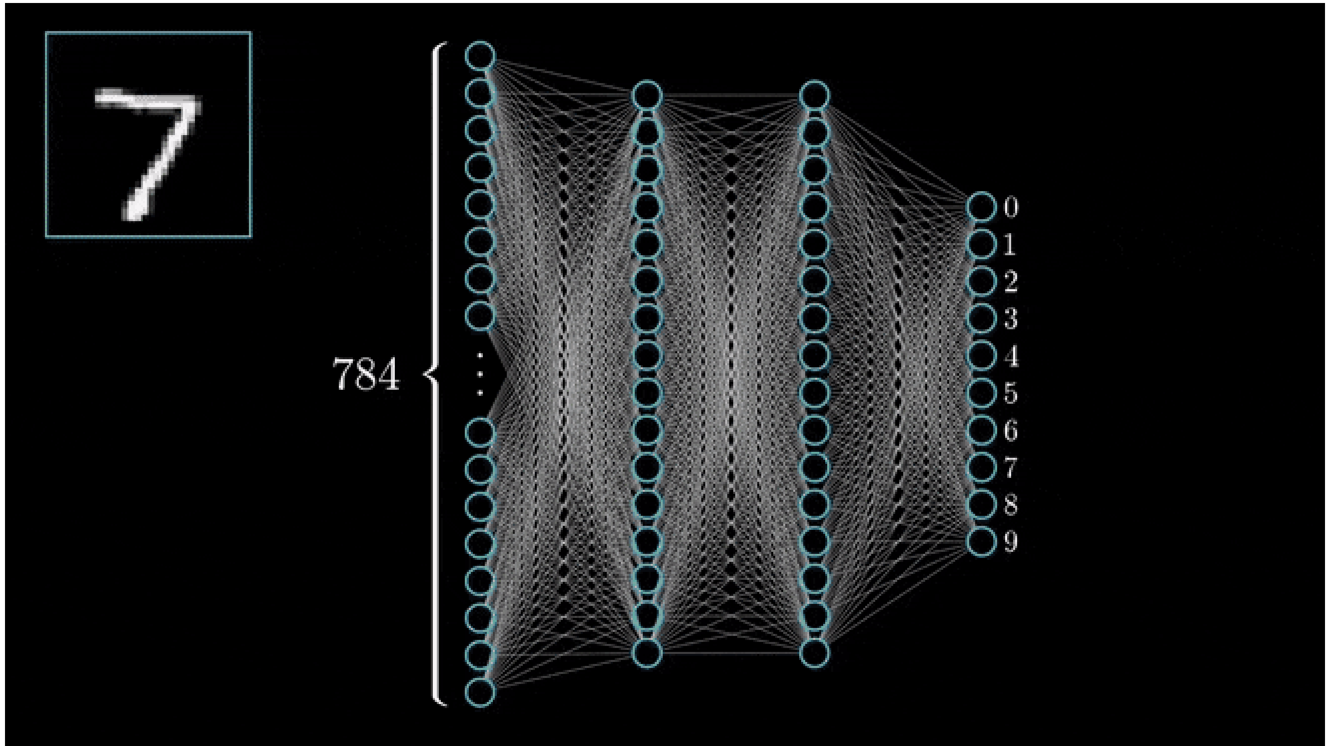
WORKSHOP
INTRODUCTION TO DEEP LEARNING

Tue Vu, PhD
Research & Data Science Services
SMU OIT

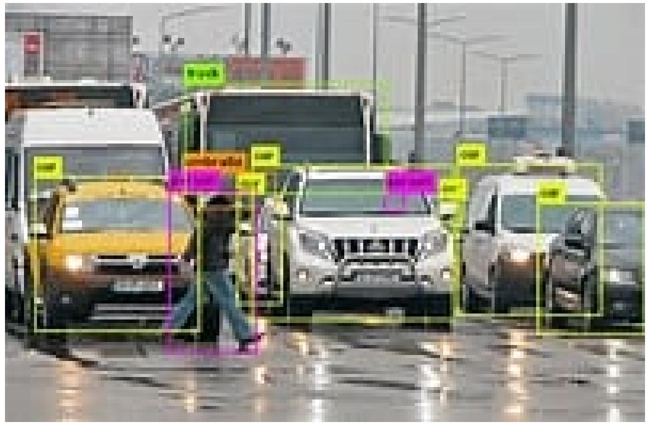
Outline

1. Convolution Neural Network
2. Recurrent Neural Network
3. Long-Short Term Memory



1. Convolution Neural Network

- CNNs are one type of ANN which utilize the neuron, kernel, activation function.
- Inputs must be in images (or assumed to be images in 2D format)
- Using Forward & Backpropagation technique with certain property to process it faster
- CNNs best for object detection, image classification, computer vision

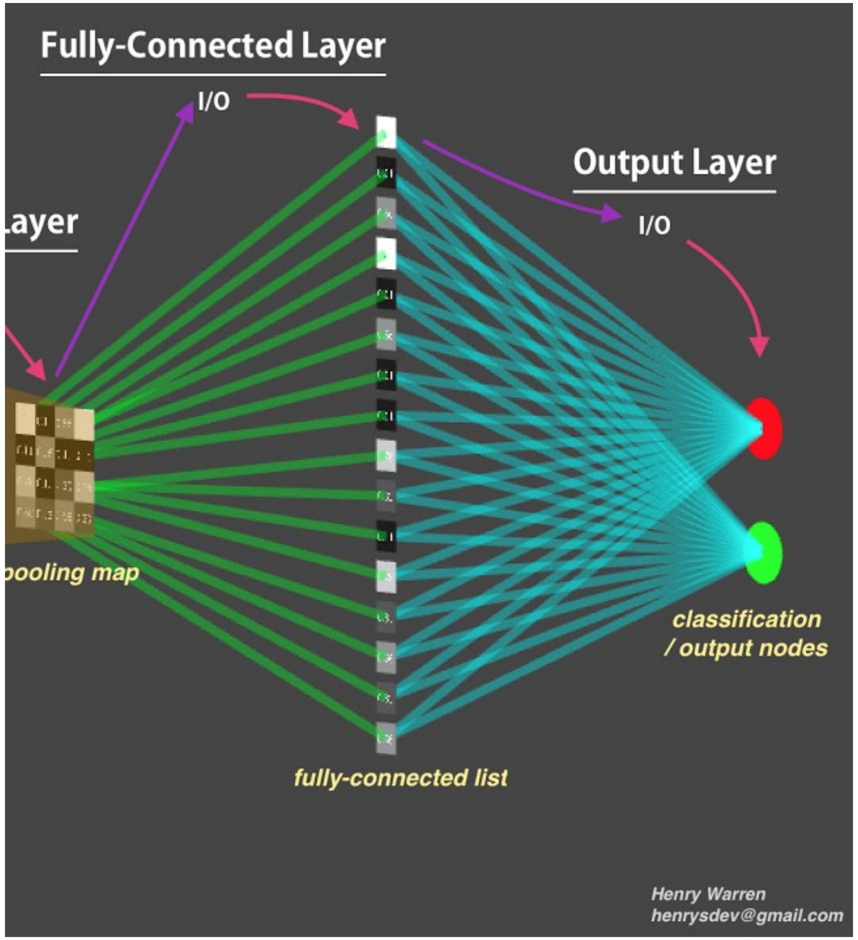


1. Convolution Neural Network

Architecture of MLPs

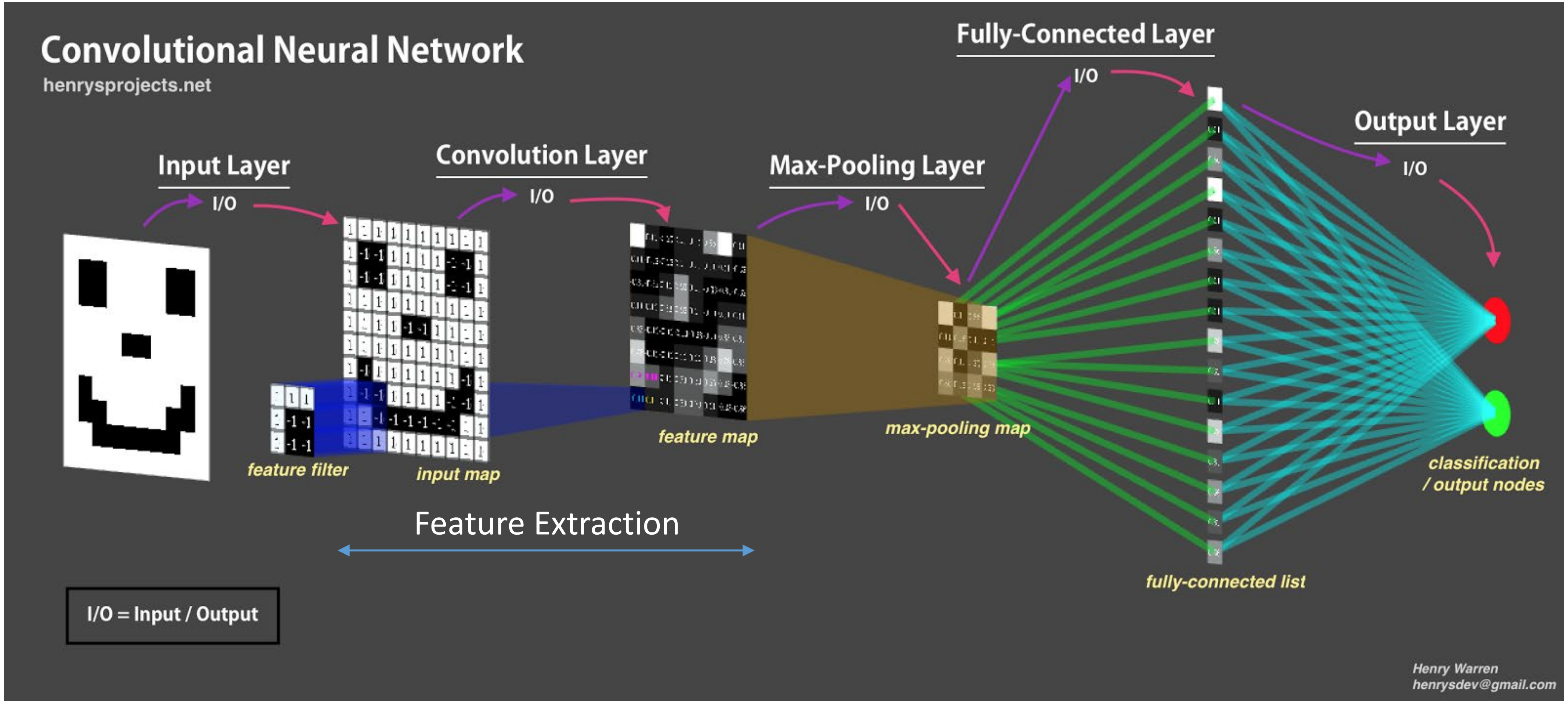
- Previous examples (MNIST, Fashion MNIST) use fully connected MLP NN to predict the images
- The accuracy/loss are ok but not so great

