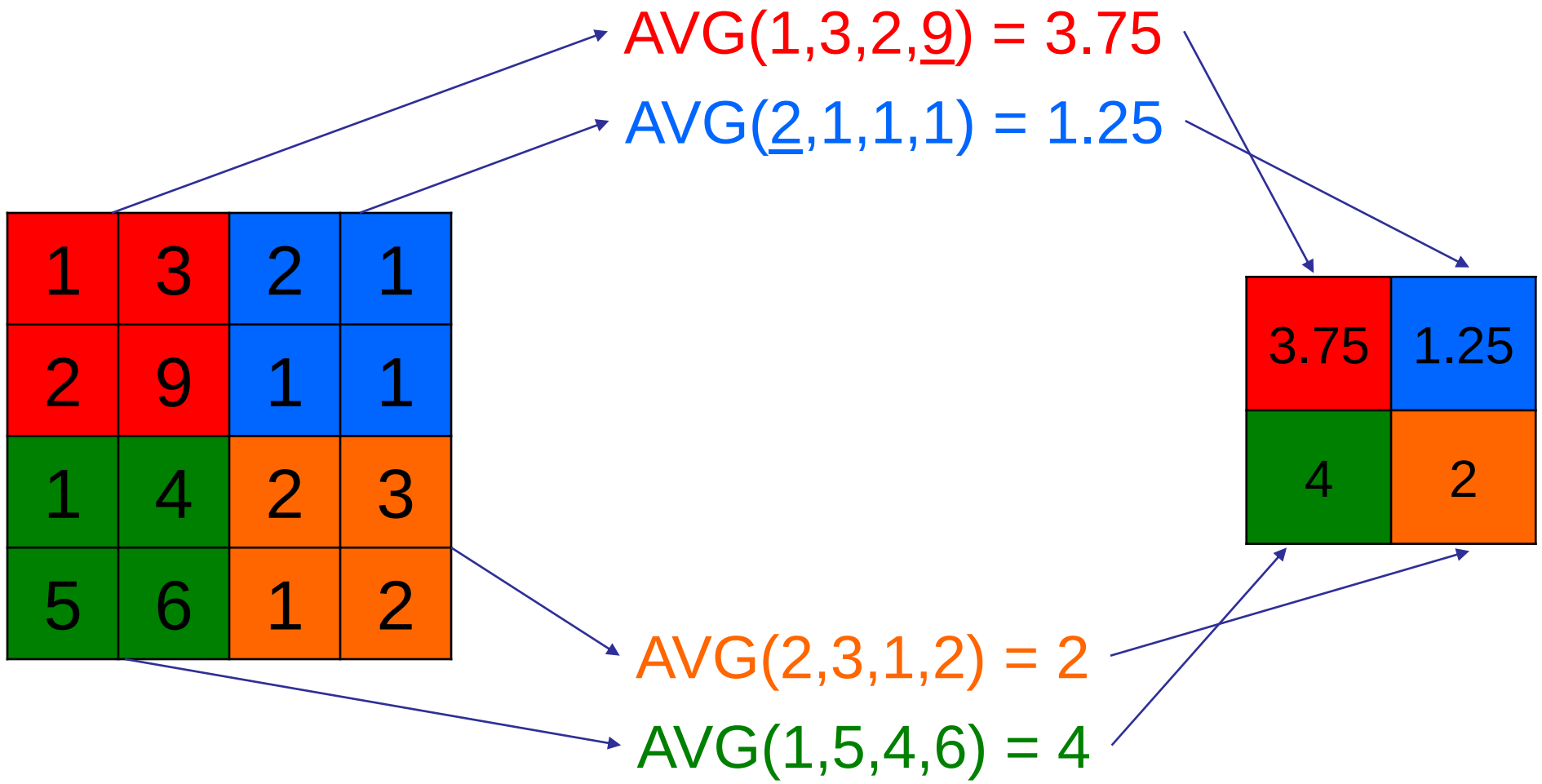
Kiến trúc lớp POOLING

- Lớp Pooling dùng để giảm không gian của ma trận tích chập ➔ giảm số lượng tham số của mô hình.
- Lớp Pooling thường được đặt sau lớp CONV.
- Có 2 dạng lớp Pooling thường gặp:
  + Max Pooling.
  + Average Pooling.
- Các lớp Pooling không học tham số.

