# Assignment 10

## Cloth Classification using RNN and CNN

Computing Lab - II, Spring 2021

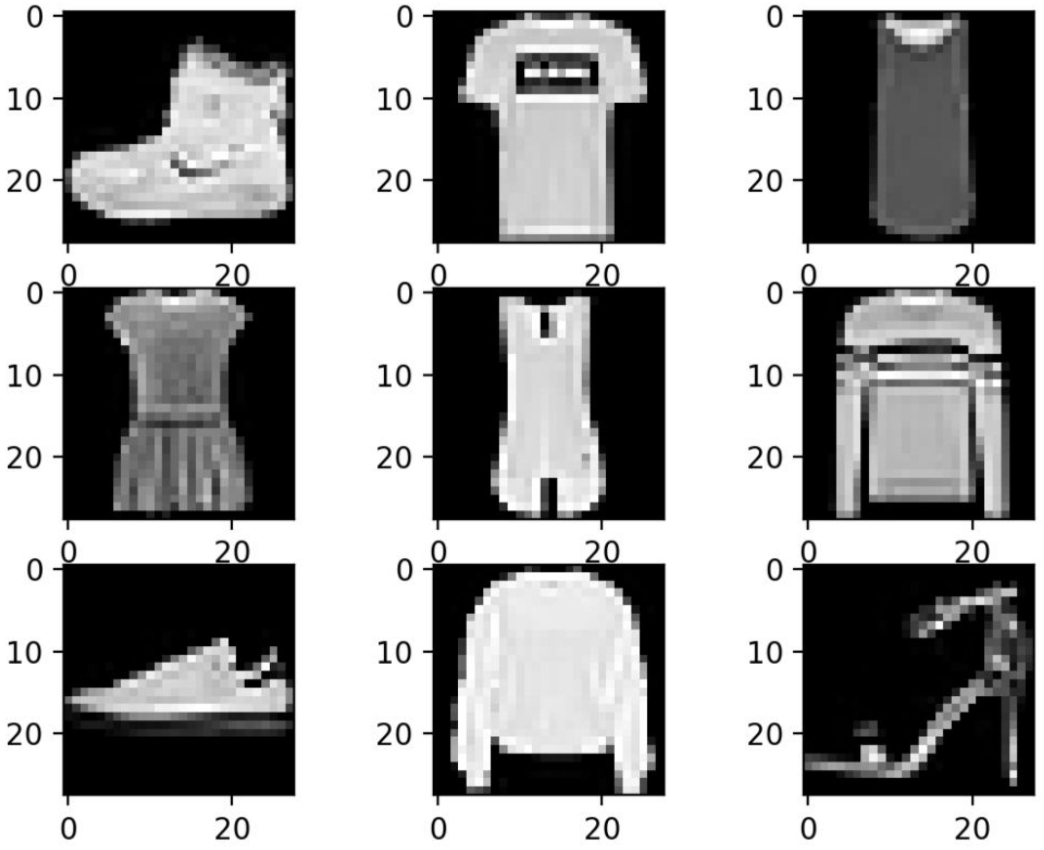
# Fashion MNIST Dataset

**60,000 small square 28×28 pixel grayscale images of items of 10 types of clothing, such as shoes, t-shirts, dresses, and more. The mapping of all 0-9 integers to class labels is listed below.**

## Task: Image Classification

- 0: T-shirt/top
- 1: Trouser
- 2: Pullover
- 3: Dress
- 4: Coat
- 5: Sandal
- 6: Shirt
- 7: Sneaker
- 8: Bag
- 9: Ankle boot

# Images



Plot of a Subset of Images From the Fashion-MNIST Dataset

# Tasks for the assignment

1. Use an RNN for the classification problem
2. Use a CNN for the same task

# Recurrent Neural Networks

Task 1

# Recurrent Neural Network

We can process a sequence of vectors x by applying a recurrence formula at each step:

$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

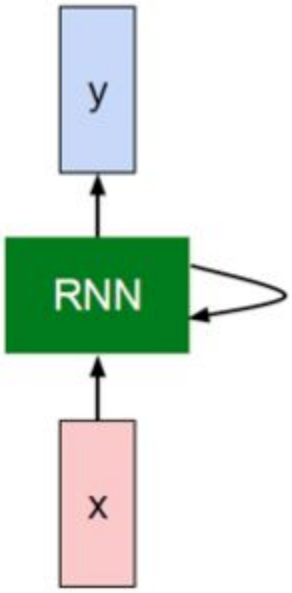
new state

some function
with parameters W

old state

input vector at
some time step

Notice: the same function and the same set of parameters are used at every time step.



