



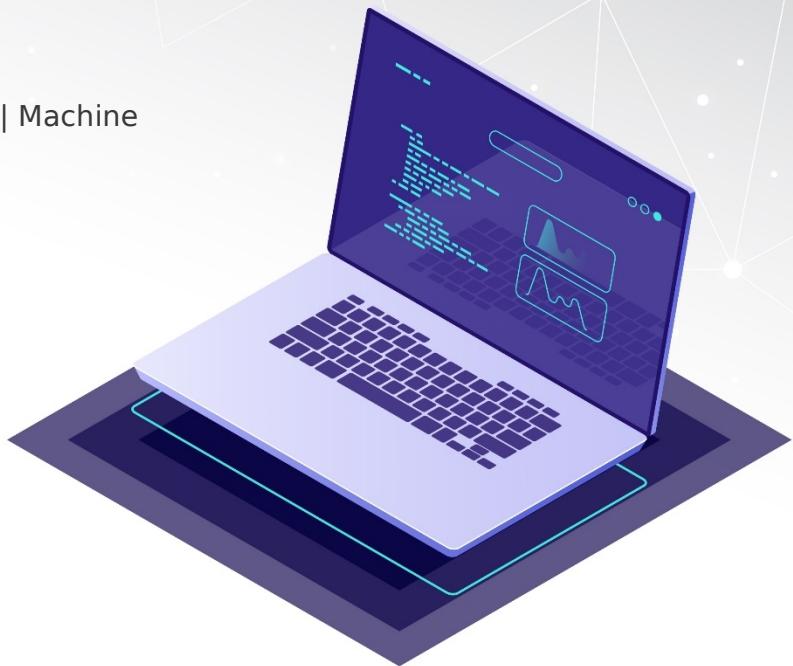
# DIGITAL TALENT SCHOLARSHIP 2019



Program Fresh Graduate Academy Digital Talent Scholarship 2019 | Machine Learning

# Deep Learning: Convolutional Neural Networks

Nama pembicara dengan gelar





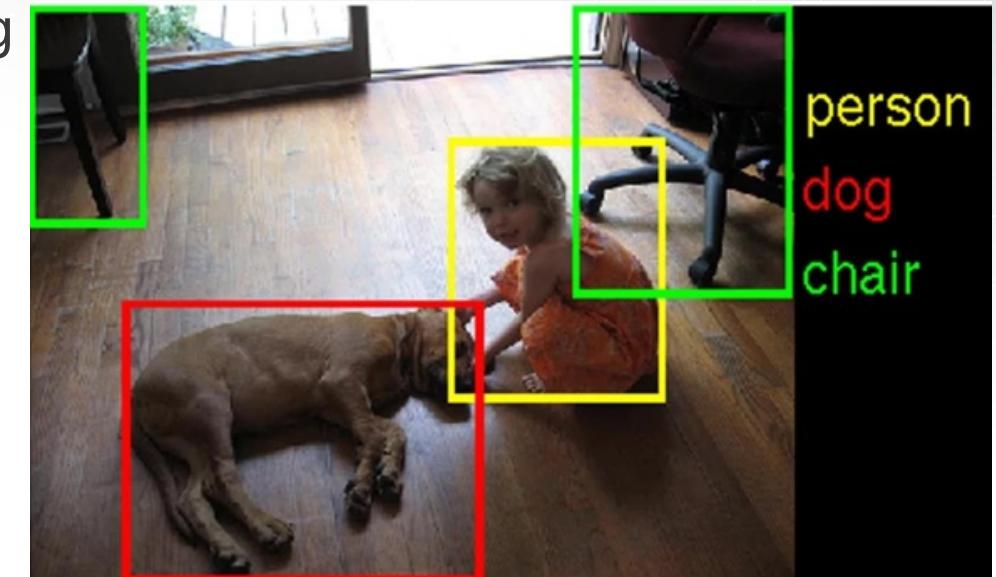
Bagian 1

# Introduction to Convolutional Networks



# Applications

- Signal and image processing
- Handwritten text/digits recognition
- **Natural object classification (photos and videos)**
- Segmentation
- Face detection
- Recommender systems
- Speech recognition
- Natural Language Processing



One of the CNN applications: Natural object classification

# Original goal of CNN

*“How to create **good representations of the visual world** in a way it could be used **to support recognition?**”*

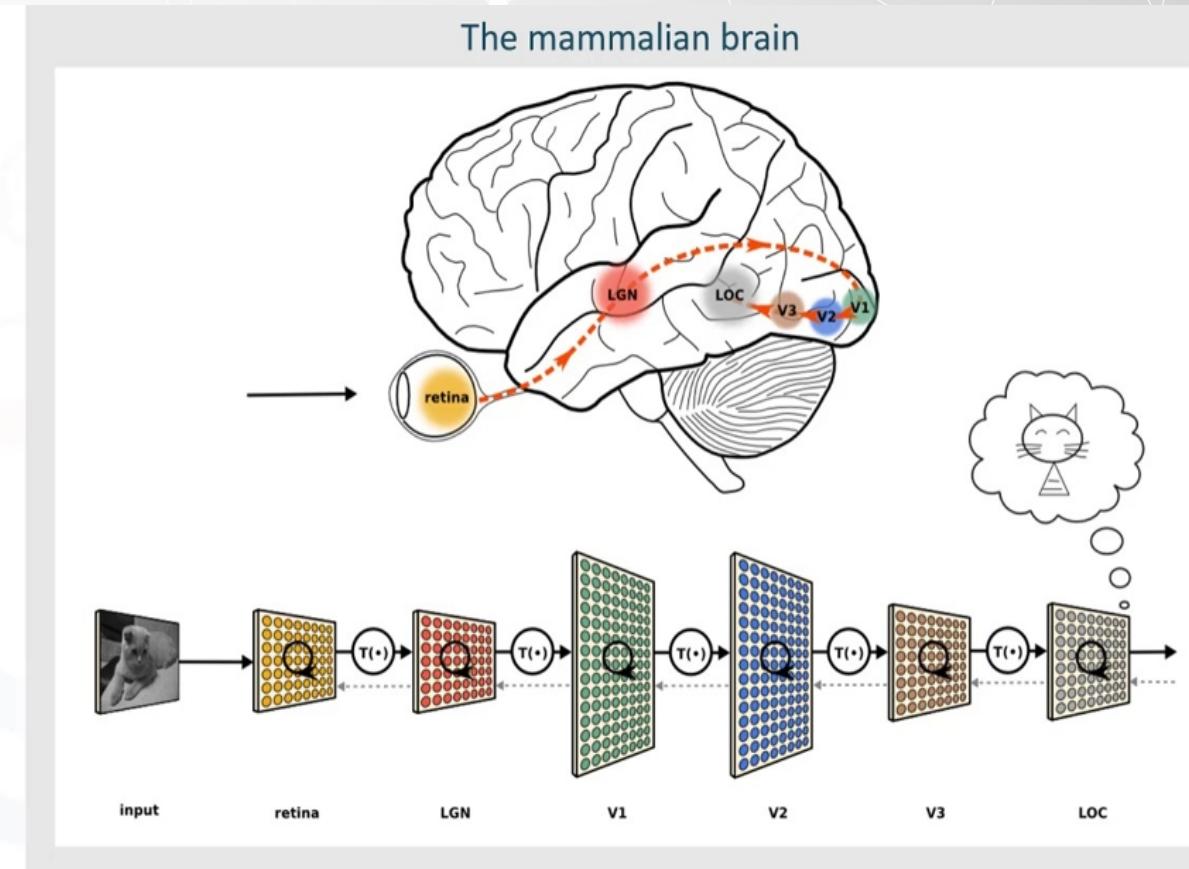
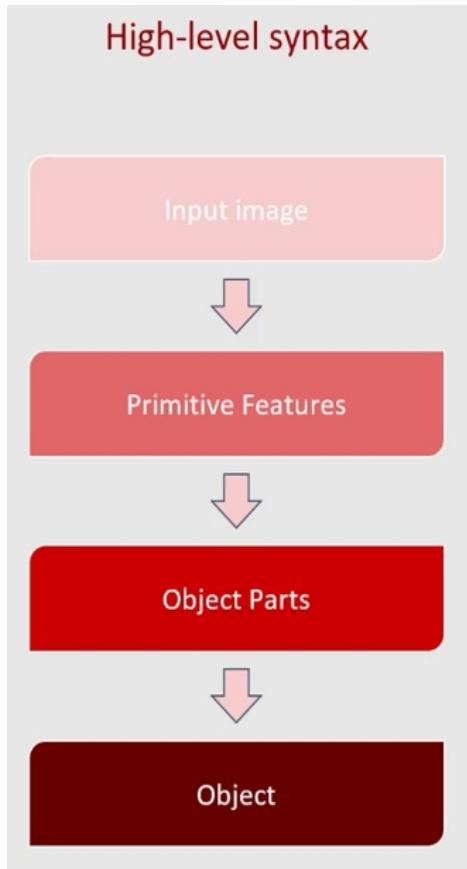
## Key features:

- **Detect and classify** objects into categories
- **Independence** from pose, scale, illumination, conformation and clutter



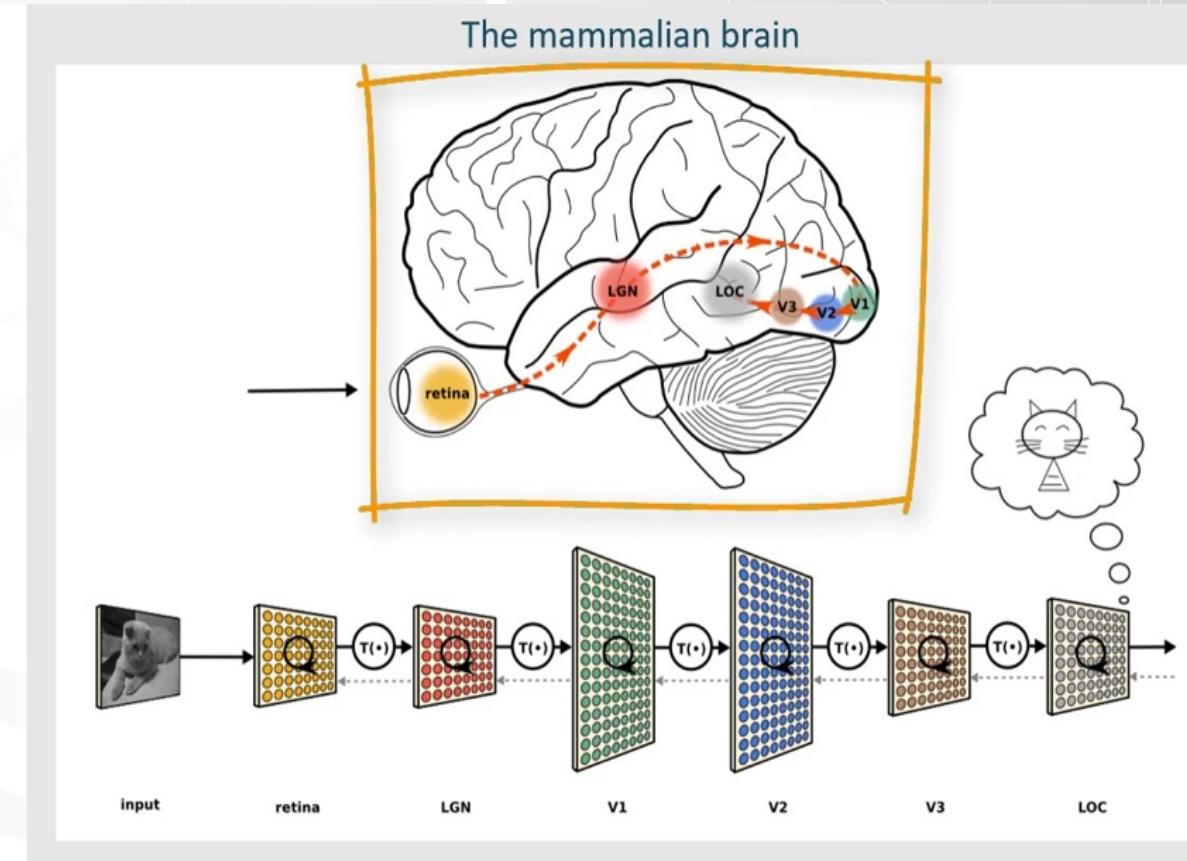
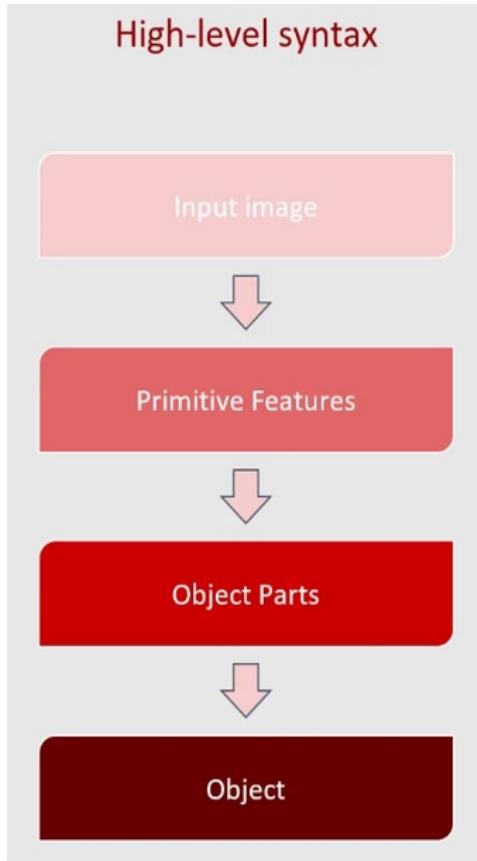


# Inheritance from the real world





# Inheritance from the real world





# Inheritance from the real world

## High-level syntax

Input image



Primitive Features

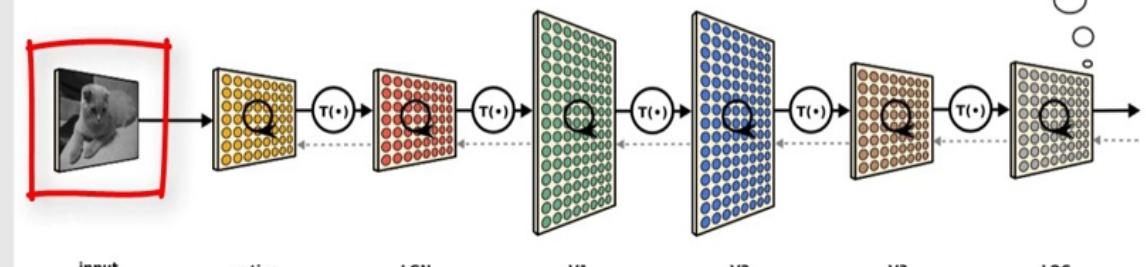
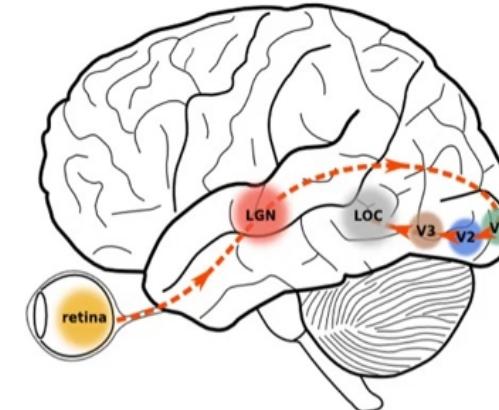


Object Parts



Object

## The mammalian brain





# Inheritance from the real world

## High-level syntax

Input image



Primitive Features

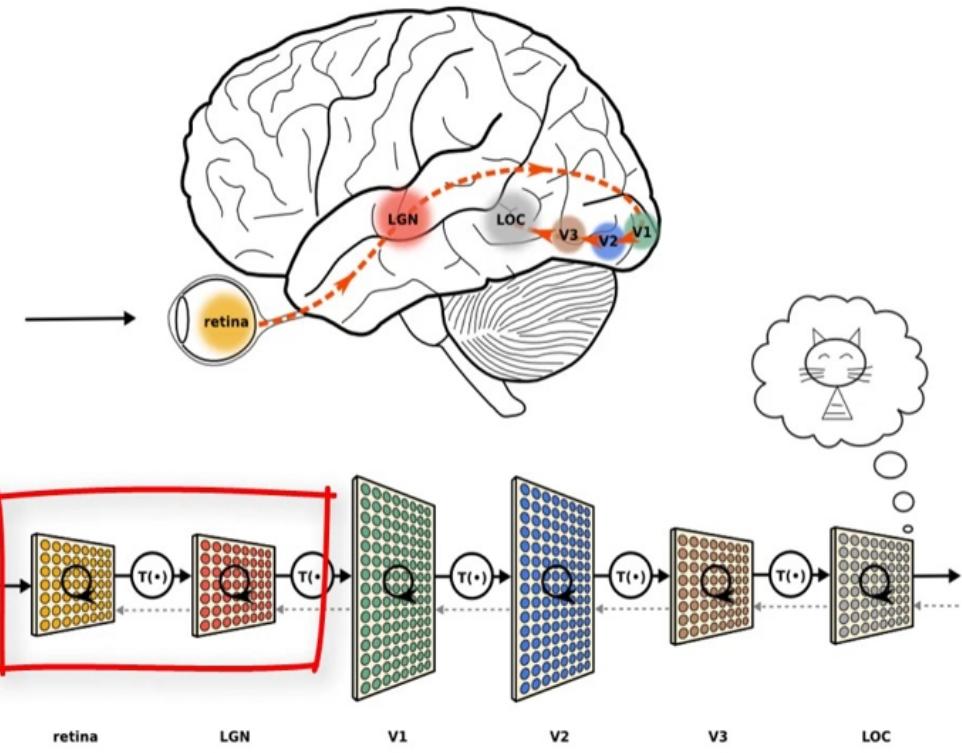


Object Parts



Object

## The mammalian brain



# Inheritance from the real world

## High-level syntax

Input image



Primitive Features

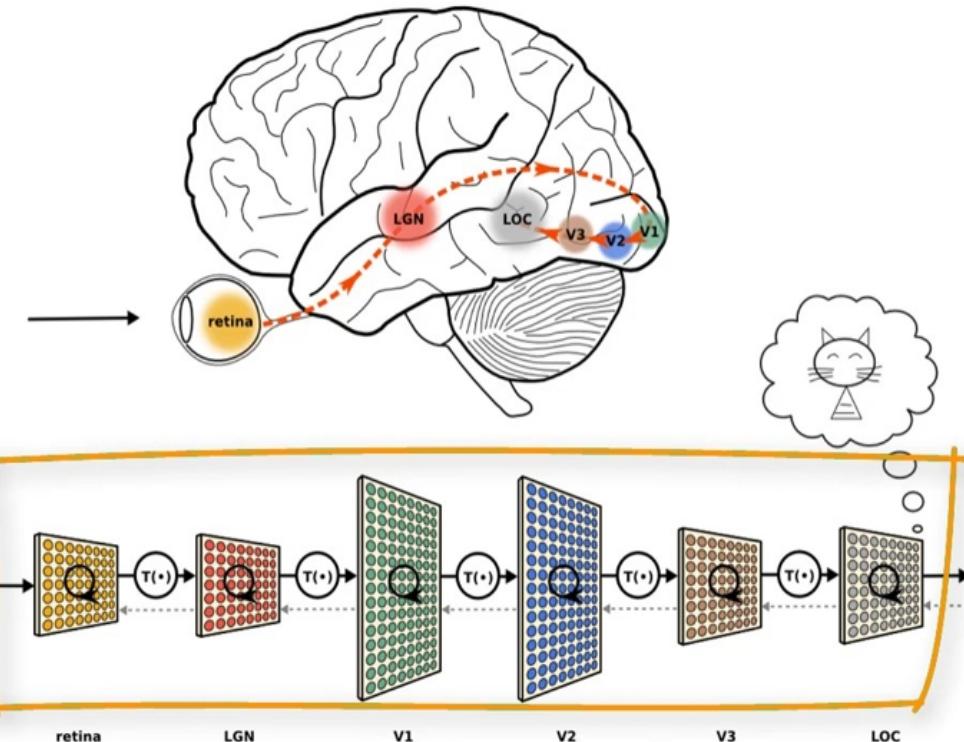


Object Parts



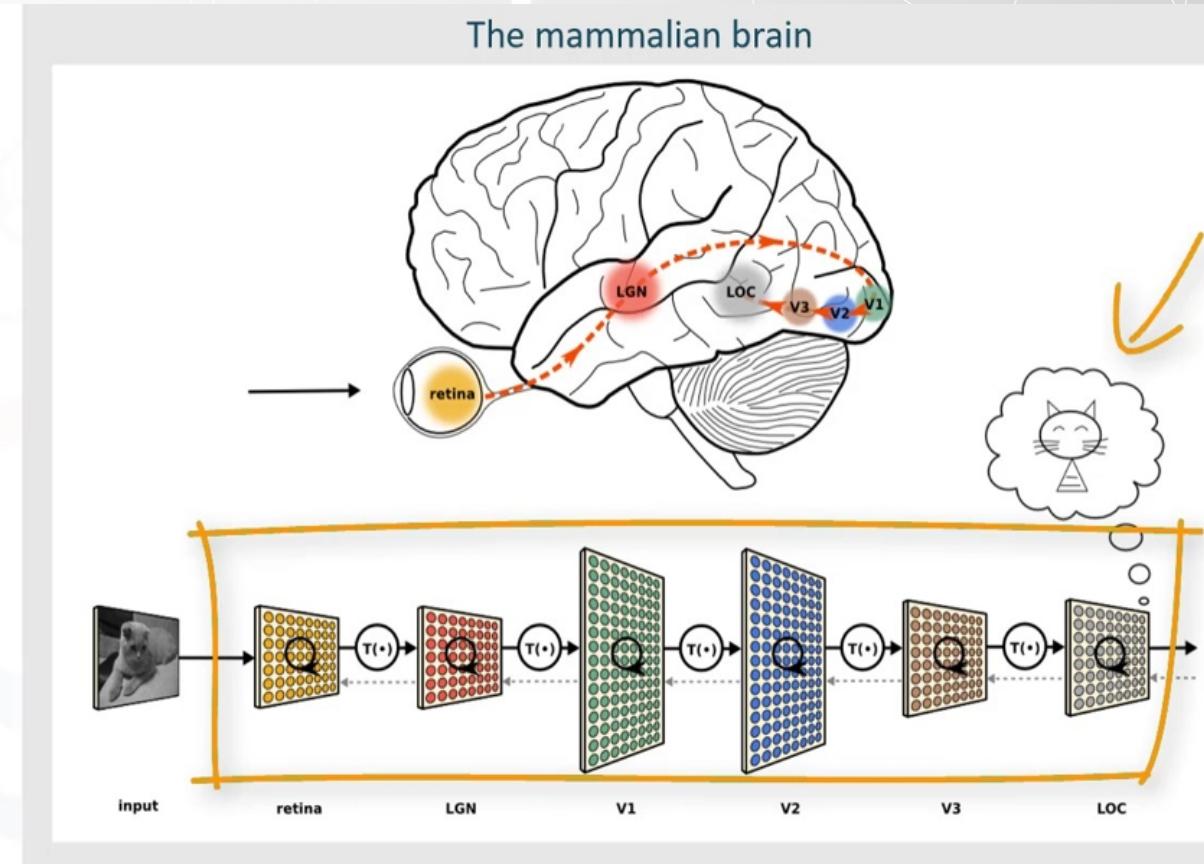
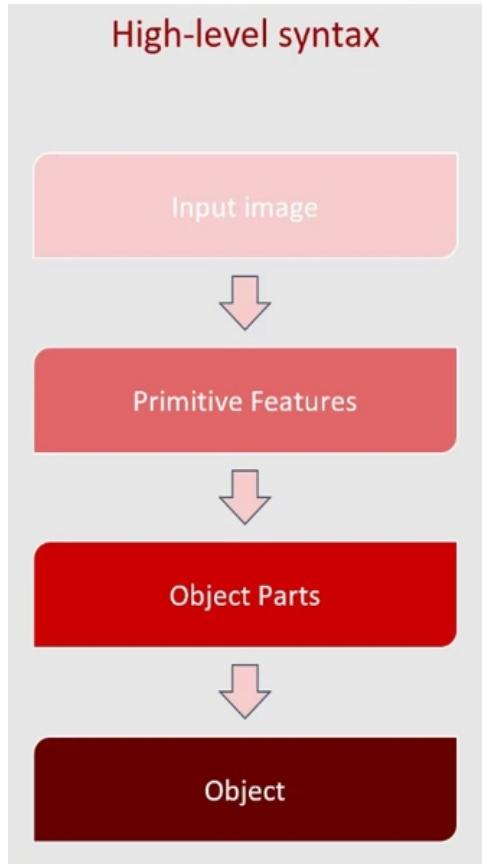
Object

## The mammalian brain



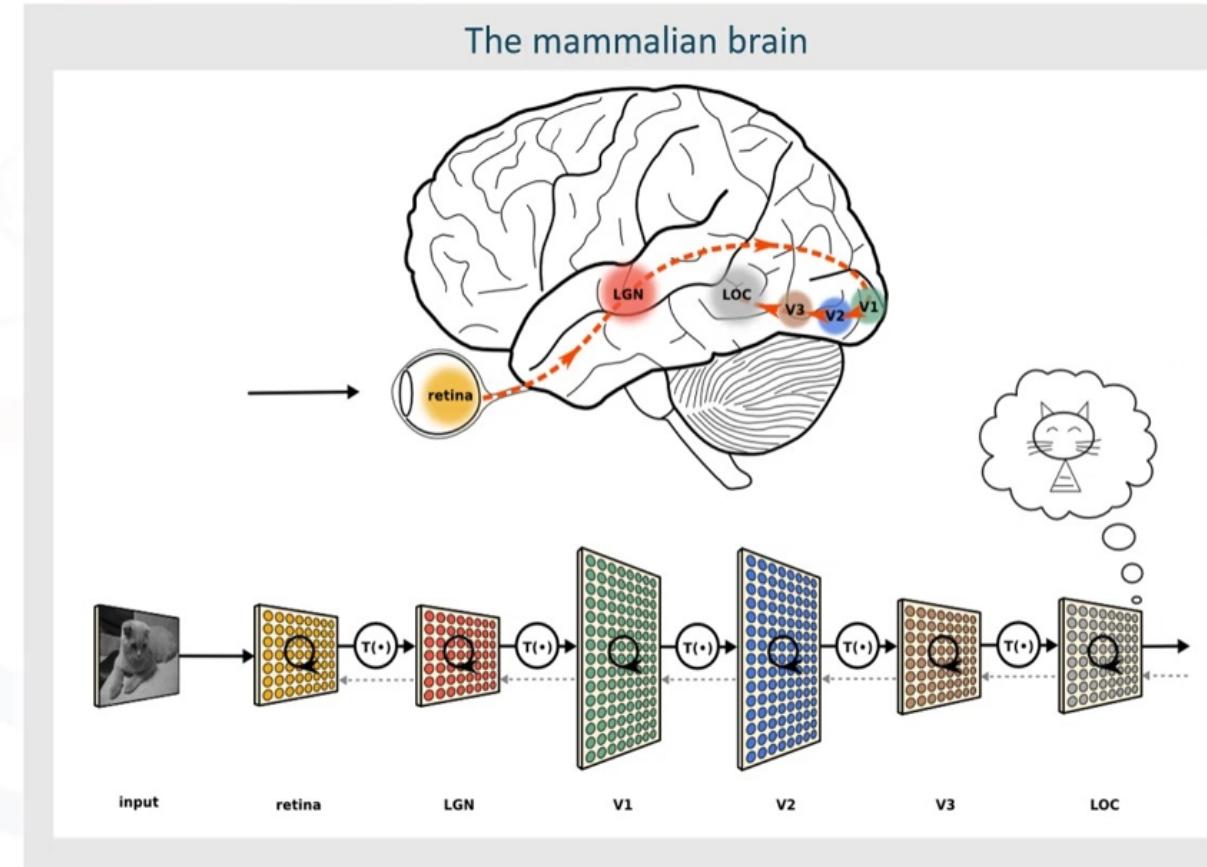
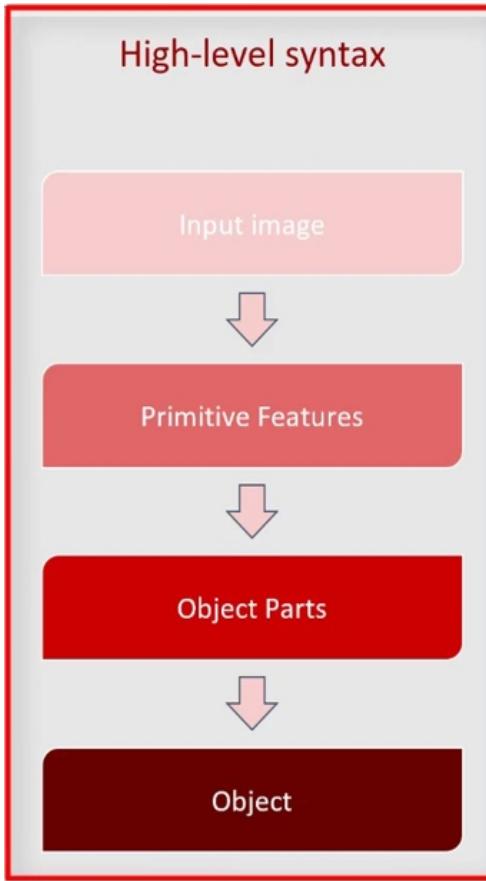


# Inheritance from the real world





# Inheritance from the real world





# Inheritance from the real world

## High-level syntax

Input image



Primitive Features

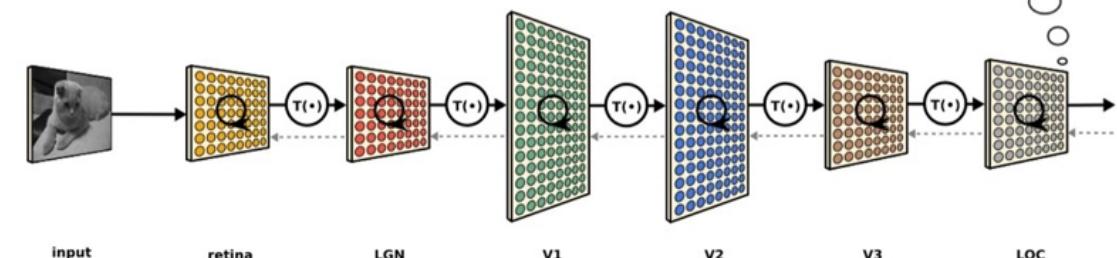
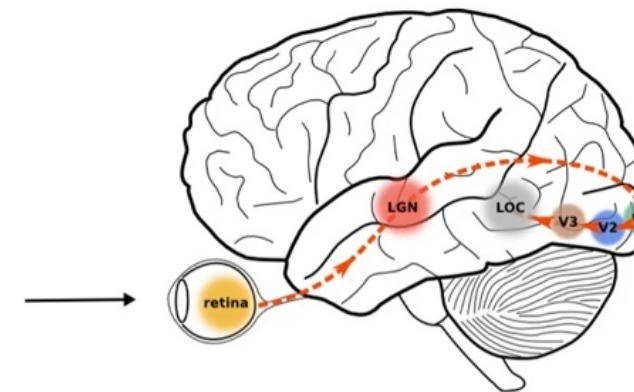


Object Parts



Object

## The mammalian brain





# Inheritance from the real world

## High-level syntax

Input image



Primitive Features

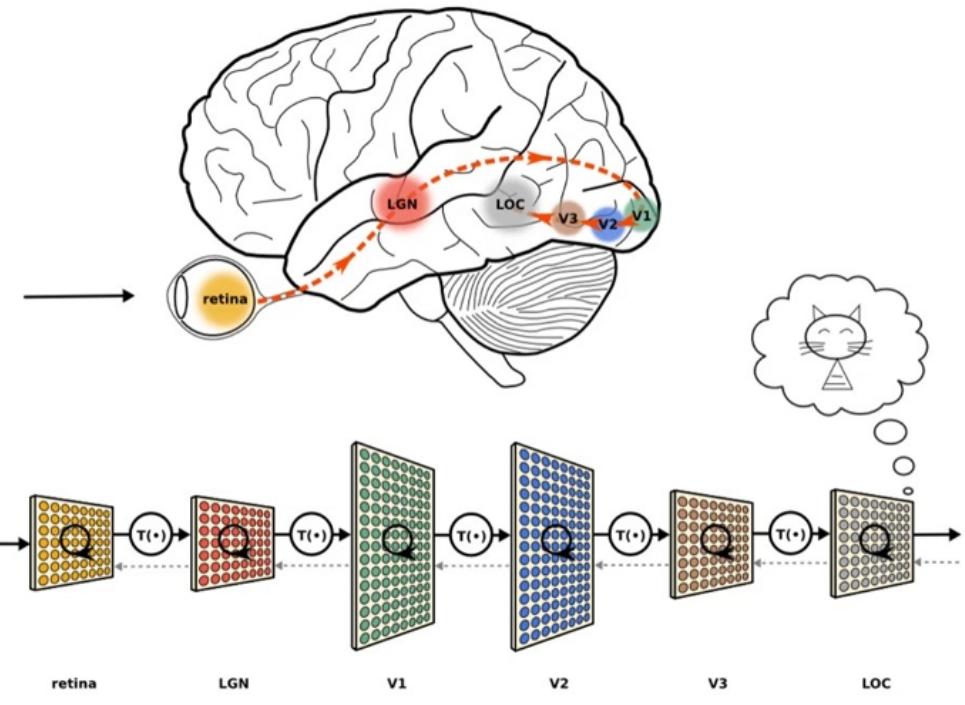


Object Parts



Object

## The mammalian brain



# Inheritance from the real world

## High-level syntax

Input image



Primitive Features

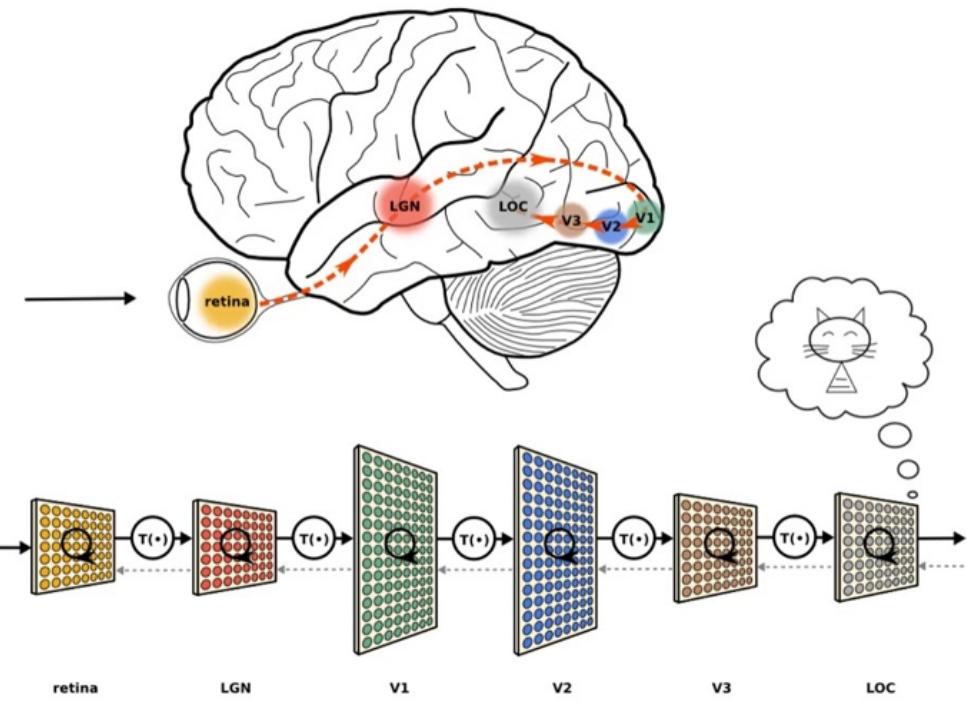


Object Parts

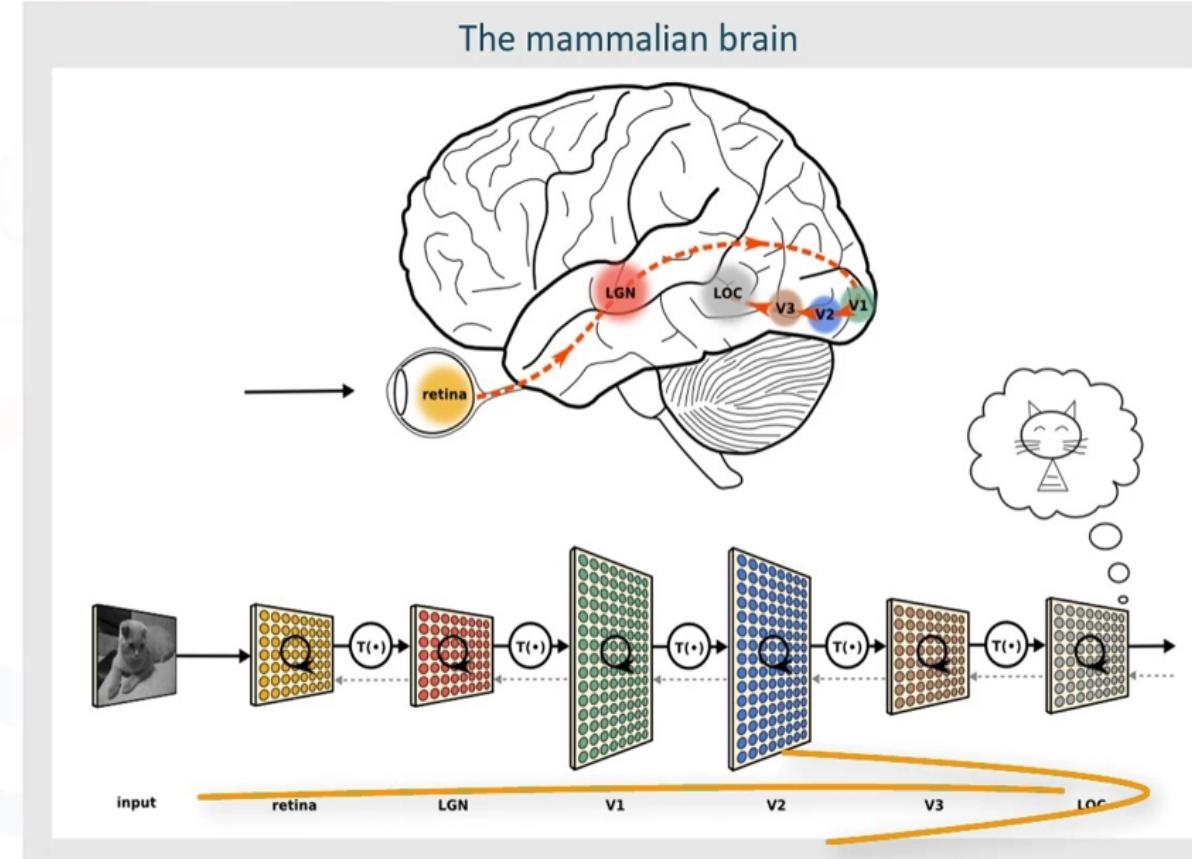
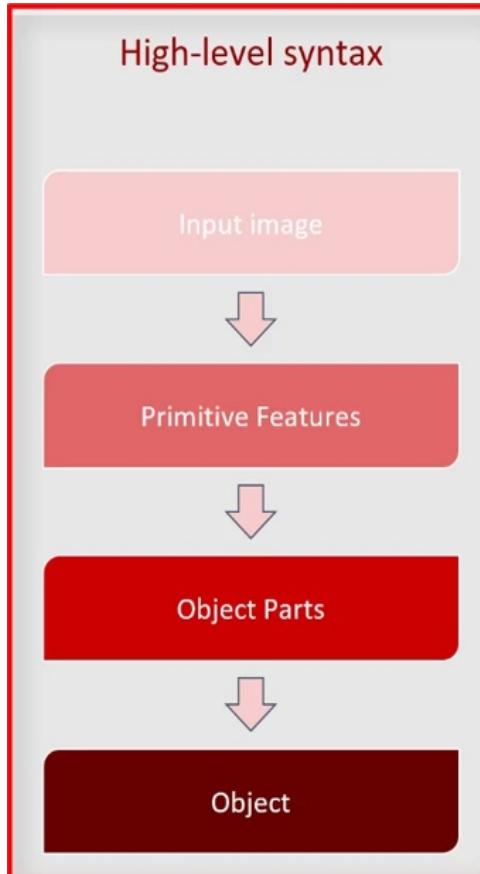