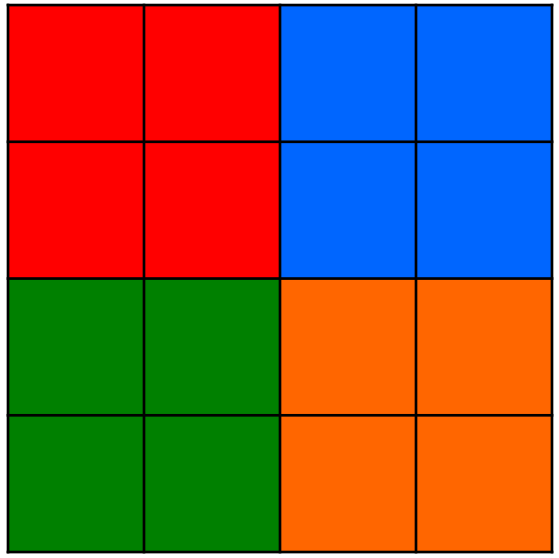
Ví dụ 1: Max pooling



Ví dụ 2: Average pooling

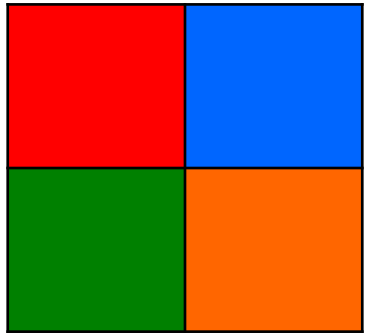


Công thức tính ma trận đầu ra



4x4

F = 2
S = 2



2x2

$$P = \frac{N - F}{S} + 1$$

Ma trận Pooling

Ma trận đầu vào: N = 4 (4x4)
F = 2, S = 2
➔ $P = \frac{4-2}{2} + 1 = 2$
➔ Ma trận Pooling: 2x2

FULLY CONNECTED Layers (FC Layer)

FC layers

- Lớp Fully connected là một lớp có chức năng kết nối tất cả các neural trong lớp hiện tại với lớp trước đó.
- Fully connected layer sẽ nhận giá trị đầu vào ở lớp trước đó, làm phẳng (flattern) và chuyển thành vector 1 chiều.
- Đầu ra của lớp FC là một vector 1 chiều.
- Fully connected layer là lớp chiếm trọng số nhiều nhất trong mô hình mạng tích chập.