# RNN - further details

# RNN update equations (forward pass)

*Update Equations*

Initial state - $h^{(0)}$

From $t = 1$ to $t = \tau$, the following update equation is applied:

$$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)}$$

$$h^{(t)} = tanh(a^{(t)})$$

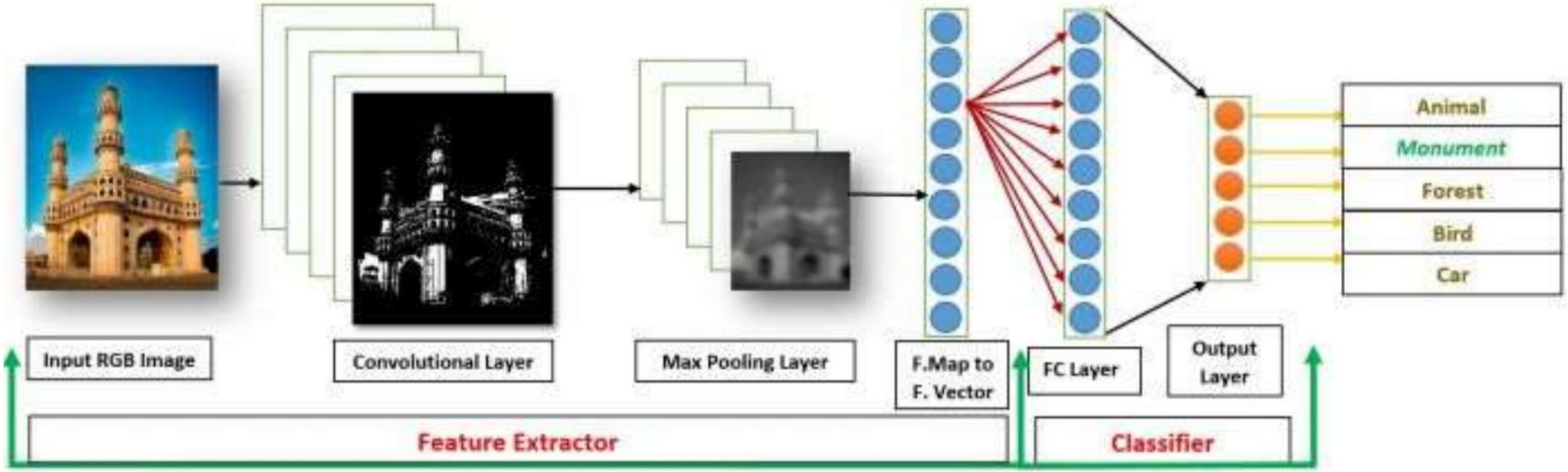
$$o^{(t)} = c + Vh^{(t)}$$

$$\hat{y}^{(t)} = softmax(o^{(t)})$$