Fully connected (Dense) Layer



1. Convolution Neural Network

Architecture of CNNs



1. Convolution Neural Network

Architecture of CNNs

- Convolutional Layers
- Pooling Layers
- Flatten Layer

1. Convolution Neural Network

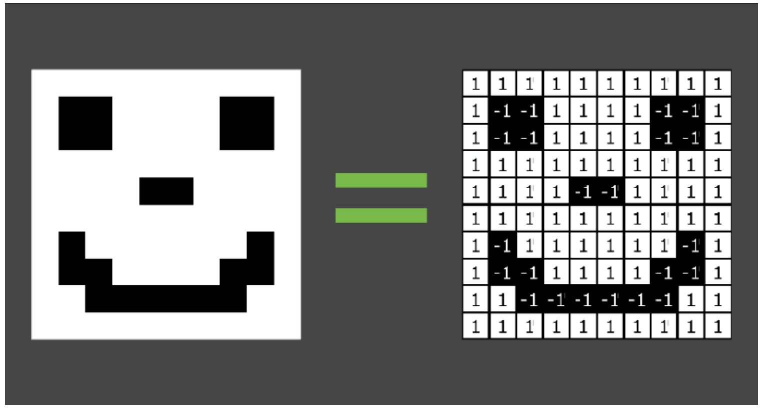
Hyper-parameters of Convolutional Layers (Conv2D):

- Depth
- Filter/kernel
- Stride
- Padding

1. Convolution Neural Network

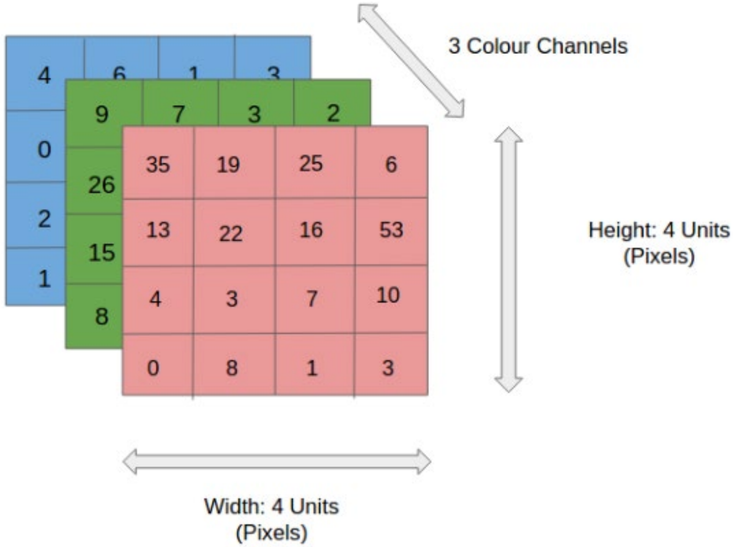
Hyper-Parameters of Conv2D: depth

Depth = 1



MNIST
Fashion MNIST

Depth = 3
(RGB)



Regular images

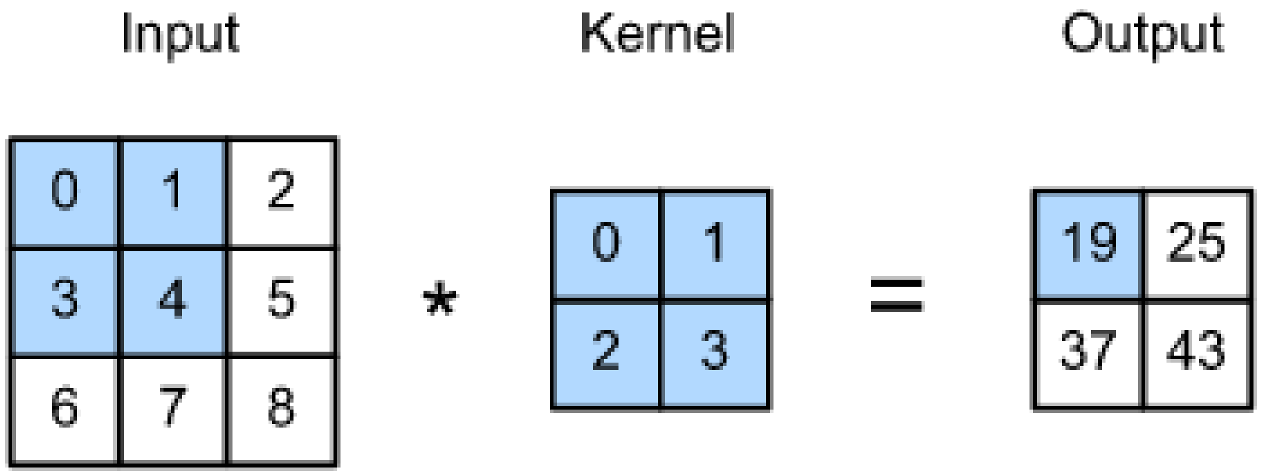
1. Convolution Neural Network

Hyper-Parameters of Conv2D: filter & kernel

1. Convolution Neural Network

Hyper-Parameters of Conv2D: filter & kernel

- dot product



1. Convolution Neural Network

Hyper-Parameters of Conv2D : filter & kernel

Kernel size (3,3)

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Blur filter

1. Convolution Neural Network

Hyper-Parameters of Conv2D : filter & kernel

Kernel size (3,3)

0	-1	0
-1	5	-1
0	-1	0

Sharp filter

1. Convolution Neural Network

Hyper-Parameters of Conv2D : filter & kernel

Kernel size (3,3)

-1	-2	-1
0	0	0
1	2	1

Horizontal

-1	0	1
-2	0	2
-1	0	1

Vertical

Edge detection

1. Convolution Neural Network

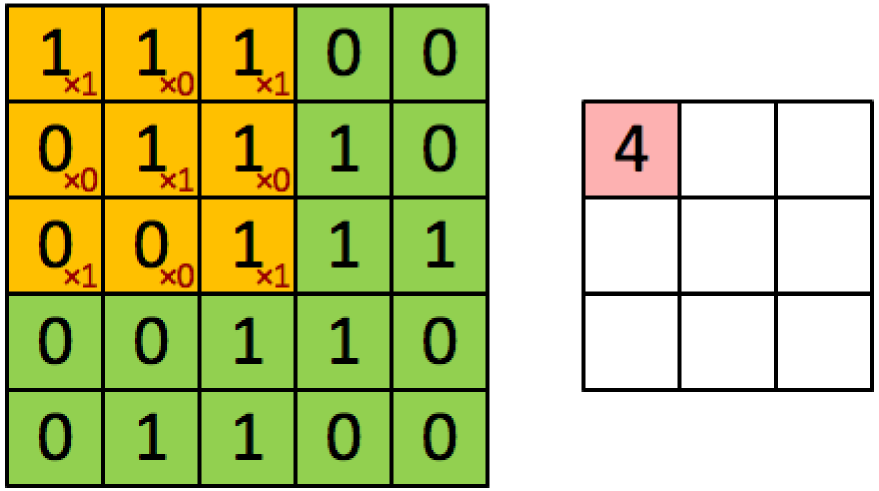
Hyper-Parameters of Conv2D : filter & kernel

Convolved Feature with filter

- CNN uses the Convolved Feature to reduce the image size by dot product with given kernel (filter)

- The image reduction without losing features and easier to process for good prediction

- In CNNs, filters are not defined. The value of each filter is learned during the training process.

