Deep Learning

Chapter 4 Tensorflow

Dr. Van-Toi NGUYEN

EEE, Phenikaa University

Chapter 3: Convolutional Neural Network

1. Convolutional operator
2. History of CNN
3. Layers in CNN
4. Deep Convolutional Models
5. Applications of CNN
6. Practice

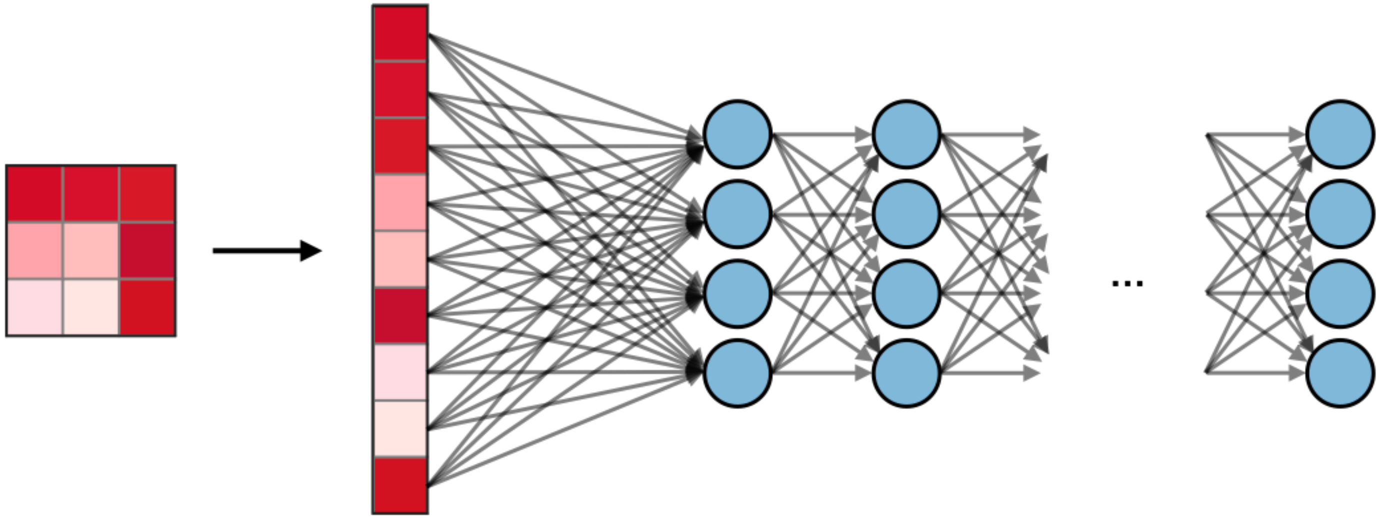
3.4 Deep Convolutional Models

Learning Objectives

- ✓ Be able to explain what a convolutional layer does and how it's different from a fully-connected layer
- ✓ Understand the assumptions and trade-offs that are being made when using convolutional architectures
- ✓ Be able to build a convolutional architecture using Tensorflow 2.0 and Keras Layers
- ✓ Be able to use Keras to train a model on a dataset
- ✓ Implement either network together

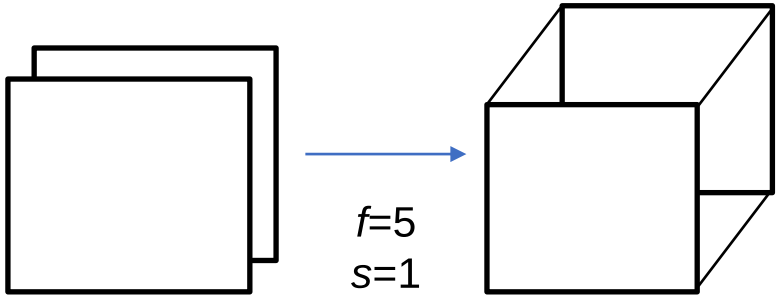
3.4 Deep Convolutional Models

The fully connected layer (FC) operates on a flattened input where each input is connected to all neurons. If present, FC layers are usually found towards the end of CNN architectures and can be used to optimize objectives such as class scores.