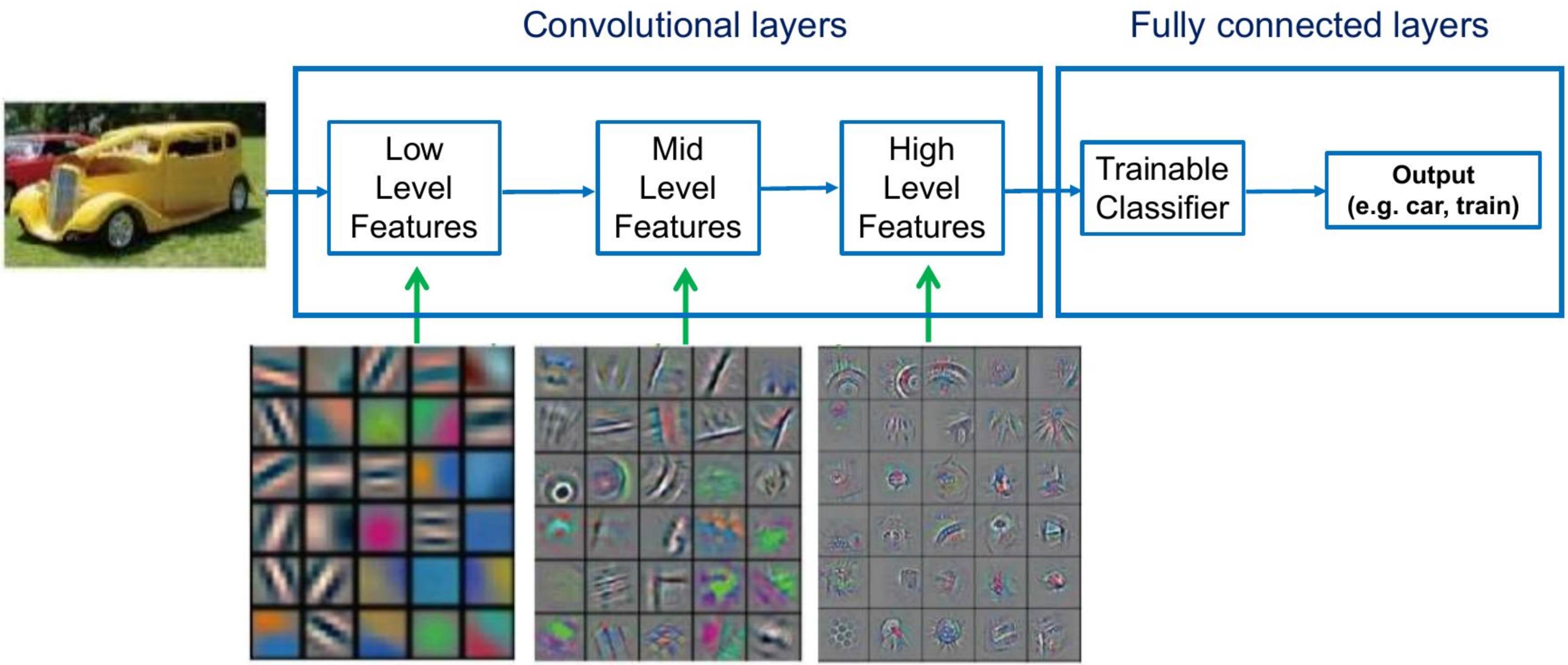
# Convolutional Neural Networks

# Introduction

Convolutional Neural Networks (**CNNs**) **learns multi-level features and classifier in a joint fashion** and performs much **better than traditional approaches** for various image classification and segmentation problems.



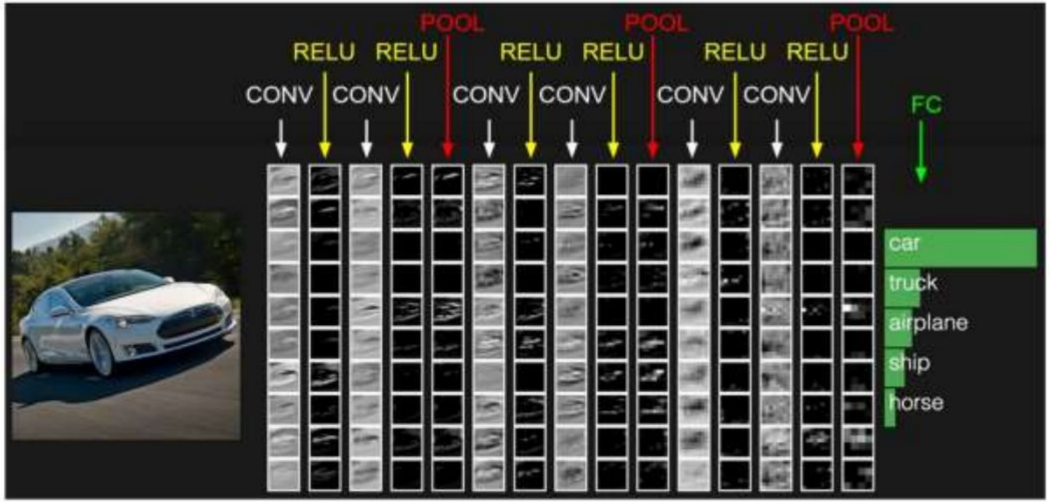
# CNN – What do they learn?