Ví dụ về Flattern

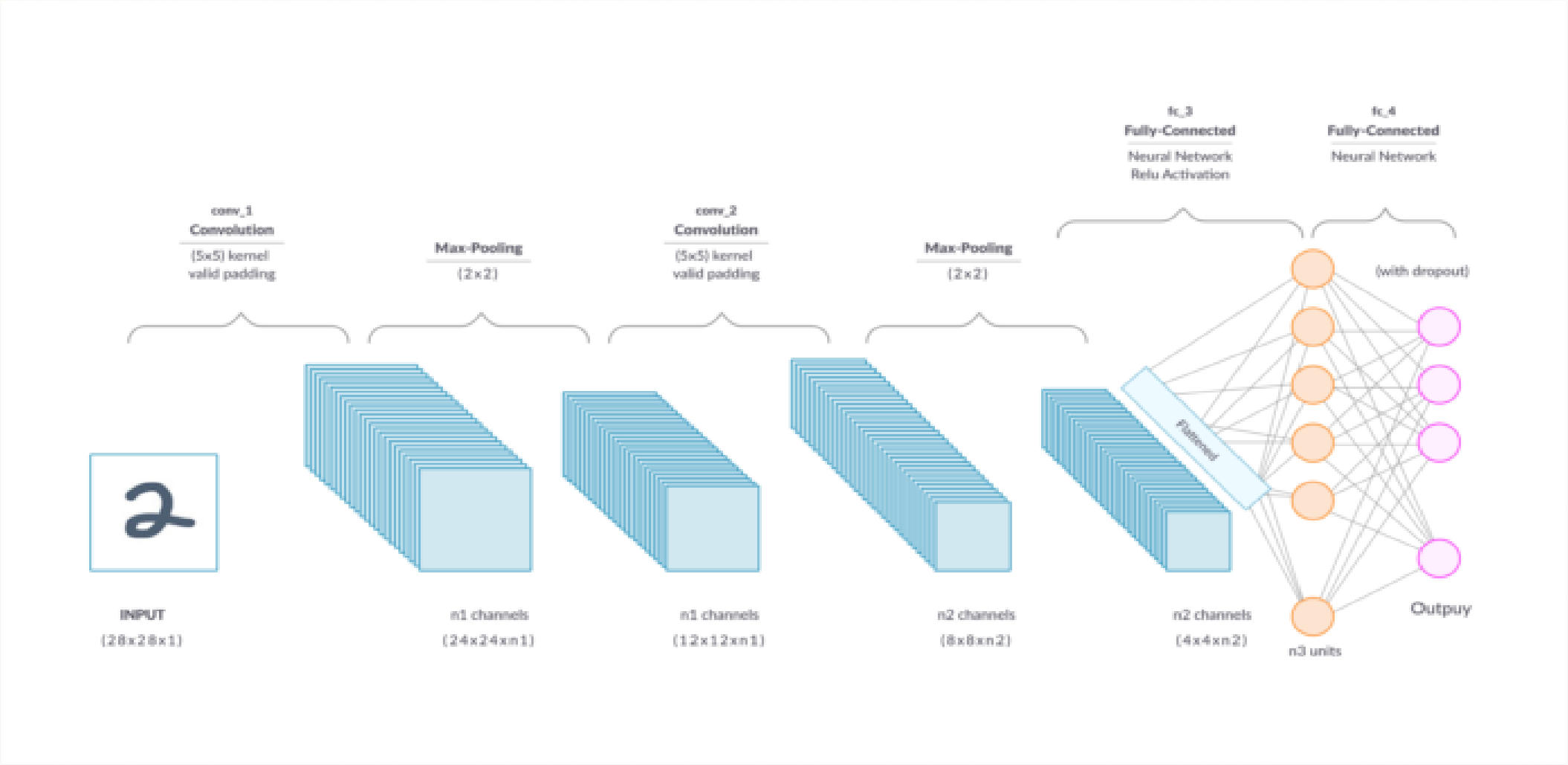
1	1	0
4	2	1
0	2	1

3x3x1

1
1
0
4
2
1
0
2
1

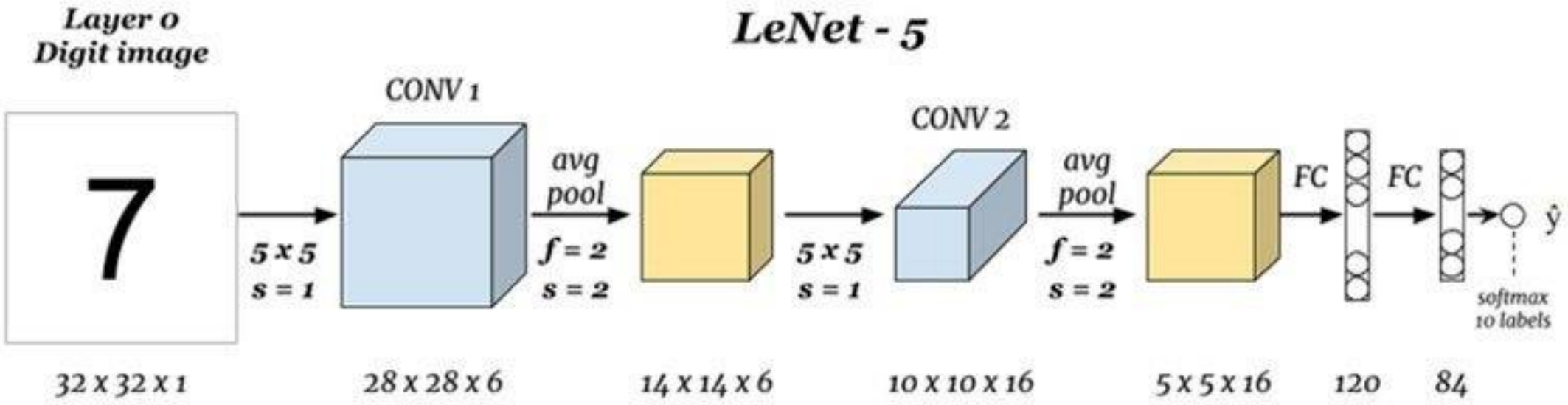
9x1x1

Ví dụ về Fully connected layers



KIẾN TRÚC MỘT MẠNG TÍCH CHẬP