3.4 Deep Convolutional Models

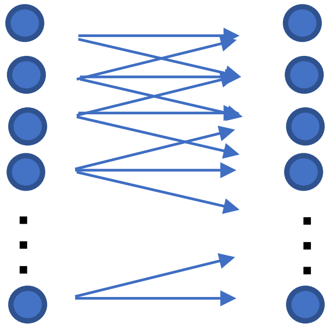
CONV1



f=5
s=1
6 filters

32 x 32 x 3
= 3,072

28 x 28 x 6
= 4,704



3,072 4,704

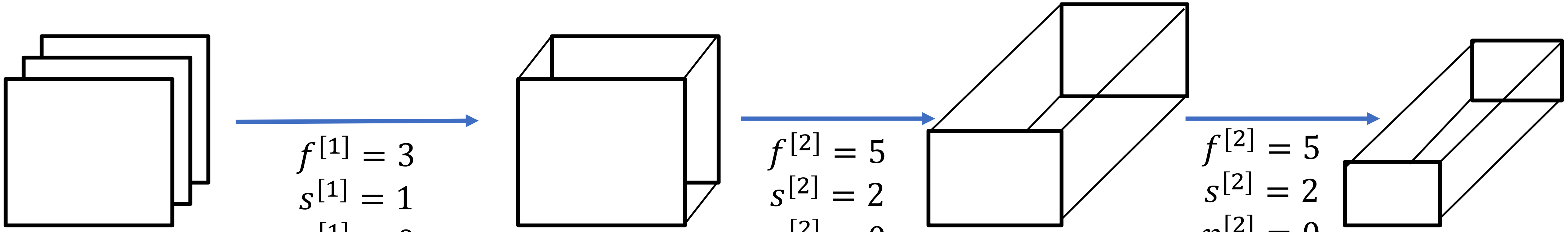
6 x (5 x 5 + 1) = 156 parameters

3,072 x 4,704 ≈ 14M

Large
Difference!!