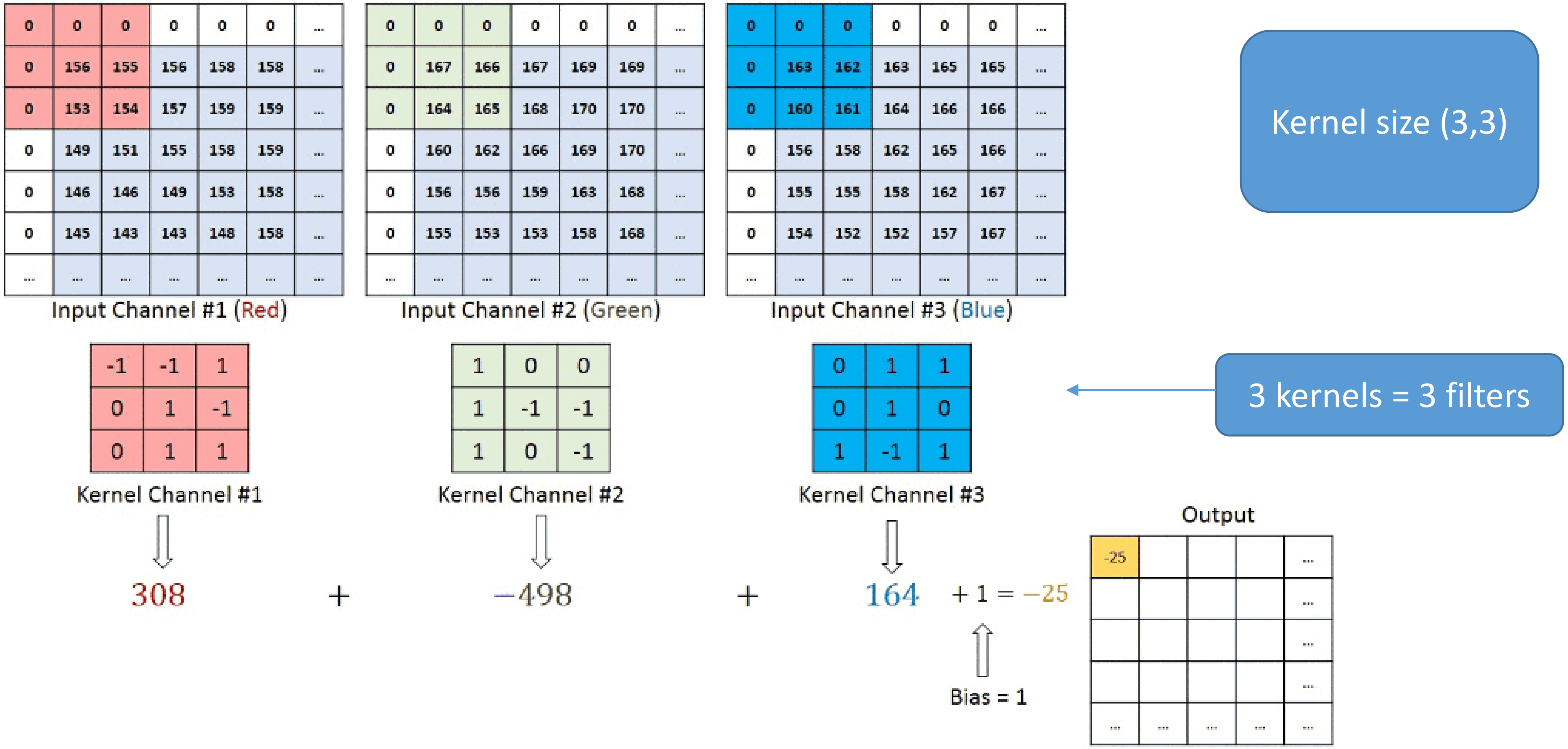


Image

Convolved Feature

1. Convolution Neural Network

Hyper-Parameters of Conv2D : filter & kernel

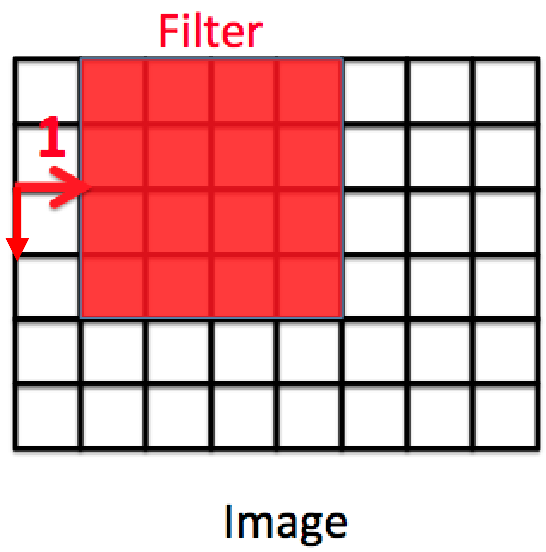


1. Convolution Neural Network

Hyper-Parameters of Conv2D : stride

- Stride tuned for the compression of images and video data

Stride = (1,1)

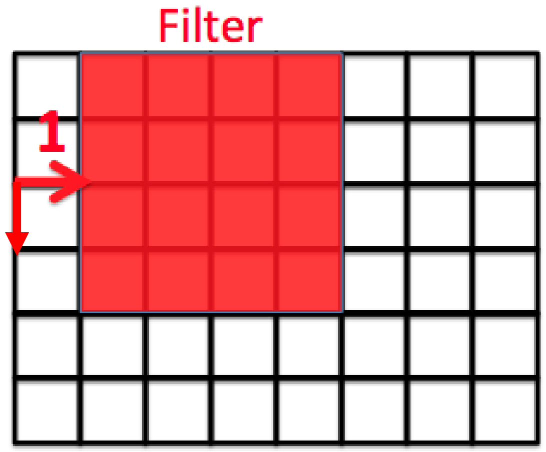


1. Convolution Neural Network

Hyper-Parameters of Conv2D : stride

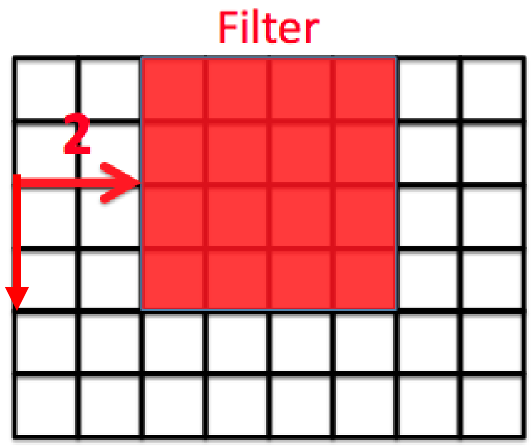
- Stride tuned for the compression of images and video data

Stride = (1,1)



Image

Stride = (2,2)



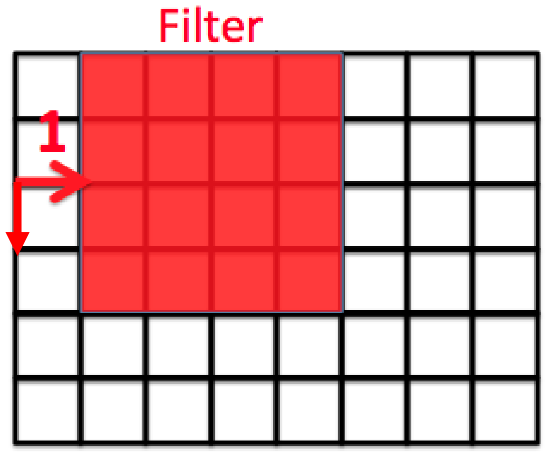
Image

1. Convolution Neural Network

Hyper-Parameters of Conv2D : stride

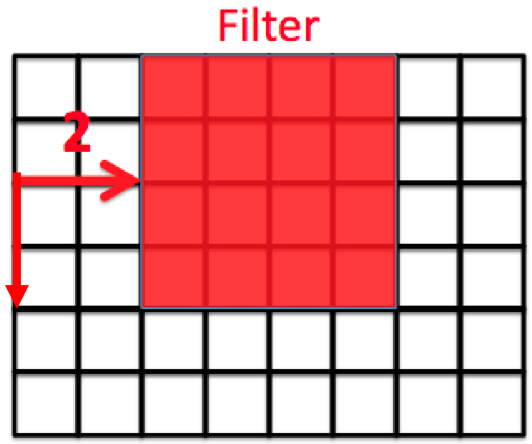
- Stride tuned for the compression of images and video data

Stride = (1,1)



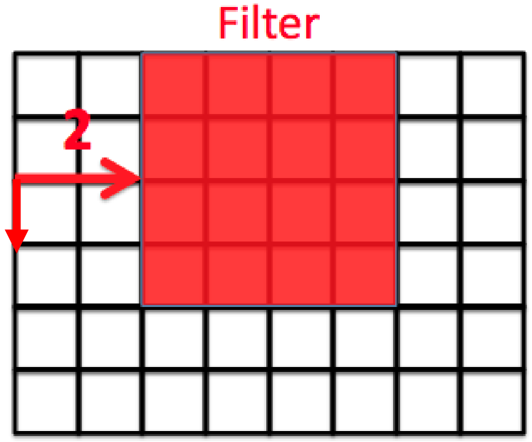
Image

Stride = (2,2)



Image

Stride = (2,1) or (1,2)?
Yes but not common

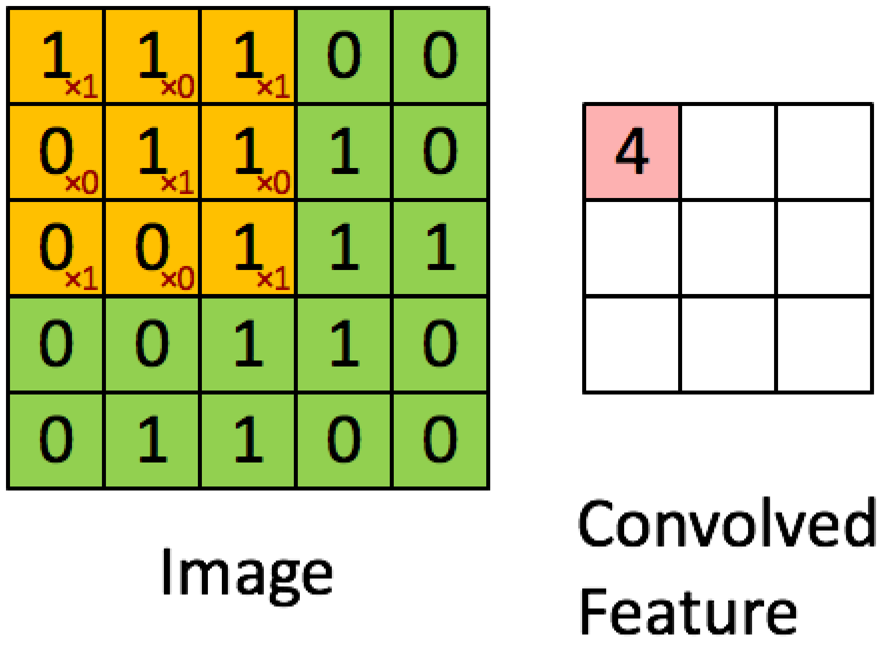


Image

1. Convolution Neural Network

Hyper-Parameters of Conv2D : padding

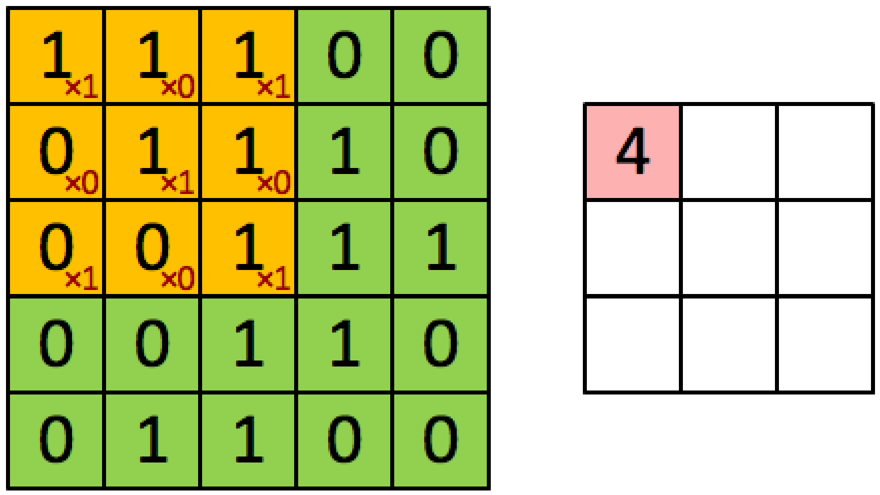
- The pixels located on the corners and the edges are used much less than those in the middle => the information on borders and edges are note preserved



1. Convolution Neural Network

Hyper-Parameters of Conv2D : padding

- The pixels located on the corners and the edges are used much less than those in the middle => the information on borders and edges are note preserved



Image

Convolved Feature

Solution?

1. Convolution Neural Network

Hyper-Parameters of Conv2D : padding

- The pixels located on the corners and the edges are used much less than those in the middle => the information on borders and edges are not preserved

3	5	9	1	10
13	2	4	6	11
16	24	9	13	1
7	1	6	8	3
8	4	9	1	9

(W, H)

padding

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	3	5	9	1	10	0	0
0	0	13	2	4	6	11	0	0
0	0	16	24	9	13	1	0	0
0	0	7	1	6	8	3	0	0
0	0	8	4	9	1	9	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

(W + 2P, H + 2P)

Solution?