Kiến trúc LeNET

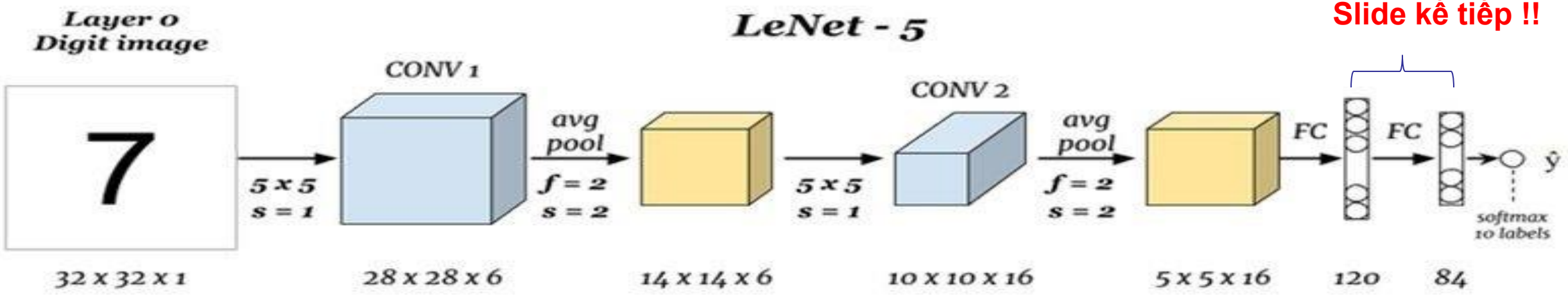


INPUT → CONV1 → POOL → CONV2 → POOL → FC → FC → SOFTMAX

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, november 1998

Tính số lượng tham số

Slide kế tiếp !!