Object

## The mammalian brain



# Inheritance from the real world





# A visual explanation



# A visual explanation



# A visual explanation

Input image

Primitive Features

Object Parts

Object

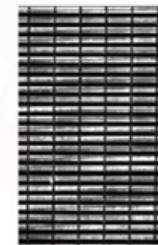


# A visual explanation

Input image



Primitive Features



Object Parts



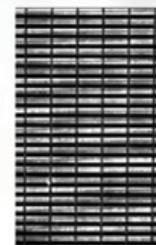
Object

# A visual explanation

Input image



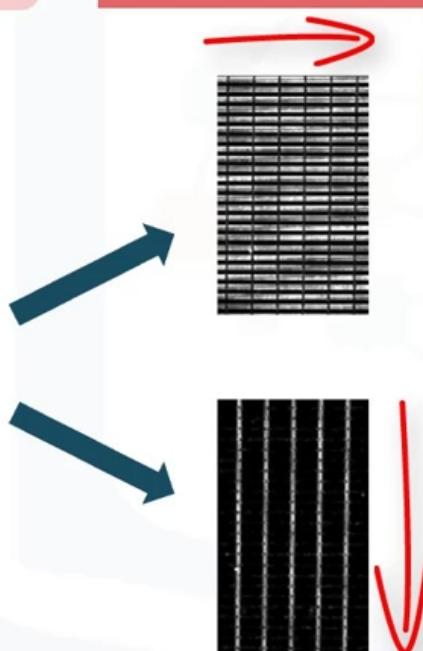
Primitive Features



Object Parts

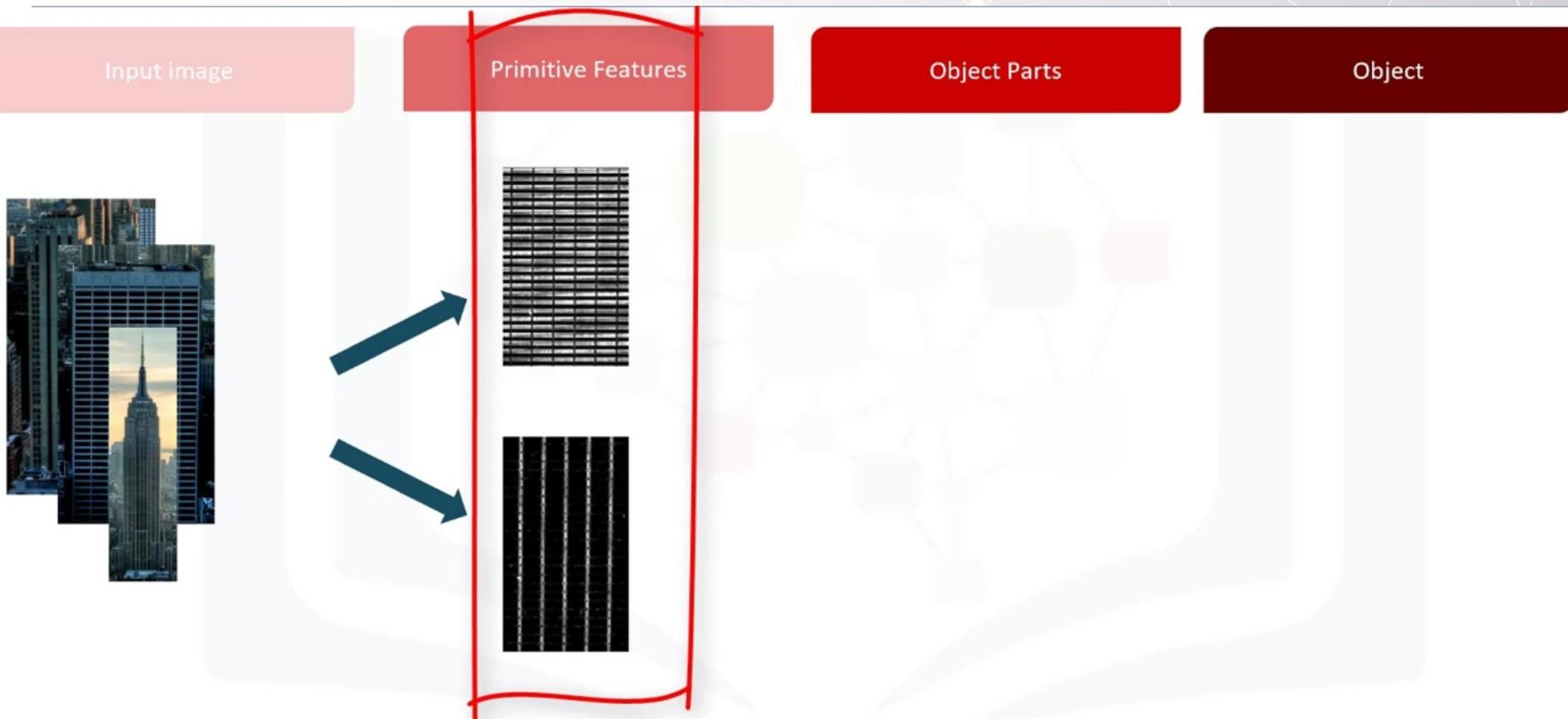


Object





# A visual explanation



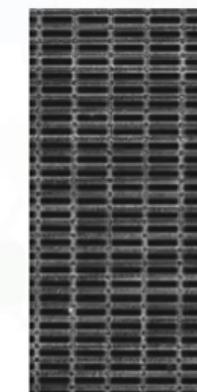
# A visual explanation

Input image

Primitive Features

Object Parts

Object



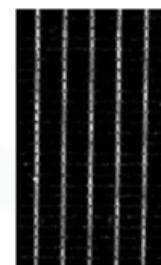


# A visual explanation

Input image



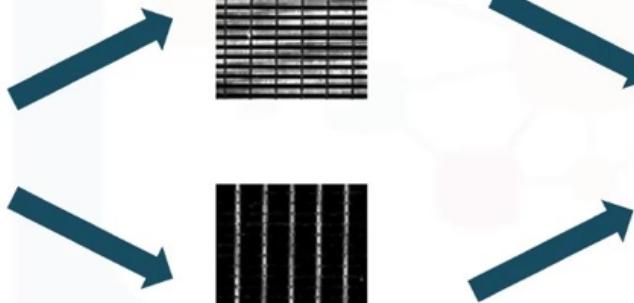
Primitive Features



Object Parts



Object



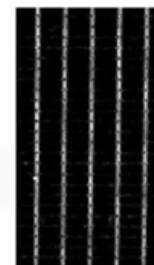
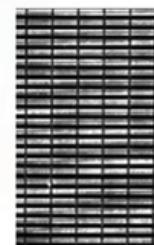
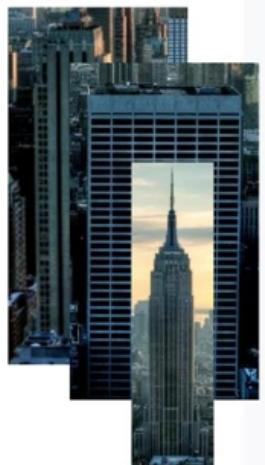
# A visual explanation

Input image

Primitive Features

Object Parts

Object



Building

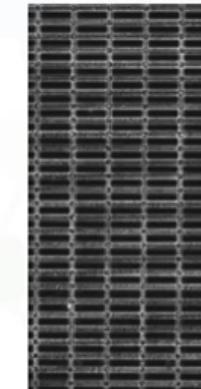
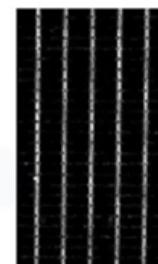
# A visual explanation

Input image

Primitive Features

Object Parts

Object





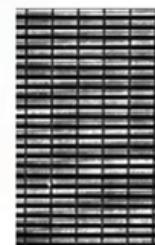
# A visual explanation

Input image

Primitive Features

Object Parts

Object



Building



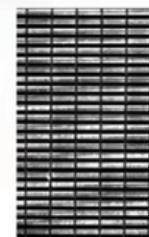
# A visual explanation

Input image

Primitive Features

Object Parts

Object



Building



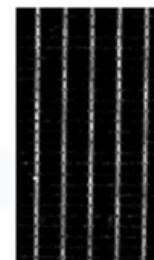
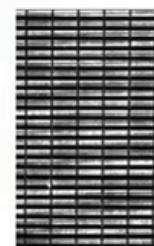
# A visual explanation

Input image

Primitive Features

Object Parts

Object



Building

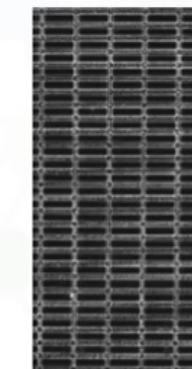
# A visual explanation

Input image

Primitive Features

Object Parts

Object



Building

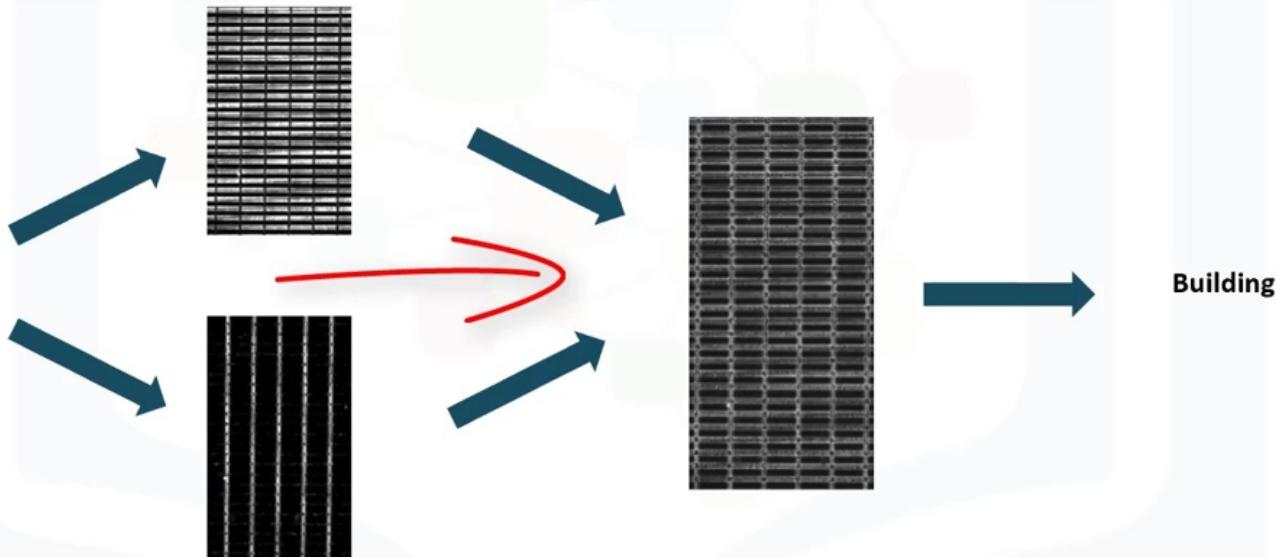
# A visual explanation

Input image

Primitive Features

Object Parts

Object





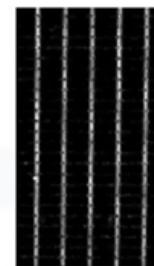
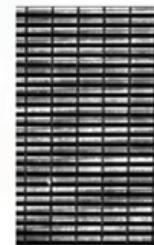
# A visual explanation

Input image

Primitive Features

Object Parts

Object



Building



Bagian 2



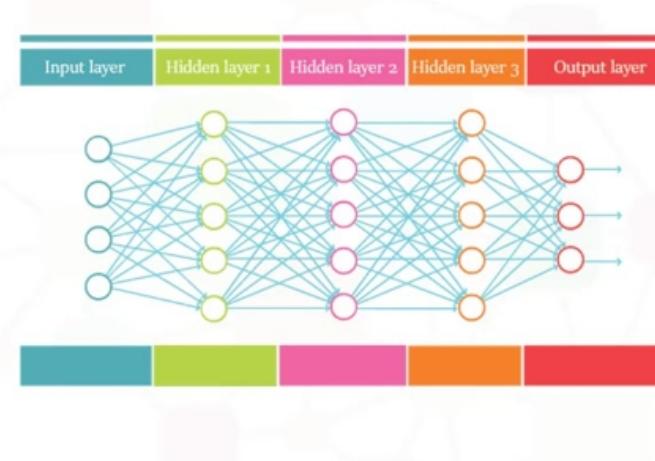
# CNN for Classification



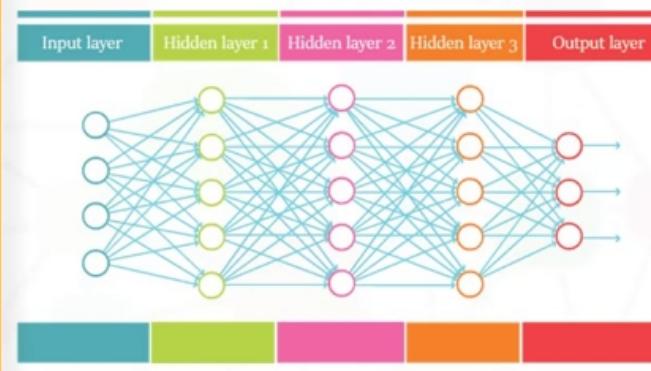
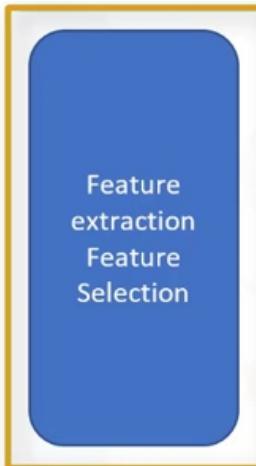
# Shallow Neural Networks, why not?



Feature extraction  
Feature Selection



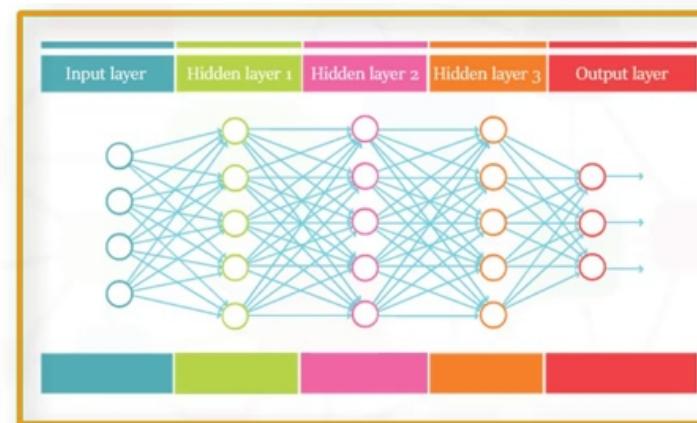
# Shallow Neural Networks, why not?



# Shallow Neural Networks, why not?



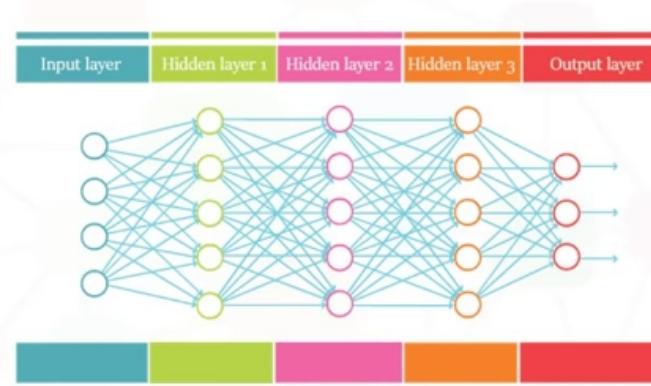
Feature extraction  
Feature Selection



# Shallow Neural Networks, why not?



Feature extraction  
Feature Selection



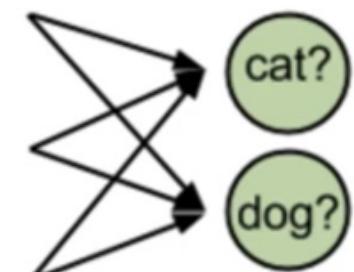
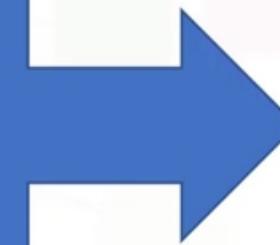
- The process of selecting the best features is hard
- Extending the features to other types of images is not possible



# Convolutional Neural Networks (CNNs)



Automatic  
Feature extractor  
Feature selector

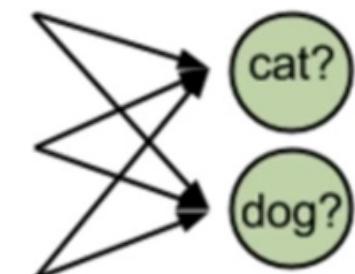
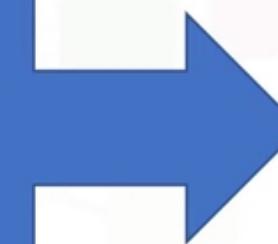




# Convolutional Neural Networks (CNNs)



Automatic  
Feature extractor  
Feature selector

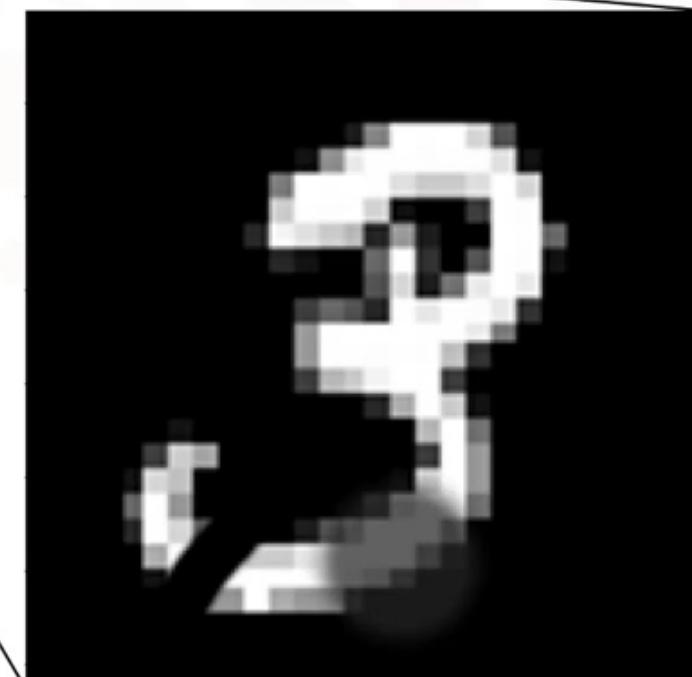




# Datasets

- **MNIST Dataset:** database of handwritten digits

0	4	1	9	2	1	3	1	4	3
5	3	6	1	7	2	8	6	9	4
0	9	1	1	2	4	3	2	7	3
8	6	9	0	5	6	0	7	6	1
8	7	9	3	9	8	5	9	3	3
0	7	4	9	8	0	9	4	1	4
4	6	0	4	5	6	1	0	0	1
7	1	6	3	0	2	1	1	7	9
0	2	6	7	8	3	9	0	4	6
7	4	6	8	0	7	8	3	1	5

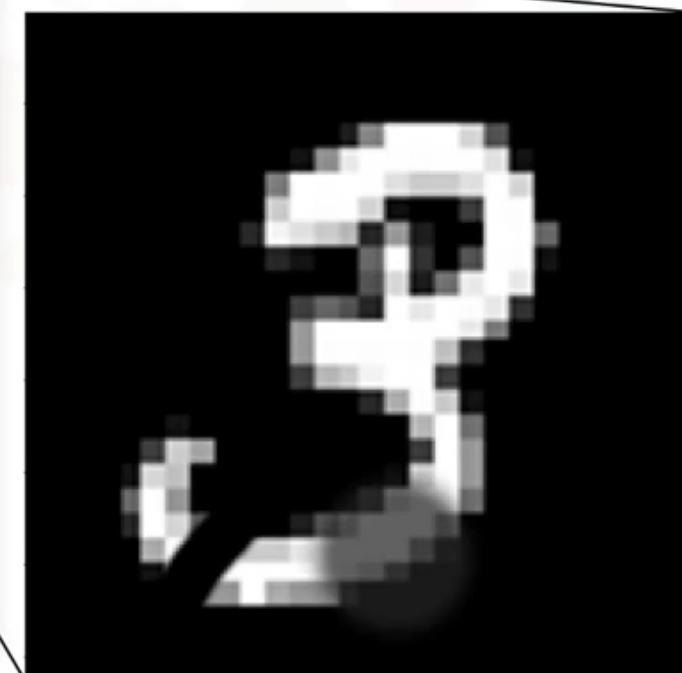




# Datasets

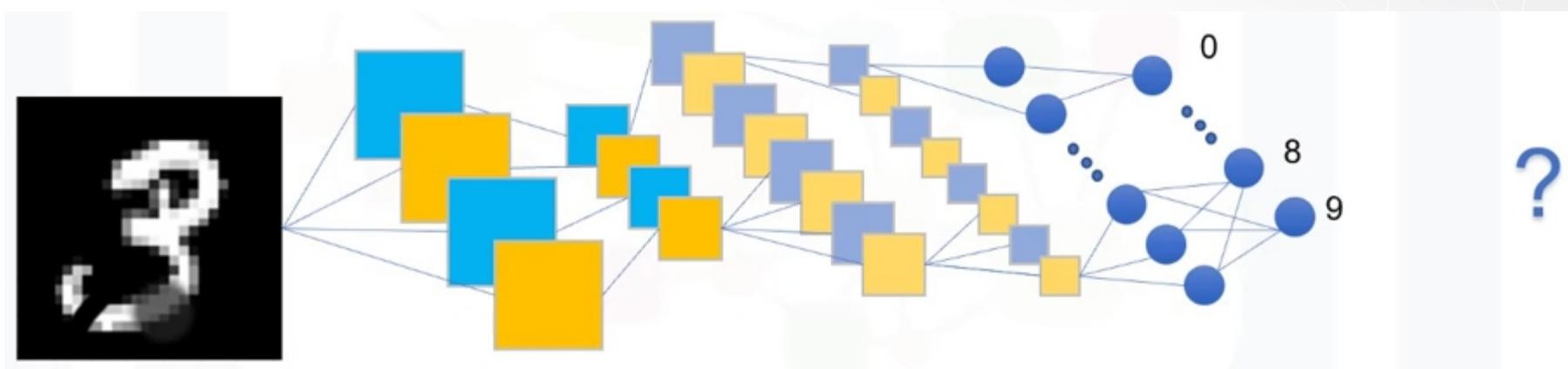
- **MNIST Dataset:** database of handwritten digits

0	4	1	9	2	1	3	1	4	3
5	3	6	1	7	2	8	6	9	4
0	9	1	1	2	4	3	2	7	3
8	6	9	0	5	6	0	7	6	1
8	7	9	3	9	8	5	9	3	3
0	7	4	9	8	0	9	4	1	4
4	6	0	4	5	6	1	0	0	1
7	1	6	3	0	2	1	1	1	9
0	2	6	7	8	3	9	0	4	6
7	4	6	8	0	7	8	3	1	5



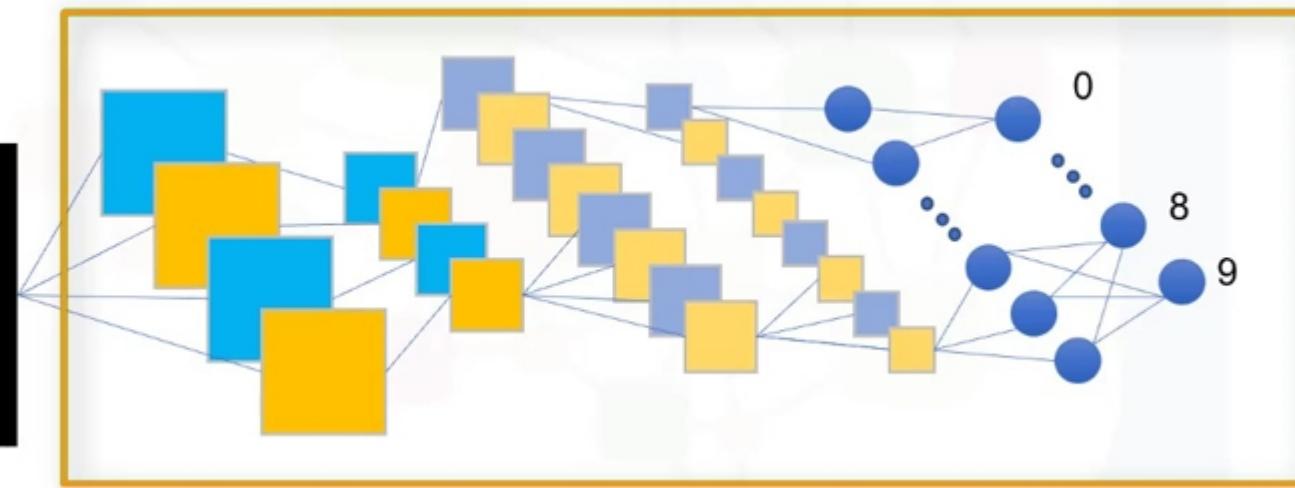
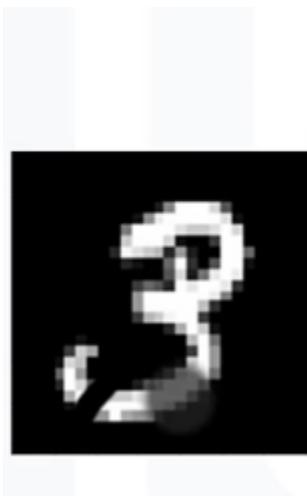


# Digit Recognition





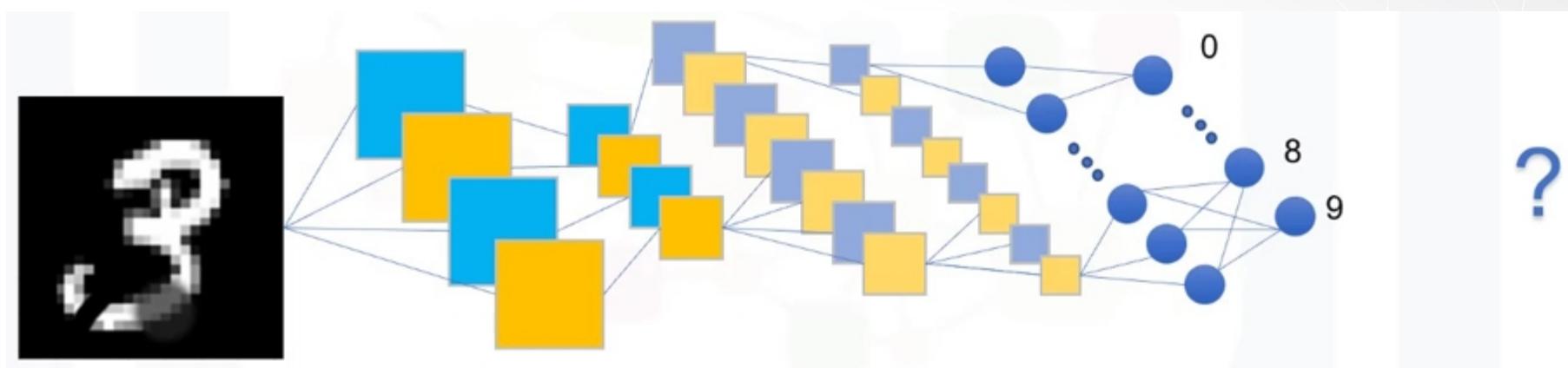
# Digit Recognition



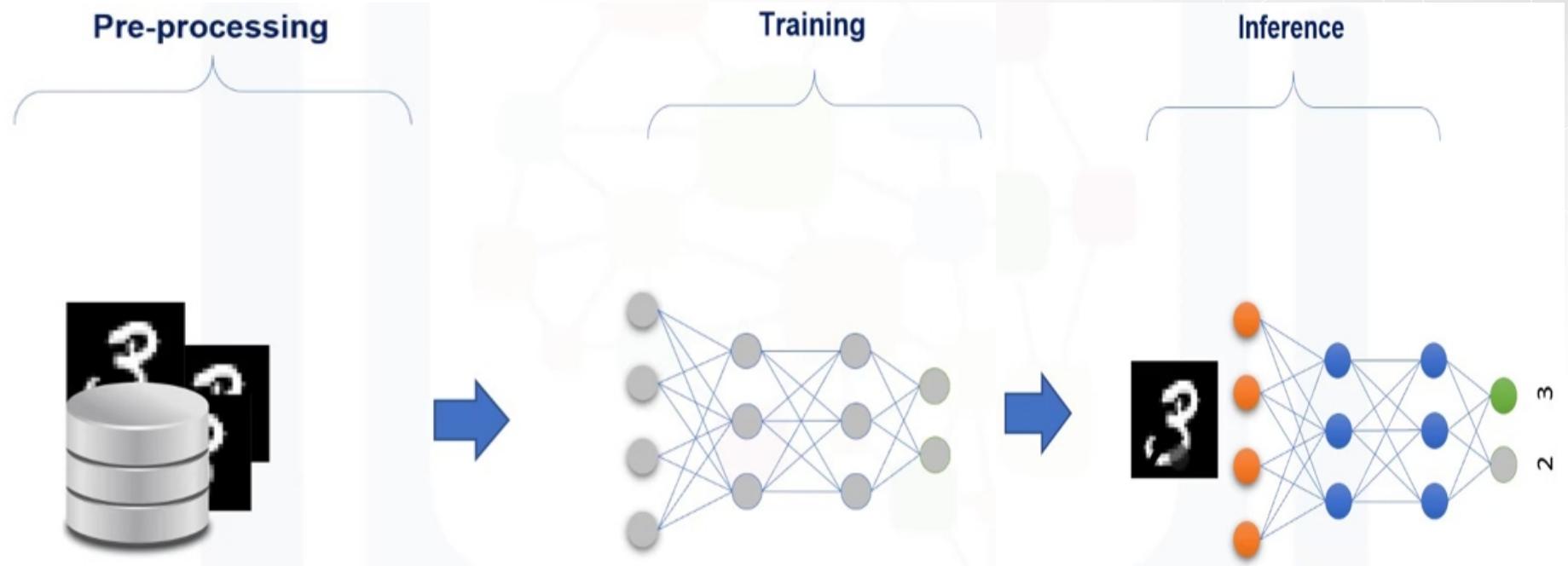
?