3.4 Deep Convolutional Models

Example



39 x 39 x 3

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

$n_C^{[0]} = 3$

$f^{[1]} = 3$

$s^{[1]} = 1$

$p^{[1]} = 0$

10 filters

37 x 37 x 10

$n_H^{[1]} = n_W^{[1]} = 37$

$n_C^{[1]} = 10$

$f^{[2]} = 5$

$s^{[2]} = 2$

$p^{[2]} = 0$

20 filters

17 x 17 x 20

$n_H^{[2]} = n_W^{[2]} = 17$

$n_C^{[2]} = 20$

$f^{[2]} = 5$

$s^{[2]} = 2$

$p^{[2]} = 0$

40 filters

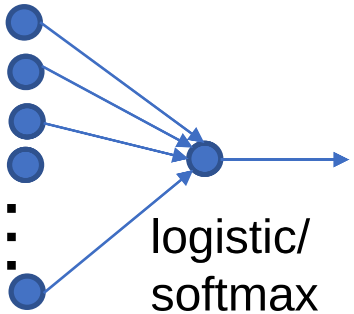
7 x 7 x 40

$n_H^{[3]} = n_W^{[3]} = 7$

$n_C^{[3]} = 40$

$$\frac{n + 2p - f}{s} + 1 = \frac{39 + 0 - 3}{1} + 1 = 37$$

7 x 7 x 40 =1960

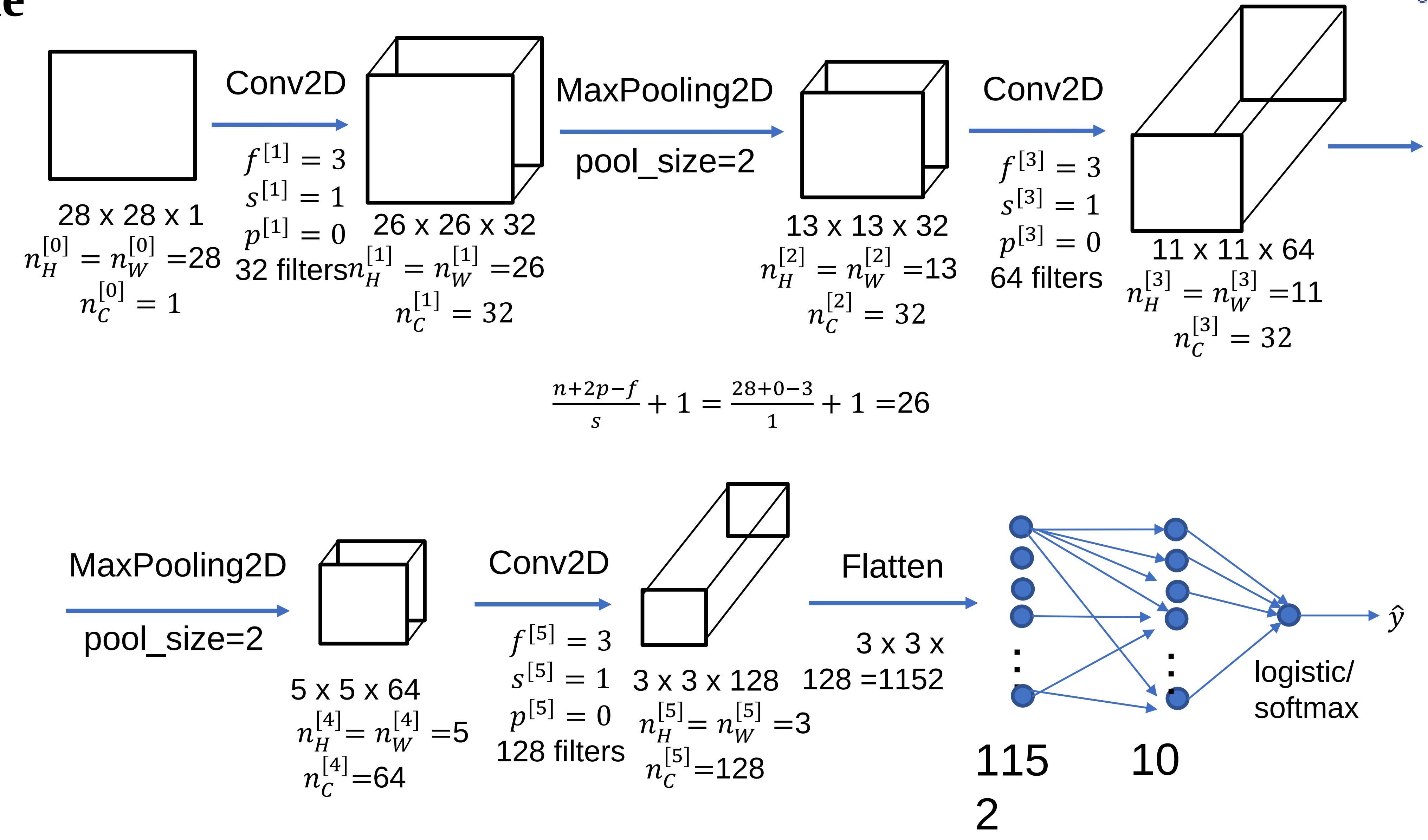


logistic/ softmax

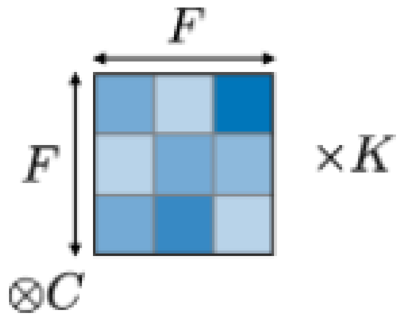
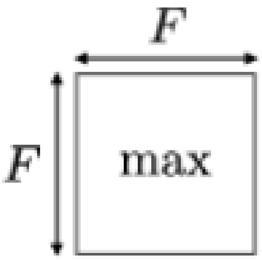
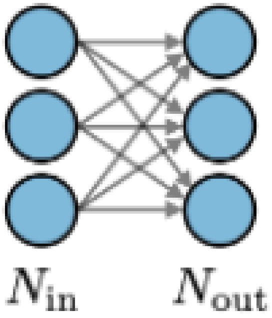
$\hat{y}$