# CNN - Components

There are four main components in the CNN:

1. Convolution
2. Non-Linearity
3. Pooling or Sub Sampling
4. Classification (Fully Connected Layer)

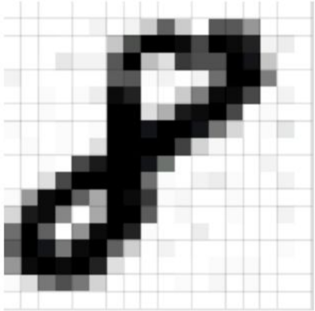


# Input

- An Image is a matrix of pixel values.

- If we consider a gray scale image, the value of each pixel in the matrix will range from 0 to 255.

- If we consider an RGB image, each pixel will have the combined values of R, G and B.

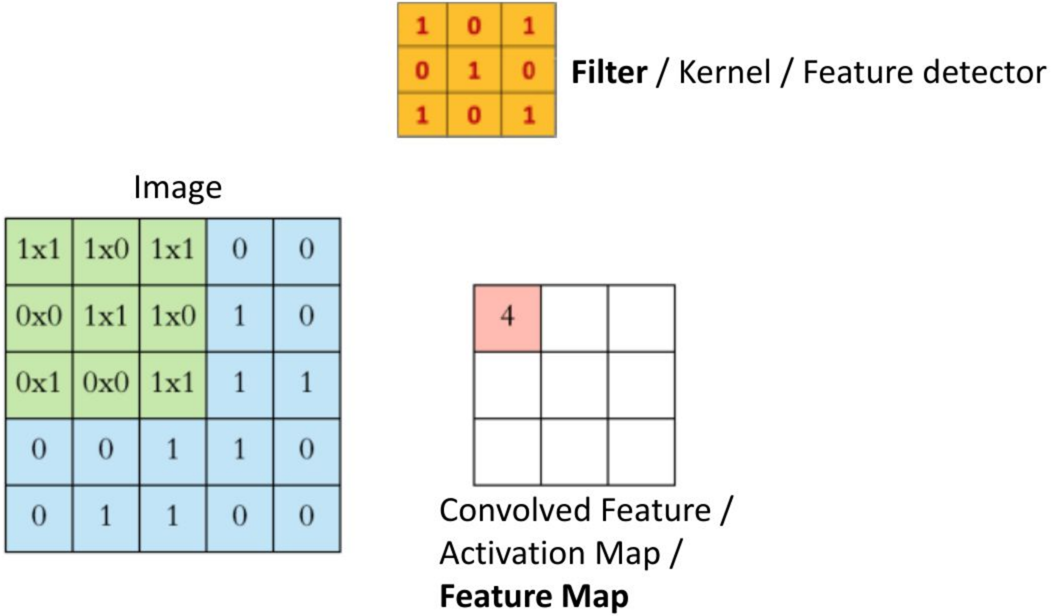


What We See

What Computers See

# Convolution

The primary purpose of Convolution in case of a CNN is to extract features from the input image.



# Convolution...

- The size of the output volume is controlled by three parameters that we need to decide before the convolution step is performed:

  ✓**Depth:** Depth corresponds to the number of filters we use for the convolution operation.

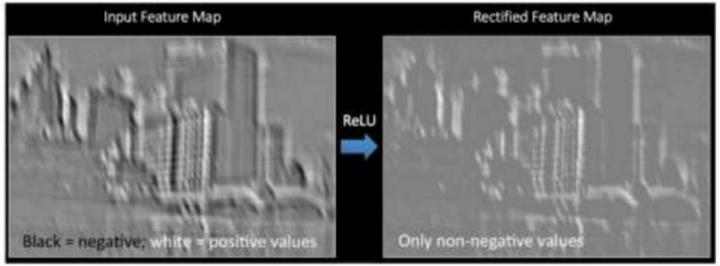
  ✓**Stride:** Stride is the number of pixels by which we slide our filter matrix over the input matrix.