- Add rows and columns of 0 to the input images
- The image on left was added with padding parameter P=2

1. Convolution Neural Network

Hyper-Parameters of Conv2D : padding

- The pixels located on the corners and the edges are used much less than those in the middle => the information on borders and edges are not preserved

3	5	9	1	10
13	2	4	6	11
16	24	9	13	1
7	1	6	8	3
8	4	9	1	9

(W, H)

padding

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	3	5	9	1	10	0	0
0	0	13	2	4	6	11	0	0
0	0	16	24	9	13	1	0	0
0	0	7	1	6	8	3	0	0
0	0	8	4	9	1	9	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

(W + 2P, H + 2P)

In Keras

- padding = "valid": no padding
- padding = "same": padding with 0 evenly left/right, up/down
- padding = "same" with strides = (1,1): output has same size as input

1. Convolution Neural Network

Hyper-Parameters of Conv2D :

- Depth (L): 3 or 1
- Number of Filter: F
- kernel: (K, K): (3,3) or (5,5)
- Stride (S, S): (1,1) or (2,2)
- Padding (P)

Formulation to compute the output size of a convolutional layer from an image with size (W, H)?

W

H

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...	...	...	...	...	...	...

1. Convolution Neural Network

Hyper-Parameters of Conv2D :

- Depth (L): 3 or 1
- Number of Filter: F
- kernel: (K, K): (3,3) or (5,5)
- Stride (S, S): (1,1) or (2,2)
- Padding (P)

Formulation to compute the output size of a convolutional layer from an image with size (W, H)?

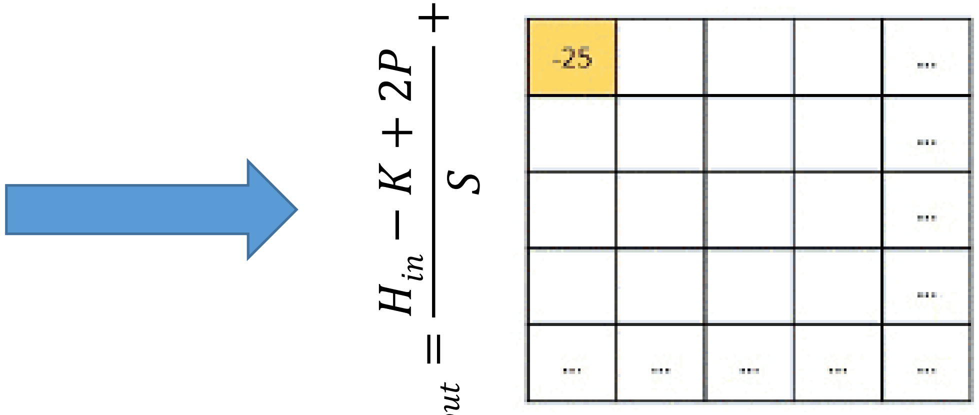
W_{in}

H_{in}

156	155	156	158	158
153	154	157	159	159
149	151	155	158	159
146	146	149	153	158
145	143	143	148	158

$$W_{out} = \frac{W_{in} - K + 2P}{S} + 1$$

$$H_{out} = \frac{H_{in} - K + 2P}{S} + 1$$

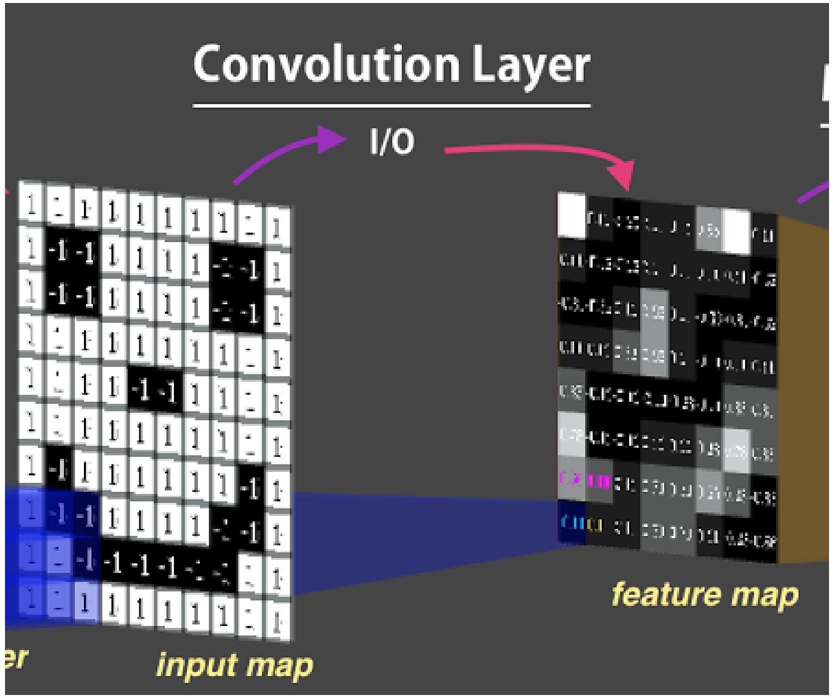


Convoluted features

1. Convolution Neural Network

How to add Conv2D in keras?

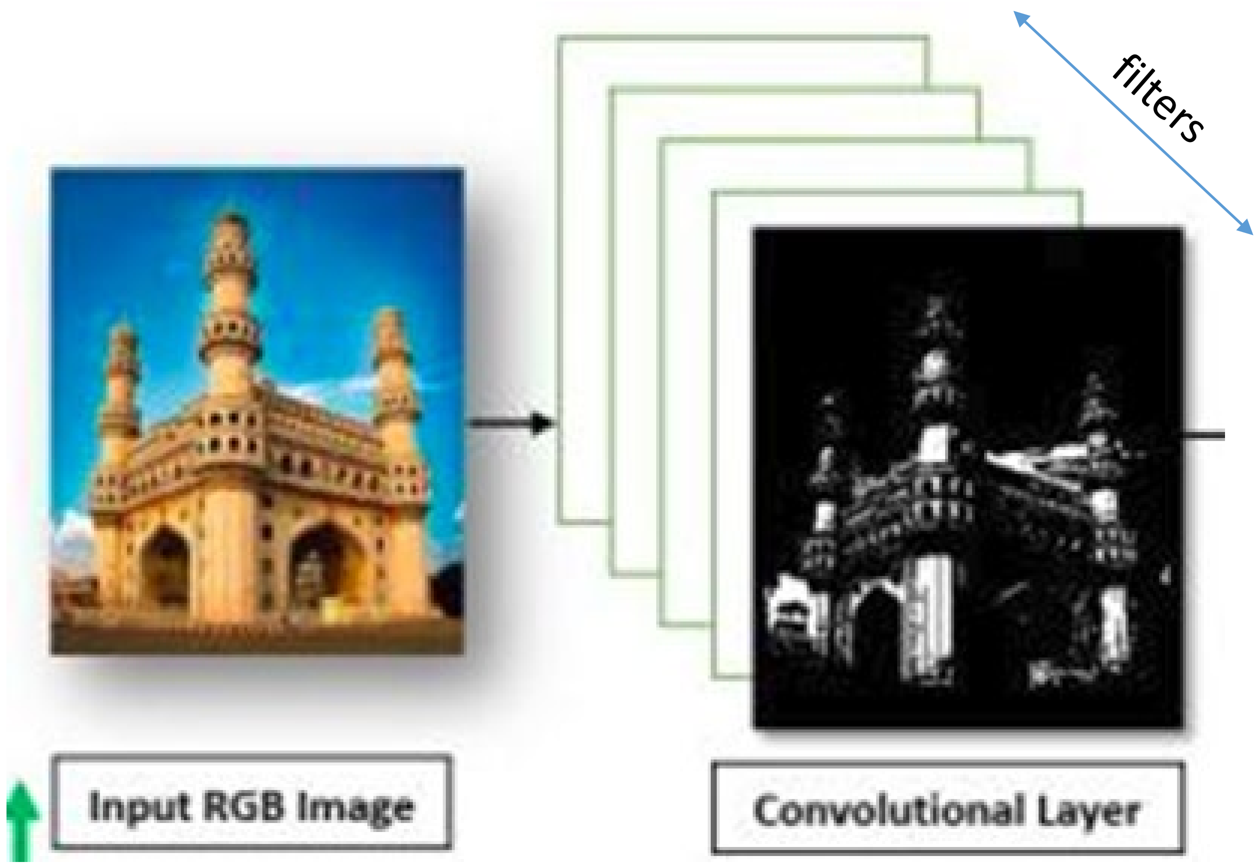
```
model = Sequential()
model.add(Conv2D(F, (K, K), strides=(S, S), activation='relu’, padding=“same”, input_shape=(32, 32, L)))
```



1. Convolution Neural Network

Convolutional Layer (CNN or ConvNet):

- The CNN will reduce the original RGB images to its Convolutional Layer
- Multiple layers can be applied