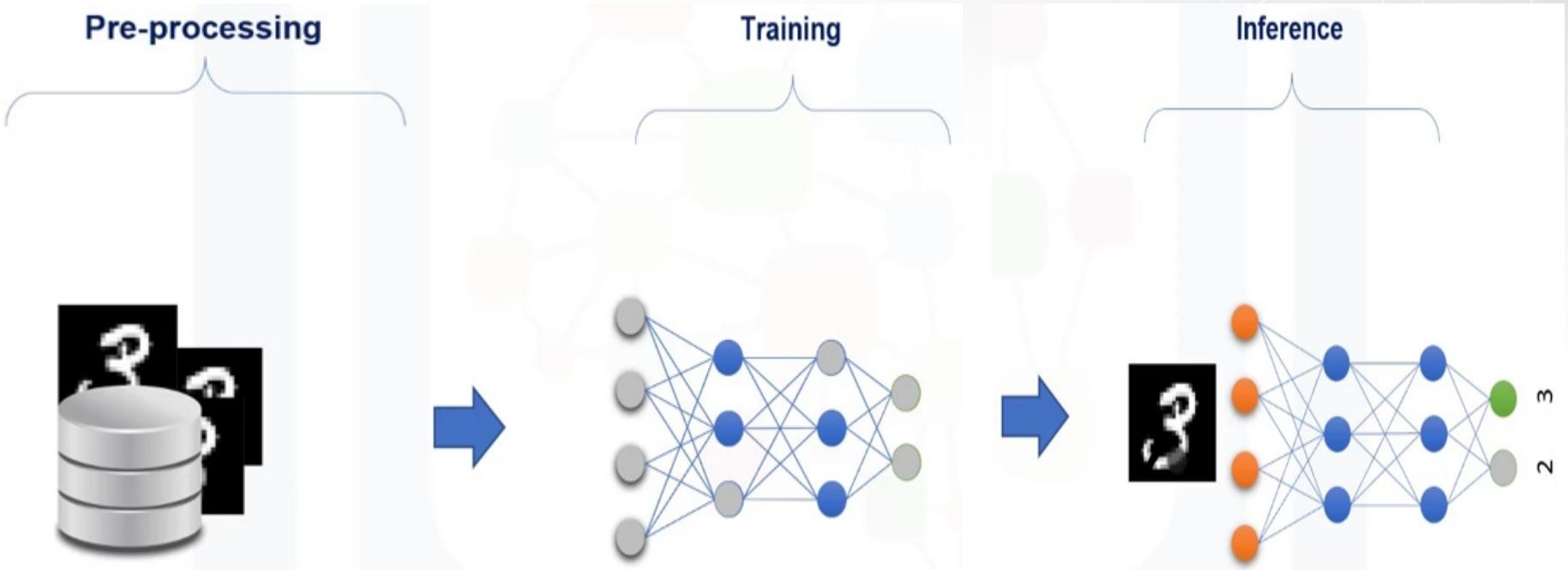
# Digit Recognition



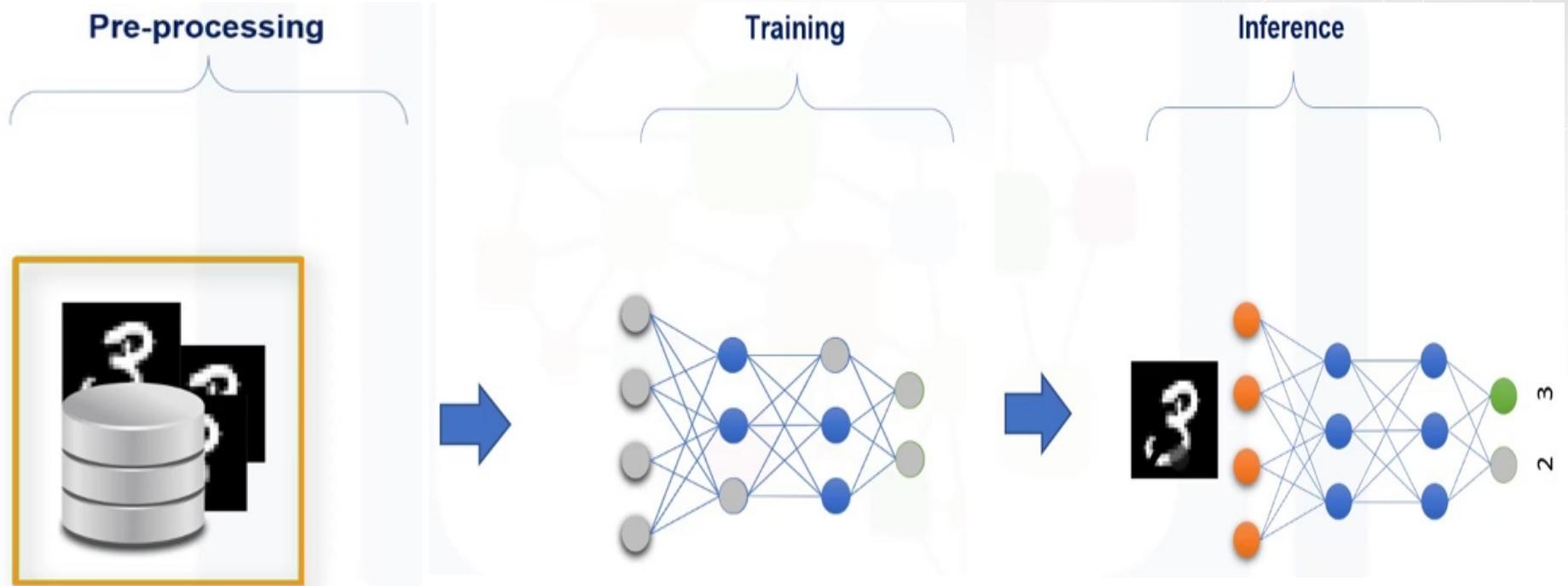
# Training and Inference



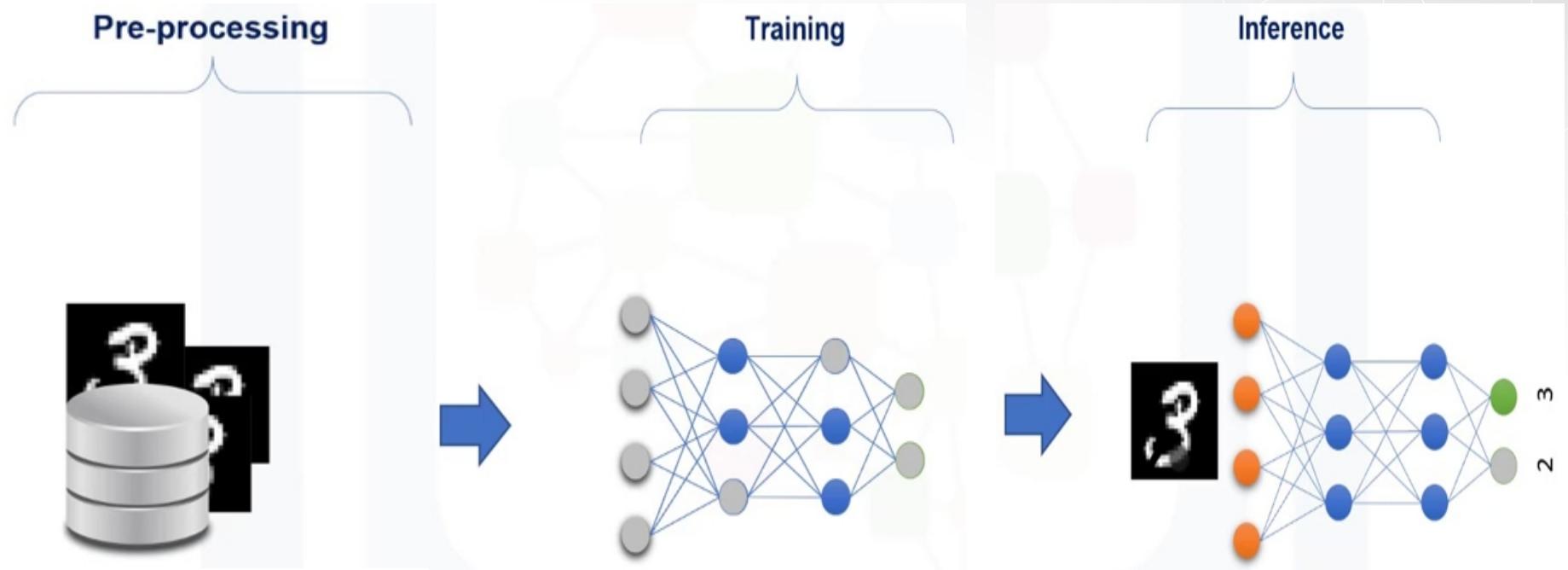
# Training and Inference



# Training and Inference

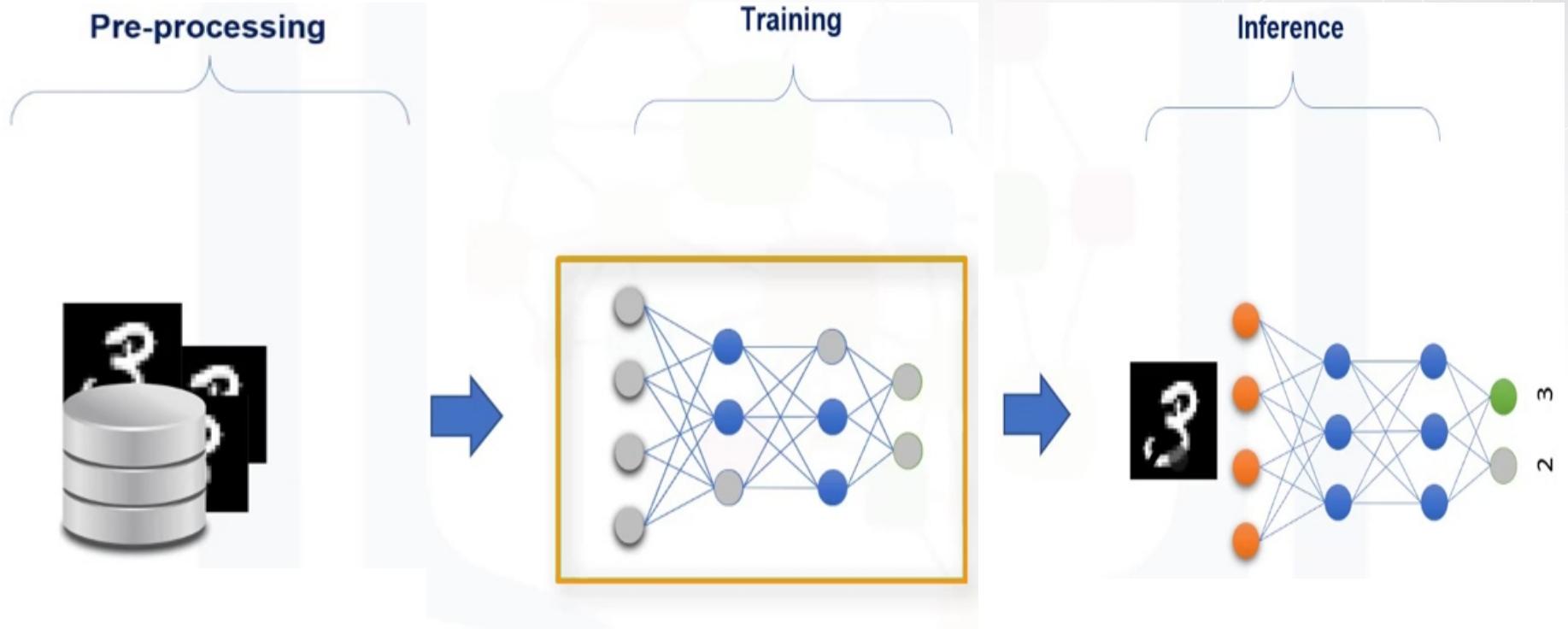


# Training and Inference



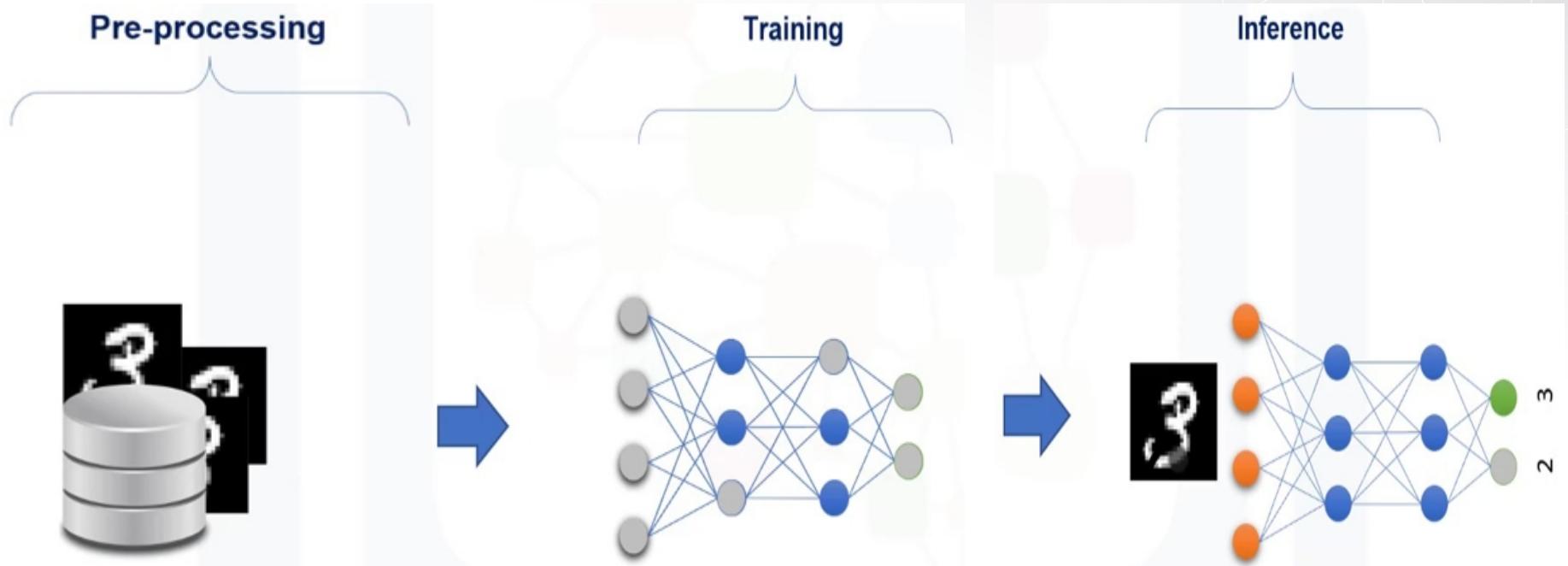


# Training and Inference



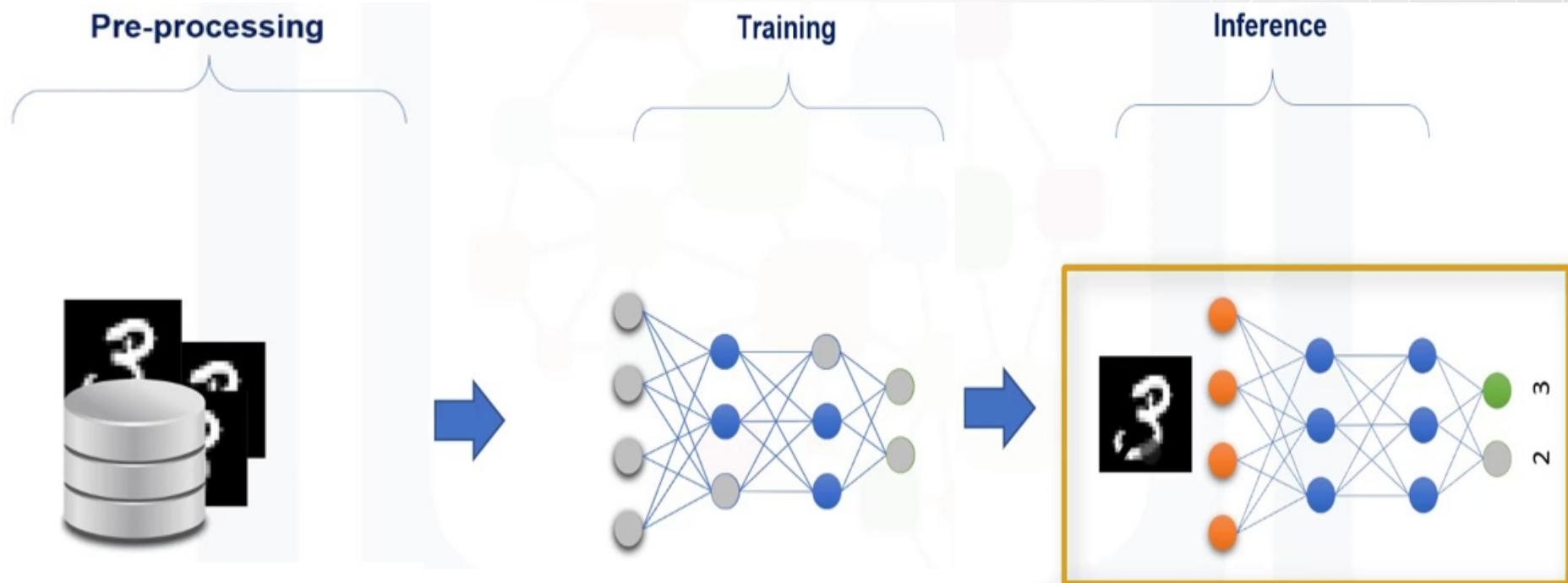


# Training and Inference

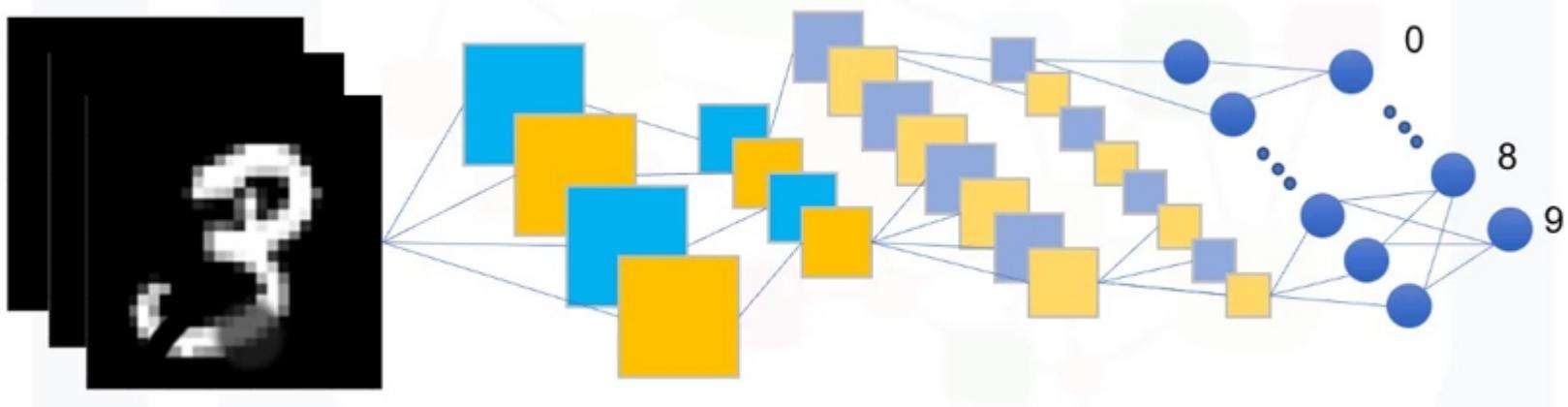




# Training and Inference

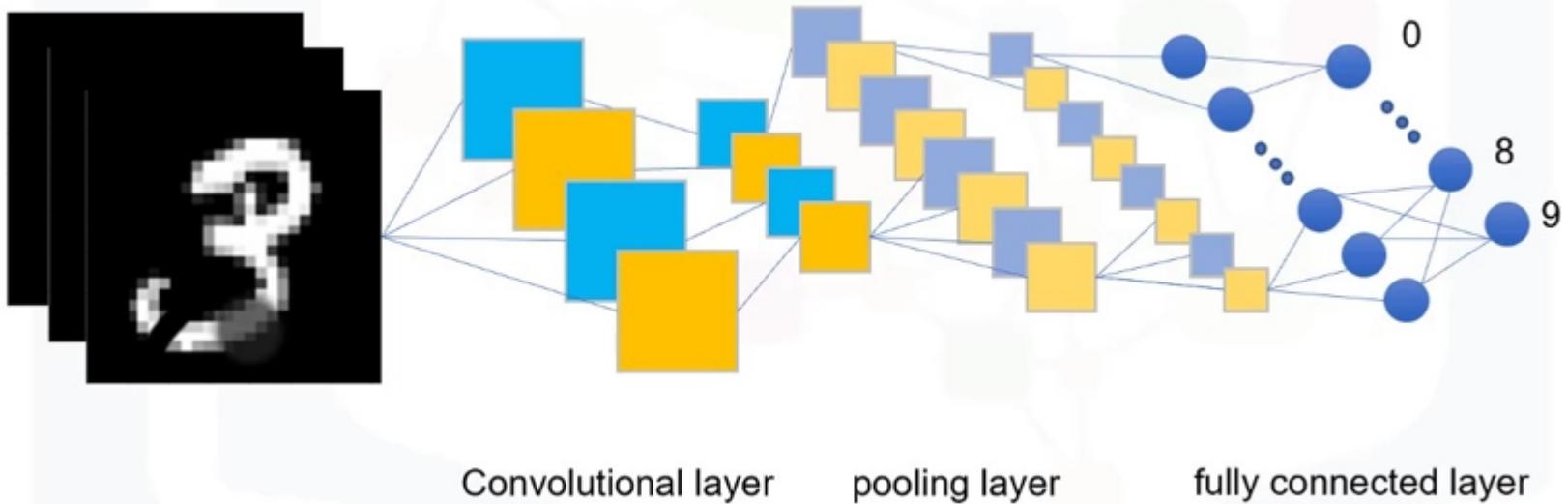


# Digit Recognition

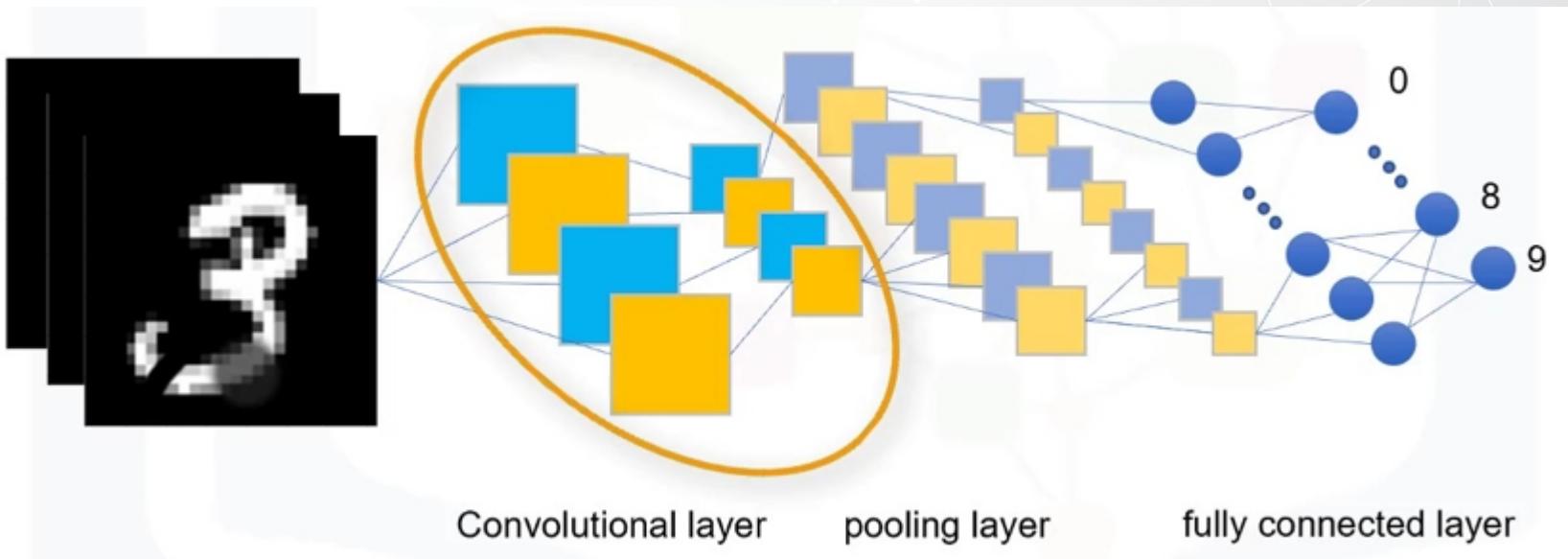




# Digit Recognition

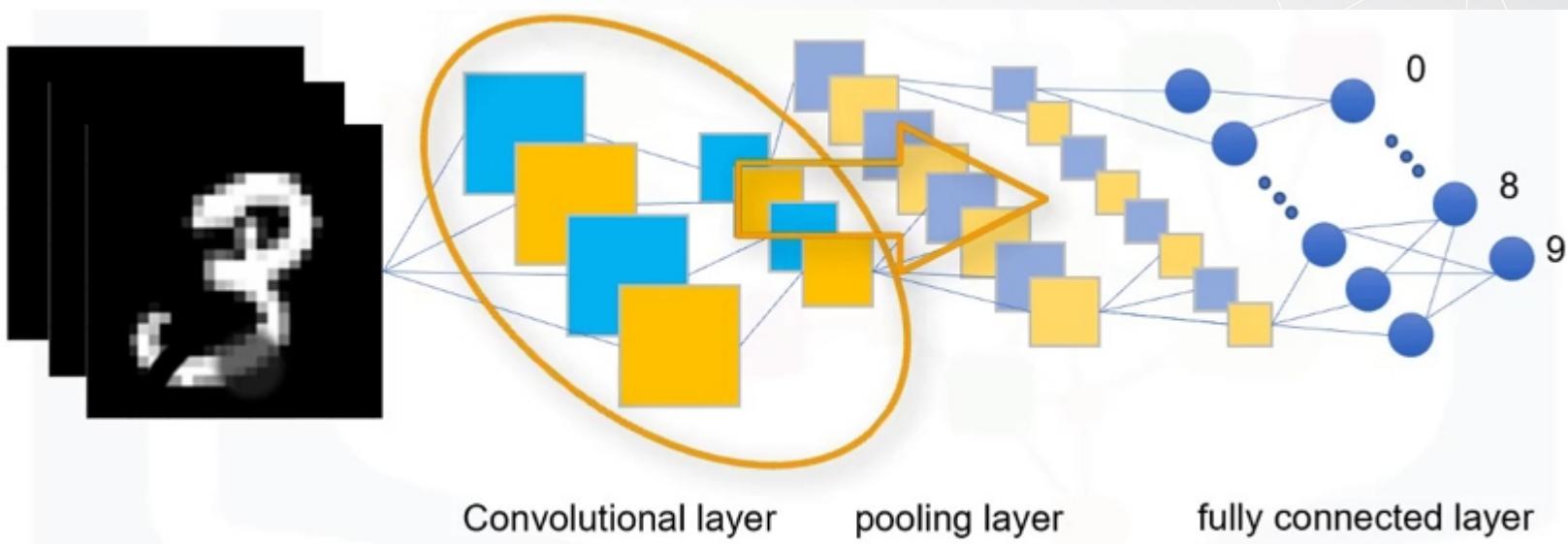


# Digit Recognition



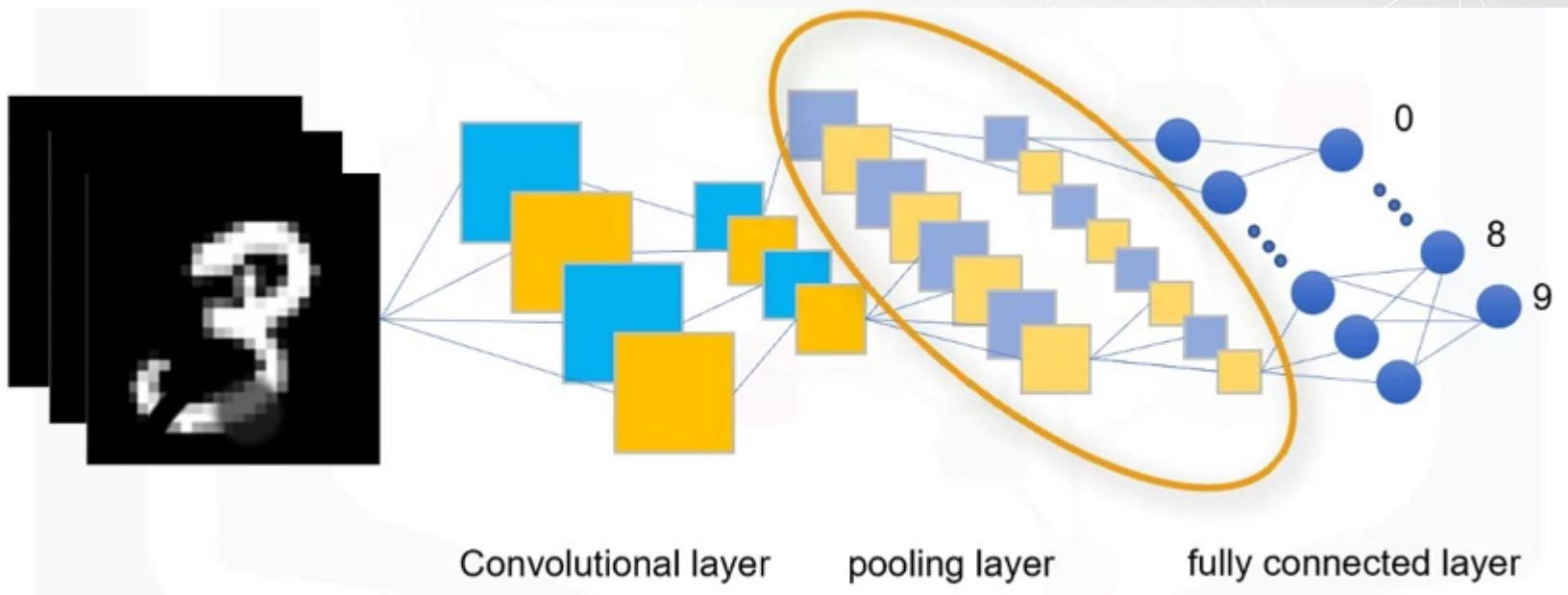


# Digit Recognition



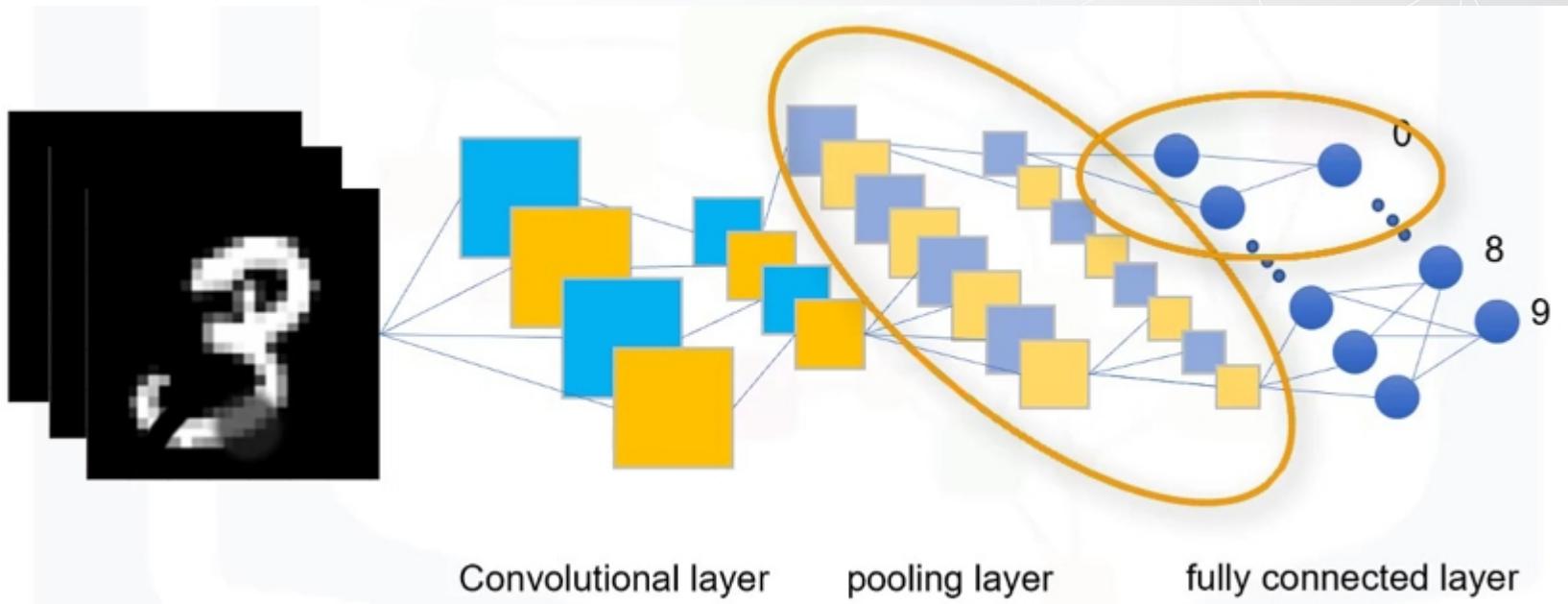


# Digit Recognition



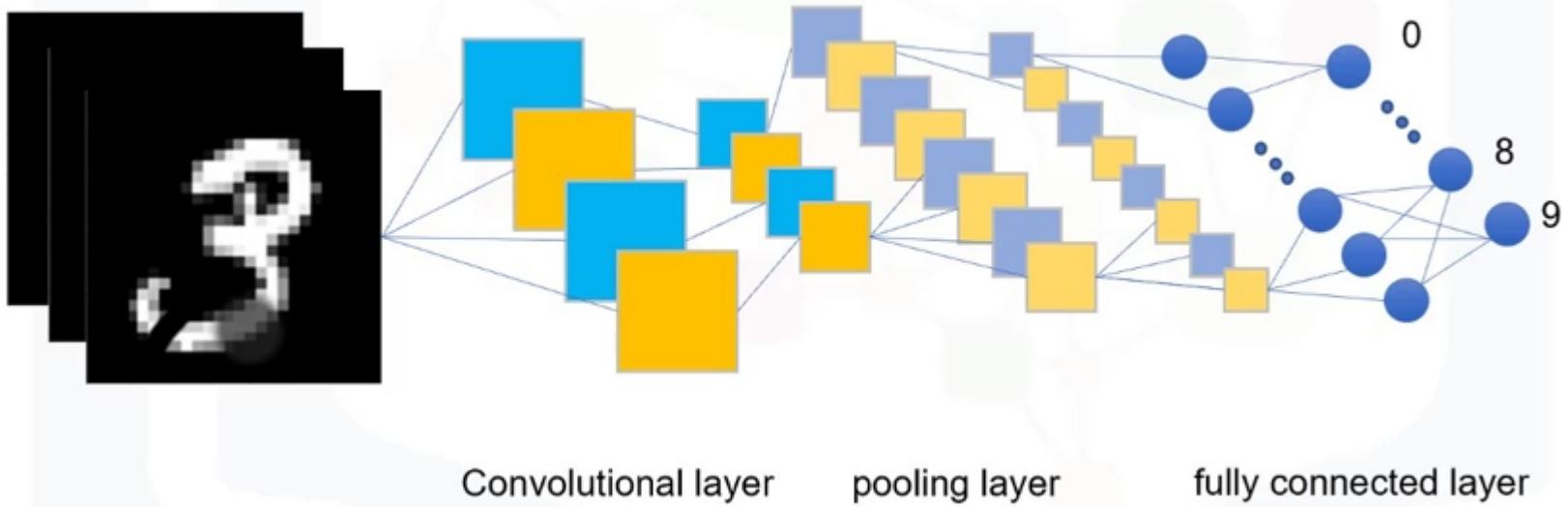


# Digit Recognition





# Digit Recognition



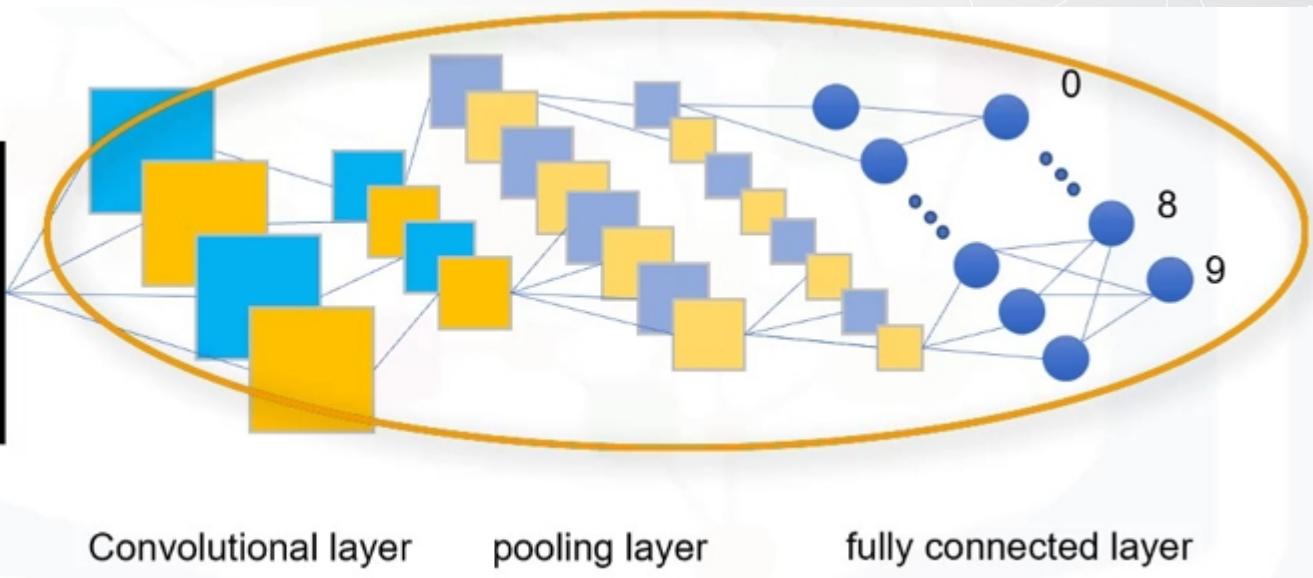
Convolutional layer

pooling layer

fully connected layer



# Digit Recognition



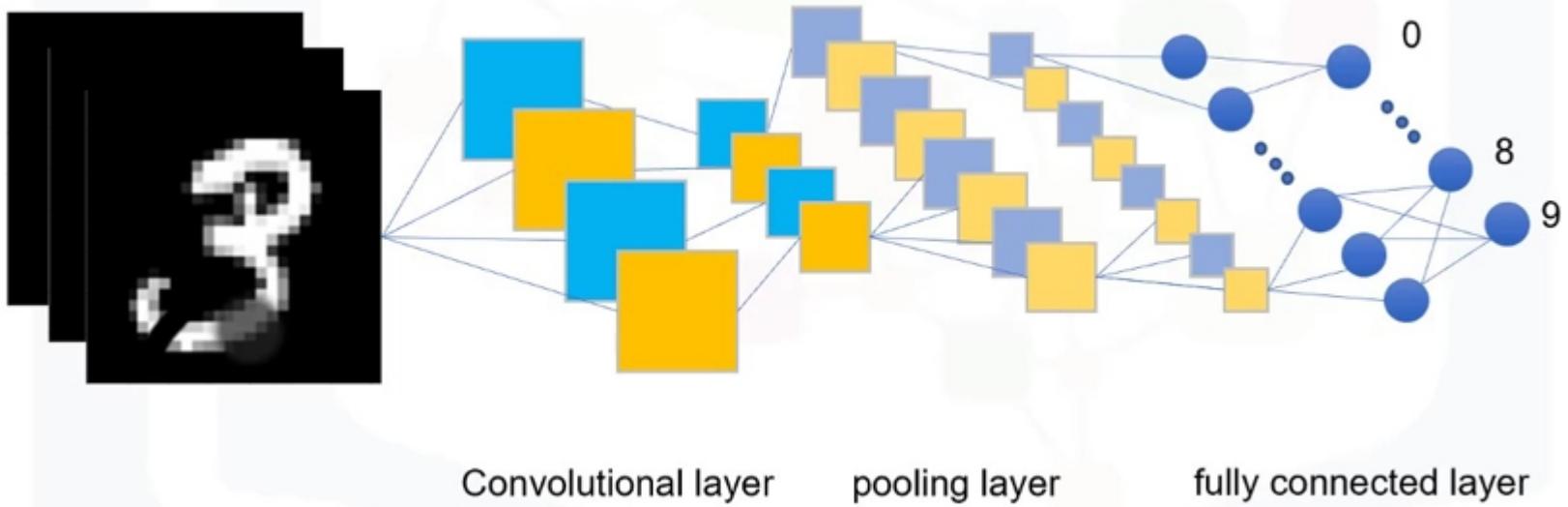
Convolutional layer

pooling layer

fully connected layer



# Digit Recognition



Convolutional layer

pooling layer

fully connected layer



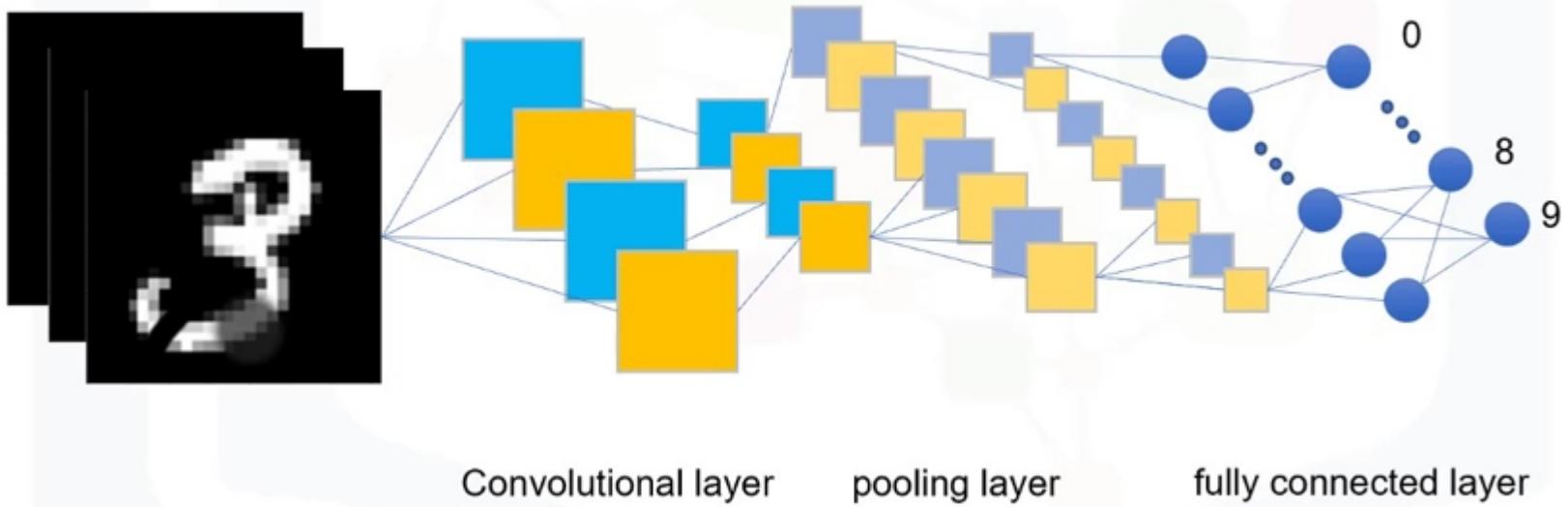
Bagian 3



# Convolutional Neural Networks



# Digit Recognition



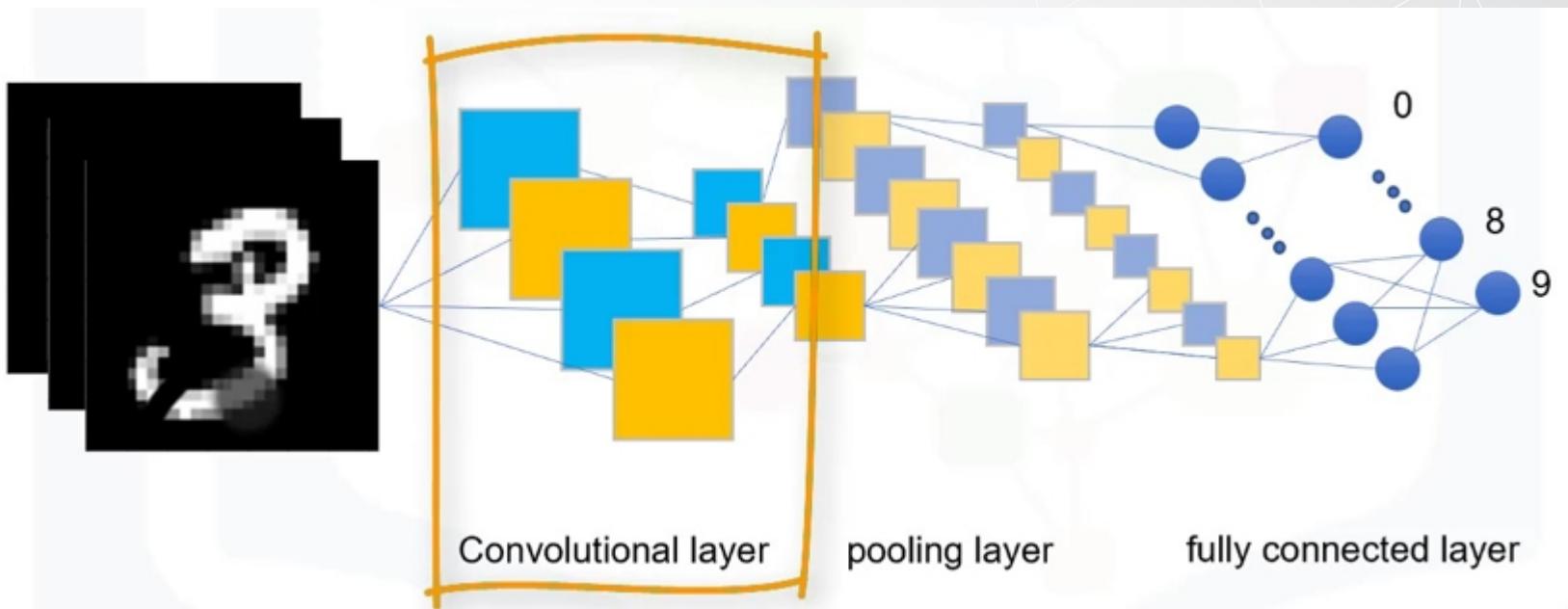
Convolutional layer

pooling layer

fully connected layer

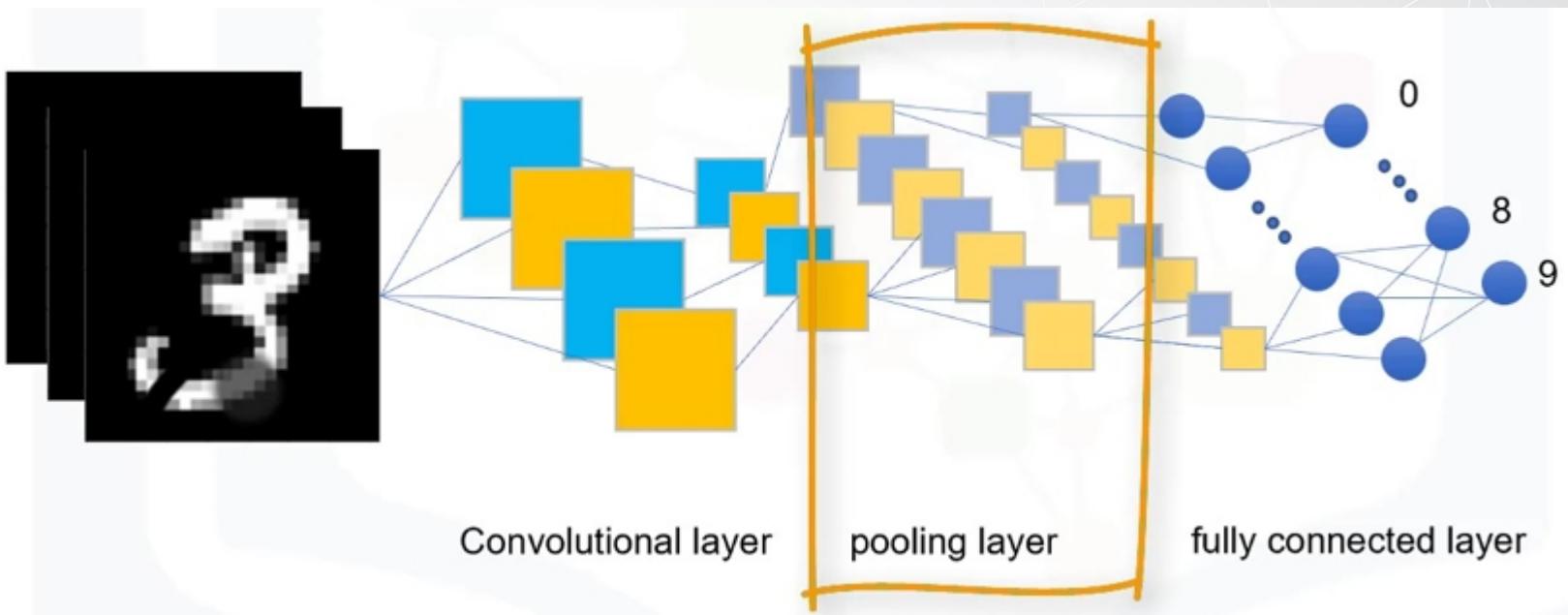


# Digit Recognition



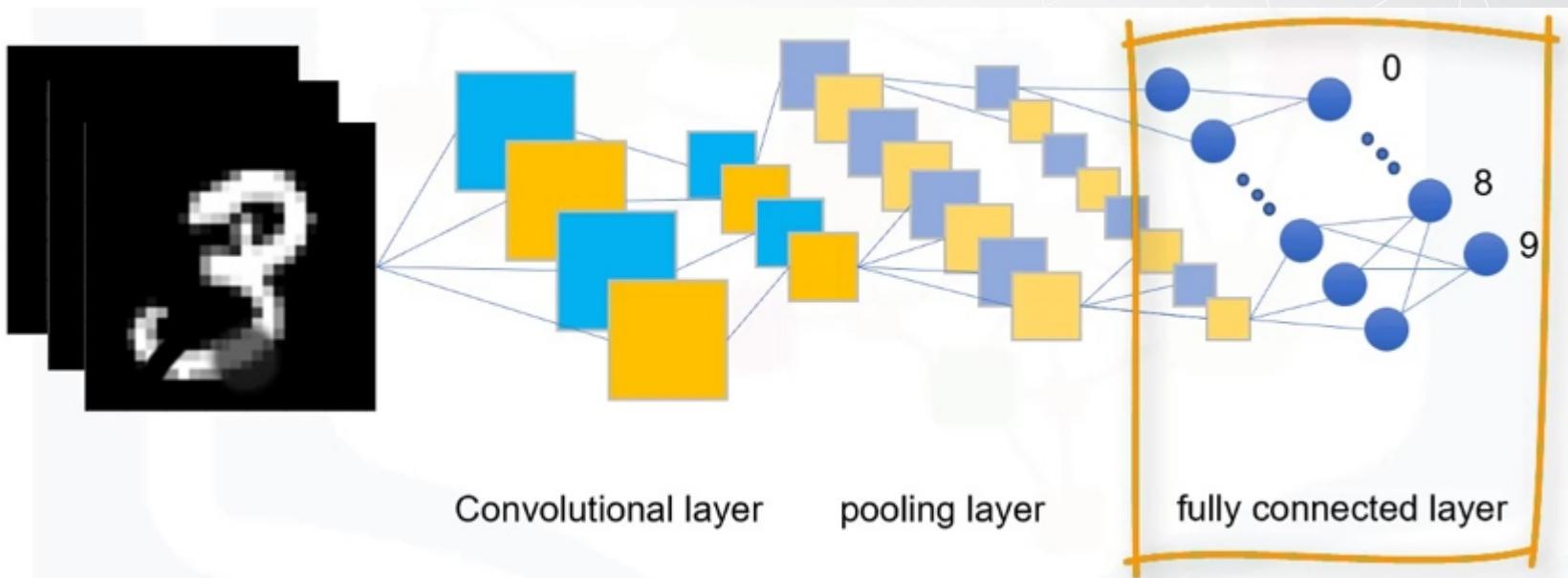


# Digit Recognition



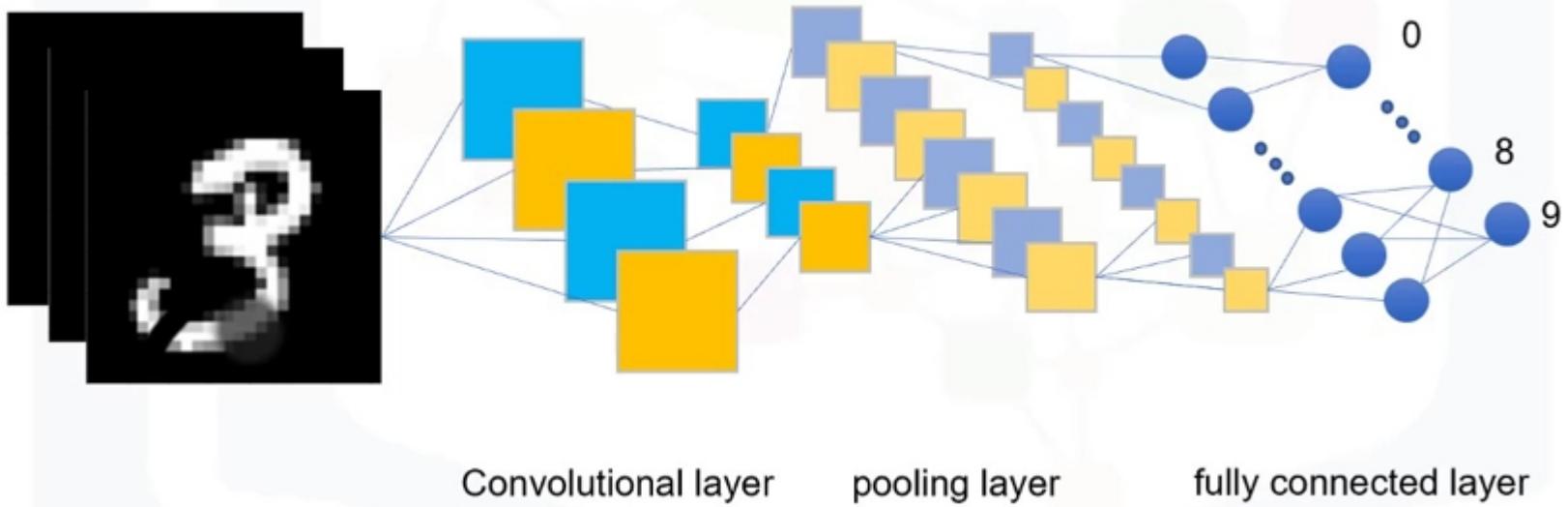


# Digit Recognition





# Digit Recognition



Convolutional layer

pooling layer

fully connected layer



# Understanding Convolution





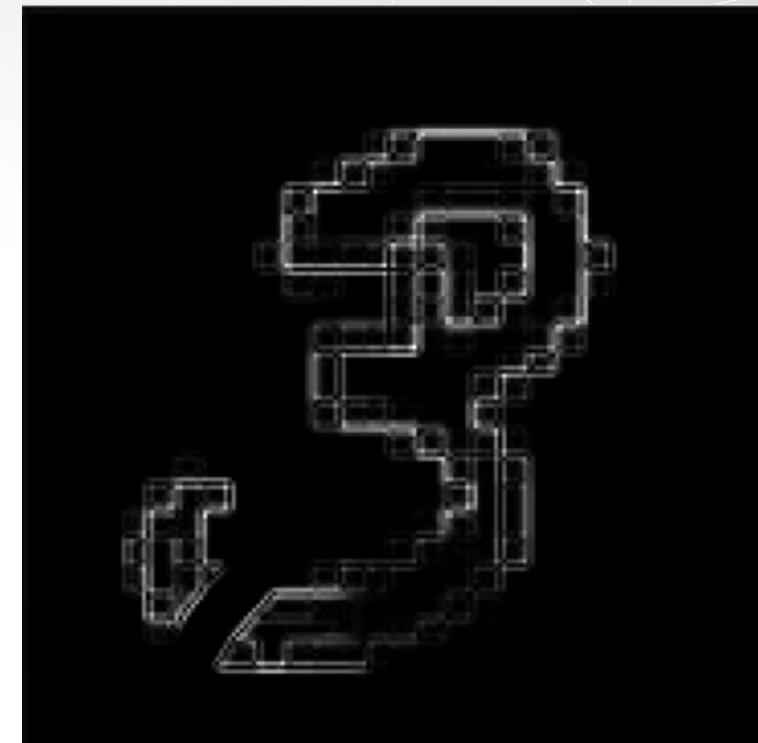
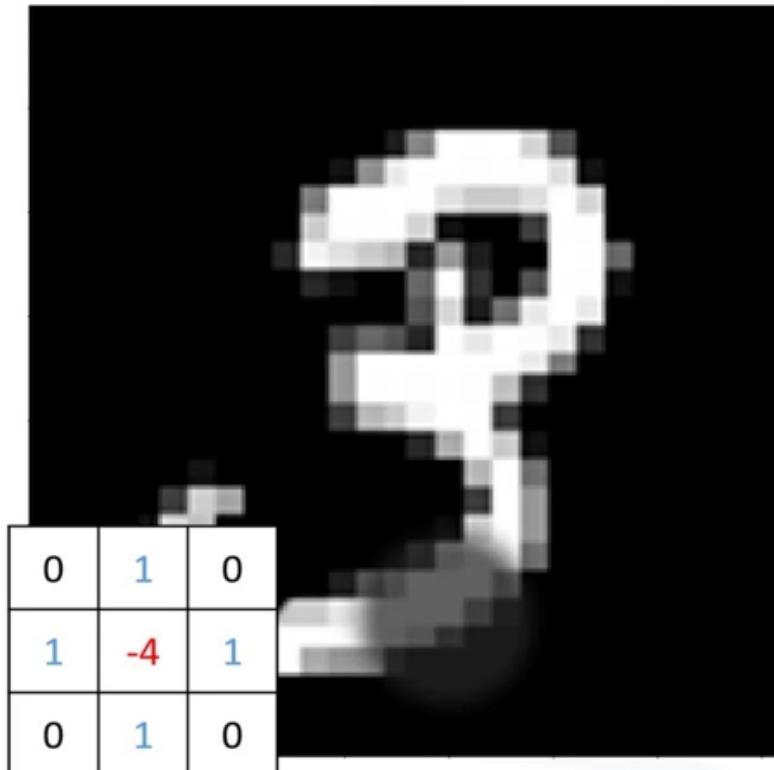
# Understanding Convolution



0	1	0
1	-4	1
0	1	0

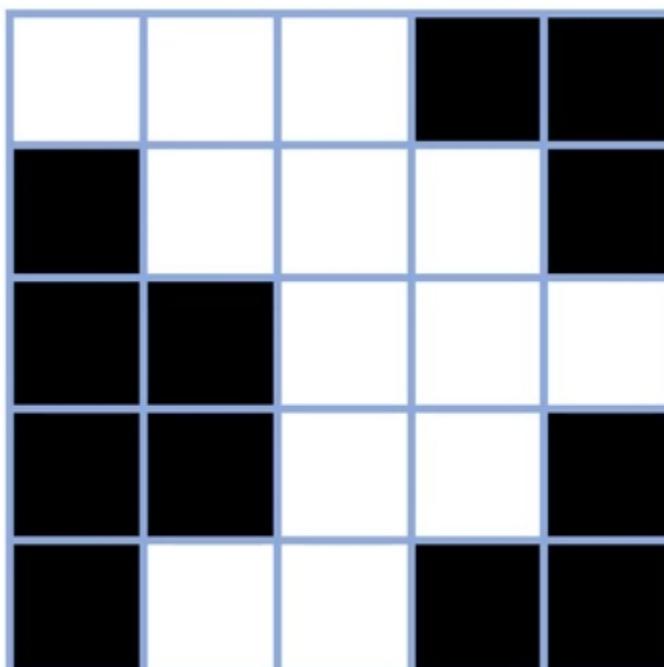


# Understanding Convolution





# Understanding Convolution



Image

# Understanding Convolution

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image

# Understanding Convolution

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image



# Understanding Convolution

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image



# Understanding Convolution

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image

0	0	0
0	-1	1
0	0	0



# Understanding Convolution

1	0	1	0	1	0	0	0
0	0	1	1	1	1	1	0
0	0	-1	1	1	1	1	1
0	0	0	1	0	1	1	0
0	0	1	1	1	0	0	0

Image



# Understanding Convolution

$$0*1+0*1+0*1 + 0*0+ -1*1+1*1+0*0+0*0+0*1 = 0$$

1	0	1	0	1	0	0	0
0	0	1	-1	1	1	1	0
0	0	0	1	0	1	1	1
0	0	1	1	1	0	0	0
0	1	1	0	0	0	0	0

0							



# Understanding Convolution

$$0*1+0*1+0*0 + 0*1+ -1*1+1*1+0*0+0*1+0*1 = 0$$



1	1	0	1	0	0	0
0	1	1	1	1	0	0
0	0	1	1	1	0	1
0	0	1	1	1	0	0
0	1	1	0	0	0	0

0	0					

# Understanding Convolution

## 28x28 Matrix

## image

```
tf.nn.conv2d(image, W_kernel)
```

# Understanding Convolution

## 28x28 Matrix

## 5x5 Matrix

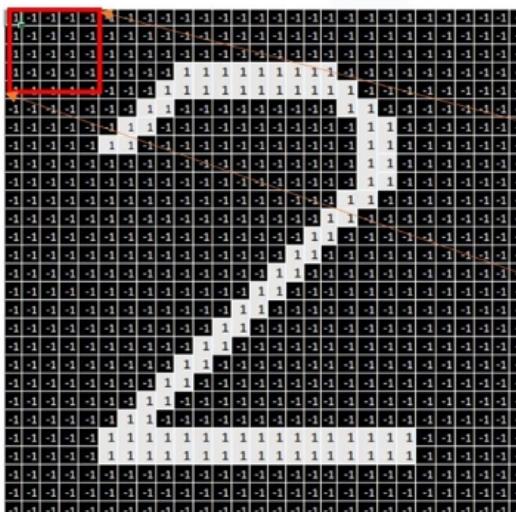
1	-1	-1	-1
-1	1	-1	-1
-1	-1	1	-1
-1	-1	-1	1
-1	-1	-1	-1

W\_kerne

```
tf.nn.conv2d(image, W_kernel)
```

# Understanding Convolution

## 28x28 Matrix



## image

## 5x5 Matrix



## W\_kerne

```
tf.nn.conv2d(image, W_kernel)
```

# Understanding Convolution

## 28x28 Matrix

5x5 Matrix



W\_kerne

## 28x28 Matrix

```
tf.nn.conv2d(image, W_kernel)
```

# Understanding Convolution

## 28x28 Matrix

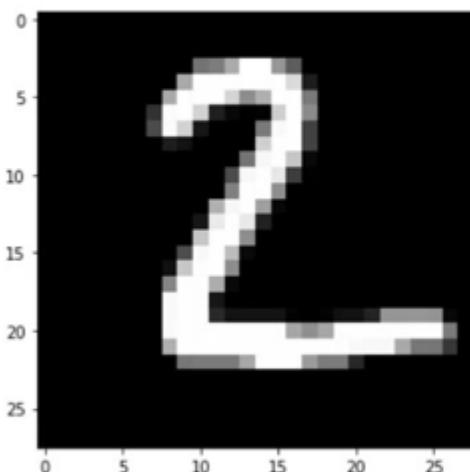
[5x5]x8 Matrix



[28x28]x8 Matrix  
Feature map



# Convolution Layer



$\times$

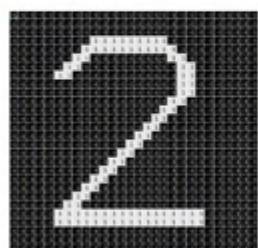


$=$



# CNN Architecture

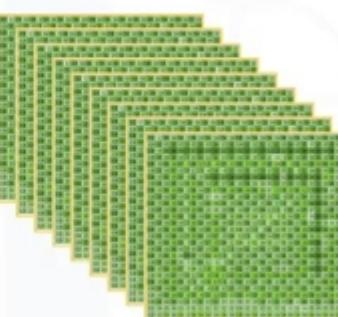
Convolution



28x28 Image



[5x5]x8 kernel [28x28]x8 Image



# CNN Architecture

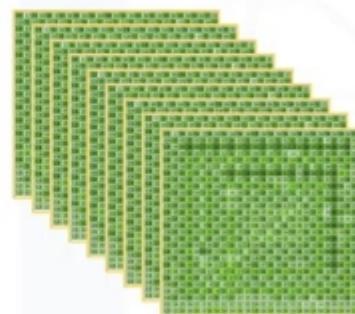
Convolution



28x28 Image

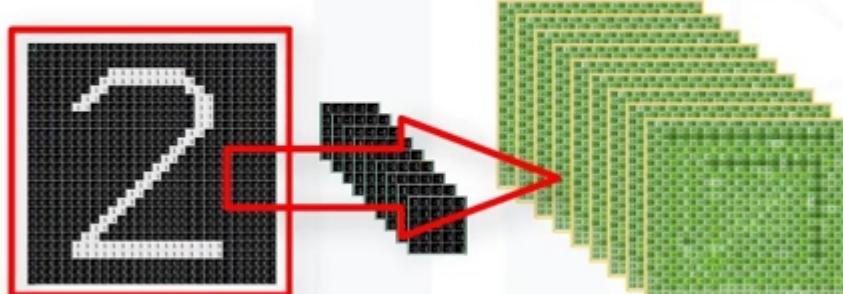


[5x5]x8 kernel [28x28]x8 Image



# CNN Architecture

Convolution

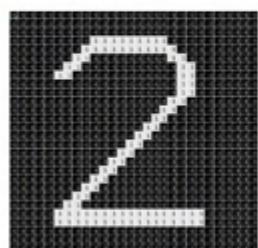


28x28 Image

[5x5]x8 kernel [28x28]x8 Image

# CNN Architecture

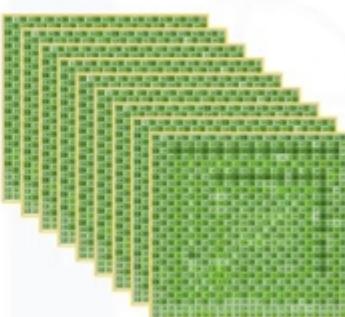
Convolution



28x28 Image

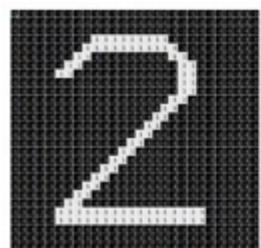


[5x5]x8 kernel [28x28]x8 Image



# CNN Architecture

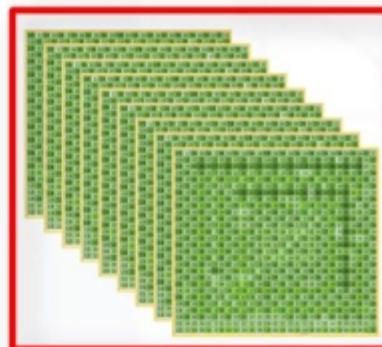
Convolution



28x28 Image



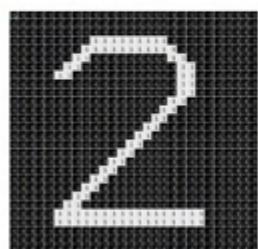
[5x5]x8 kernel



[28x28]x8 Image

# CNN Architecture

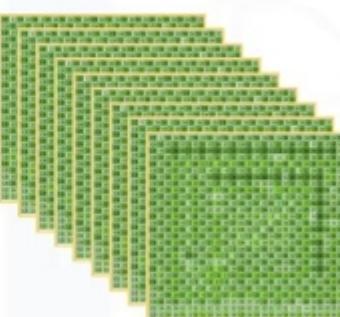
Convolution



28x28 Image

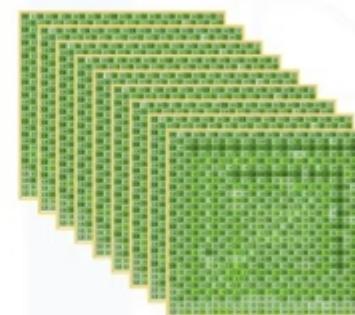
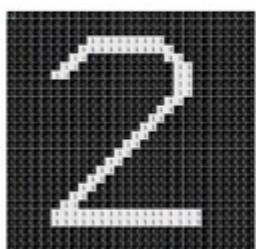


[5x5]x8 kernel [28x28]x8 Image



# CNN Architecture

Convolution



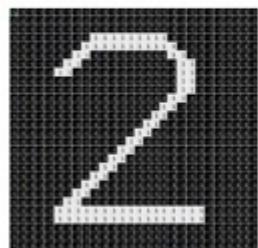
28x28 Image

[5x5]x8 kernel

[28x28]x8 Image

# CNN Architecture

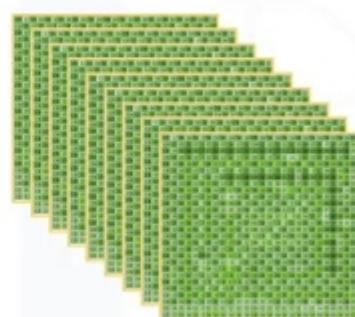
Convolution



28x28 Image

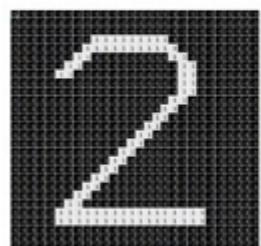


[5x5]x8 kernel [28x28]x8 Image



# CNN Architecture

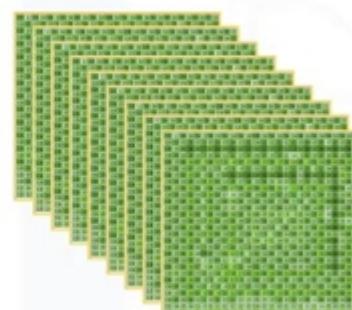
Convolution



28x28 Image



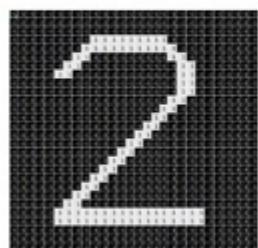
[5x5]x8 kernel



[28x28]x8 Image

# CNN Architecture

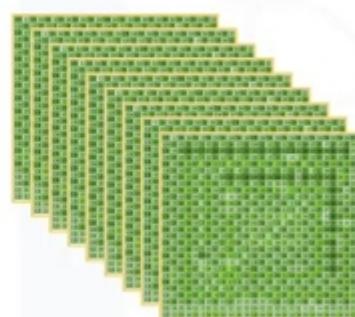
Convolution



28x28 Image



[5x5]x8 kernel [28x28]x8 Image