  ✓**Zero-padding:** Sometimes, it is convenient to pad the input matrix with zeros around the border, so that we can apply the filter to bordering elements of our input image matrix.

    - With zero-padding *wide convolution*
    - Without zero-padding *narrow convolution*

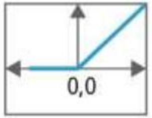
# Non-Linearity (ReLU)

- Replaces all negative pixel values in the feature map by zero.
- The purpose of ReLU is to introduce non-linearity in CNN, since most of the real-world data would be non-linear.
- Other non-linear functions such as tanh (-1,1) or sigmoid (0,1) can also be used instead of ReLU (0,input).




| | | | |
|---|---|---|---|
| 15 | 20 | -10 | 35 |
| 18 | -110 | 25 | 100 |
| 20 | -15 | 25 | -10 |
| 101 | 75 | 18 | 23 |

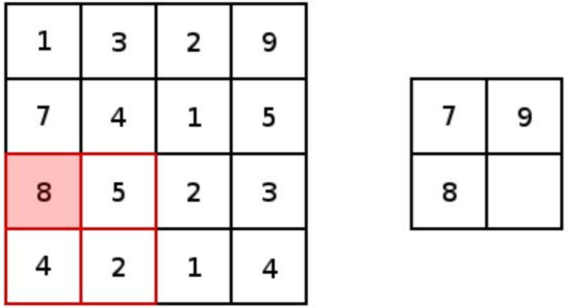
Transfer Function



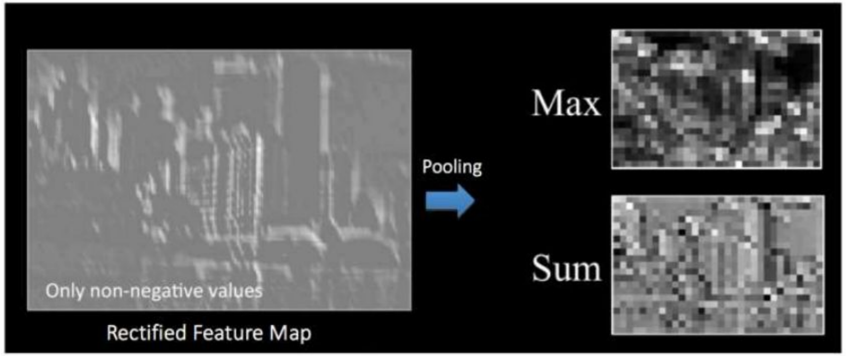

| | | | |
|---|---|---|---|
| 15 | 20 | 0 | 35 |
| 18 | 0 | 25 | 100 |
| 20 | 0 | 25 | 0 |
| 101 | 75 | 18 | 23 |

# Pooling

Reduces the dimensionality of each feature map but retains the most important information. Pooling can be of different types: Max, Average, Sum etc.

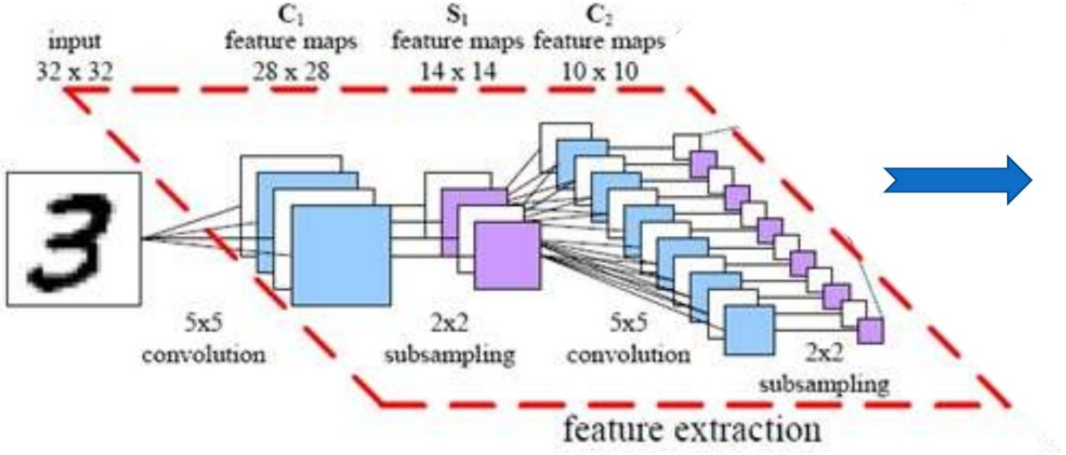



2×2 region



# Story so far

- Together these layers extract the useful features from the images.
- The output from the convolutional and pooling layers represent high-level features of the input image.