3.4 Deep Convolutional Models

Example



Complexity

	CONV	POOL	FC
Illustration			
Input size	$I \times I \times C$	$I \times I \times C$	N_{in}
Output size	$O \times O \times K$	$O \times O \times C$	N_{out}
Number of parameters	$(F \times F \times C + 1) \cdot K$	0	$(N_{\text{in}} + 1) \times N_{\text{out}}$

3.4 Deep Convolutional Models

Visualization example

https://www.cs.ryerson.ca/~aharley/vis/conv/

https://poloclub.github.io/cnn-explainer/

3.5 Applications of CNN

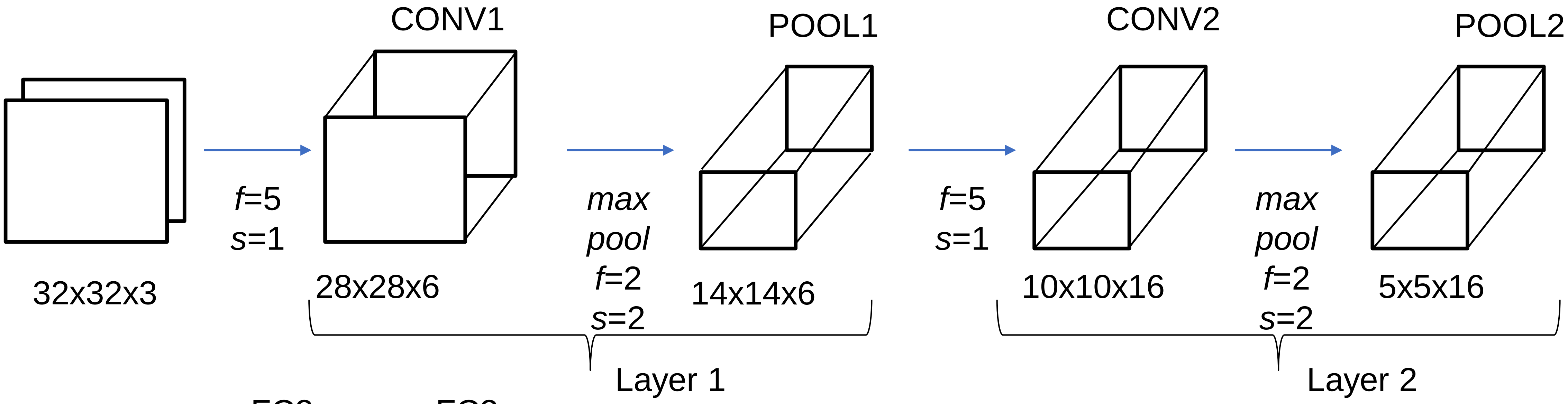
MNIST

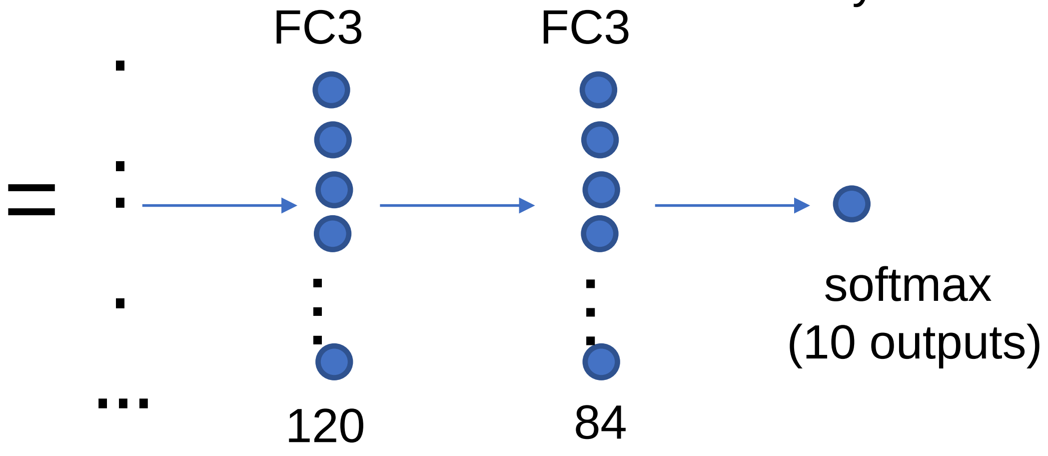
Test accuracy: 0.991

```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 28, 28, 1)]       0
_________________________________________________________________
conv2d (Conv2D)              (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 3, 3, 128)         73856
_________________________________________________________________
flatten (Flatten)            (None, 1152)              0
_________________________________________________________________
dense (Dense)                (None, 10)                11530
=================================================================
Total params: 104,202
Trainable params: 104,202
Non-trainable params: 0
```

```
Epoch 1/5