# ReLU

0.2	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
0.3	-0.2	0.5	0.5	0.5	0.5	0.6	0.5	0.4	0.3	0.2	0.2	0.2	0.2		
0.5	0.6	-0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1	0.1		
0.5	0.6	0.6	0.6	0.6	0.7	0.5	0.4	0.2	0.2	0.1	0.1	0.1	0.1		
0.5	0.6	0.6	0.7	0.5	0.4	0.2	0	-0	0.1	0.1	0.1	0.1	0.1		
0.5	0.6	0.7	0.5	0.4	0.2	0	-0	-0	0.1	0.1	0.1	0.1	0.1		
0.5	0.6	0.5	0.5	0.4	0.2	0.1	0.1	0.2	0.3	0.4	0.4	0.4	0.4		
0.5	0.6	0.5	0.4	0.4	0.3	0.3	0.4	0.5	0.7	0.6	0.6	0.6	0.6		
0.5	0.6	0.5	0.4	0.3	0.4	0.4	0.5	0.7	0.6	0.6	0.6	0.6	0.6		
0.5	0.6	0.5	0.4	0.4	0.4	0.6	0.7	0.6	0.6	0.6	0.6	0.6	0.6		
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6		
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6

convolved1

0.2	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
0.3	0	0.5	0.5	0.5	0.5	0.6	0.5	0.4	0.3	0.2	0.2	0.2	0.2	0.2	0.2
0.5	0.6	0	0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.2	0.2	0.1	0.1	0.1
0.5	0.6	0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.7	0.5	0.4	0.2	0	-0	0.1	0.1	0.1	0.1	0.1		
0.5	0.6	0.7	0.5	0.4	0.2	0	-0	-0	0.1	0.1	0.1	0.1	0.1		
0.5	0.6	0.5	0.5	0.4	0.2	0.1	0.1	0.2	0.3	0.4	0.4	0.4	0.4		
0.5	0.6	0.5	0.4	0.4	0.3	0.3	0.4	0.5	0.7	0.6	0.6	0.6	0.6		
0.5	0.6	0.5	0.4	0.3	0.4	0.4	0.5	0.7	0.6	0.6	0.6	0.6	0.6		
0.5	0.6	0.5	0.4	0.3	0.4	0.4	0.5	0.7	0.6	0.6	0.6	0.6	0.6		
0.5	0.6	0.5	0.4	0.4	0.4	0.6	0.7	0.6	0.6	0.6	0.6	0.6	0.6		
0.5	0.6	0.5	0.4	0.4	0.4	0.6	0.7	0.6	0.6	0.6	0.6	0.6	0.6		
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6

# ReLU

0.2	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
0.3	-0.2	0.5	0.5	0.5	0.5	0.6	0.5	0.4	0.3	0.2	0.2	0.2	0.2	0.2	0.2	0.2
0.5	0.6	-0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.6	0.6	0.7	0.5	0.4	0.2	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.7	0.5	0.4	0.2	0	-0	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.7	0.5	0.4	0.2	0	-0	-0	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.5	0.5	0.4	0.2	0.1	0.1	0.2	0.3	0.4	0.4	0.4	0.4	0.4	0.4	0.4
0.5	0.6	0.5	0.4	0.4	0.3	0.3	0.4	0.5	0.7	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.5	0.4	0.3	0.4	0.4	0.5	0.7	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.5	0.4	0.4	0.4	0.6	0.7	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6

convolved1

0.2	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
0.3	0	0.5	0.5	0.5	0.5	0.6	0.5	0.4	0.3	0.2	0.2	0.2	0.2	0.2	0.2	0.2
0.5	0.6	0	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.7	0.5	0.4	0.2	0	-0	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.7	0.5	0.4	0.2	0	-0	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.7	0.5	0.4	0.2	0	-0	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2

tf.nn.relu(convolved1)



# ReLU

0.2	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
0.3	-0.2	0.5	0.5	0.5	0.5	0.6	0.5	0.4	0.3	0.2	0.2	0.2	0.2	0.2	0.2
0.5	0.6	-0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.6	0.7	0.5	0.4	0.2	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.7	0.5	0.4	0.2	0	-0	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.7	0.5	0.4	0.2	0	-0	-0	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.5	0.5	0.4	0.2	0.1	0.1	0.2	0.3	0.4	0.4	0.4	0.4	0.4	0.4
0.5	0.6	0.5	0.4	0.4	0.3	0.3	0.4	0.5	0.7	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.5	0.4	0.3	0.4	0.4	0.5	0.7	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.5	0.4	0.4	0.4	0.6	0.7	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.4

convolved1

0.2	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
0.3	0	0.5	0.5	0.5	0.5	0.5	0.6	0.5	0.4	0.3	0.2	0.2	0.2	0.2	0.2
0.5	0.6	0	0.6	0.6	0.6	0.7	0.5	0.5	0.5	0.5	0.5	0.4	0.2	0.1	0.1
0.5	0.6	0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.7	0.5	0.4	0.2	0.2	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.7	0.5	0.4	0.2	0	-0	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.7	0.5	0.4	0.2	0	-0	-0	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.6	0.5	0.4	0.2	0	-0	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.5	0.5	0.4	0.2	0.1	0.1	0.2	0.3	0.4	0.4	0.4	0.4	0.4	0.4
0.5	0.6	0.5	0.4	0.4	0.3	0.3	0.4	0.5	0.7	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.5	0.4	0.3	0.4	0.4	0.5	0.7	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.5	0.4	0.4	0.3	0.3	0.4	0.5	0.7	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.5	0.4	0.4	0.4	0.6	0.7	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.5	0.4	0.4	0.4	0.6	0.7	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.5
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.4

tf.nn.relu(convolved1)

# ReLU

0.2	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
0.3	-0.2	0.5	0.5	0.5	0.5	0.6	0.5	0.4	0.3	0.2	0.2	0.2	0.2	0.2	0.2	0.2
0.5	0.6	-0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.6	0.6	0.7	0.5	0.4	0.2	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.7	0.5	0.4	0.2	0	-0	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.7	0.5	0.4	0.2	0	-0	-0	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.5	0.5	0.4	0.2	0.1	0.1	0.2	0.3	0.4	0.4	0.4	0.4	0.4	0.4	0.4
0.5	0.6	0.5	0.4	0.4	0.3	0.3	0.4	0.5	0.7	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.5	0.4	0.3	0.4	0.4	0.5	0.7	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.5	0.4	0.4	0.4	0.6	0.7	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6