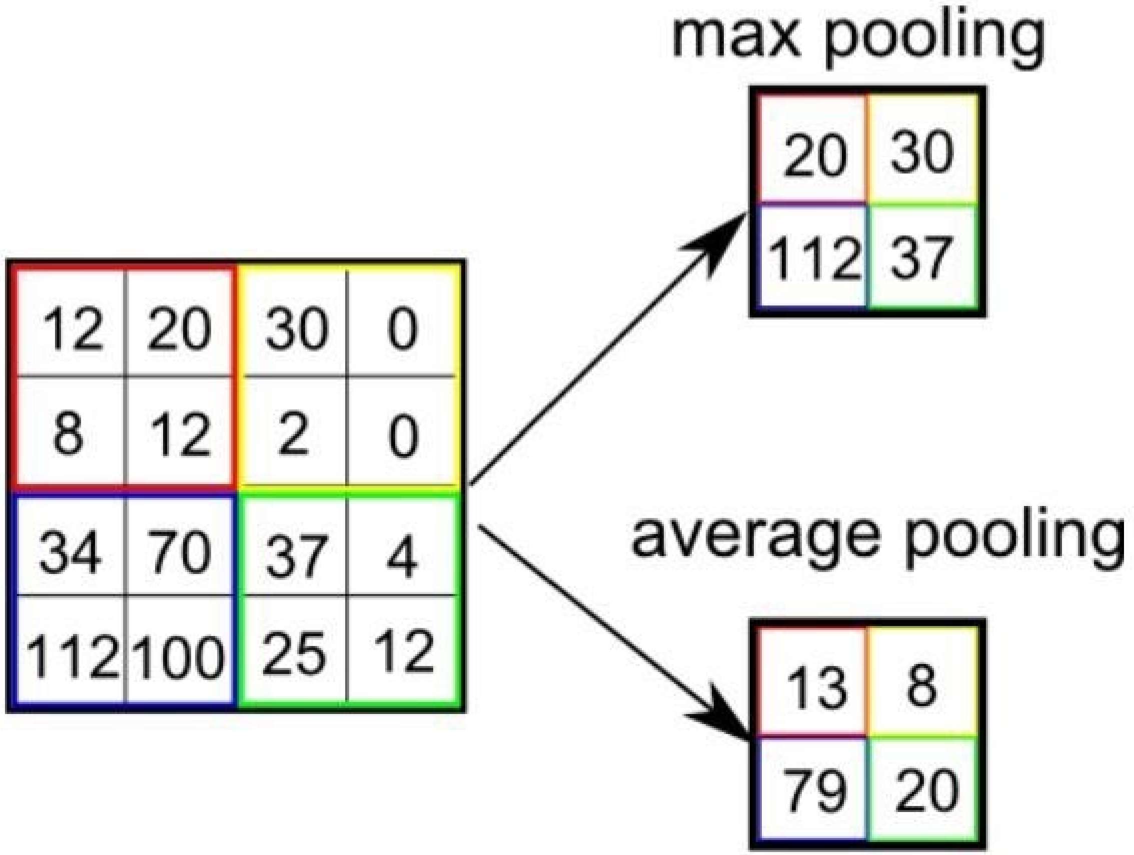
1. Convolution Neural Network

Pooling Layer

- Pooling Layer should follow Convolutional Layer
- Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature.
- This is to decrease the computational power required to process the data through dimensionality reduction
- Two types of Pooling: Max Pooling & Average Pooling.

1. Convolution Neural Network

Pooling Layer

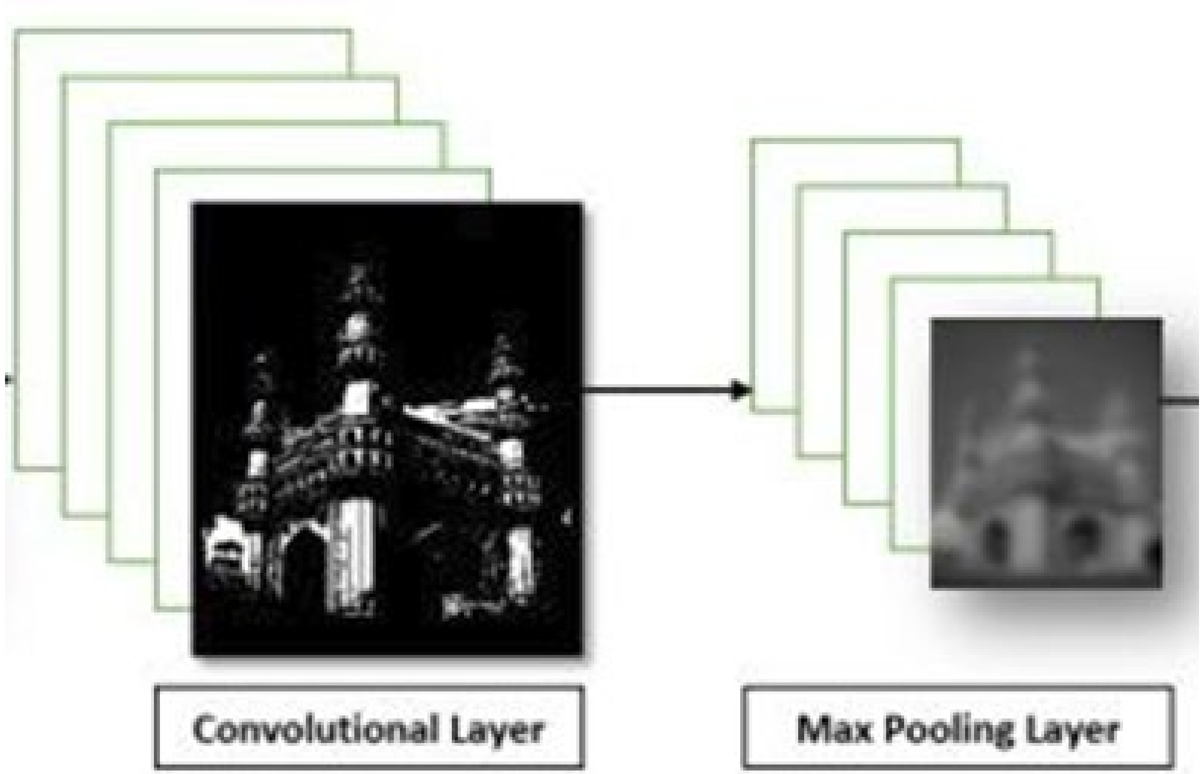


In which Max Pooling performs a lot better than Average Pooling.

1. Convolution Neural Network

Pooling Layer

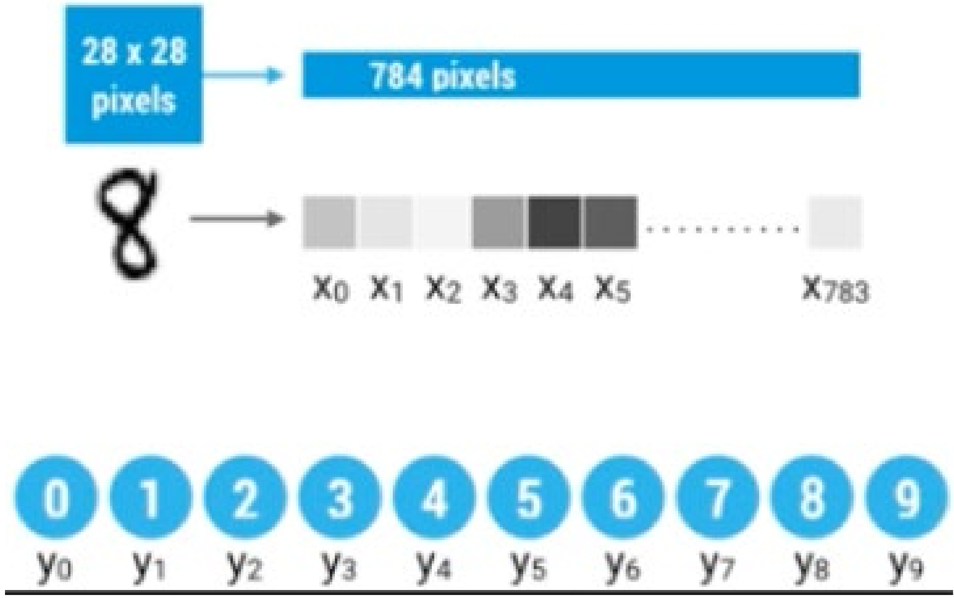
- The image after Max Pooling layer would look like:



1. Convolution Neural Network

Flatten Layer

- Once the images have passed through Convolution Layer and Pooling Layer, its size has been reduced greatly and ready for MLP training (or to another Convolution steps).
- The image is then flatten to a column vector and passed through feed-forward NN and BackPropagation applied to every iteration.
- Softmax activation function is applied to classified the multi-output/multi-labels



1. Convolution Neural Network

Some other useful layers?

1. Convolution Neural Network

Batch Normalization

- A process to make Deep neural networks faster and more stable through adding extra layers in a deep neural network.
- The new layer performs the standardizing and normalizing operations on the input of a layer coming from a previous layer.
- Normalize the amounts of weights trained between layers during training
- It usually goes after Conv2D layers or after Dense layer

```
from tensorflow.keras.layers BatchNormalization
model = Sequential()
model.add(Conv2D(75, (3, 3), strides=1, activation="relu", input_shape=(28, 28, 1)))
model.add(BatchNormalization())
model.add(MaxPool2D((2, 2), strides=2))
```

1. Convolution Neural Network

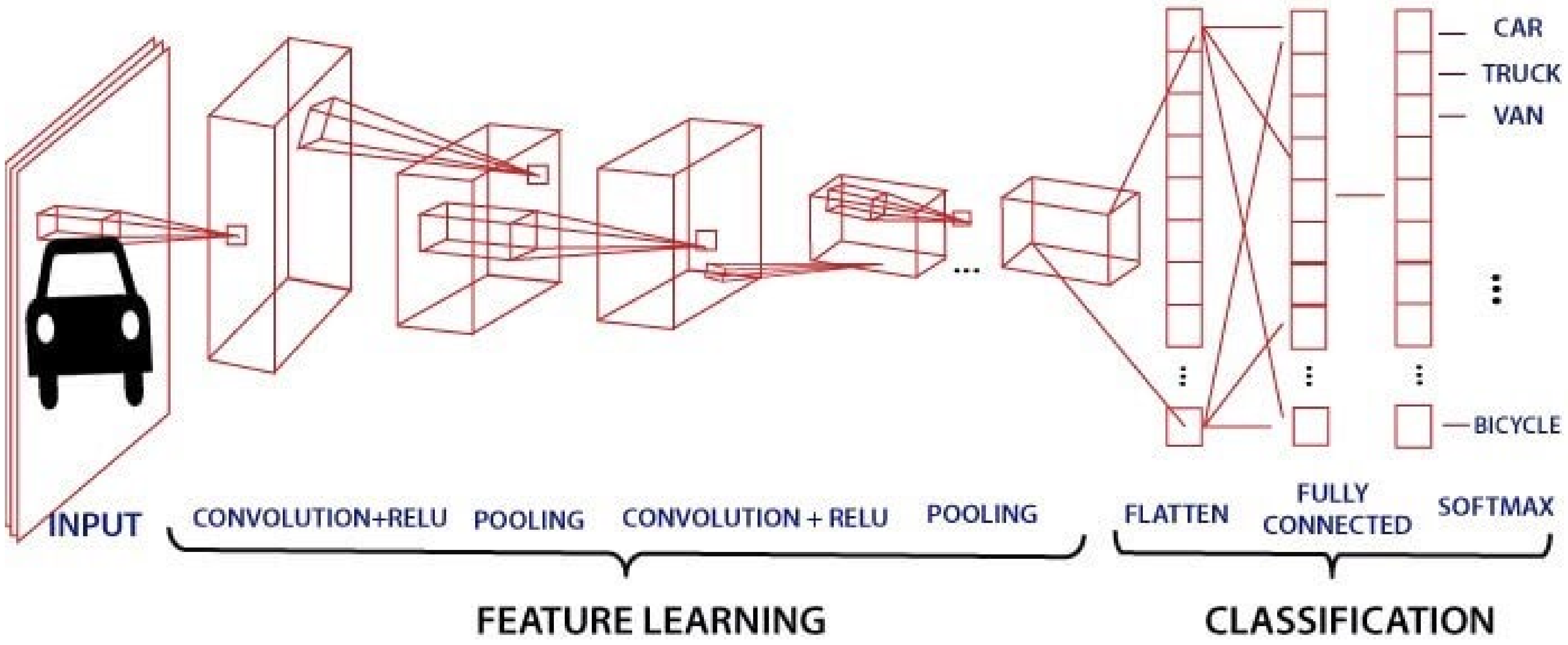
Dropout

- Dropout is a regularization method that approximates training a large number of neural networks with different architectures in parallel.
- Dropout helps to avoid Overfitting
- Dropout is implemented per layer in the NN
- Dropout is not used after training when making a prediction with the fit network.

```
from tensorflow.keras.layers BatchNormalization
model = Sequential()
model.add(Conv2D(75, (3, 3), strides=1, activation="relu", input_shape=(28, 28, 1)))
model.add(Dropout(0.2))
```