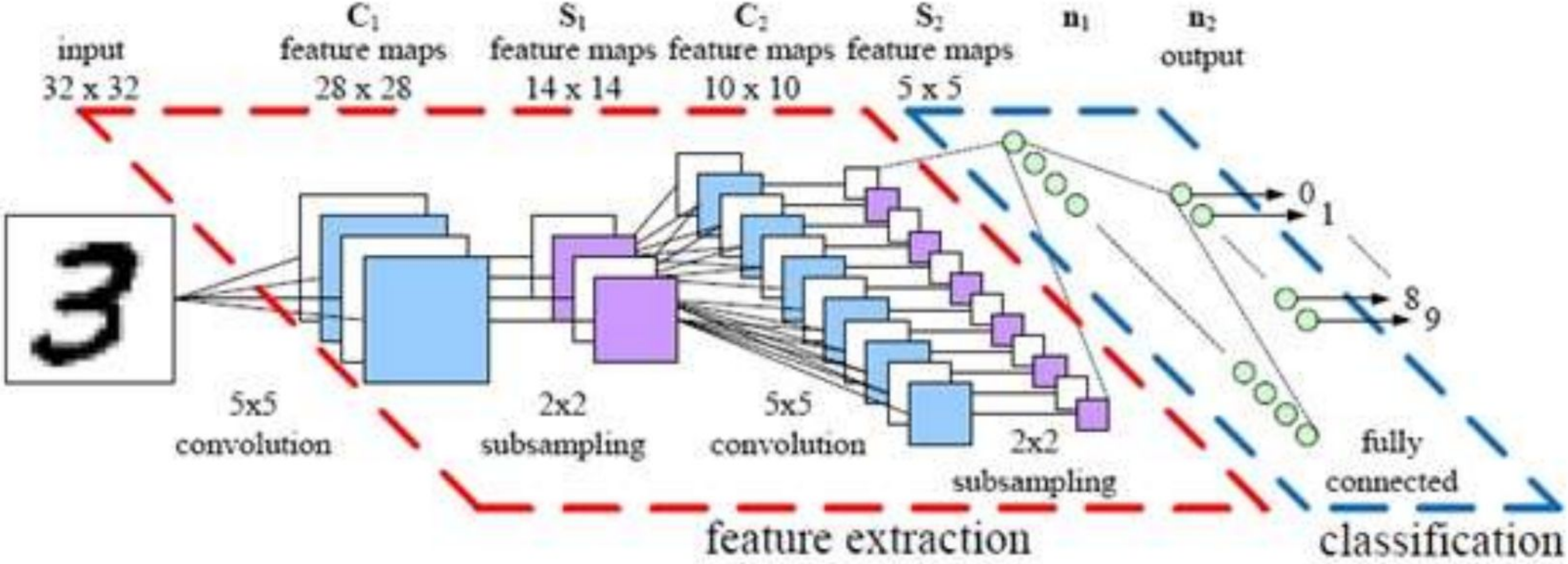



*High-level features*

# Fully Connected Layer

- A traditional Multi-Layer Perceptron.
- The term “Fully Connected” implies that every neuron in the previous layer is connected to every neuron on the next layer.
- Their activations can hence be computed with a matrix multiplication followed by a bias offset.
- The purpose of the Fully Connected layer is to use the high-level features for classifying the input image into various classes based on the training dataset.

# Overall CNN Architecture



**Summary**. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters $K$,
  - their spatial extent $F$,
  - the stride $S$,
  - the amount of zero padding $P$.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and $K$ biases.
- In the output volume, the $d$-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the $d$-th filter over the input volume with a stride of $S$, and then offset by $d$-th bias.