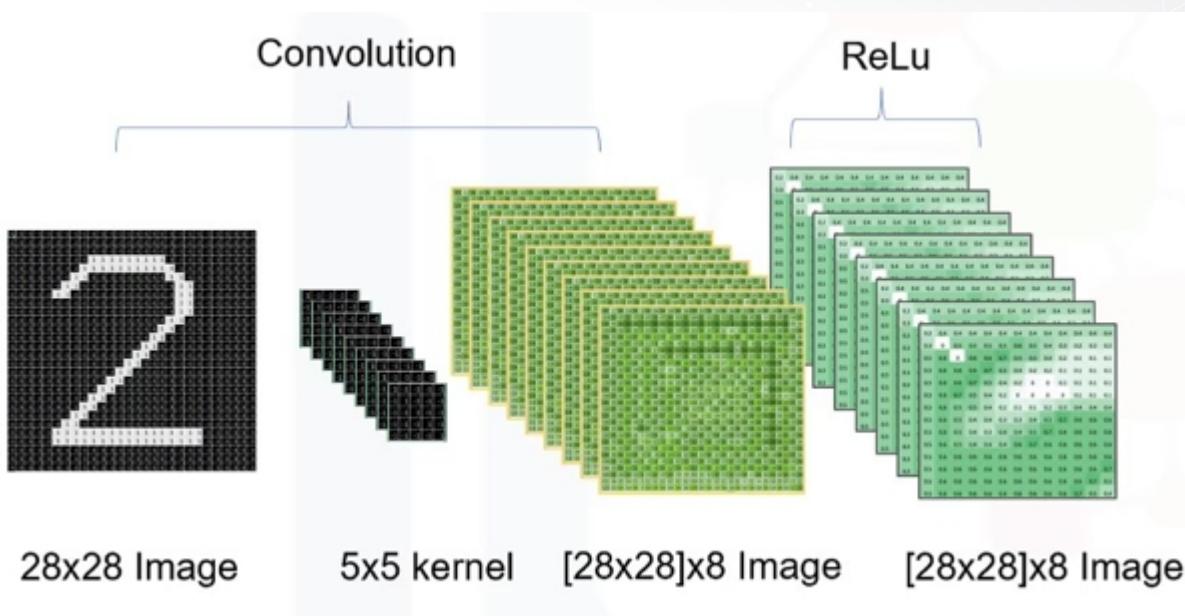
convolved1

0.2	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
0.3	0	0.5	0.5	0.5	0.5	0.6	0.5	0.4	0.3	0.2	0.2	0.2	0.2	0.2	0.2	0.2
0.5	0.6	0	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.7	0.5	0.4	0.2	0	-0	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.7	0.5	0.4	0.2	0	-0	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.7	0.5	0.4	0.2	0	-0	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2

tf.nn.relu(convolved1)



# CNN Architecture - ReLu



# Max Pooling

0.1	0.2	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
0.2	0.3	0.5	0.5	0.5	0.5	0.5	0.6	0.5	0.4	0.3	0.2	0.2	0.2	0.2	0.2	0.2
0.4	0.5	0.6	0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1	0.1	0.1	0.1
0.4	0.5	0.6	0.6	0.6	0.7	0.5	0.4	0.2	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.4	0.5	0.6	0.6	0.7	0.5	0.4	0.2	0	0	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.4	0.5	0.6	0.7	0.5	0.4	0.2	0	0	0	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.4	0.5	0.6	0.7	0.5	0.4	0.2	0	0	0	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.4	0.5	0.6	0.5	0.5	0.4	0.2	0.1	0.1	0.2	0.3	0.4	0.4	0.4	0.4	0.4	0.4
0.4	0.5	0.6	0.5	0.4	0.4	0.3	0.3	0.4	0.5	0.7	0.6	0.6	0.6	0.6	0.6	0.6
0.4	0.5	0.6	0.5	0.4	0.3	0.4	0.4	0.5	0.7	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.4	0.5	0.6	0.5	0.4	0.4	0.4	0.6	0.7	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.4	0.5	0.6	0.5	0.4	0.4	0.4	0.6	0.7	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.4	0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7
0.4	0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5
0.4	0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.5	0.4
0.4	0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5	0.4	0.2	0	0
0.4	0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5	0.4	0.2	0	0	0

conv1\_ReLu

0.3	0.5	0.5	0.6	0.5	0.4	0.4	0.4	0.4
0.5	0.6	0.7	0.7	0.5	0.2	0.1	0.1	0.1
0.5	0.7	0.7	0.4	0	0.1	0.1	0.1	0.1
0.5	0.6	0.5	0.3	0.5	0.7	0.6	0.6	0.6
0.5	0.6	0.4	0.6	0.7	0.6	0.6	0.6	0.7
0.5	0.6	0.6	0.6	0.6	0.6	0.7	0.7	0.7
0.5	0.6	0.6	0.6	0.6	0.7	0.7	0.7	0.4
0.5	0.6	0.6	0.6	0.7	0.7	0.4	0.2	0

```
tf.nn.max_pool(conv1_ReLu, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1])
```



# Max Pooling

0.1	0.2	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
0.2	0.3	0.5	0.5	0.5	0.5	0.5	0.6	0.5	0.4	0.3	0.2	0.2	0.2	0.2	0.2	0.2
0.4	0.5	0.6	0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1	0.1	0.1	0.1
0.4	0.5	0.6	0.6	0.6	0.7	0.5	0.4	0.2	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.4	0.5	0.6	0.6	0.7	0.5	0.4	0.2	0	0	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.4	0.5	0.6	0.7	0.5	0.4	0.2	0	0	0	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.4	0.5	0.6	0.5	0.5	0.4	0.2	0.1	0.1	0.2	0.3	0.4	0.4	0.4	0.4	0.4	0.4
0.4	0.5	0.6	0.5	0.4	0.4	0.3	0.3	0.4	0.5	0.7	0.6	0.6	0.6	0.6	0.6	0.6
0.4	0.5	0.6	0.5	0.4	0.3	0.4	0.4	0.5	0.7	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.4	0.5	0.6	0.5	0.4	0.4	0.4	0.6	0.7	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.4	0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.7
0.4	0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5
0.4	0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.5	0.4
0.4	0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5	0.4	0.2	0	0	0
0.4	0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5	0.4	0.2	0	0	0	0

conv1\_ReLu

0.3	0.5	0.5	0.6	0.5	0.4	0.4	0.4
0.5	0.6	0.7	0.7	0.5	0.2	0.1	0.1
0.5	0.7	0.7	0.4	0	0.1	0.1	0.1
0.5	0.6	0.5	0.3	0.5	0.7	0.6	0.6
0.5	0.6	0.4	0.6	0.7	0.6	0.6	0.7
0.5	0.6	0.6	0.6	0.6	0.6	0.7	0.7
0.5	0.6	0.6	0.6	0.6	0.7	0.7	0.4
0.5	0.6	0.6	0.6	0.7	0.7	0.4	0.2

```
tf.nn.max_pool(conv1_ReLu, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1])
```



# Max Pooling

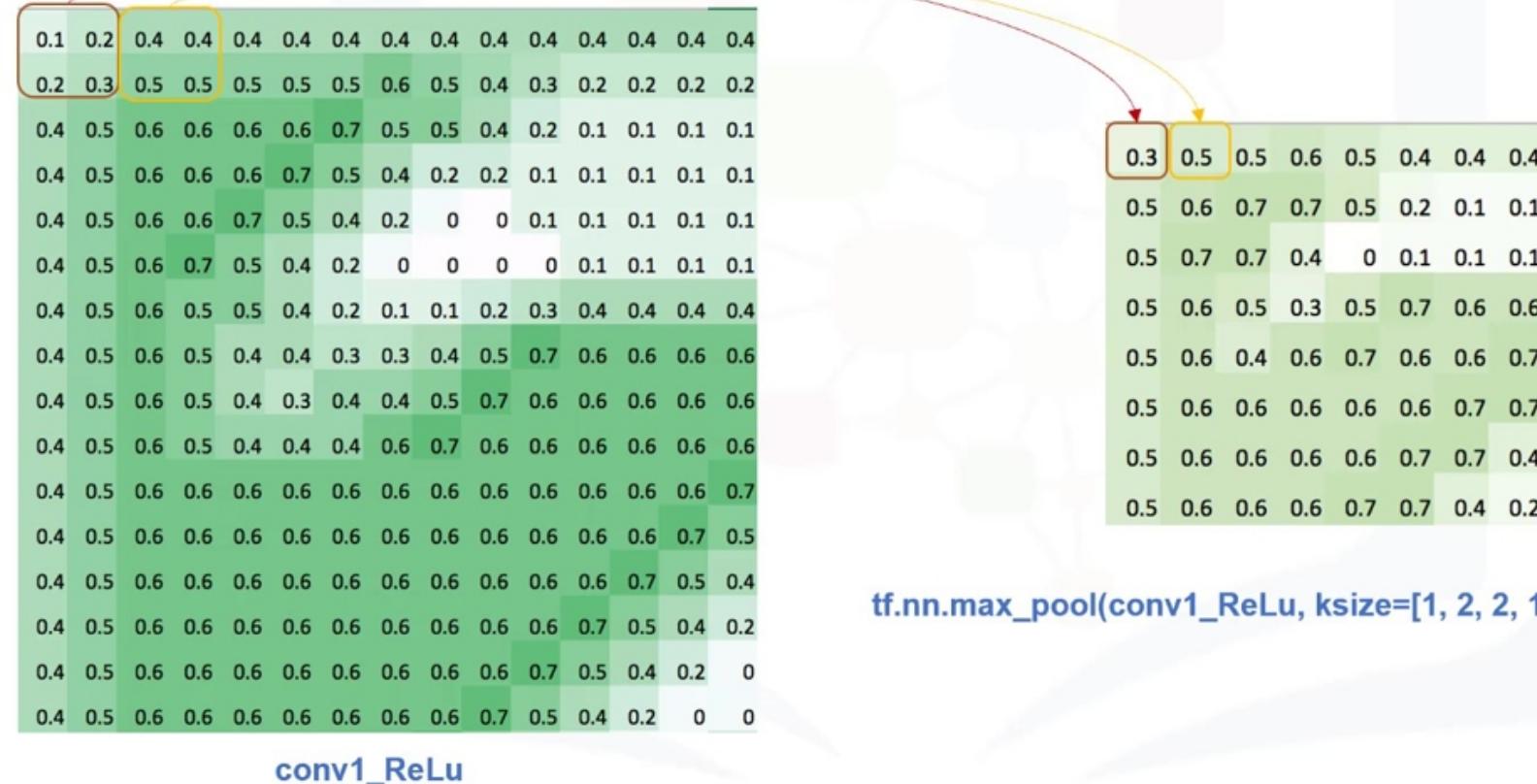
0.1	0.2	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
0.2	0.3	0.5	0.5	0.5	0.5	0.5	0.6	0.5	0.4	0.3	0.2	0.2	0.2	0.2	0.2
0.4	0.5	0.6	0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1	0.1	0.1
0.4	0.5	0.6	0.6	0.6	0.7	0.5	0.4	0.2	0.2	0.1	0.1	0.1	0.1	0.1	0.1
0.4	0.5	0.6	0.6	0.7	0.5	0.4	0.2	0	0	0.1	0.1	0.1	0.1	0.1	0.1
0.4	0.5	0.6	0.7	0.5	0.4	0.2	0	0	0	0.1	0.1	0.1	0.1	0.1	0.1
0.4	0.5	0.6	0.5	0.5	0.4	0.2	0.1	0.1	0.2	0.3	0.4	0.4	0.4	0.4	0.4
0.4	0.5	0.6	0.5	0.4	0.4	0.3	0.3	0.4	0.5	0.7	0.6	0.6	0.6	0.6	0.6
0.4	0.5	0.6	0.5	0.4	0.3	0.4	0.4	0.5	0.7	0.6	0.6	0.6	0.6	0.6	0.6
0.4	0.5	0.6	0.5	0.4	0.4	0.4	0.4	0.6	0.7	0.6	0.6	0.6	0.6	0.6	0.6
0.4	0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.4	0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.4	0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.4	0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.4	0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5	0.4	0.4	0.4	0.4
0.4	0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5	0.4	0.2	0	0

**conv1\_ReLu**

0.3	0.5	0.5	0.6	0.5	0.4	0.4	0.4
0.5	0.6	0.7	0.7	0.5	0.2	0.1	0.1
0.5	0.7	0.7	0.4	0	0.1	0.1	0.1
0.5	0.6	0.5	0.3	0.5	0.7	0.6	0.6
0.5	0.6	0.4	0.6	0.7	0.6	0.6	0.6
0.5	0.6	0.6	0.6	0.6	0.6	0.7	0.7
0.5	0.6	0.6	0.6	0.6	0.7	0.7	0.7
0.5	0.6	0.6	0.6	0.7	0.7	0.7	0.7

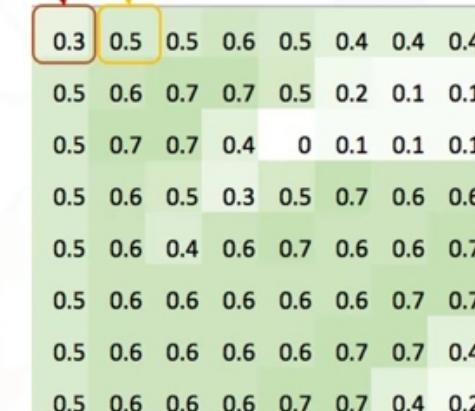
```
tf.nn.max_pool(conv1_ReLu, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1])
```

# Max Pooling



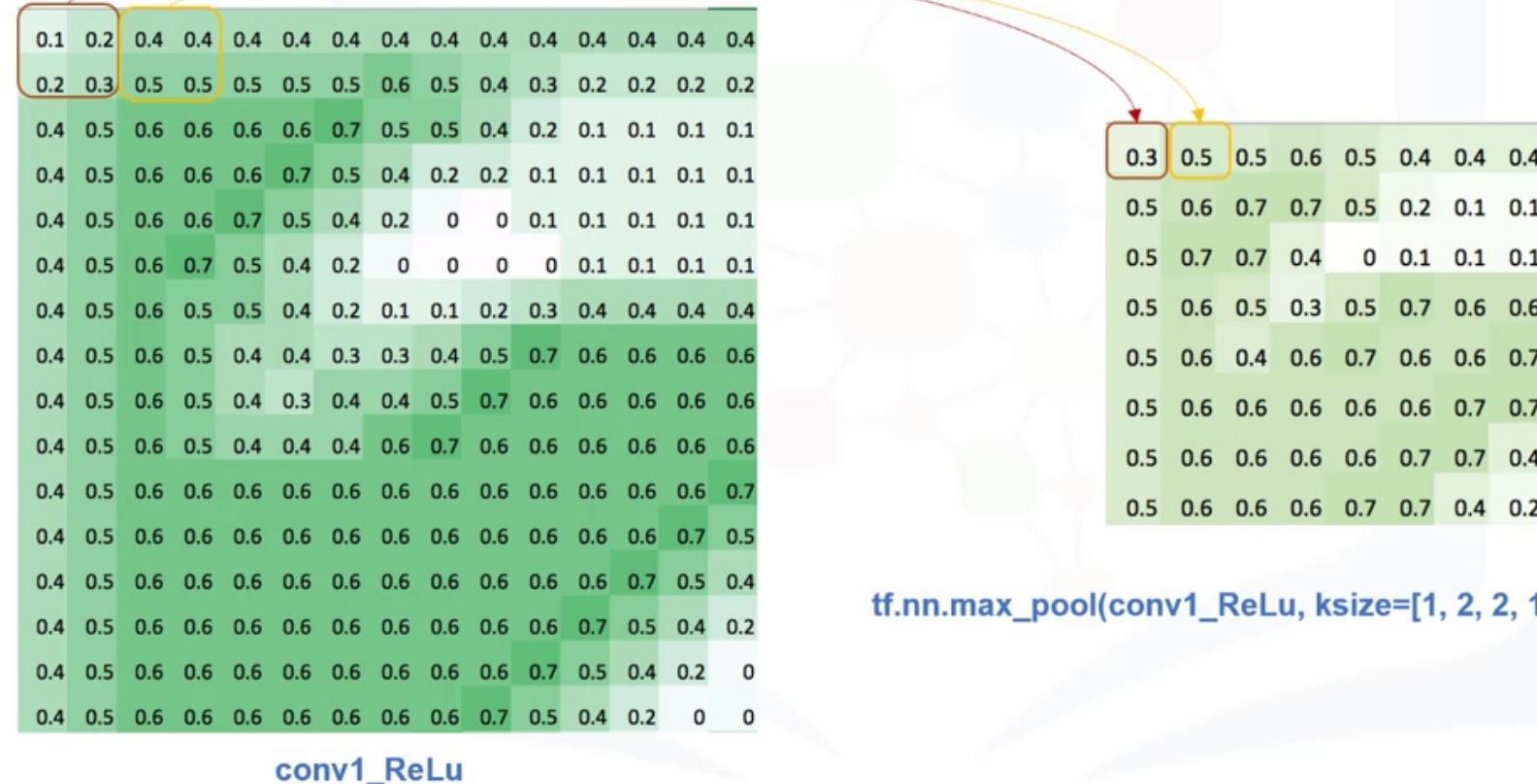


# Max Pooling

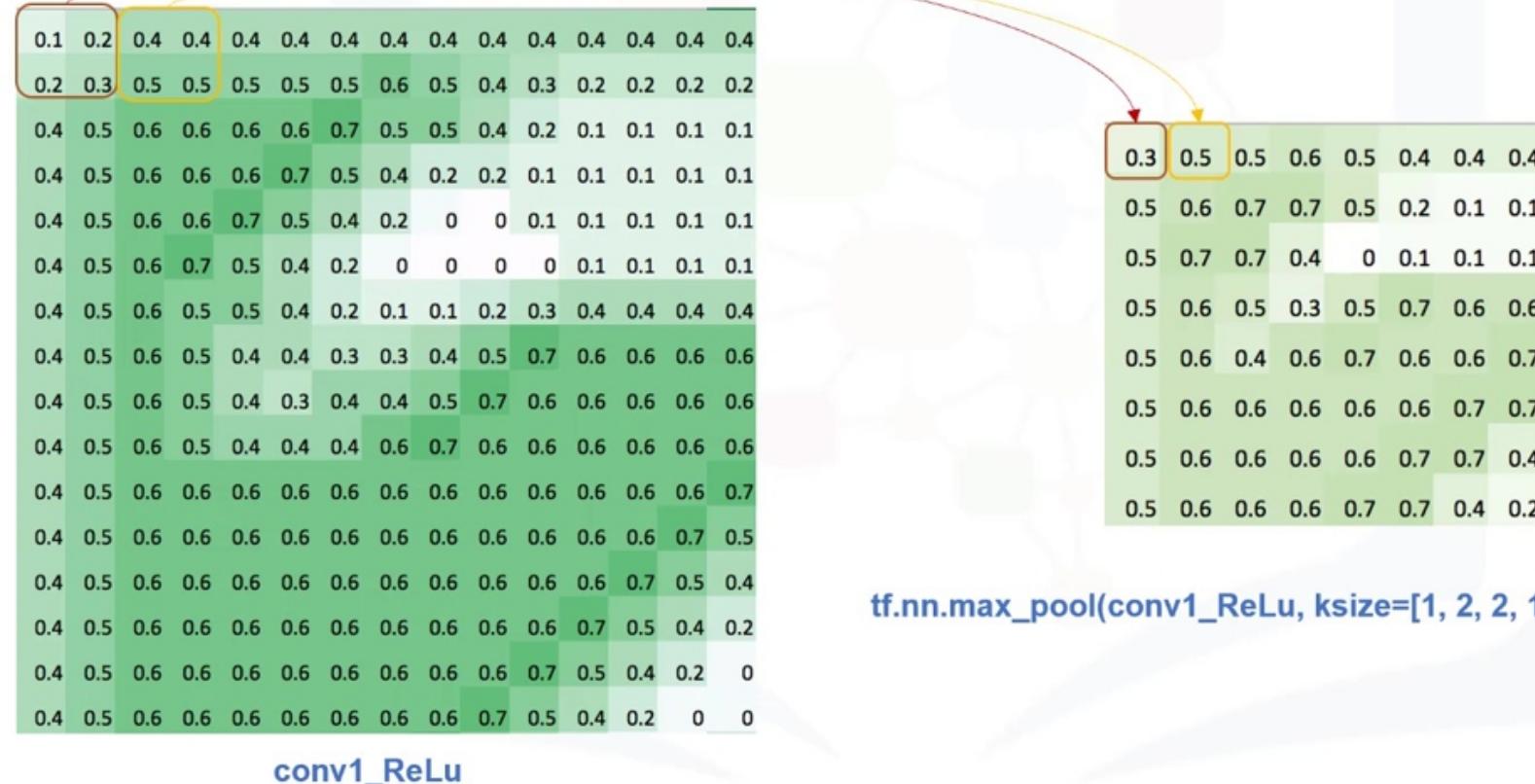


```
tf.nn.max_pool(conv1_ReLu, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1])
```

# Max Pooling

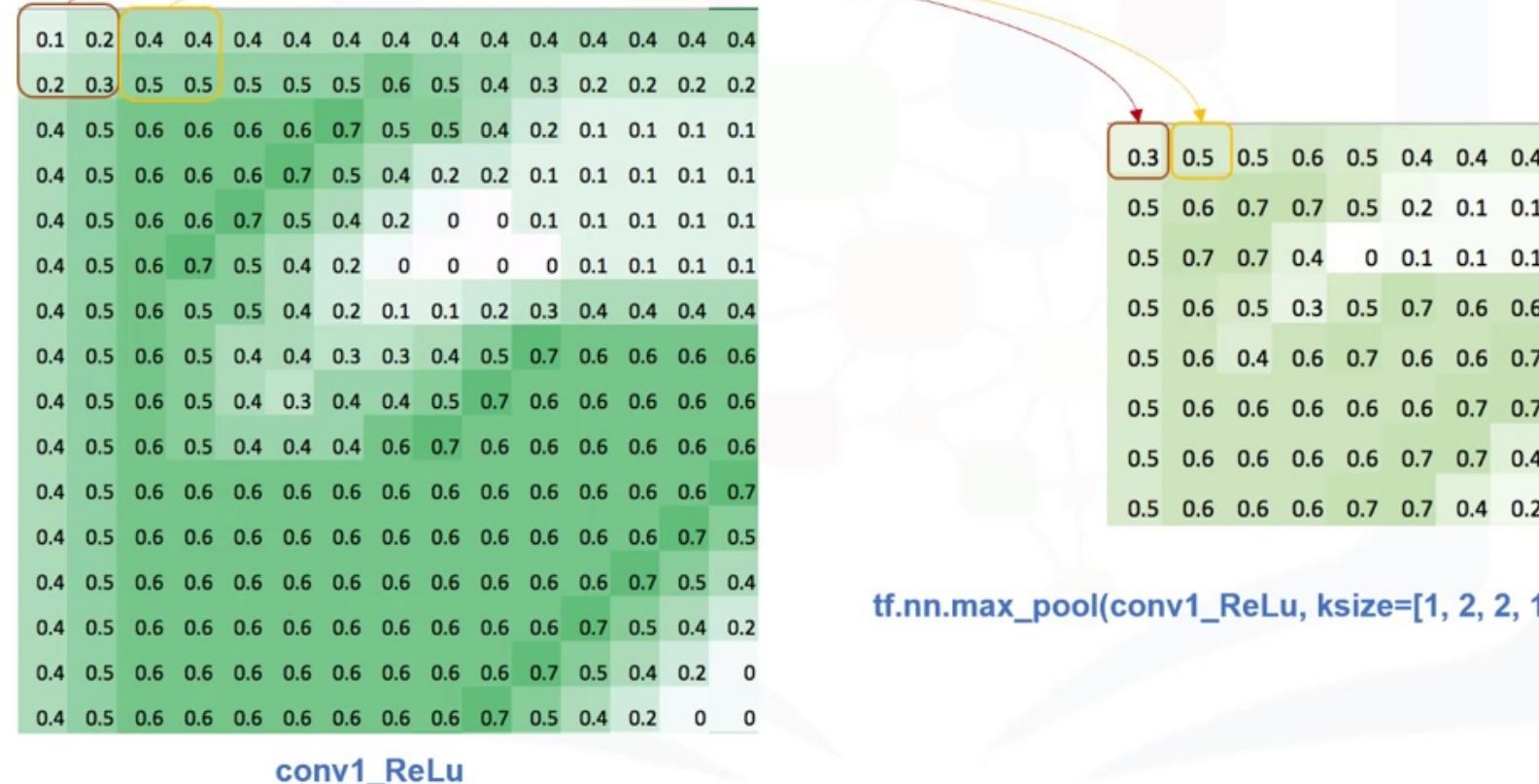


# Max Pooling



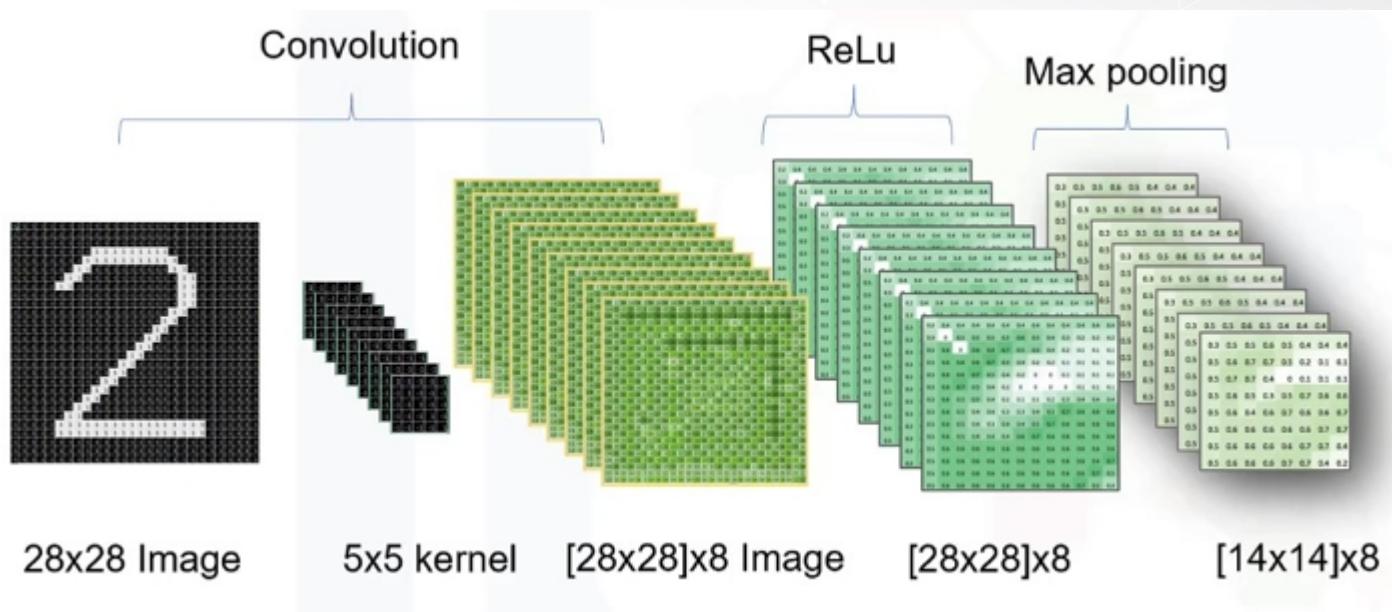
```
tf.nn.max_pool(conv1_ReLu, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1])
```

# Max Pooling



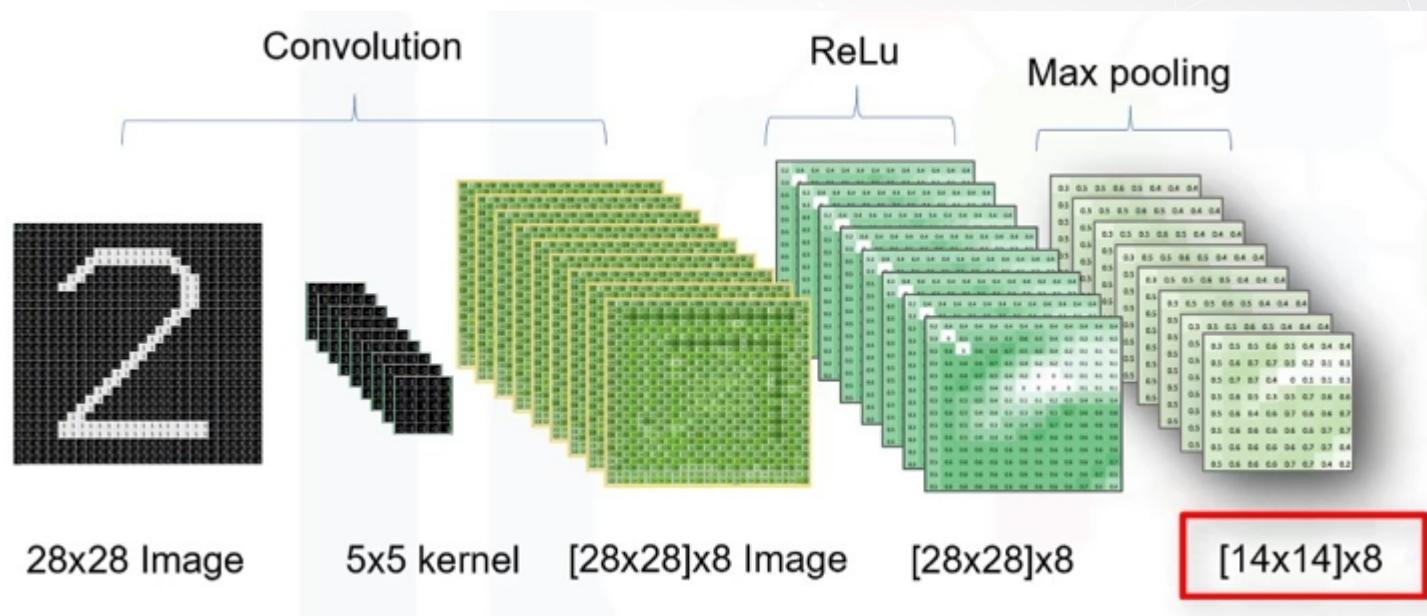


# CNN Architecture ReLu





# CNN Architecture ReLu



# Fully Connected Layer

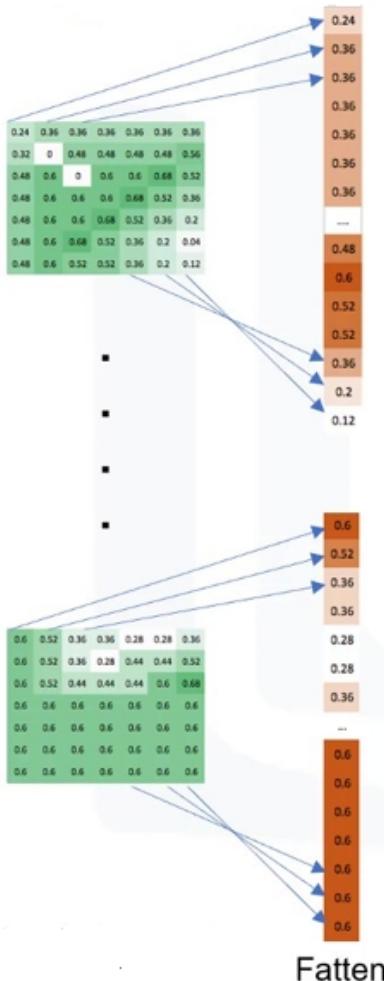
0.24	0.36	0.36	0.36	0.36	0.36	0.36	0.36
0.32	0	0.48	0.48	0.48	0.48	0.48	0.56
0.48	0.6	0	0.6	0.6	0.68	0.52	
0.48	0.6	0.6	0.6	0.68	0.32	0.36	
0.48	0.6	0.6	0.68	0.52	0.36	0.2	
0.48	0.6	0.68	0.52	0.36	0.2	0.04	
0.48	0.6	0.52	0.52	0.36	0.2	0.12	

■ ■ ■ ■ ■ ■ ■ ■

0.6	0.52	0.36	0.36	0.28	0.28	0.36	
0.6	0.52	0.36	0.28	0.44	0.44	0.52	
0.6	0.52	0.44	0.44	0.44	0.6	0.68	
0.6	0.6	0.6	0.6	0.6	0.6	0.6	
0.6	0.6	0.6	0.6	0.6	0.6	0.6	
0.6	0.6	0.6	0.6	0.6	0.6	0.6	
0.6	0.6	0.6	0.6	0.6	0.6	0.6	