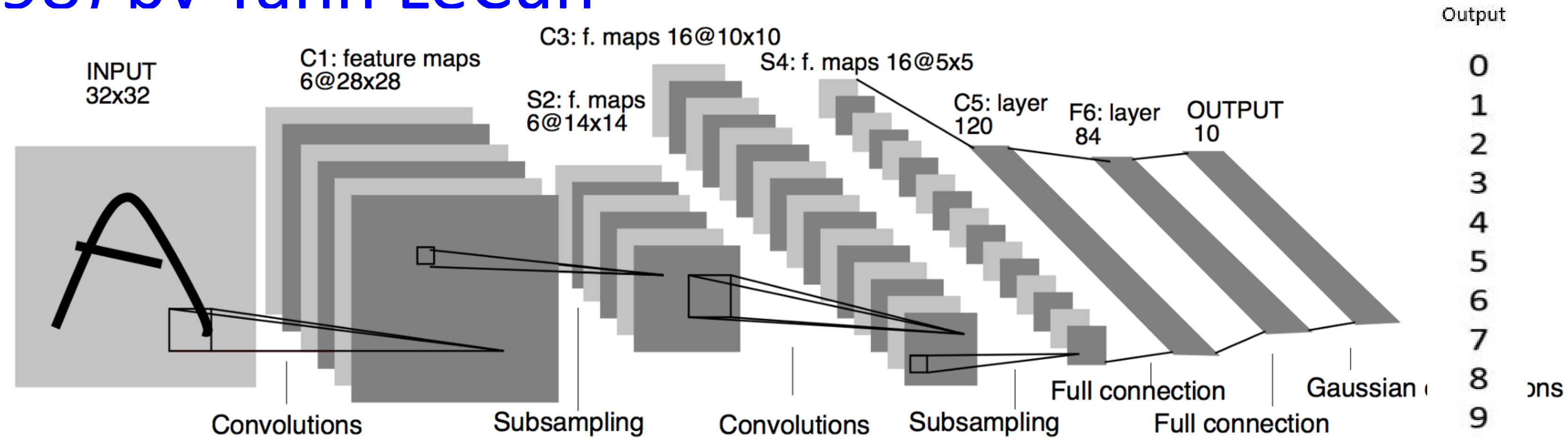
1. Convolution Neural Network

A Sample of CNN:



1. Convolution Neural Network

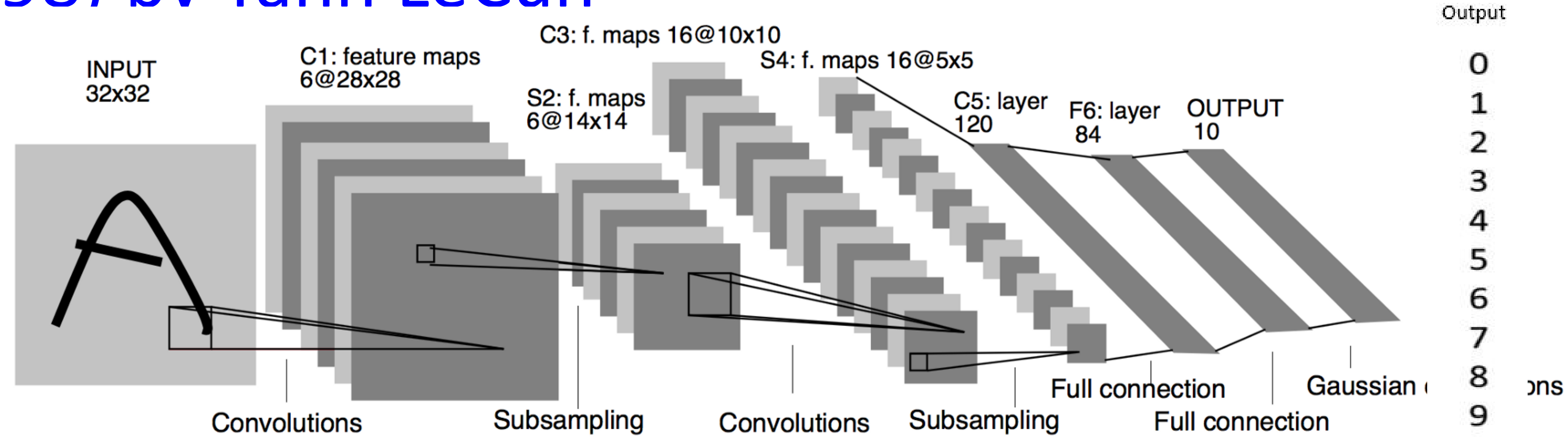
Lenet-5 (1998) bv Yann LeCun



- LeNet-5 is designed for handwritten and machine-printed character recognition
- Input of 32x32x1
- Total parameters: 60k
- Activation function: tanh

1. Convolution Neural Network

Lenet-5 (1998) by Yann LeCun

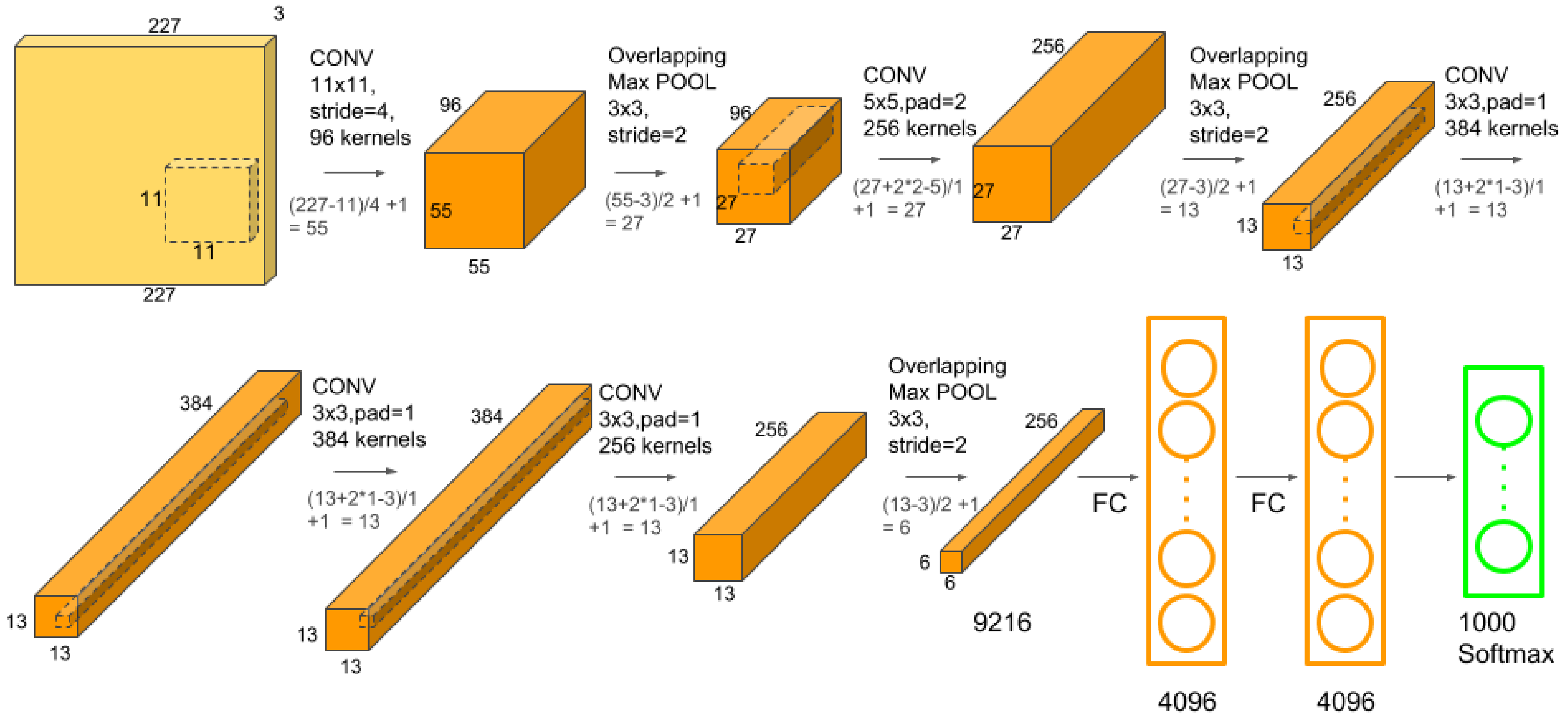



```
model = Sequential()
model.add(Conv2D(6, (5, 5), strides=(1, 1), activation=‘tanh’, padding=“valid”, input_shape=(32, 32, 1)))
model.add(AveragePooling2D(pool_size=(2,2),strides=(2,2)))
model.add(Conv2D(16, (5, 5), strides=(1, 1), activation=‘tanh’, padding=“valid”))
model.add(AveragePooling2D(pool_size=(2,2),strides=(2,2)
model.add(Conv2D(120, (5, 5), strides=(1, 1), activation=‘tanh’, padding=“valid”))
model.add(Flatten())
model.add(Dense(84,activation=‘tanh’))
model.add(Dense(10,activation=‘softmax’))
```