Shape: 32x32x1

Params: 0

#Filter (n_c): 6
Filter shape (F): 5x5
Shape: 28x28x6
(N-F+2P)/S + 1
= 28

Params:
(Weight + bias) x n_c
= (5x5x1)x6 + 1*6 = 156

Shape: 14x14x6
(N-F+2P)/S + 1
= 14

Params: 0

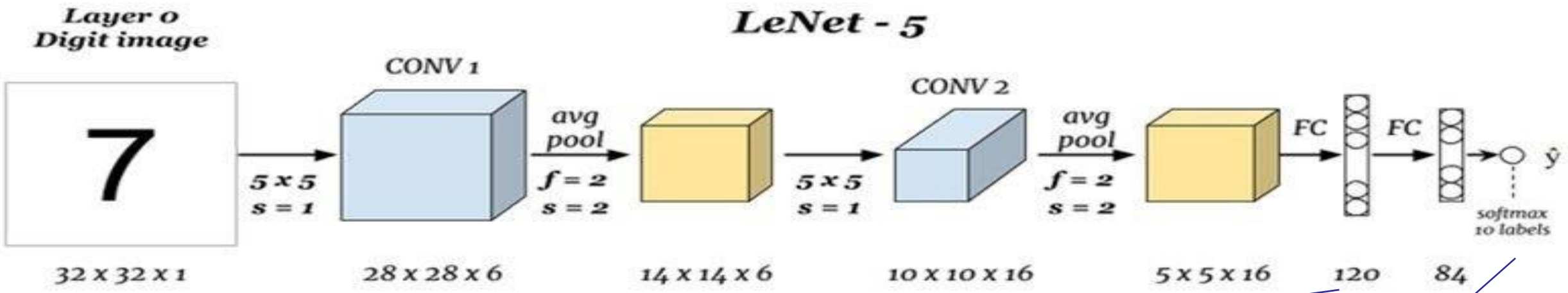
#Filter (n_c): 16
Filter shape (F): 5x5
Shape: 10x10x16
(N-F+2P)/S + 1
= 10

Params:
(Weight + bias) x n_c
= (5x5x6)x16 + 1*16 = 2,416

Shape: 5x5x16
(N-F+2P)/S + 1
= 5

Params: 0

Tính số lượng tham số (tt)



Shape: 120x1
Param: Weight + bias
(5x5x16)x120 + 1*120
= 48,120

Shape: 84x1
Param: Weight + bias
120 * 84 + 1*84
= 10,164

Shape: 10x1
Param: Weight + bias
10 * 84 + 1*10
= 850

Tổng hợp: LeNET 5

STT	Tên layer	Shape	Layer Size	#Param
1	INPUT	32x32x1	1,024	0
2	CONV1 (f=5, s=1)	28x28x6	4,704	156
3	POOL	14x14x6	1,176	0
4	CONV2 (f=2, s=2)	10x10x16	1,600	2,416
5	POOL	5x5x16	400	0
6	FC	120x1	120	48,120
7	FC	84x1	84	10,164
8	Softmax	10x1	10	850
Tổng tham số				61,706

Bài tập

- Tính tổng tham số cho mô hình LeNET 5 với bức ảnh đầu vào có màu với 3 kênh: R,G,B.

TẠI SAO TÍCH CHẬP HIỆU QUẢ

1. Parameters sharing.

   Một bộ lọc (dung để phát hiện features – feature detectors) có thể áp dụng hiệu quả cho nhiều dữ liệu khác nhau.

2. Sparsity connections.

   Ở mỗi layer, giá trị output chỉ phụ thuộc vào một phần nhỏ của input (Nhớ lại cách chạy của tích chập).

Implement LeNET-5

Giới thiệu Keras

- Keras là một framework hỗ trợ phát triển các mô hình deep learning (Deep learning framework) một cách nhanh chóng.
- Ngôn ngữ: Python.
- Tính tới thời điểm Tháng 02/2020:
  + Keras đứng vị trí số 2, sau Tensor flows.
  + Số stars trên github: 46K.

https://github.com/mbadry1/Top-Deep-Learning

Các ưu điểm của Keras

- Keras ưu tiên trải nghiệm của người lập trình.
- Keras đã được sử dụng rộng rãi trong doanh nghiệp và cộng đồng nghiên cứu.
- Keras giúp dễ dàng biến các thiết kế thành sản phẩm.
- Keras hỗ trợ huấn luyện trên nhiều GPU phân tán.
- Keras hỗ trợ đa backend engines và không giới hạn bạn vào một hệ sinh thái.