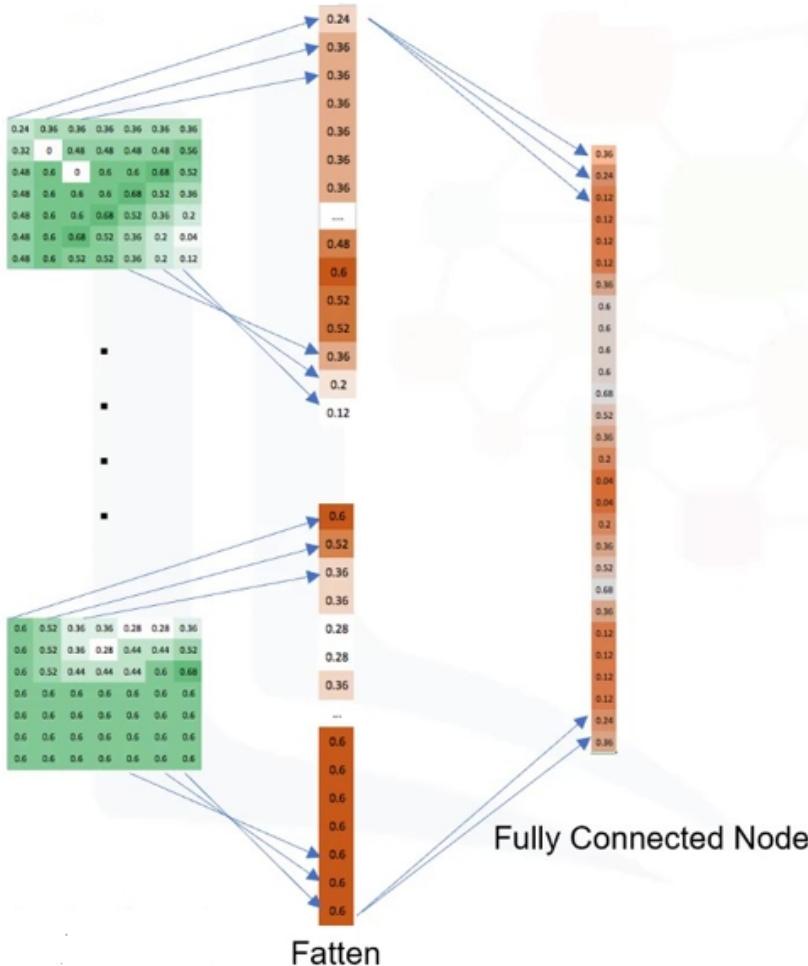


# Fully Connected Layer



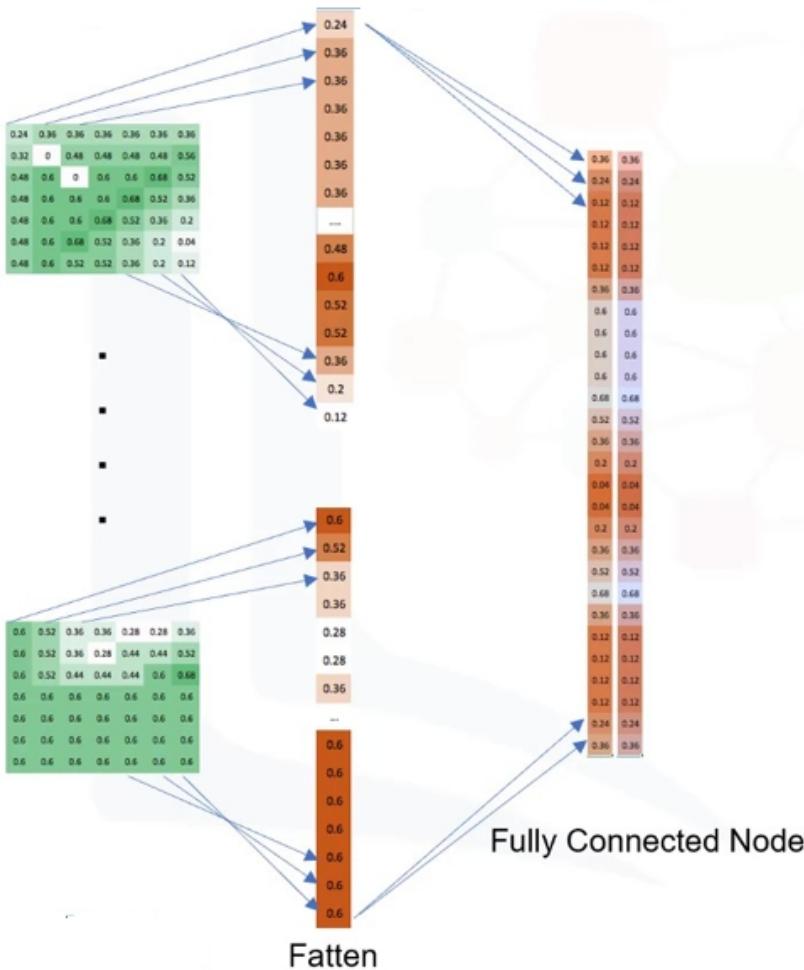


# Fully Connected Layer

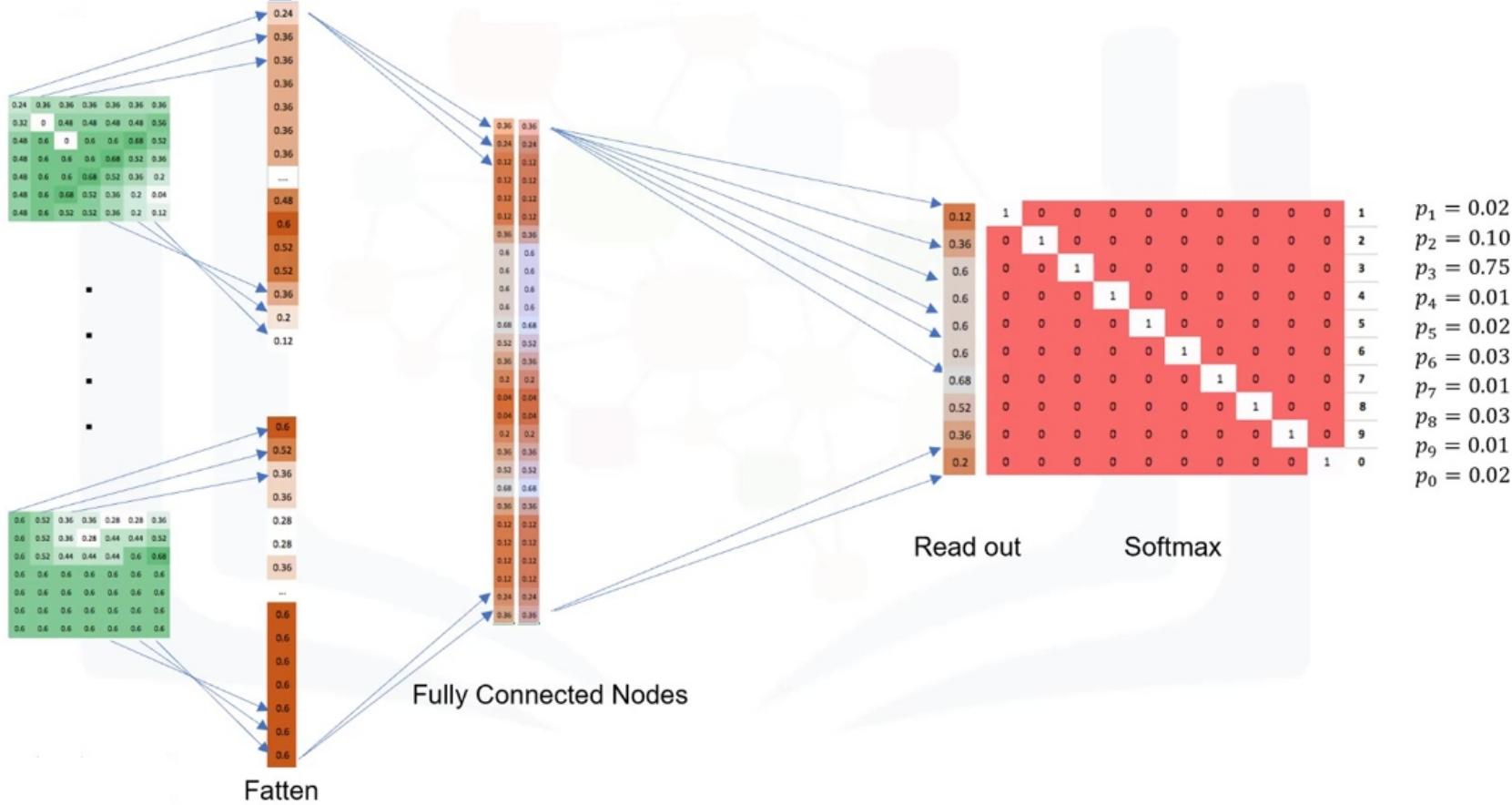




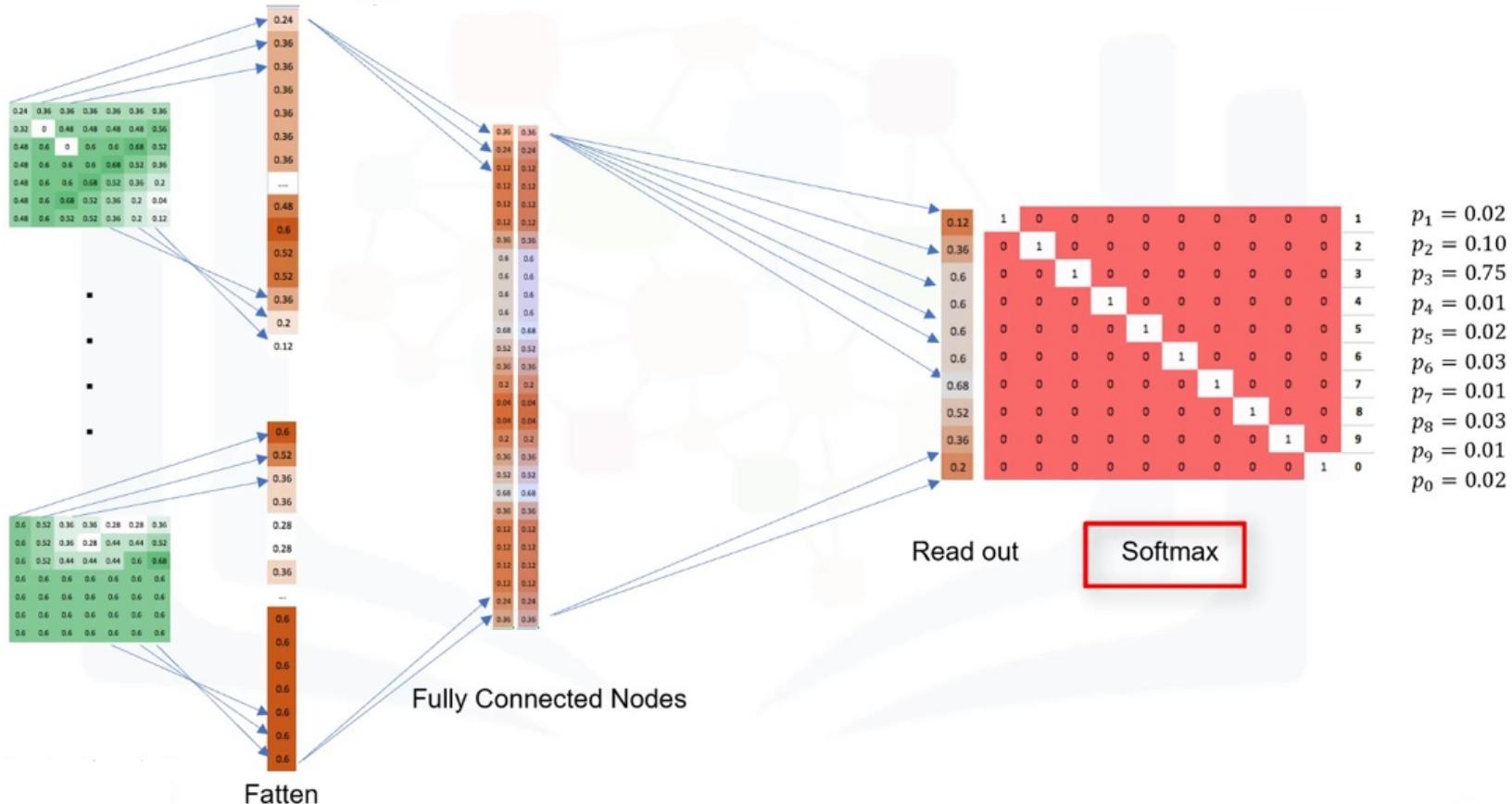
# Fully Connected Layer



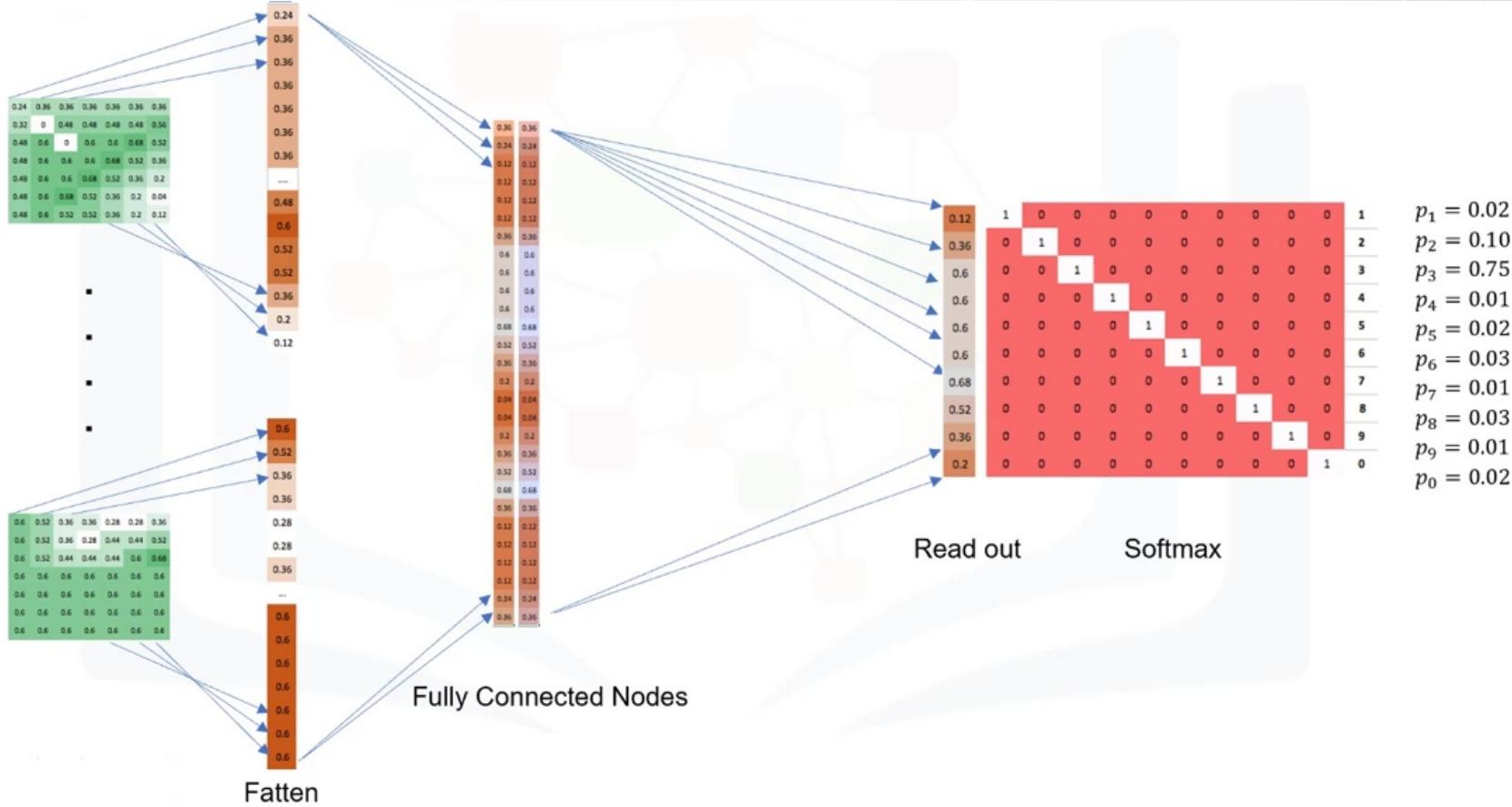
# Fully Connected Layer



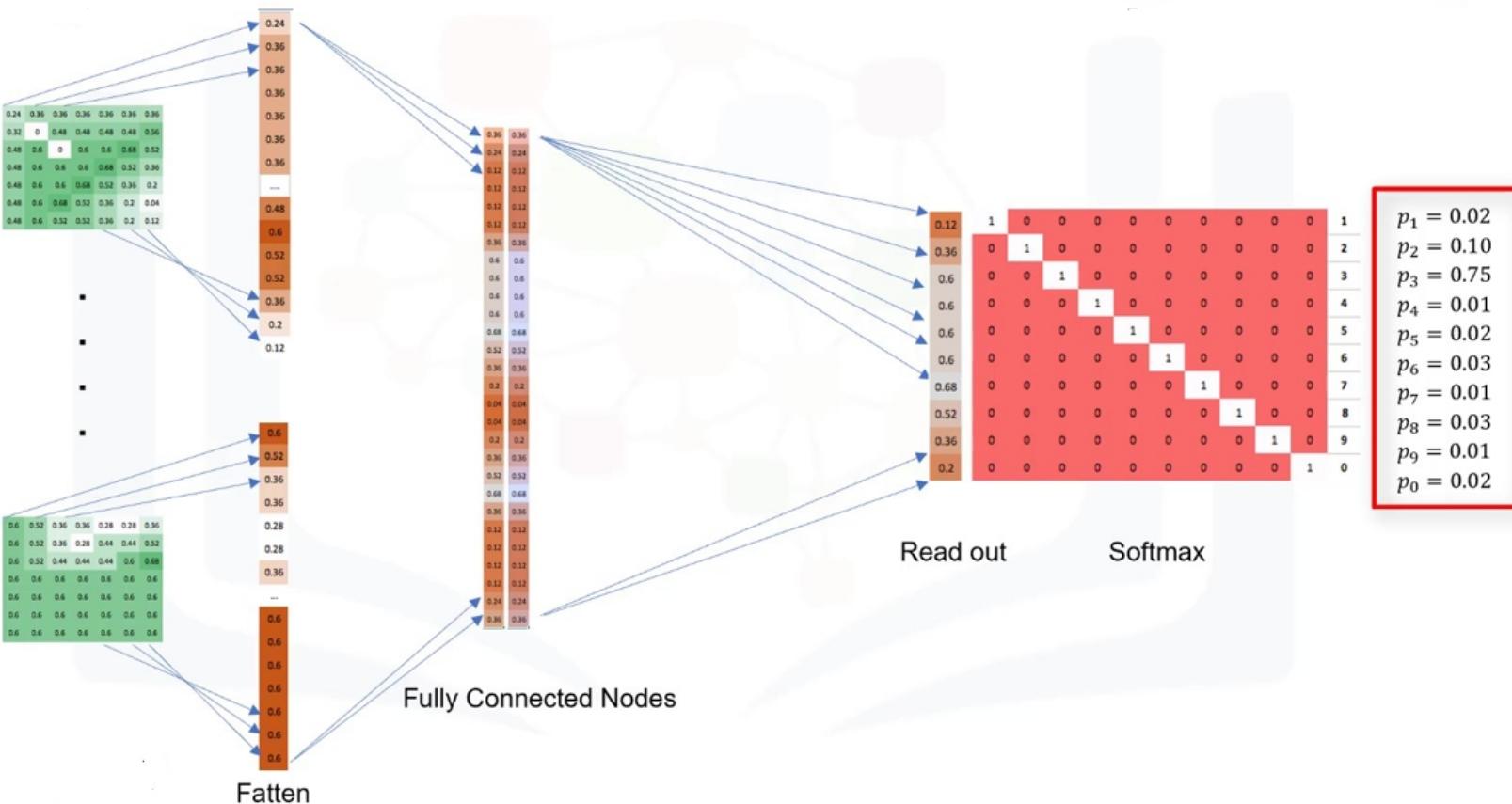
# Fully Connected Layer



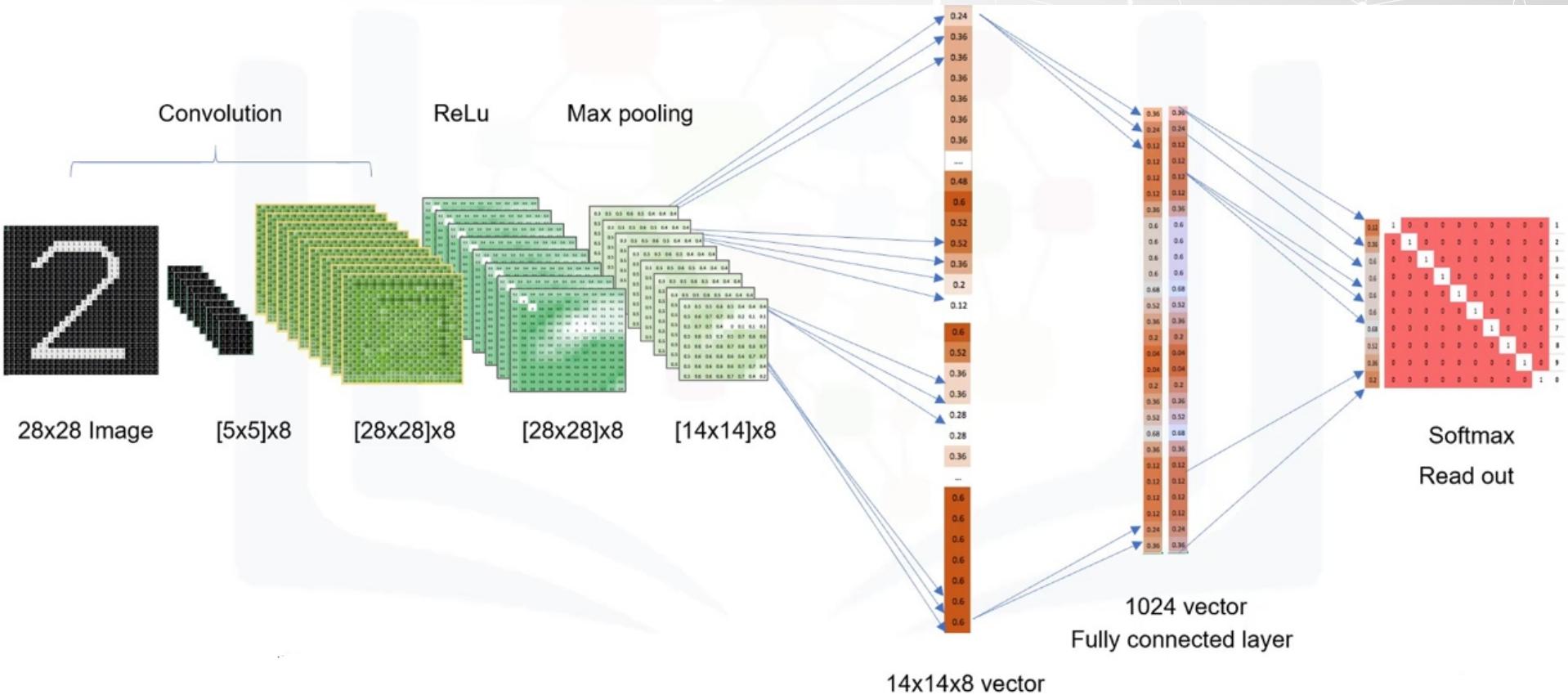
# Fully Connected Layer



# Fully Connected Layer

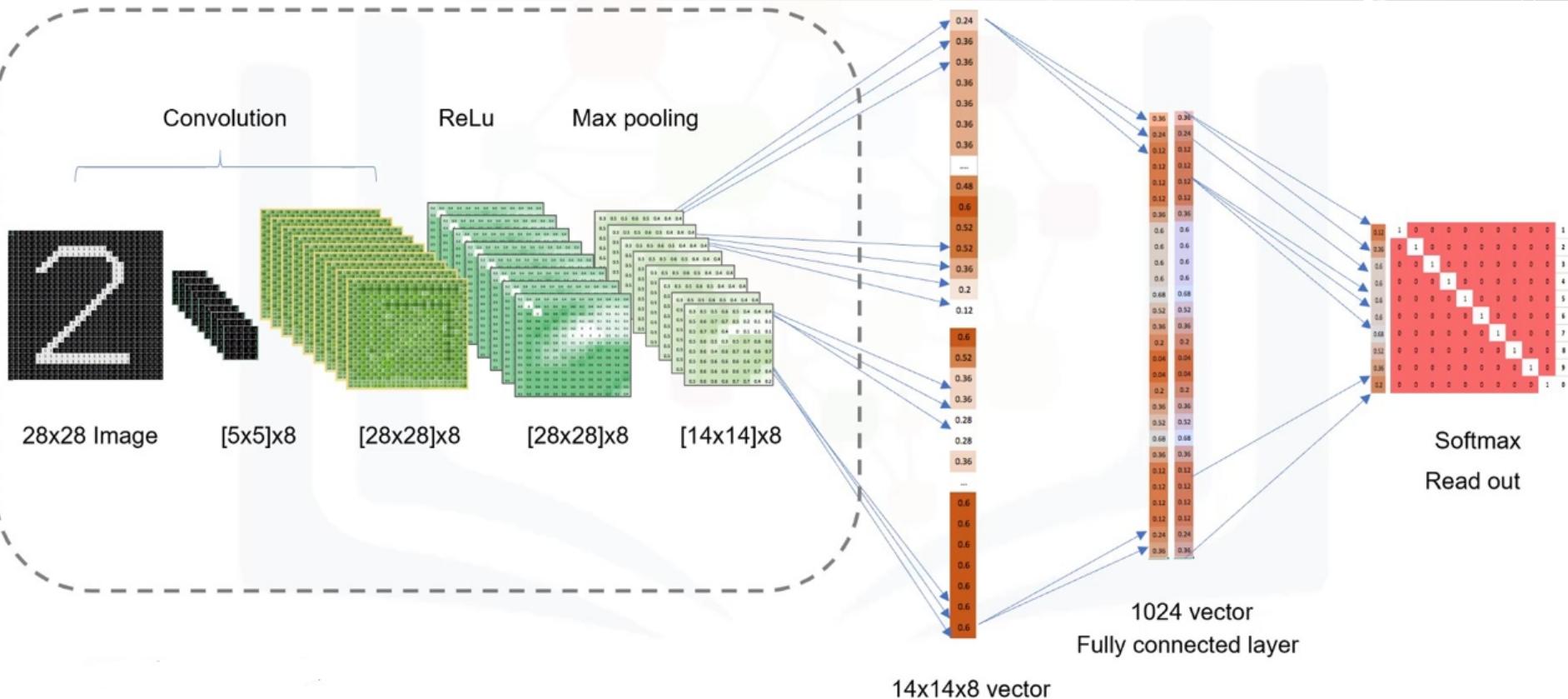


# CNN Architecture



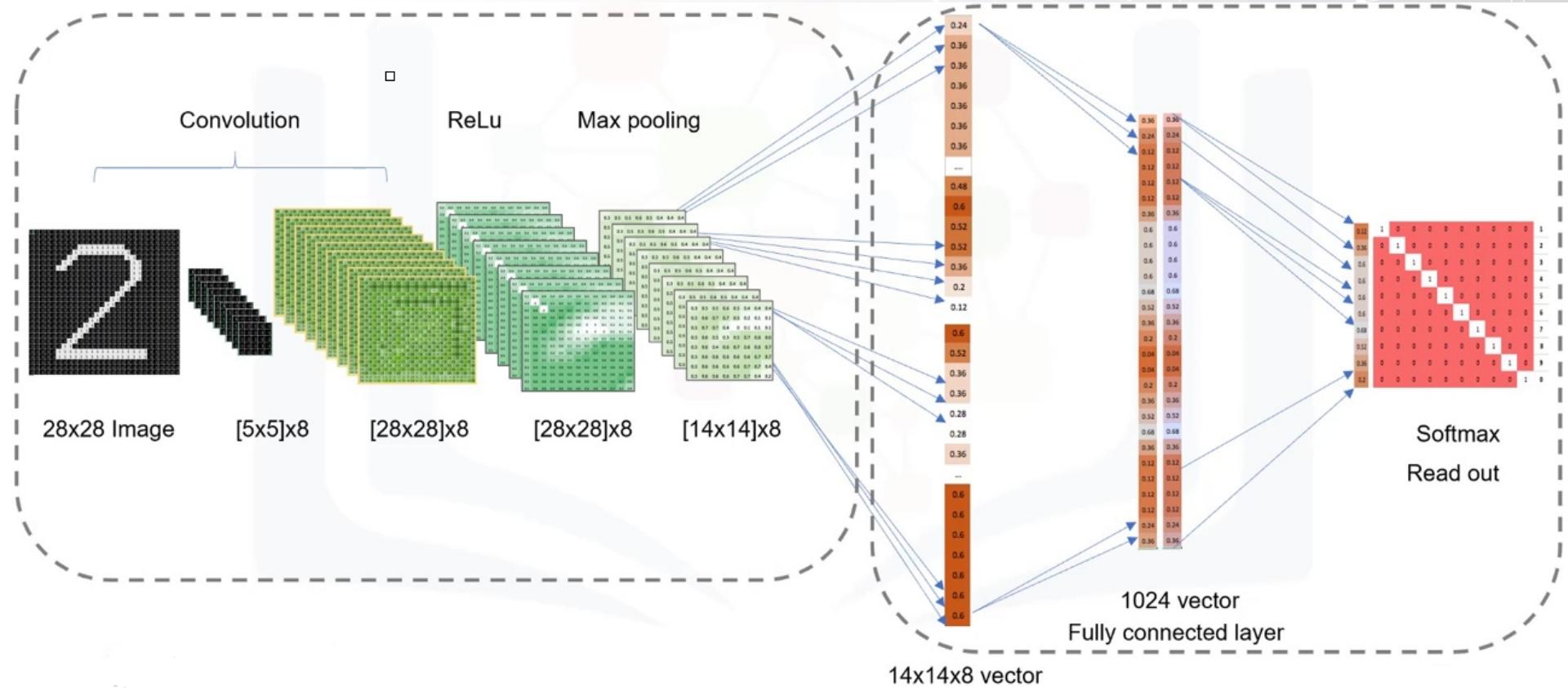


# CNN Architecture

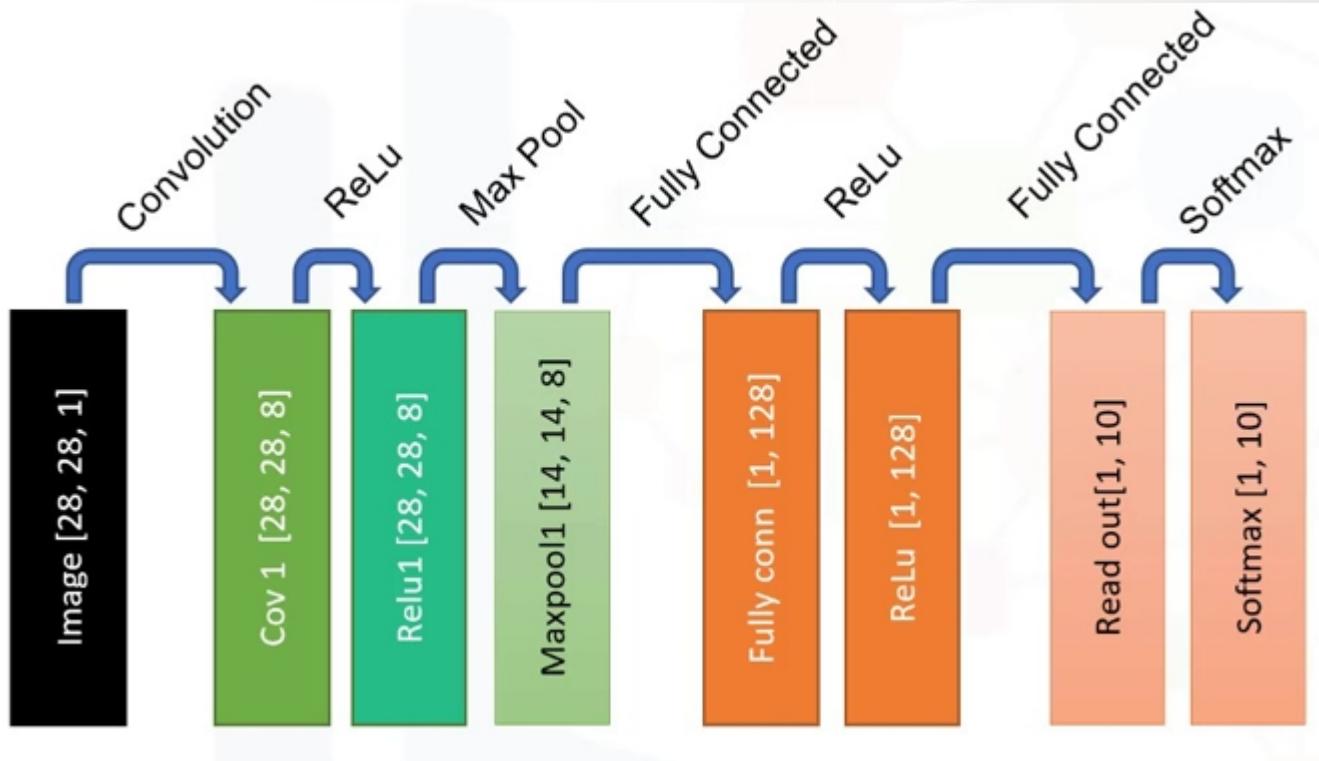




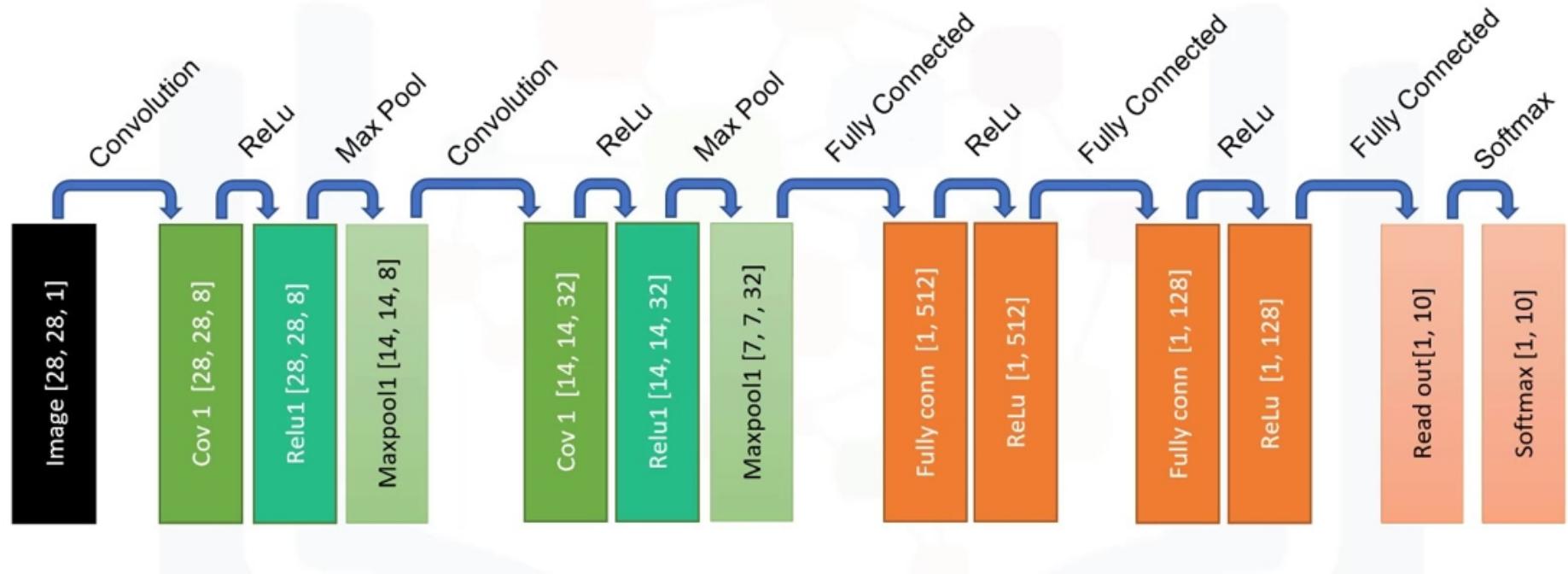
# CNN Architecture



# CNN Architecture

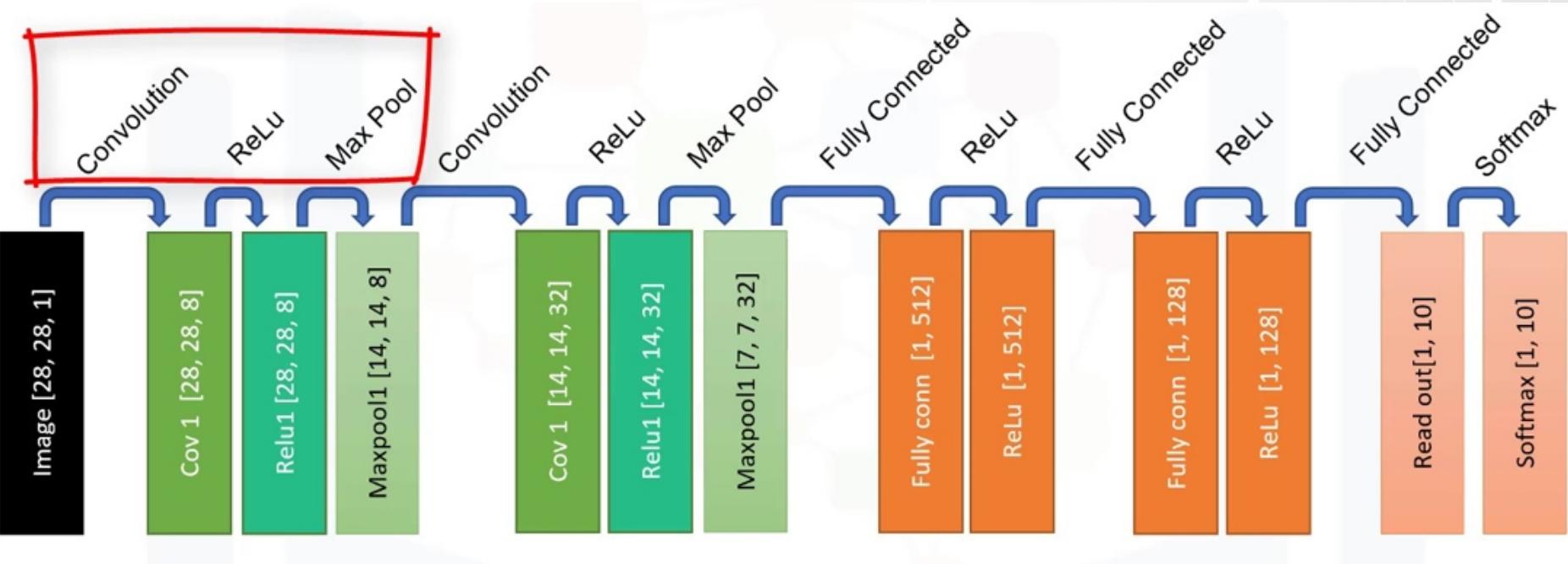


# CNN Architecture

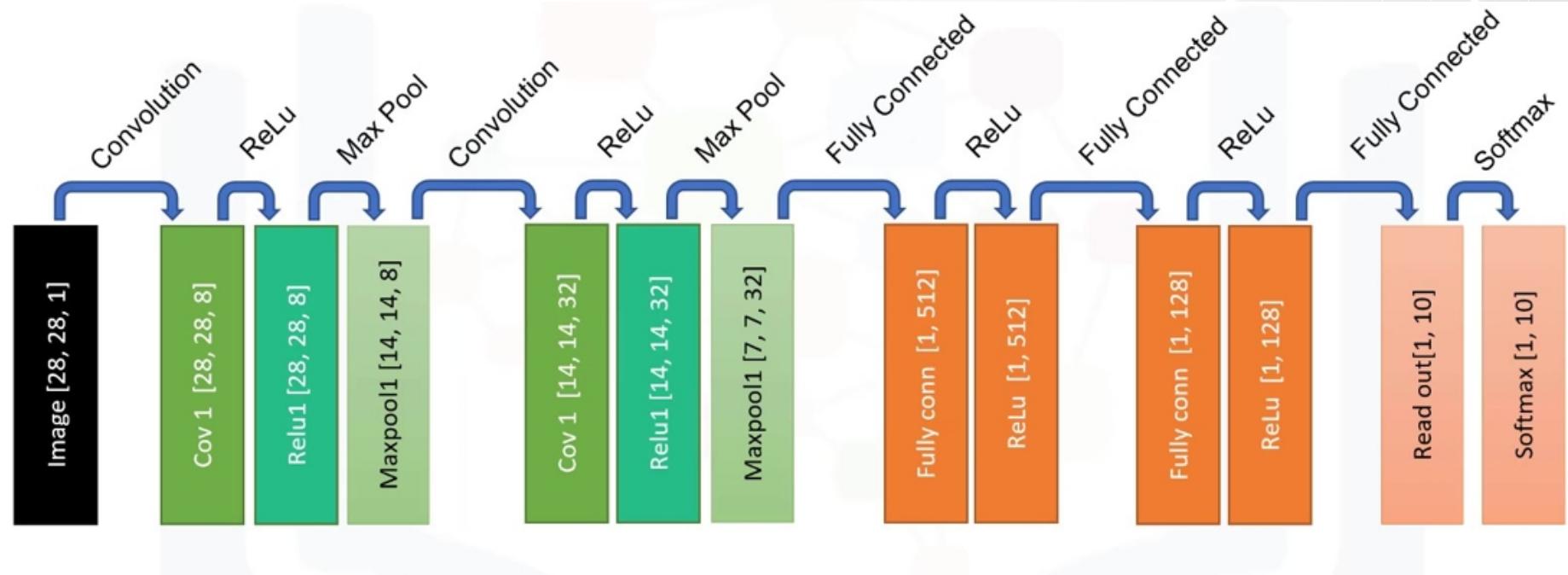




# CNN Architecture



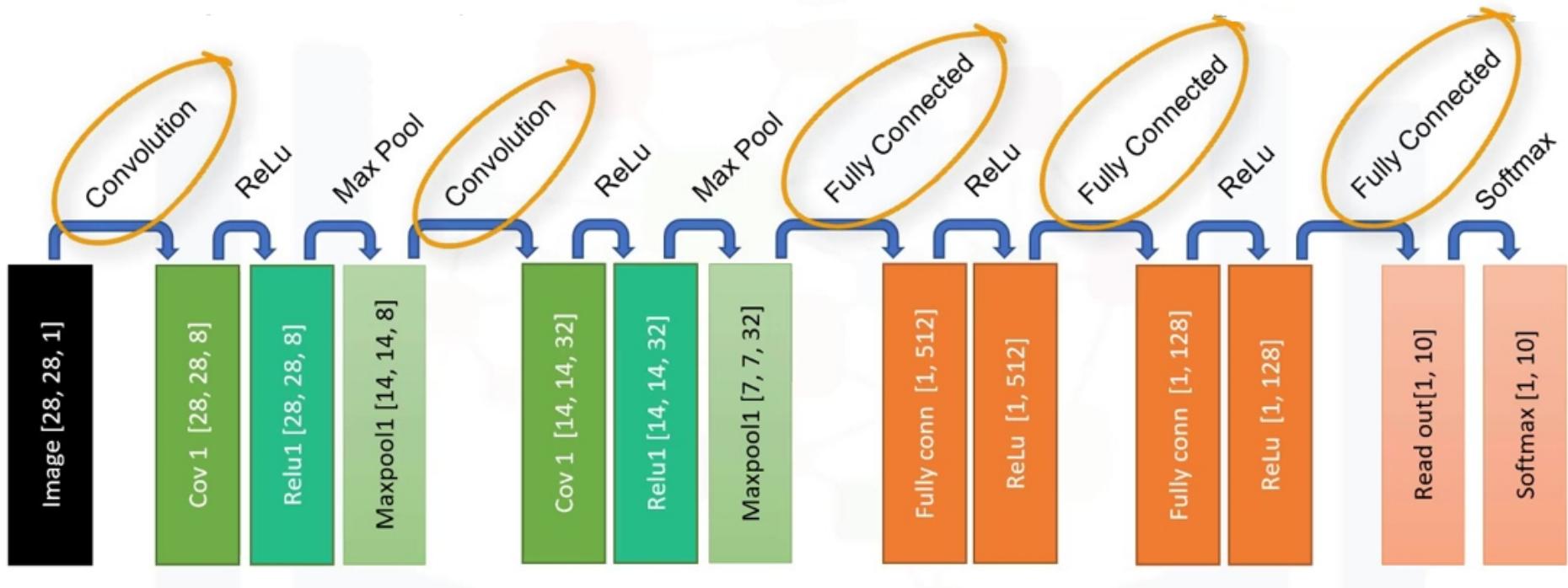
# CNN Architecture



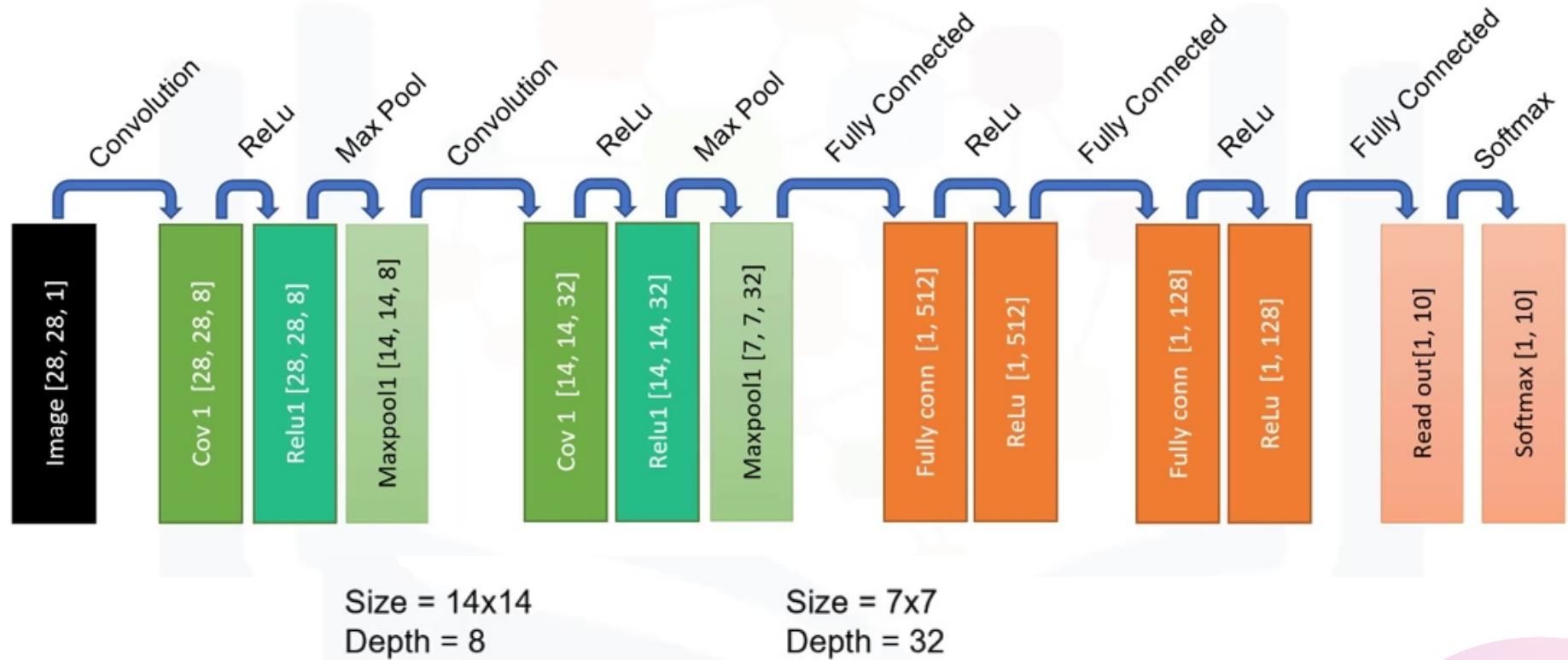
# CNN Architecture



# CNN Architecture



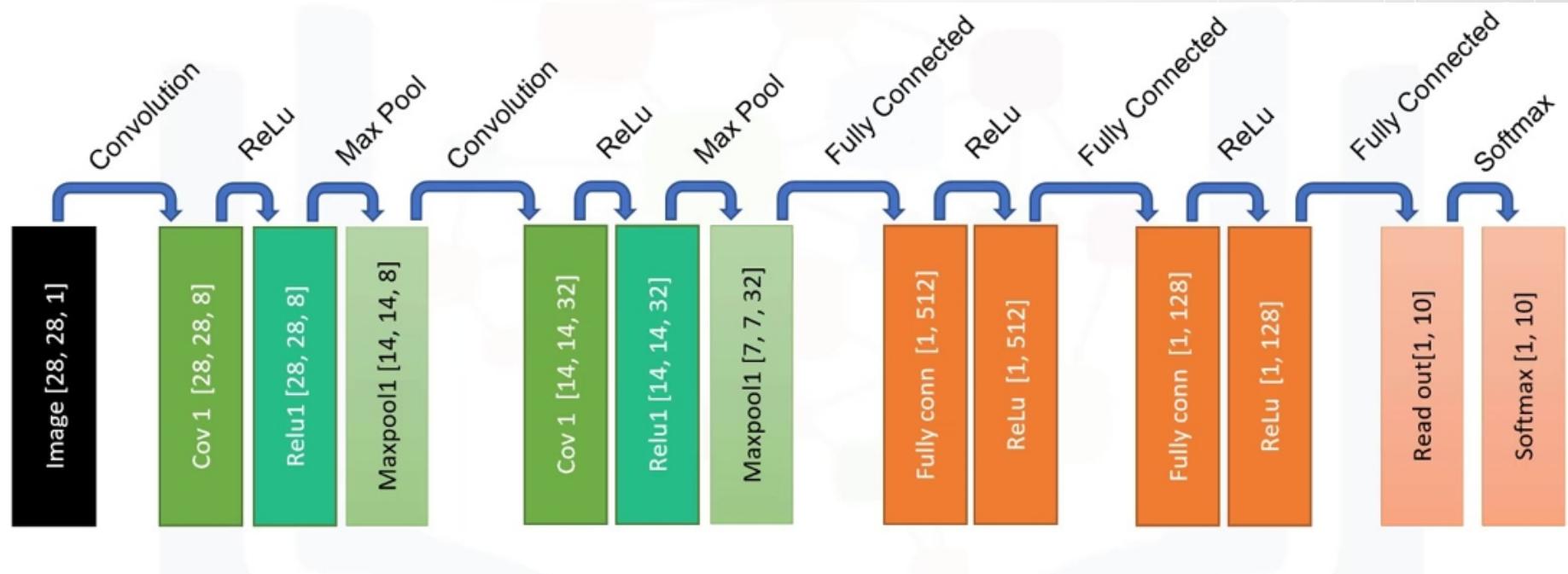