1. Convolution Neural Network

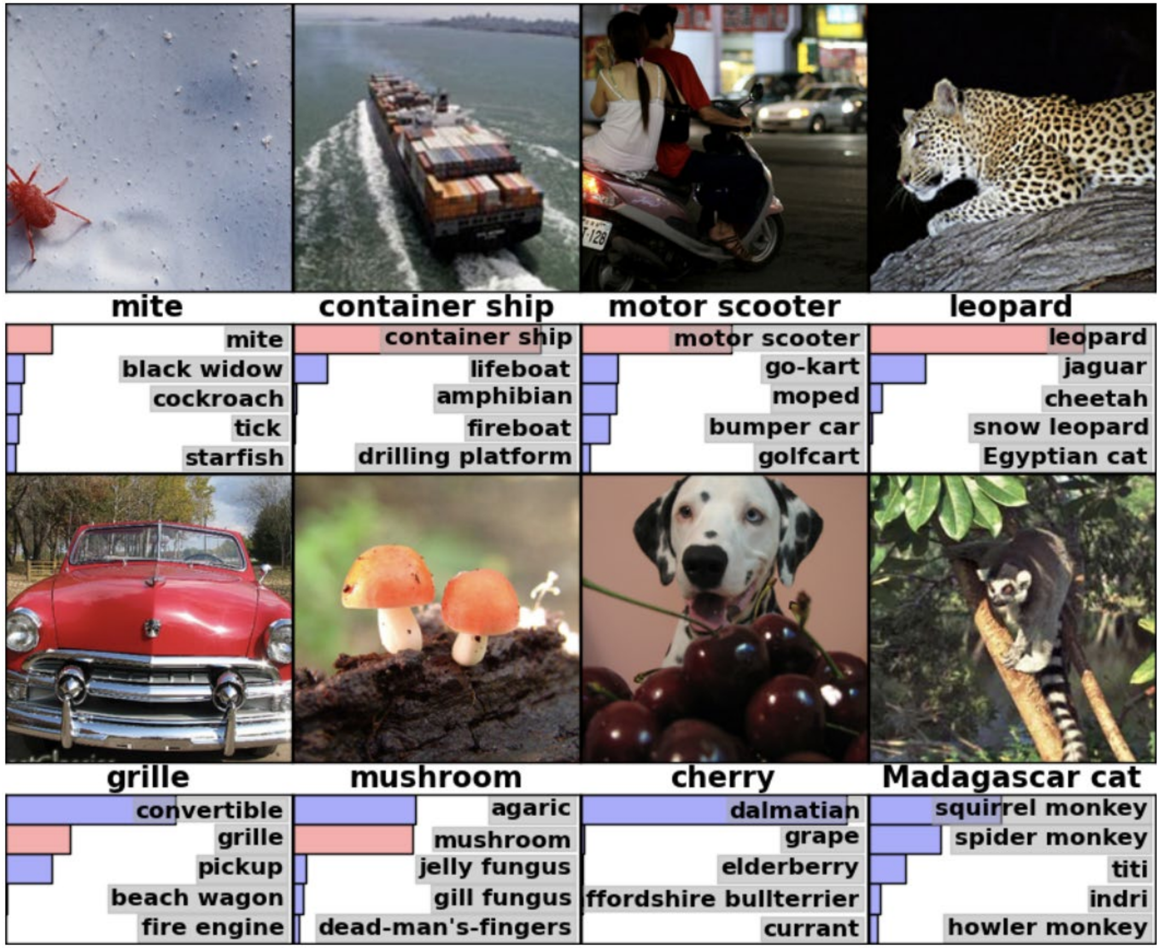
AlexNet (2012) by Hinton Alex Krizhevsky

- AlexNet won the 2012 ImageNet challenge
- Input of 227x227x3
- Total parameters: 60M
- Activation: ReLU



1. Convolution Neural Network

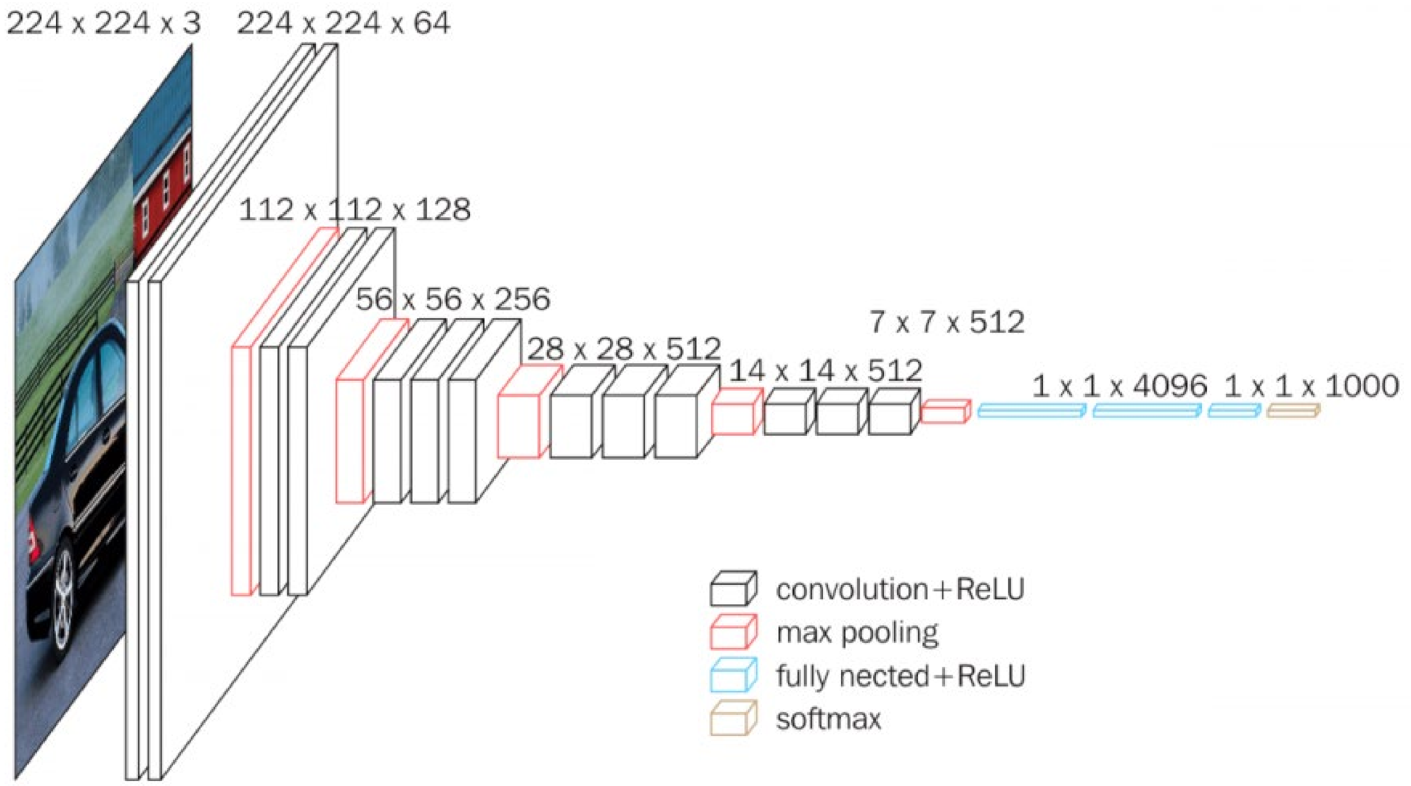
AlexNet (2012) by Hinton Alex Krizhevsky



1. Convolution Neural Network

VGG16 (2014) - Visual Geometry Group

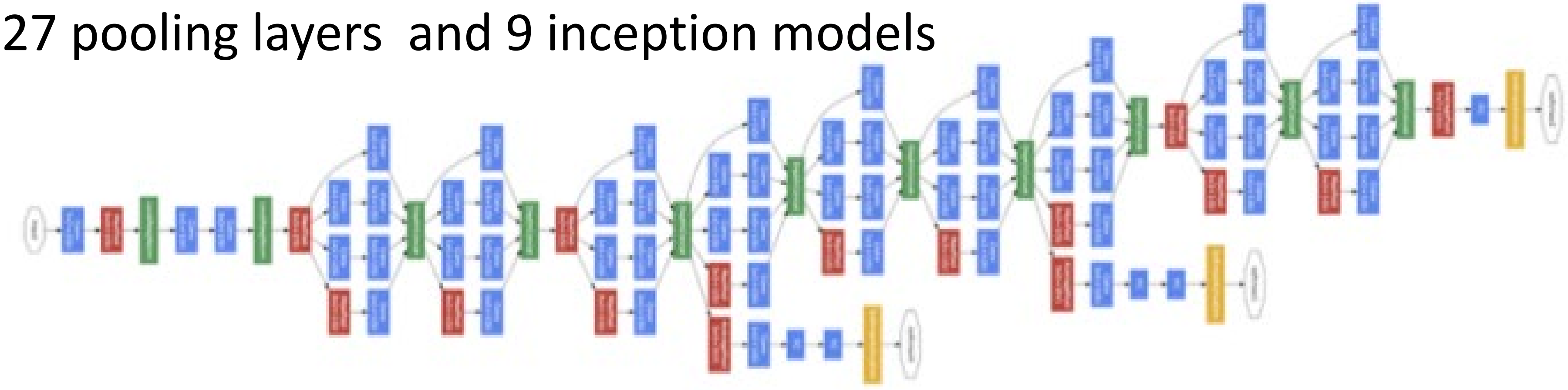
- VGG16 runner up of 2014 ImageNet challenge
- 16 layers: 13 ConvNet, 3 Fully Connected
- Total Parameters: 130M



1. Convolution Neural Network

GoogleNet (2014)

- GoogleNet won the 2014 ImageNet challenge
- Introduced Inception Network
- 22 layers deep with 27 pooling layers and 9 inception models

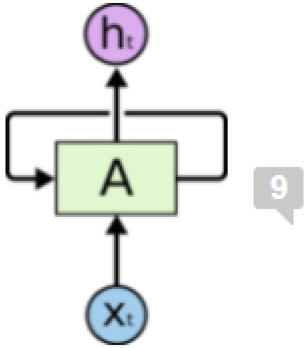


2. Recurrent Neural Network

2. Recurrent Neural Network

Introduction

- RNNs are type of Deep Learning models with built-in feedback mechanism.
- The output of a particular layer can be **re-fed** as the input in order to predict the output.
- This is different from traditional ML where output/predictand cannot be used as input

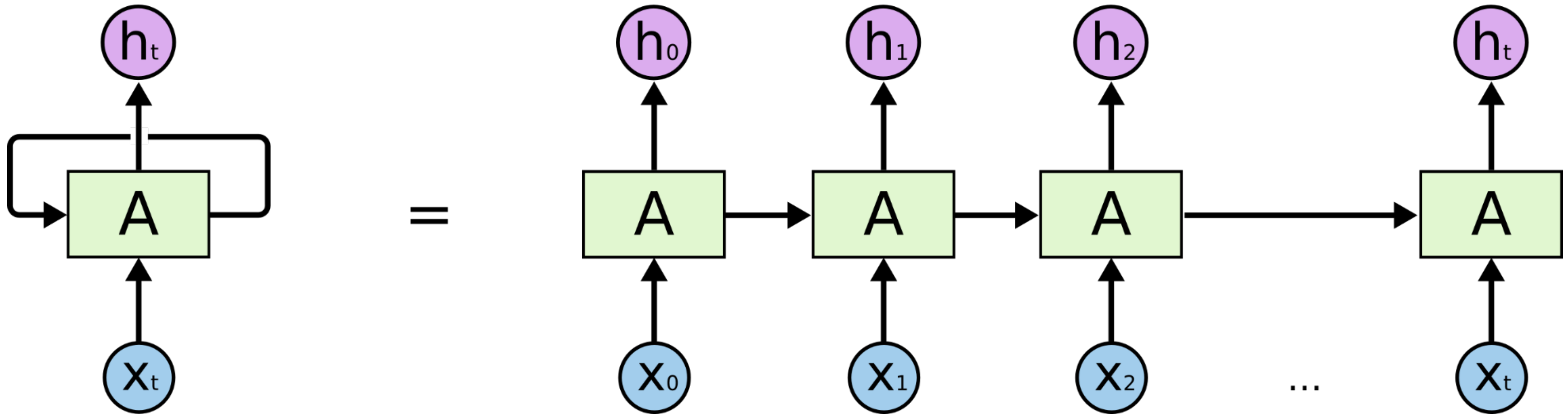


Recurrent Neural Networks have loops.