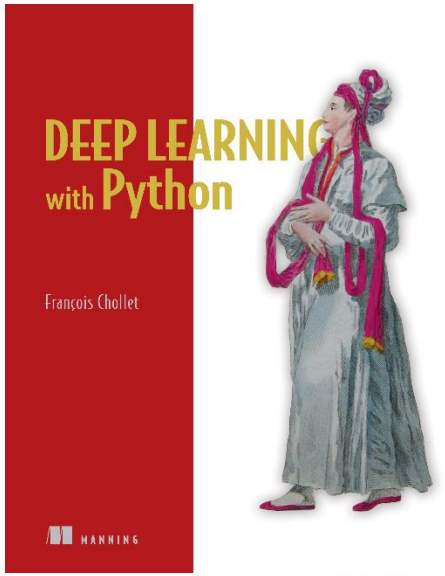
https://machinelearningcoban.com/2018/07/06/deeplearning/
https://keras.io/why_keras/

Thư viện Keras

- Cách cài đặt Keras (yêu cầu đã cài Python >3.0 trước đó):

  pip install keras.

- Đối với Google Colab (https://colab.research.google.com/), Keras được cài đặt mặc định, cùng với Numpy, tensorflow, pandas, scikit learn, ...
- Trang chủ hướng dẫn về keras: https://keras.io/
- Sách: **Deep Learning with Python by Francois Chollet**



Hiện thực LeNET-5 với Keras

```
# Khai báo thư viện
import numpy as np
import pandas as pd
import keras
from keras.models import Sequential
from keras.layers import Conv2D, Dense, Dropout, Flatten,
AveragePooling2D
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ReduceLROnPlateau
```

Hiện thực LeNET-5 với Keras (tt)

```
# Xây dựng mô hình
model = Sequential()
model.add(Conv2D(filters=6, kernel_size=(5,5), padding='same',
activation='relu', input_shape=(28, 28, 1)))
model.add(AveragePooling2D(strides=2, pool_size=(2, 2)))
model.add(Conv2D(filters=16, kernel_size=(5,5), padding='valid',
activation='relu'))
model.add(AveragePooling2D(strides=2, pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(120, activation='relu'))
model.add(Dense(84, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

Hiện thực LeNET-5 với Keras (tt)

```
# Build mô hình
model.compile()

# In ra kiến trúc mô hình đã build
model.summary()
```

Kết quả xây dựng mô hình LeNET-5

```
Model: "sequential_21"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_41 (Conv2D)           (None, 28, 28, 6)         156
_________________________________________________________________
average_pooling2d_29 (Averag (None, 14, 14, 6)         0
_________________________________________________________________
conv2d_42 (Conv2D)           (None, 10, 10, 16)        2416
_________________________________________________________________
average_pooling2d_30 (Averag (None, 5, 5, 16)          0
_________________________________________________________________
flatten_20 (Flatten)         (None, 400)               0
_________________________________________________________________
dense_60 (Dense)             (None, 120)               48120
_________________________________________________________________
dense_61 (Dense)             (None, 84)                10164
_________________________________________________________________
dense_62 (Dense)             (None, 10)                850
=================================================================
Total params: 61,706
Trainable params: 61,706
Non-trainable params: 0
_________________________________________________________________
```

Tổng kết

- Các lớp chính trong một mạng neural tích chập: CONV, POOL, Fully-connected (FC).
- Các lớp tốn tham số huấn luyện của mô hình: CONV và FC.
- Lớp FC tốn nhiều tham số nhất.
- Tính số lượng tham số cho LeNET-5 (61K).

TÀI LIỆU THAM KHẢO

1. Khoá học *Neural Network and Deep learning*, deeplearning.ai.
2. Ian Goodfellow, Yoshua Bengio, Aaron Courvile, *Deep learning*, MIT Press, 2016.
3. Andrew Ng., *Machine Learning Yearning*. Link: https://www.deeplearning.ai/machine-learning-yearning/
4. Vũ Hữu Tiệp, *Machine Learning cơ bản*, NXB Khoa học và Kỹ thuật, 2018.