# CNN Architecture



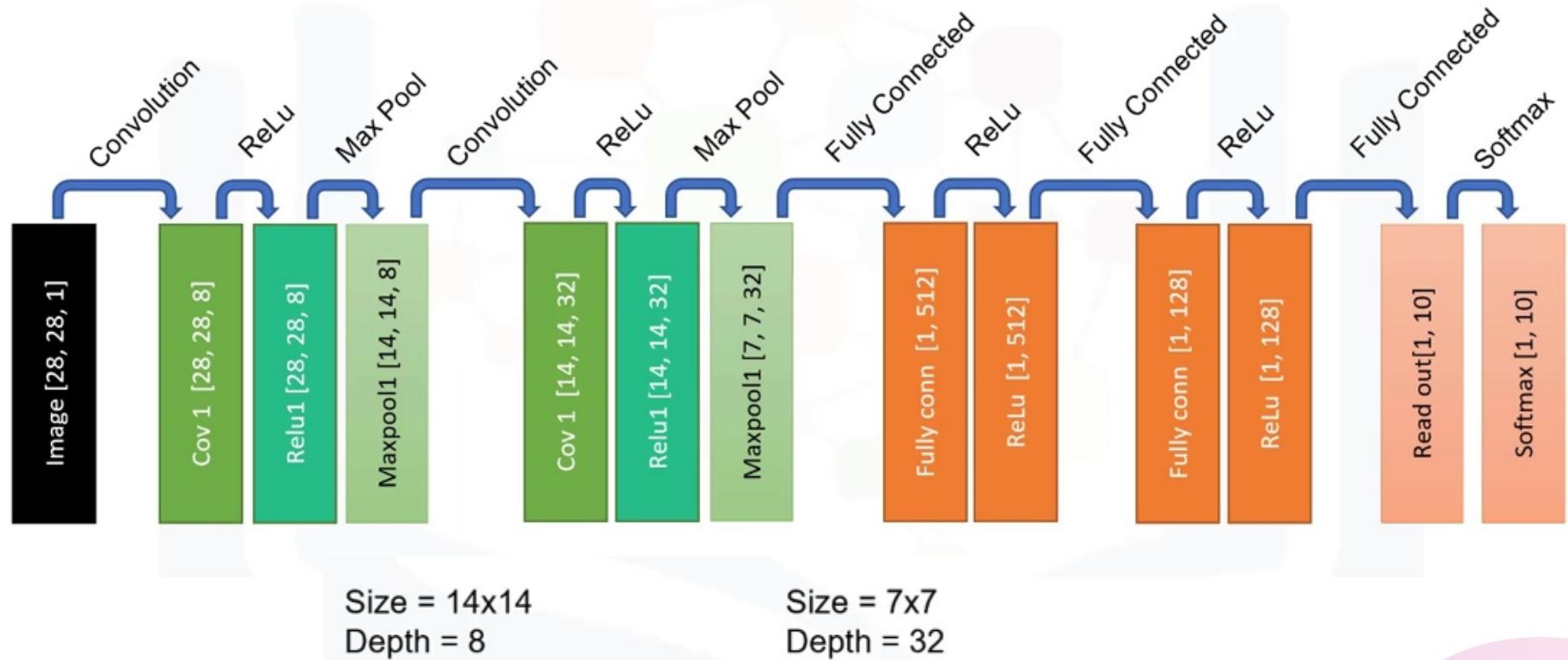
# CNN Architecture



# CNN Architecture



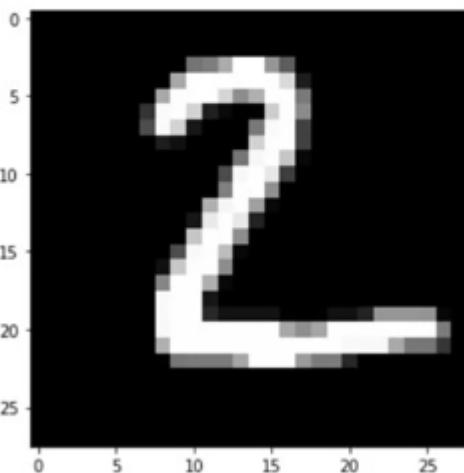
# CNN Architecture





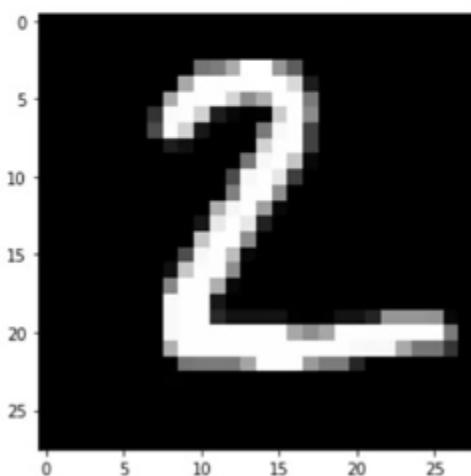
# CNN Architecture

Image [28, 28]



# CNN Architecture

Image [28, 28]

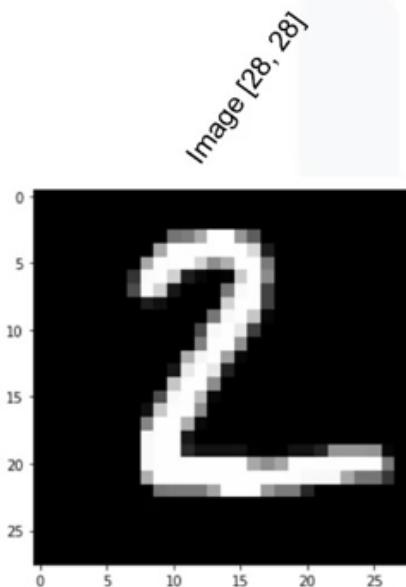


Kernel [5, 5, 8]



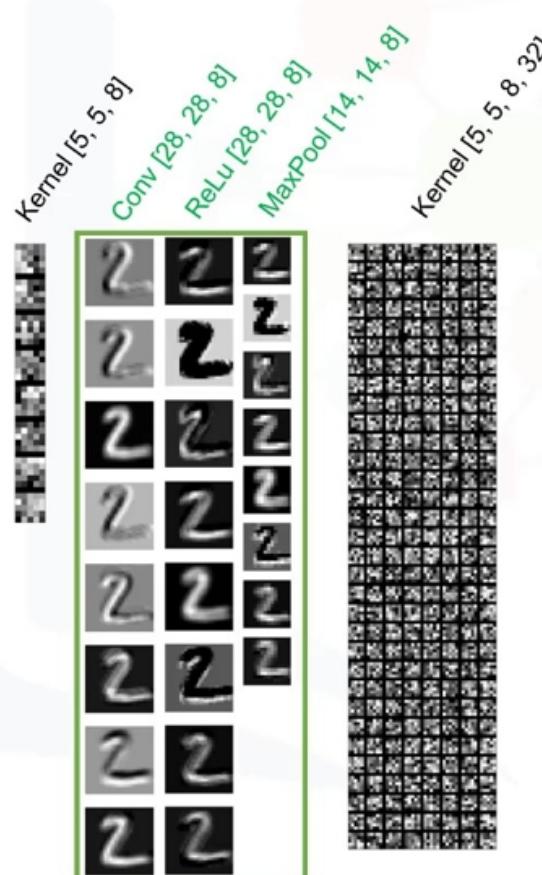
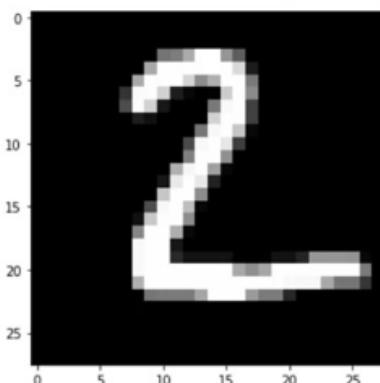


# CNN Architecture



# CNN Architecture

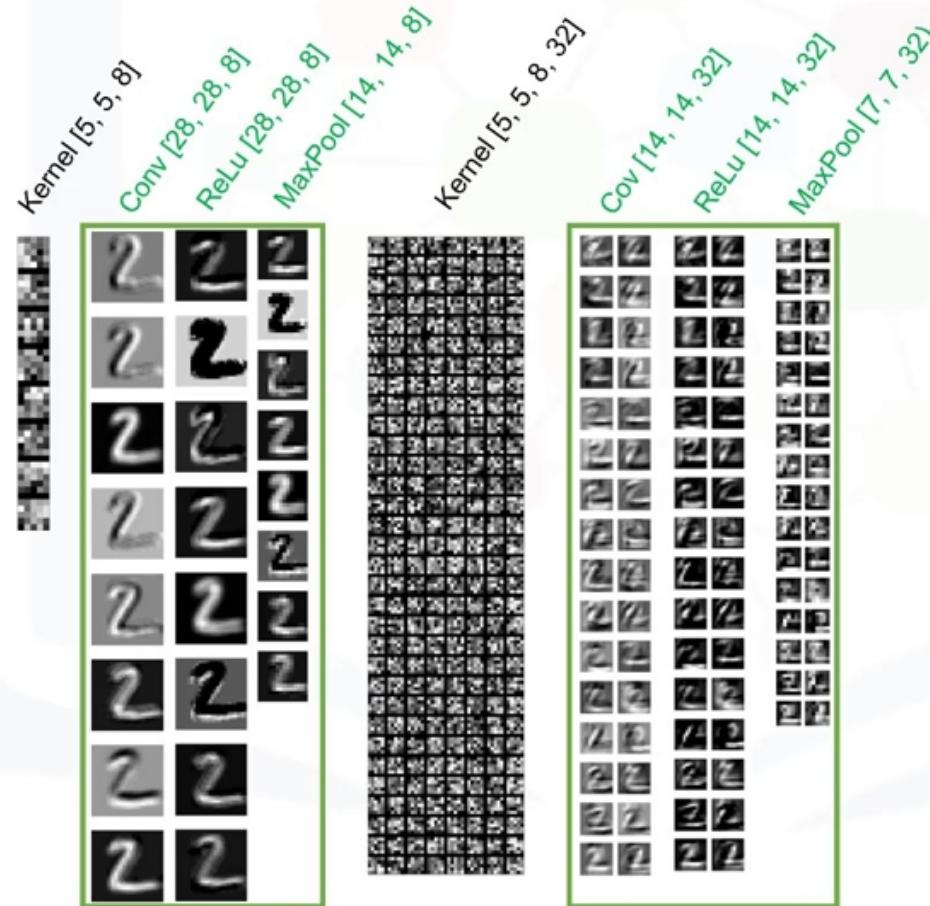
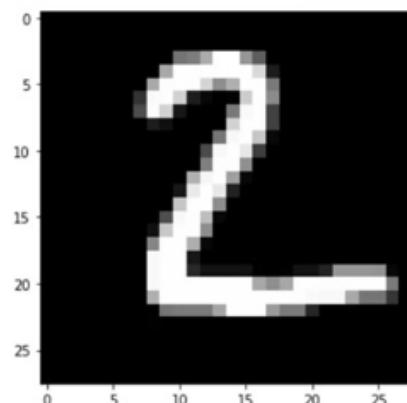
Image [28, 28]



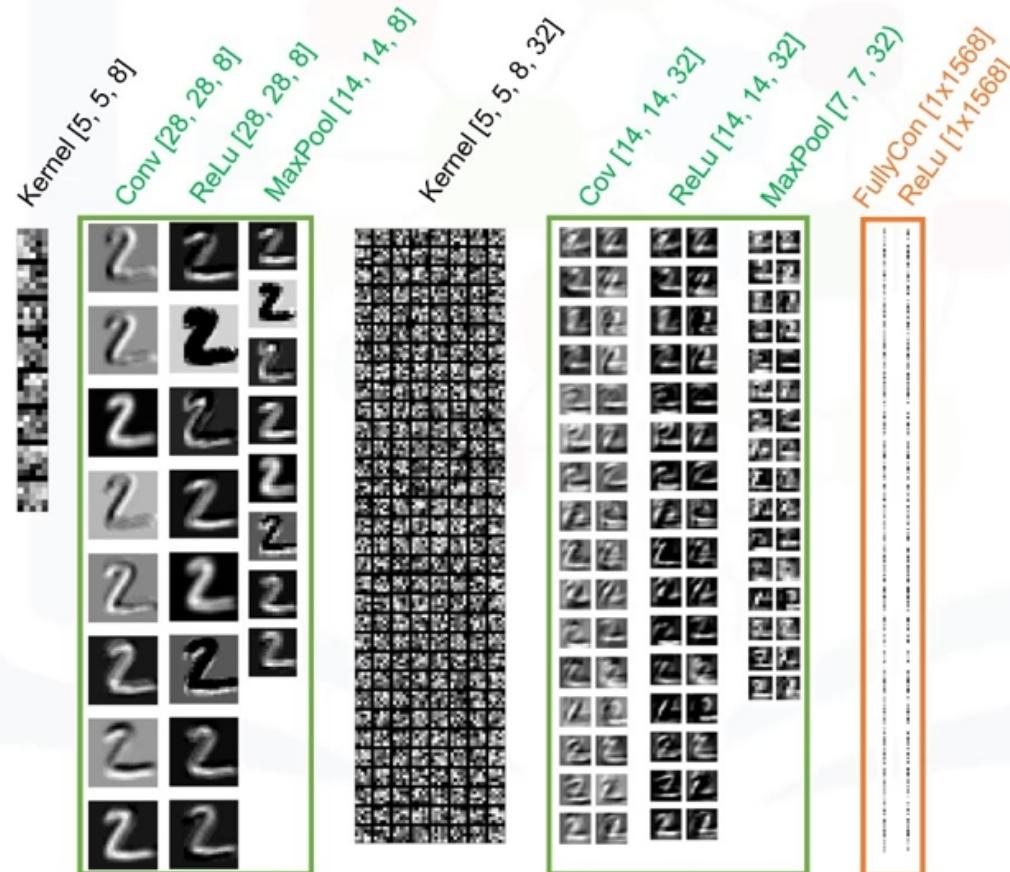
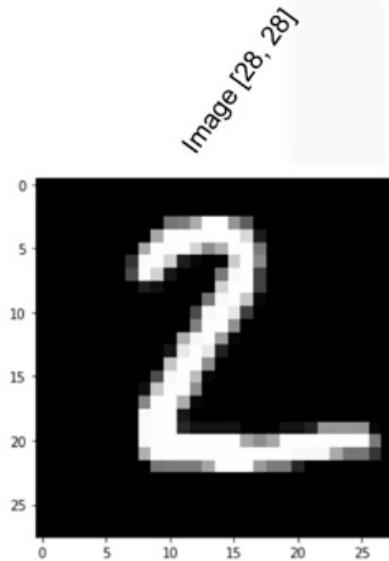


# CNN Architecture

Image [28, 28]

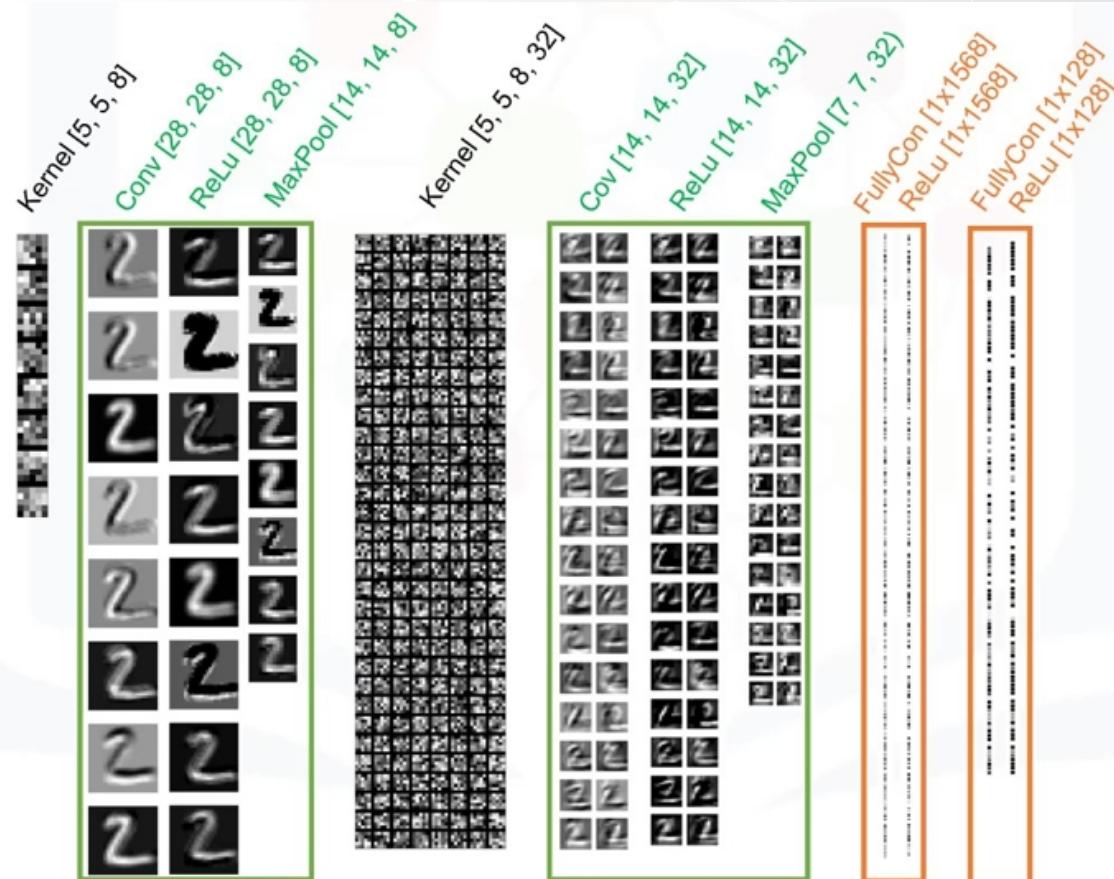
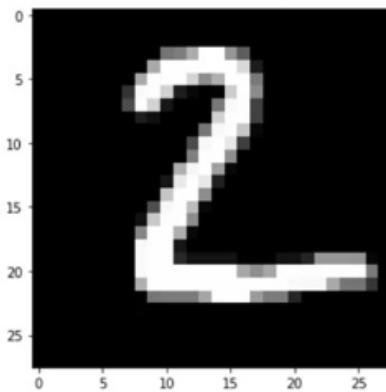


# CNN Architecture

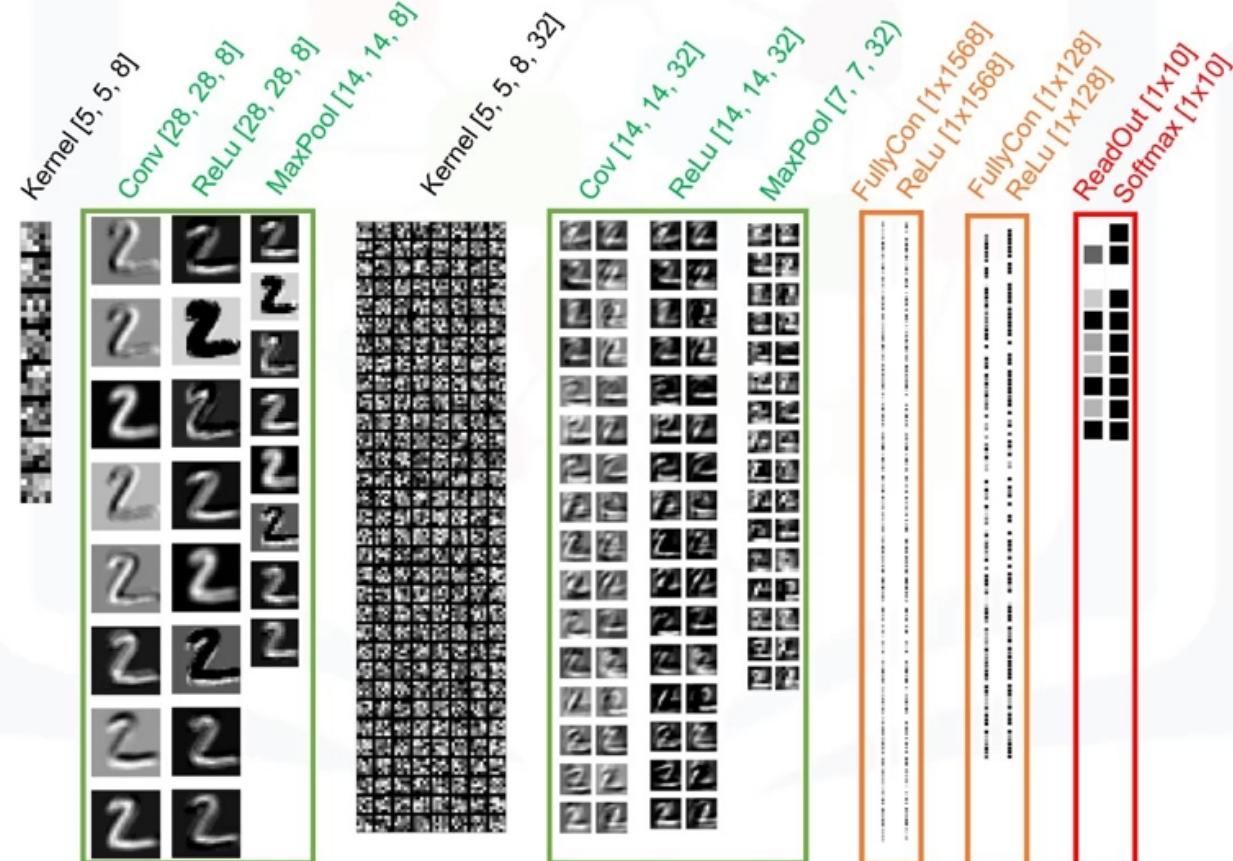
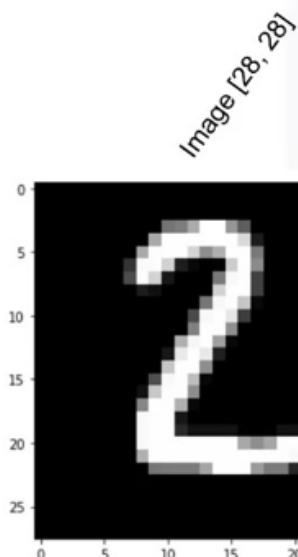


# CNN Architecture

Image [28, 28]

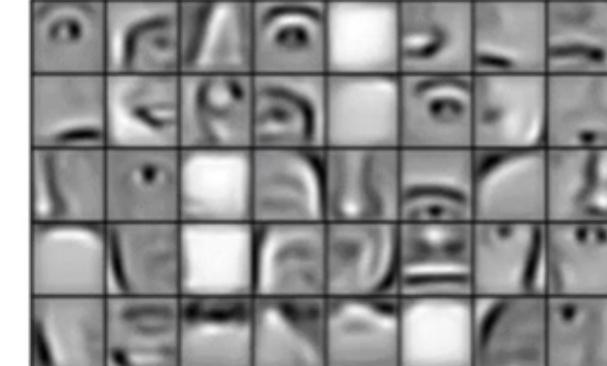
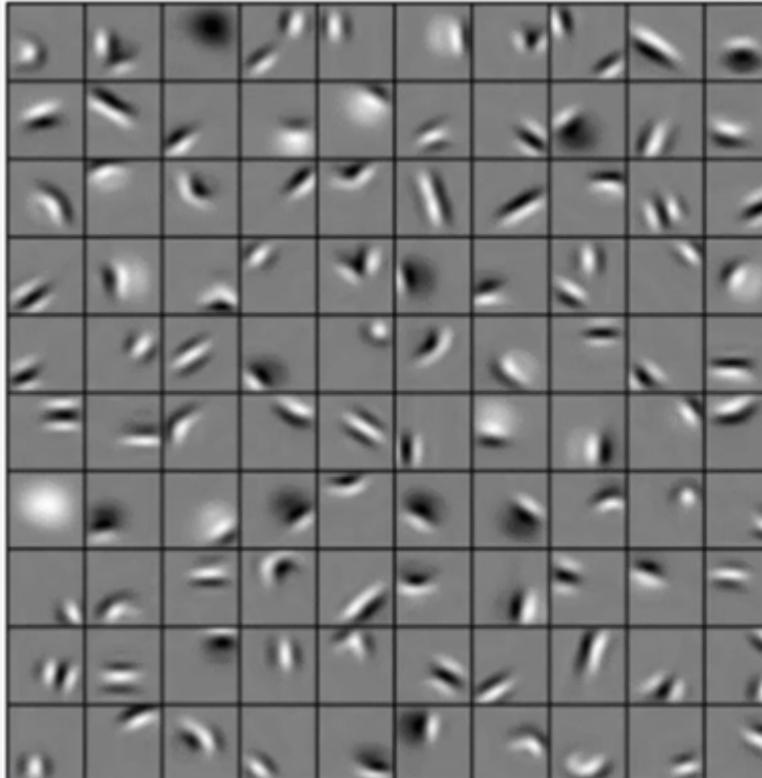