2. Recurrent Neural Network

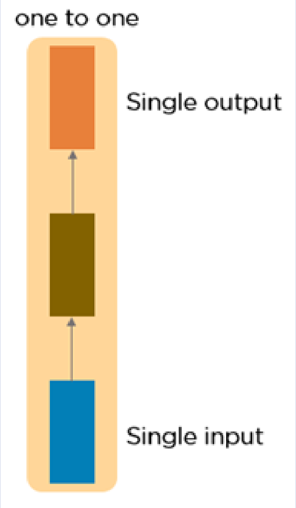
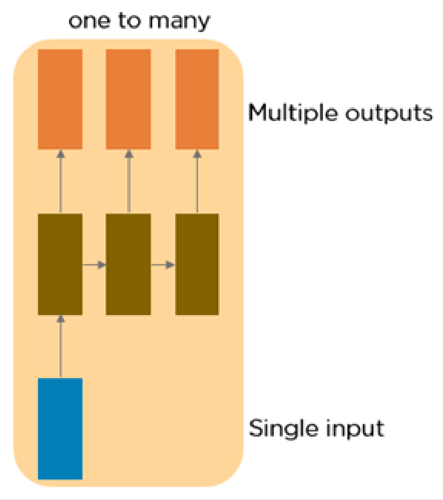
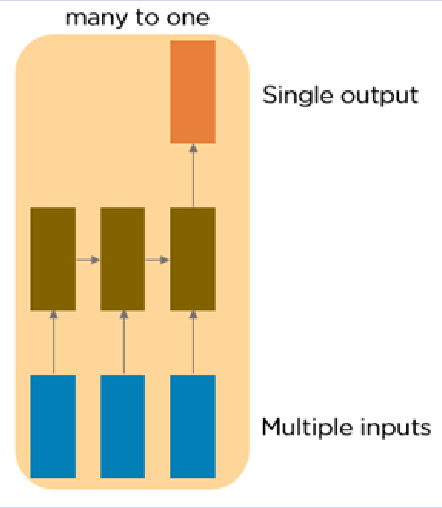
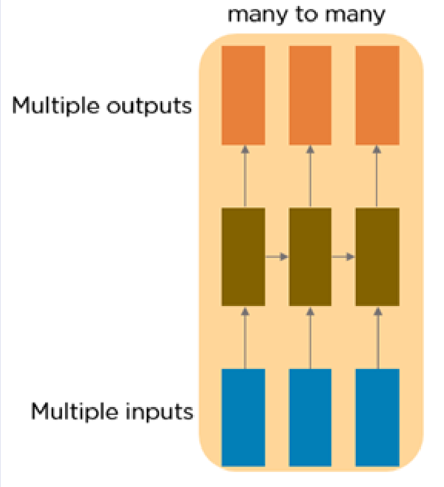
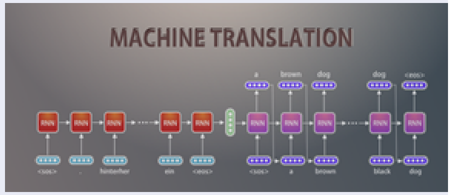
Introduction

- Unroll the RNN loop



2. Recurrent Neural Network

Type of RNNs

One to One	One to Many	Many to One	Many to Many
			
So called Vanilla NN. Similar to Backpropagation for general ML problem	Image captioning	Sentiment analysis	Machine translation
			

2. Recurrent Neural Network

Applications

It is specifically designed for Sequential problem **Weather forecast, Stock forecast, Image captioning, Natural Language Processing, Speech/Voice Recognition**

2. Recurrent Neural Network

Some Disadvantages of RNN:

- Computationally Expensive and large memory requested
- RNN is sensitive to changes in parameters and having gradient problem such as **Exploding Gradient** or **Vanishing Gradient**
- In order to resolve the **gradient problem** of RNN, a method Long-Short Term Memory (LSTM) is proposed.

In this limited workshop, we only cover LSTM for timeseries forecast problem (stock forecast and weather forecast)

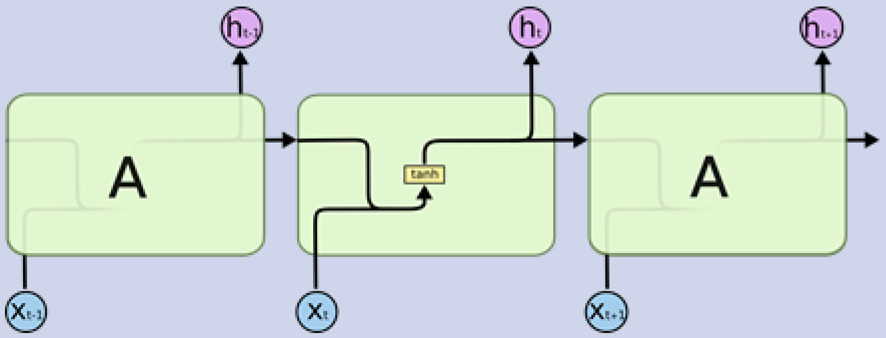
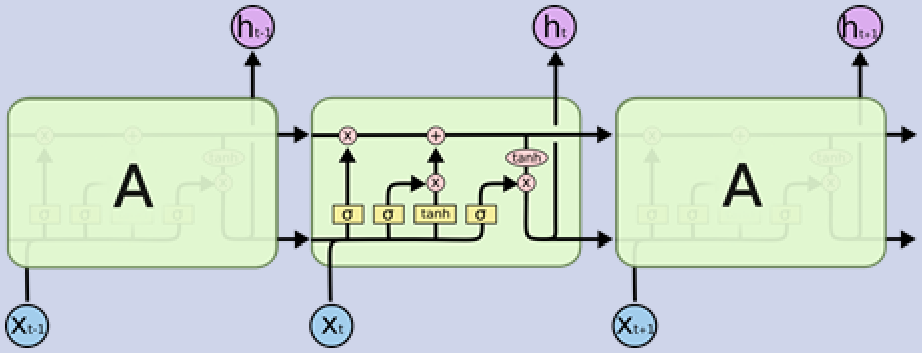
2. Recurrent Neural Network

Long-Short Term Memory model - LSTM

- LSTMs are a special kind of RNN — capable of learning long-term dependencies by remembering information for long periods is the default behavior.
- They were introduced by Hochreiter & Schmidhuber (1997) and were refined and popularized by many people
- LSTMs are explicitly designed to avoid the long-term dependency problem.

2. Recurrent Neural Network

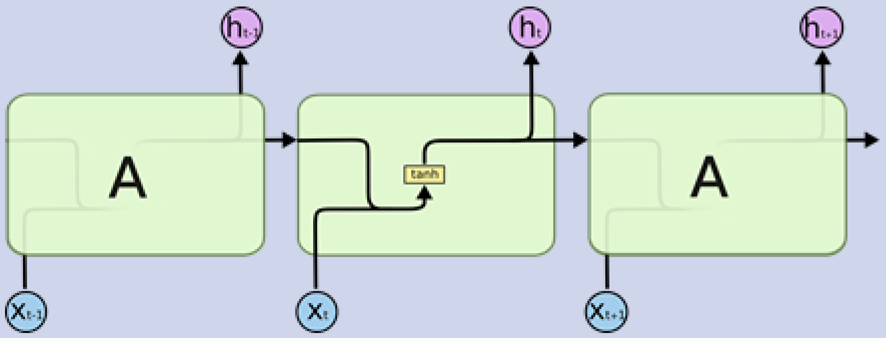
Comparison between traditional RNN and LSTM

Standard RNN structure	LSTM structure
 The repeating module in a standard RNN contains a single layer.	 The repeating module in an LSTM contains four interacting layers.

2. Recurrent Neural Network

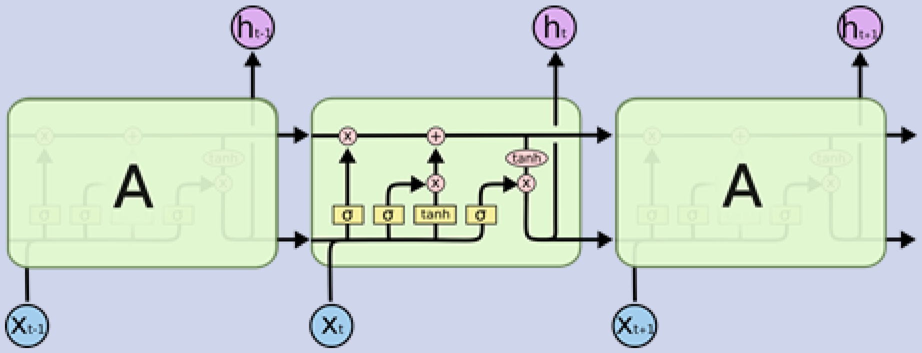
Comparison between traditional RNN and LSTM

Standard RNN structure

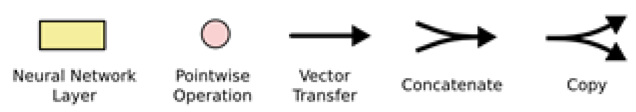


The repeating module in a standard RNN contains a single layer.

LSTM structure



The repeating module in an LSTM contains four interacting layers.




```
model = Sequential()
model.add(LSTM(128, return_sequences=True, input_shape= (x_train.shape[1], 1)))
model.add(LSTM(64, return_sequences=False))
```

2. Recurrent Neural Network

Hands-on section