938/938 [==============================] - 44s 47ms/step - loss: 0.1606 - accuracy: 0.9495
Epoch 2/5
938/938 [==============================] - 43s 46ms/step - loss: 0.0444 - accuracy: 0.9860
Epoch 3/5
938/938 [==============================] - 43s 46ms/step - loss: 0.0309 - accuracy: 0.9906
Epoch 4/5
938/938 [==============================] - 45s 48ms/step - loss: 0.0227 - accuracy: 0.9931
Epoch 5/5
938/938 [==============================] - 46s 49ms/step - loss: 0.0176 - accuracy: 0.9946
```

LeNet-5





Notes:

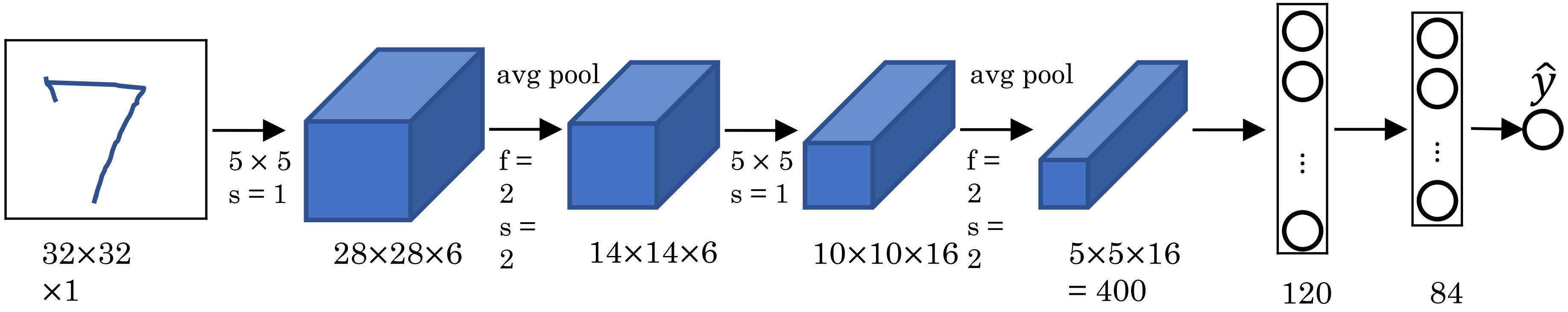
- Look up literatures and use the proven useful set of hyperparameters
- Image size ($n_H \times n_W$) decreases, while the number of channels n_C increases
- CONV-POOL-CONV-POOL-FC-FC-FC-SOFTMAX

LeNet-5

	Activation Shape	Activation Size	# Parameters
Input	(32, 32, 3)	3,072	0
CONV1 (f=5, s=1)	(28, 28, 8)	6,272	208
POOL1	(14, 14, 8)	1,568	0
CONV2 (f=5, s=1)	(10, 10, 16)	1,600	416
POOL2	(5, 5, 16)	400	0
FC3	(120, 1)	120	48,001
FC4	(84, 1)	84	10,081
Softmax	(10, 1)	10	841

3.5 Applications of CNN

LeNet-5



- 60K parameters (small by modern standards)
- n_H, n_W decreased with layers, n_C increased
- CONV – POOL – CONV – POOL – FC – FC - Output

[LeCun et al., 1998. Gradient-based learning applied to document recognition]