# CNN Architecture



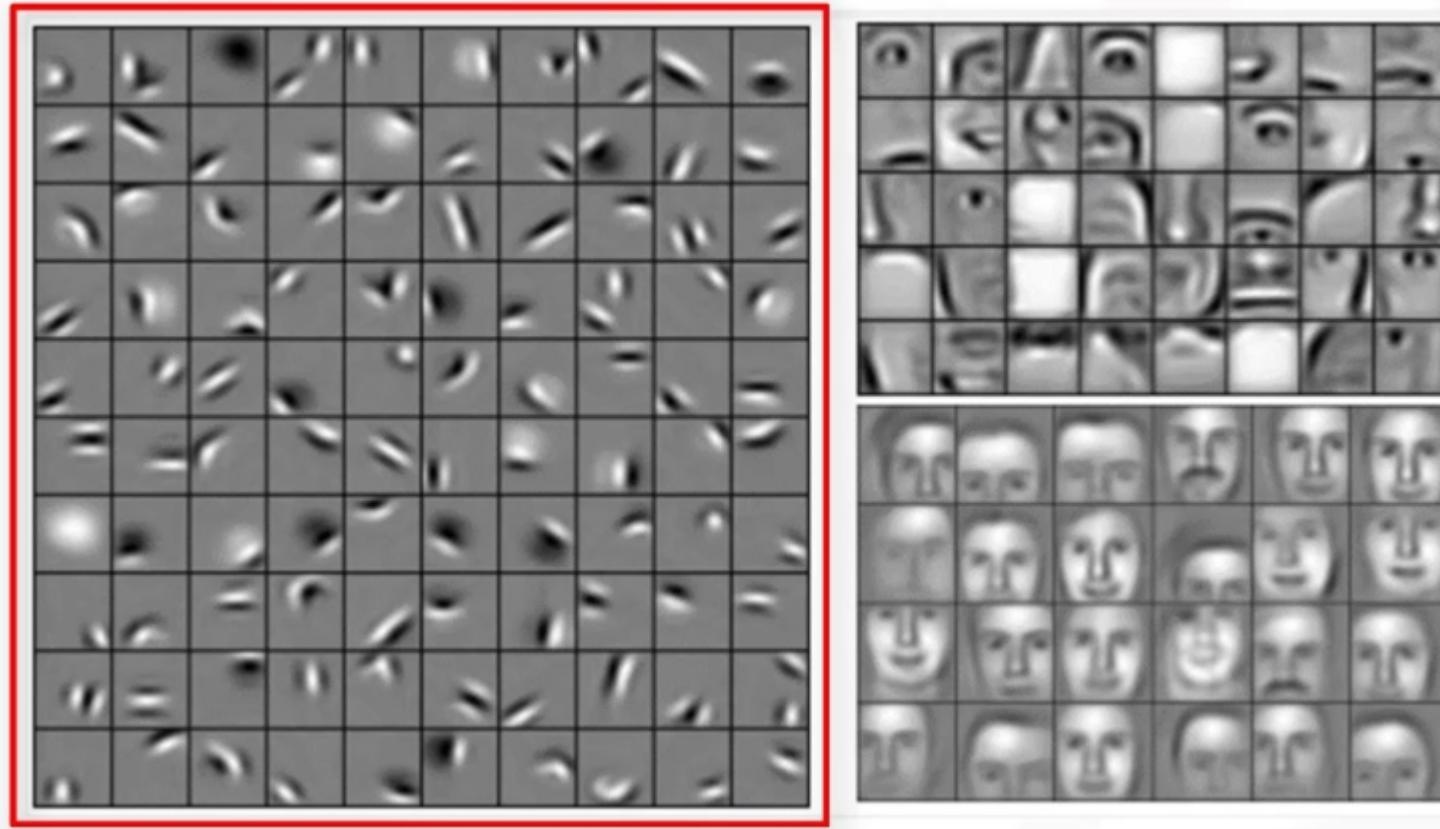


# CNN on Faces



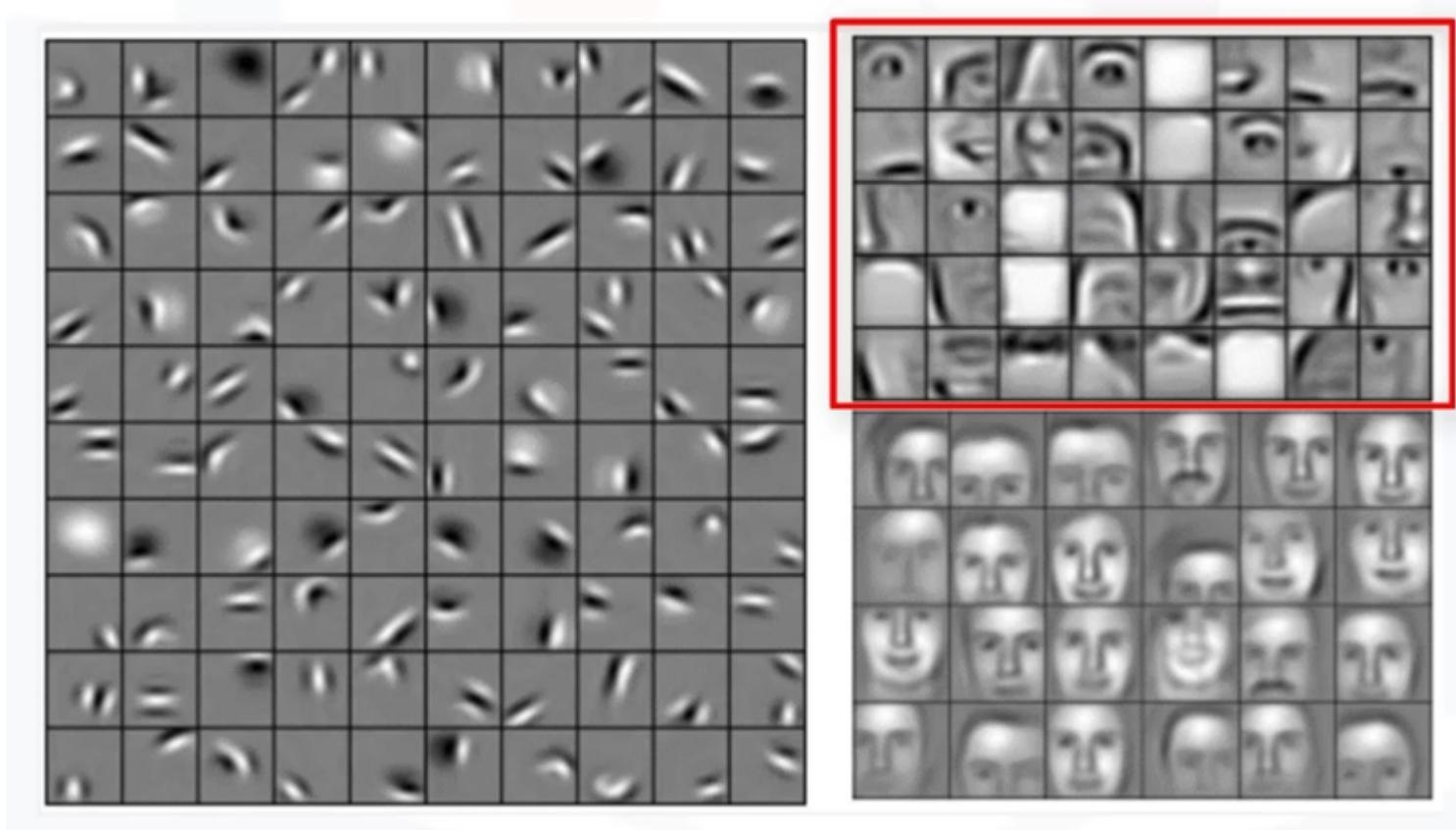


# CNN on Faces



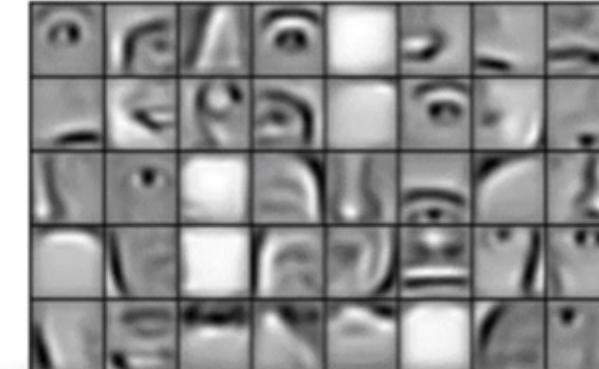
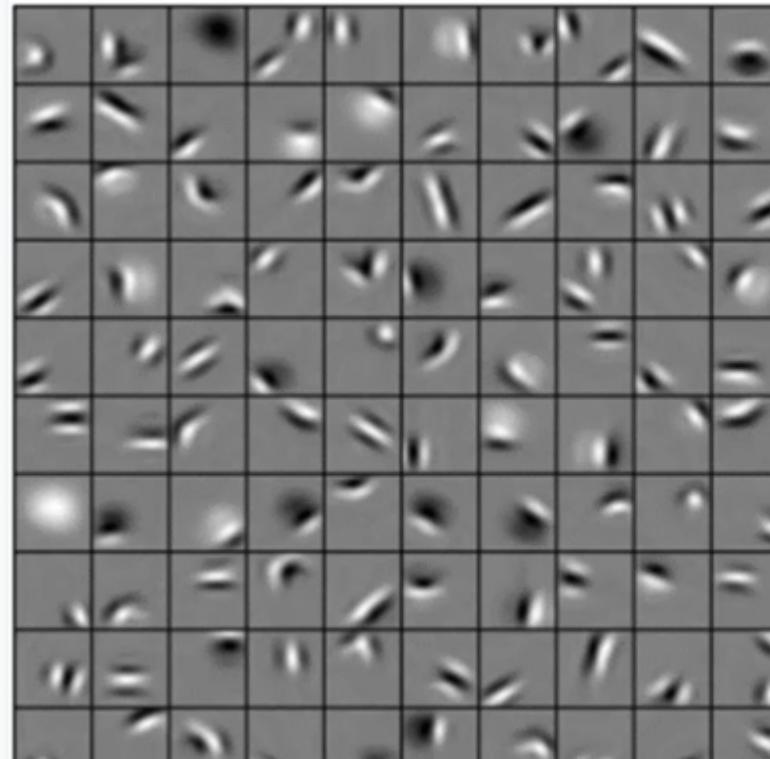


# CNN on Faces

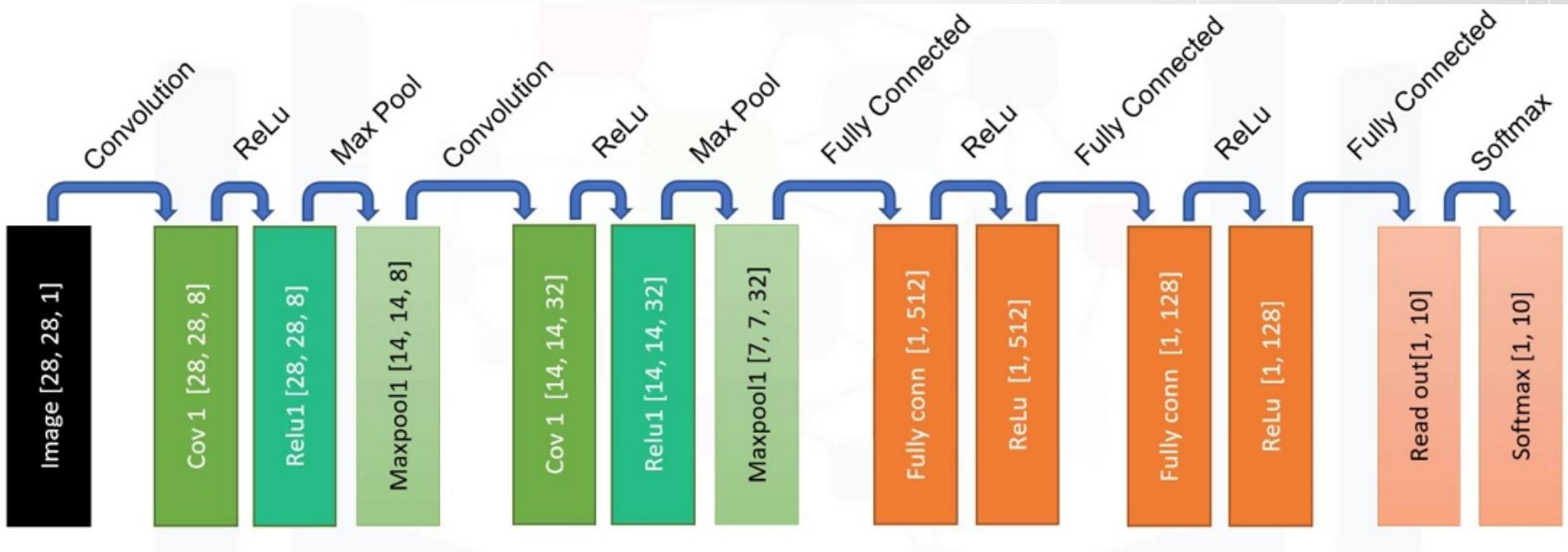




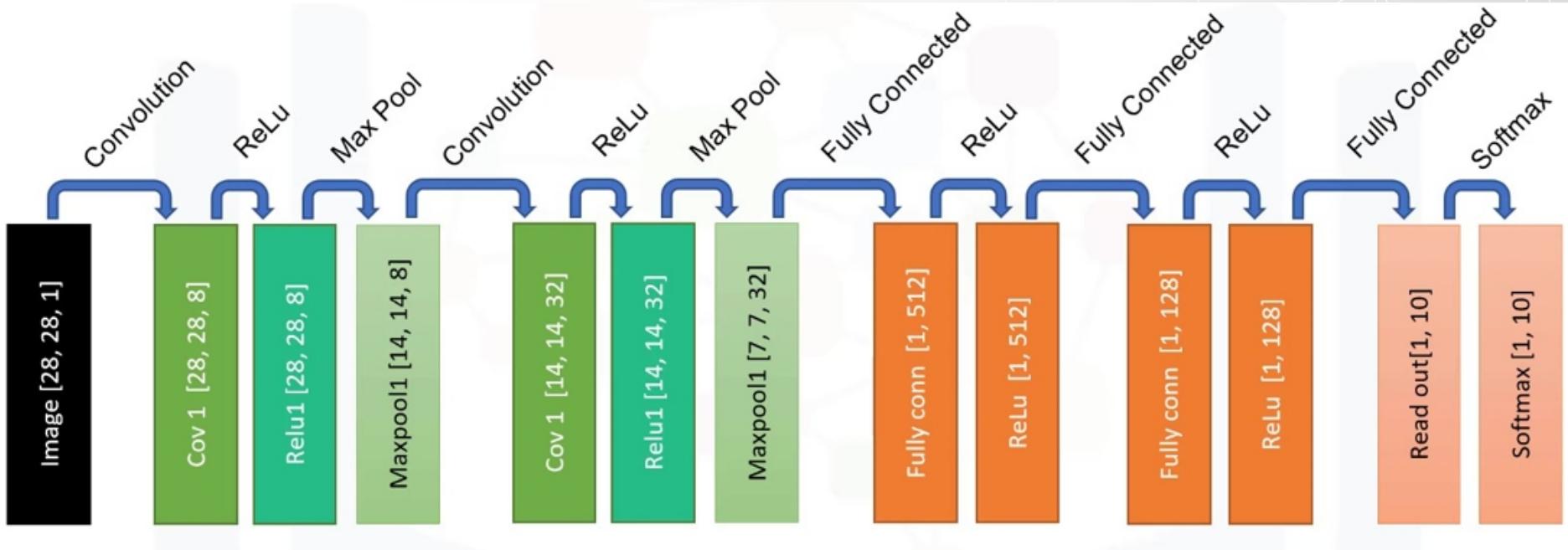
# CNN on Faces



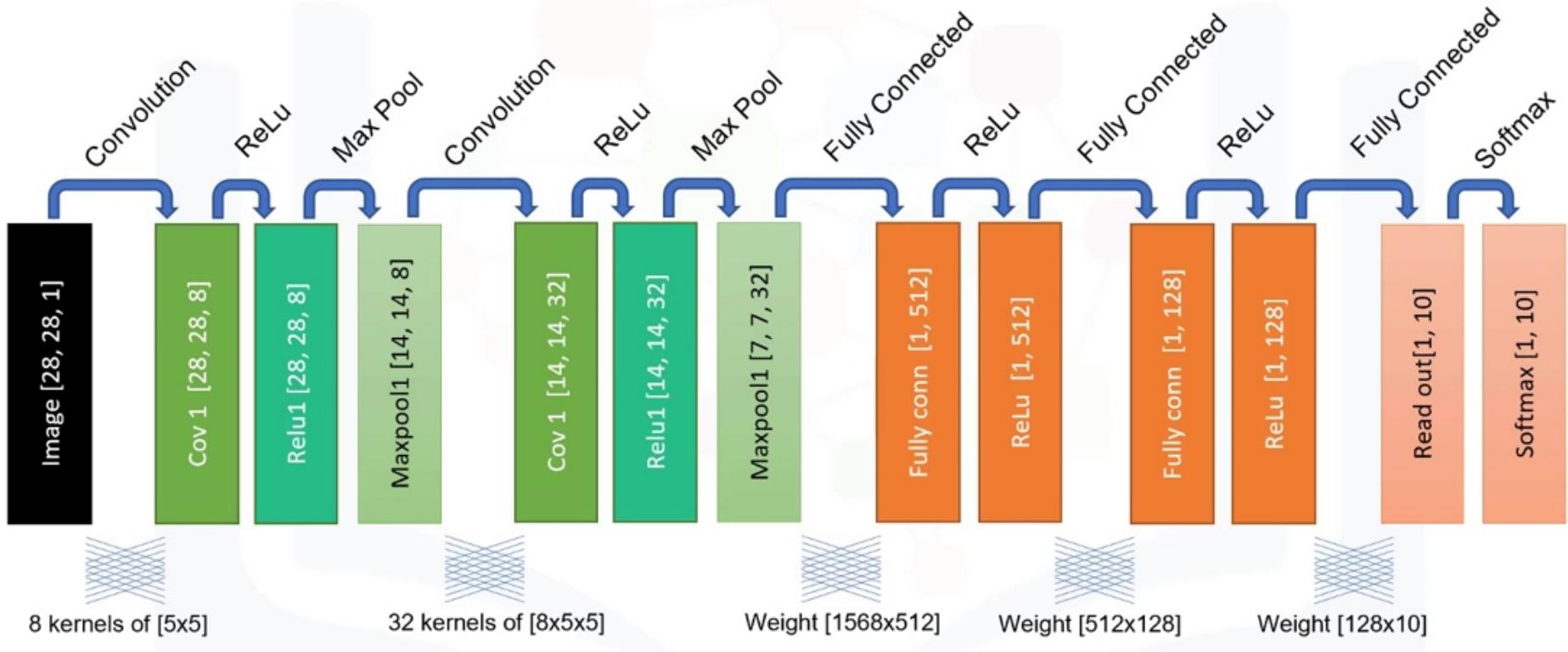
# CNN Training



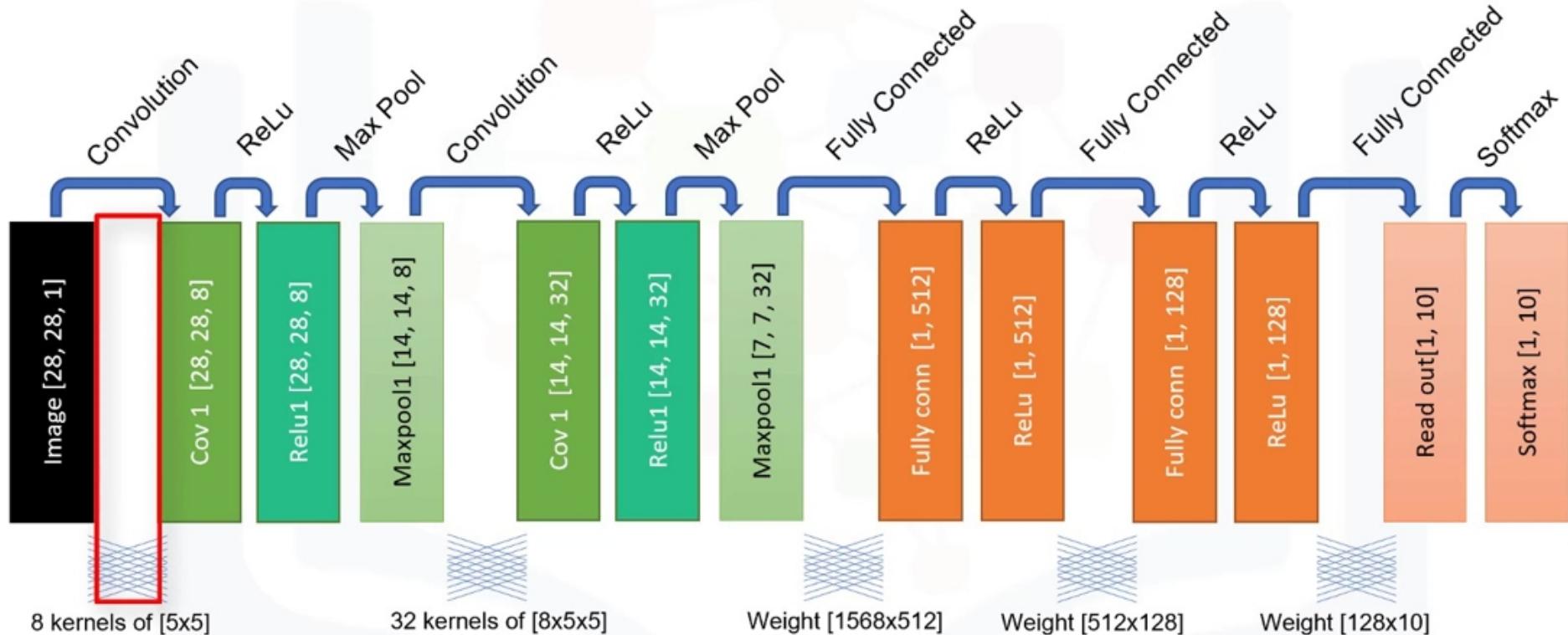
# CNN Training



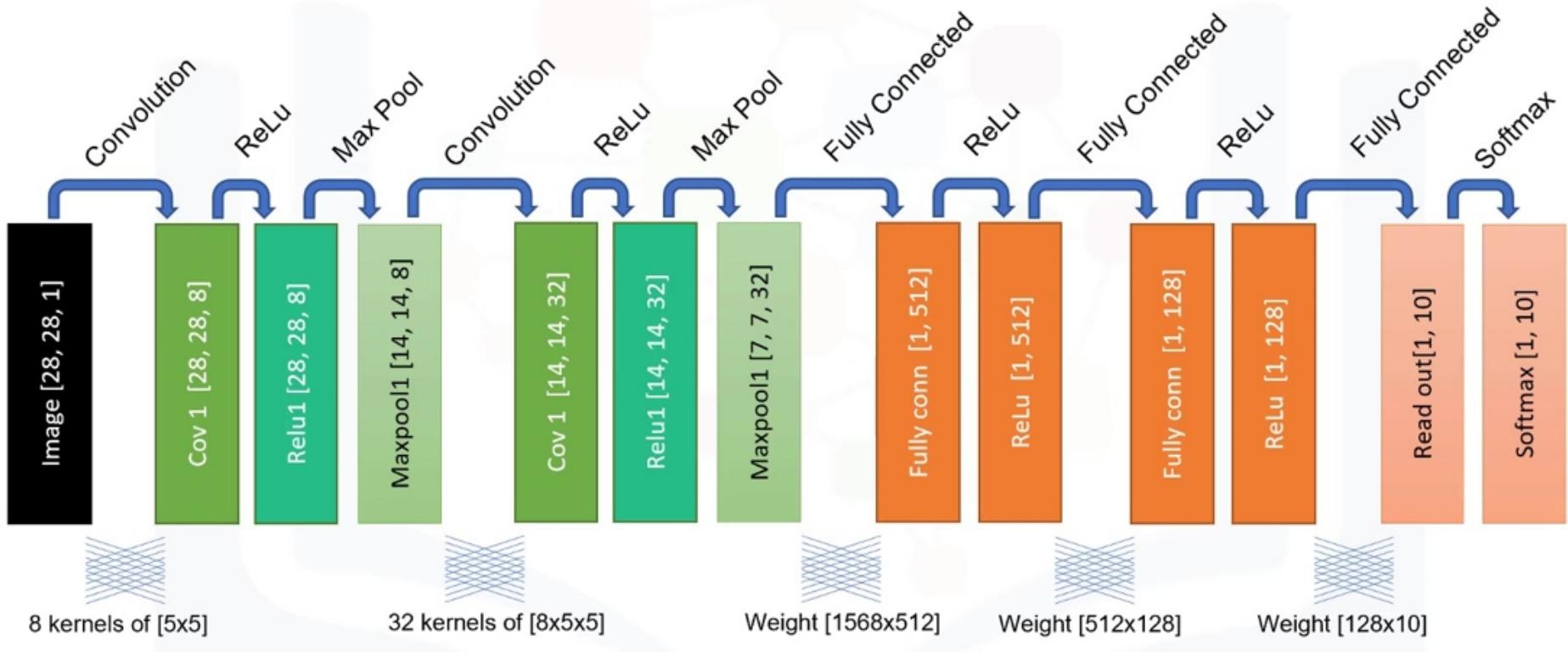
# CNN Training



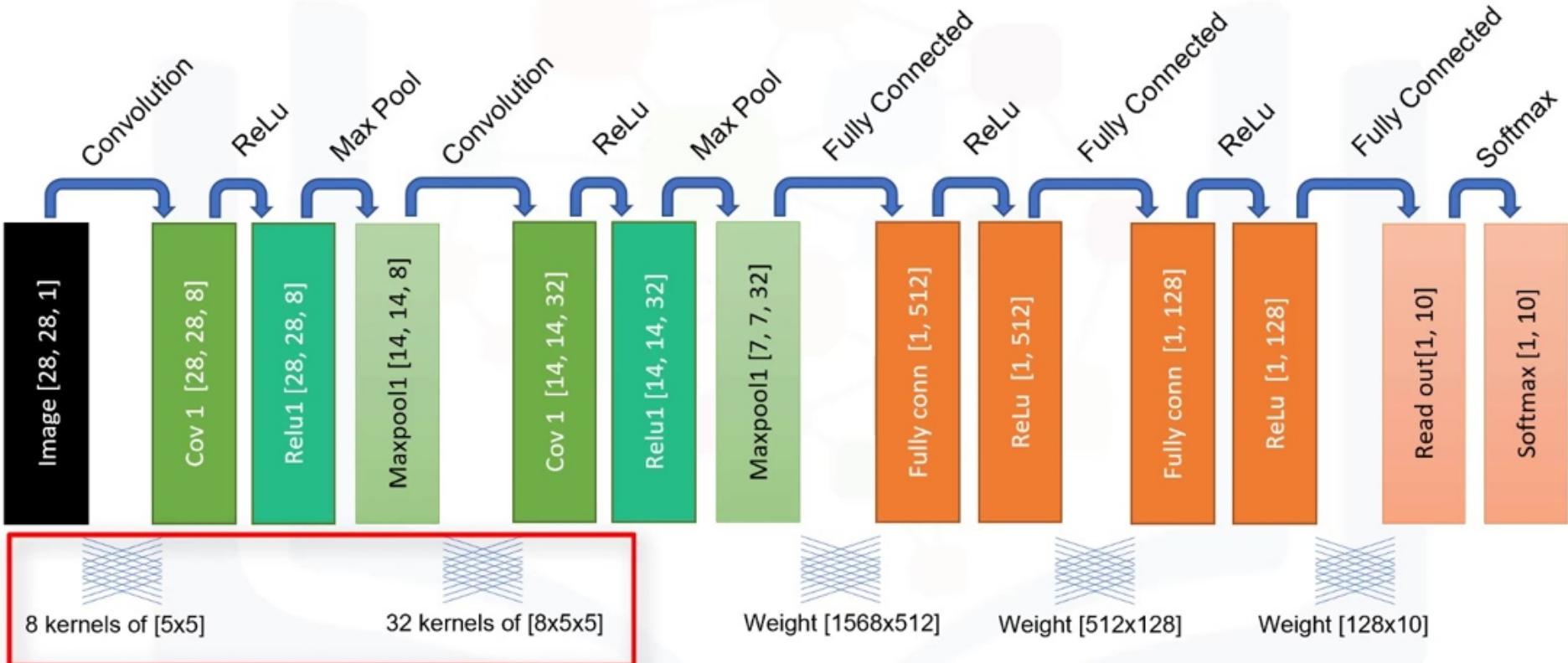
# CNN Training



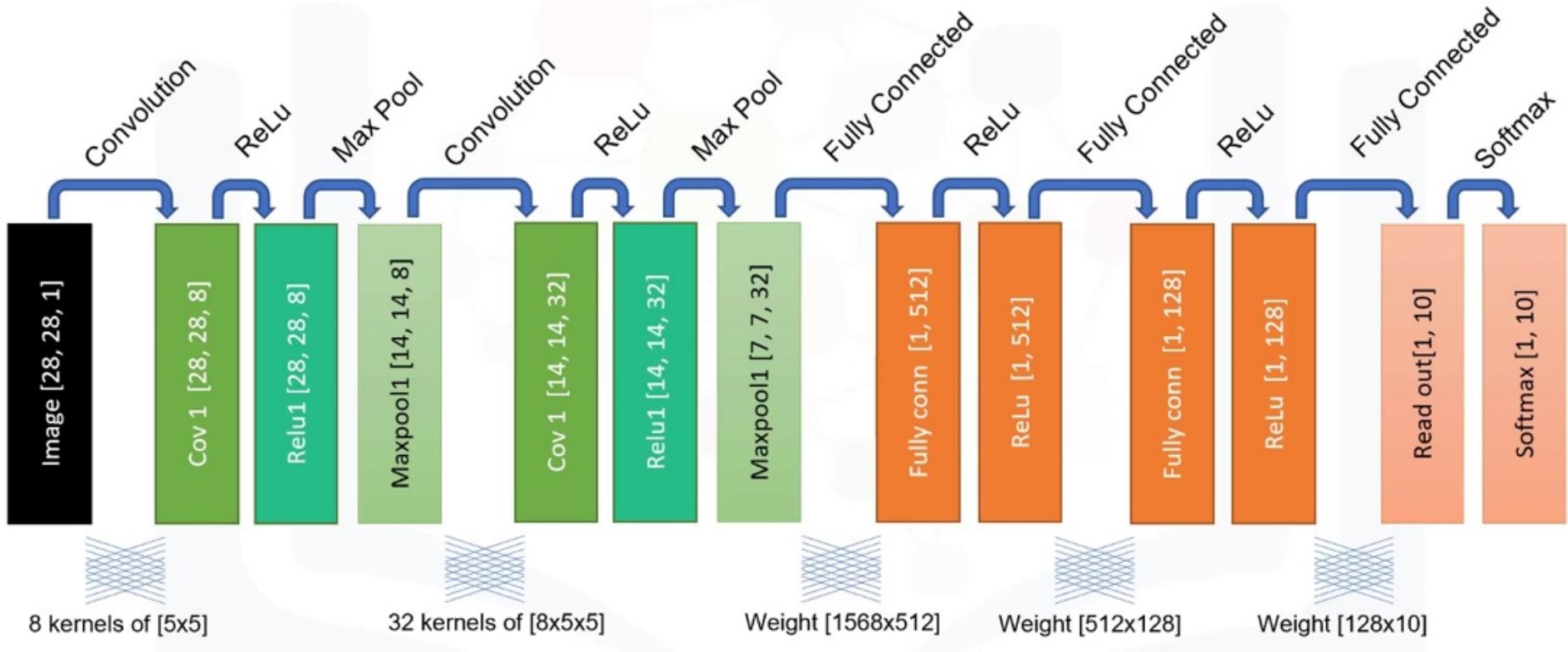
# CNN Training



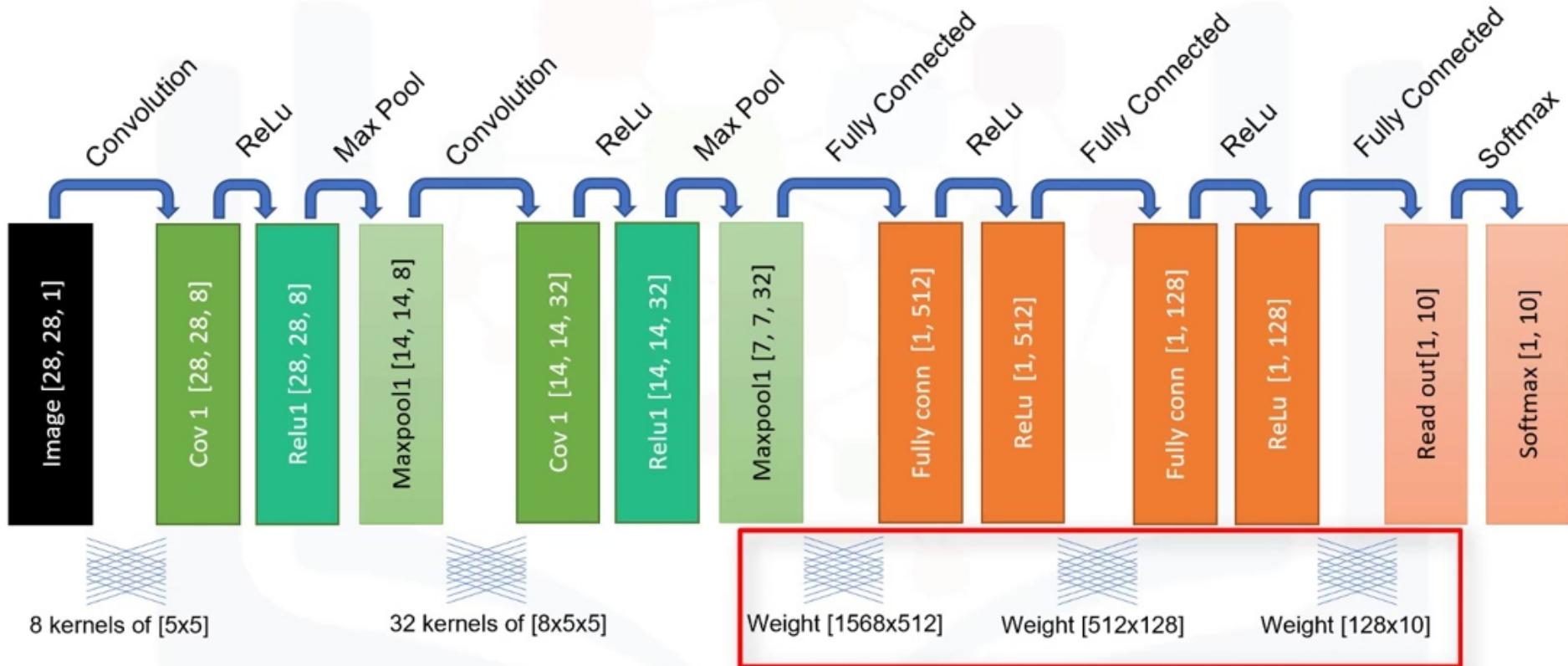
# CNN Training



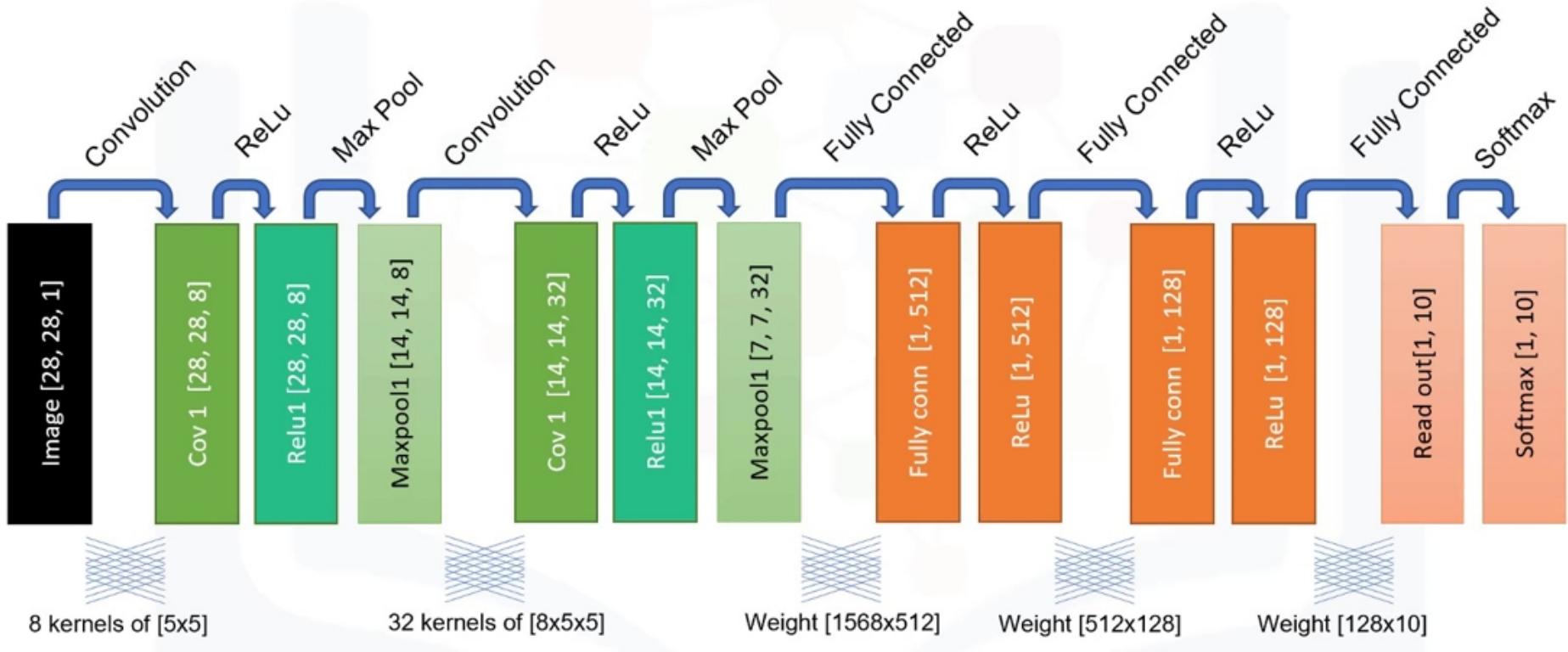
# CNN Training



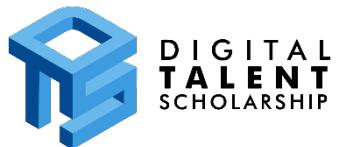
# CNN Training



# CNN Training



## IKUTI KAMI



DIGITAL  
TALENT  
SCHOLARSHIP

- [digitalentscholarship](https://www.facebook.com/digitalentscholarship)
- [digitalentscholarship](https://www.instagram.com/digitalentscholarship)
- [DTS\\_kominfo](https://twitter.com/DTS_kominfo)
- [Digital Talent Scholarship 2019](https://www.telegram.org/channels/DigitalTalentScholarship2019)

Pusat Pengembangan Profesi dan Sertifikasi  
Badan Penelitian dan Pengembangan SDM  
Kementerian Komunikasi dan Informatika  
Jl. Medan Merdeka Barat No. 9  
(Gd. Belakang Lt. 4 - 5)  
Jakarta Pusat, 10110



digitalentscholarship.go.id