GETTING STARTED WITH AI

AGENDA

- ✓ Hello World of Neural Networks
- ✓ TensorFlow Introduction
- ✓ Introduction to computer vision
- ✓ Classification with F-MNIST dataset
- ✓ Callbacks in TensorFlow
- ✓ Introduction to Convolution and Pooling
- ✓ Introduction to Stride
- ✓ Understanding CNN
- ✓ Classification in detail
- ✓ Different Loss and Activation functions
- ✓ Sample Execution with TensorFlow and Browser

- ✓ Q & A

Disclaimer: This presentation is made with best of my knowledge with reference from frequently accessed websites

HELLO WORLD OF NEURAL NETWORKS

X	-1	0	1	2	3	4	5
Y	-4	-1	2	5	8	11	?

- Are the numbers in Arithmetic Progression
- Are they multiplied by some constant
- Are they Arranged randomly

Steps to be followed:

Import Libraries: Tensorflow, keras and numpy

Define the Neural Network : One layer and it has only one neuron

Compile the Neural Network : Loss function and the optimizer. They help neural network guess the pattern, measure how well or how badly the guess performed, before trying again on the next epoch, and slowly getting more accurate.

Provide the data : Numpy arrays

Train the Neural Network & Predict Data

Solution: Y = 3*X-1

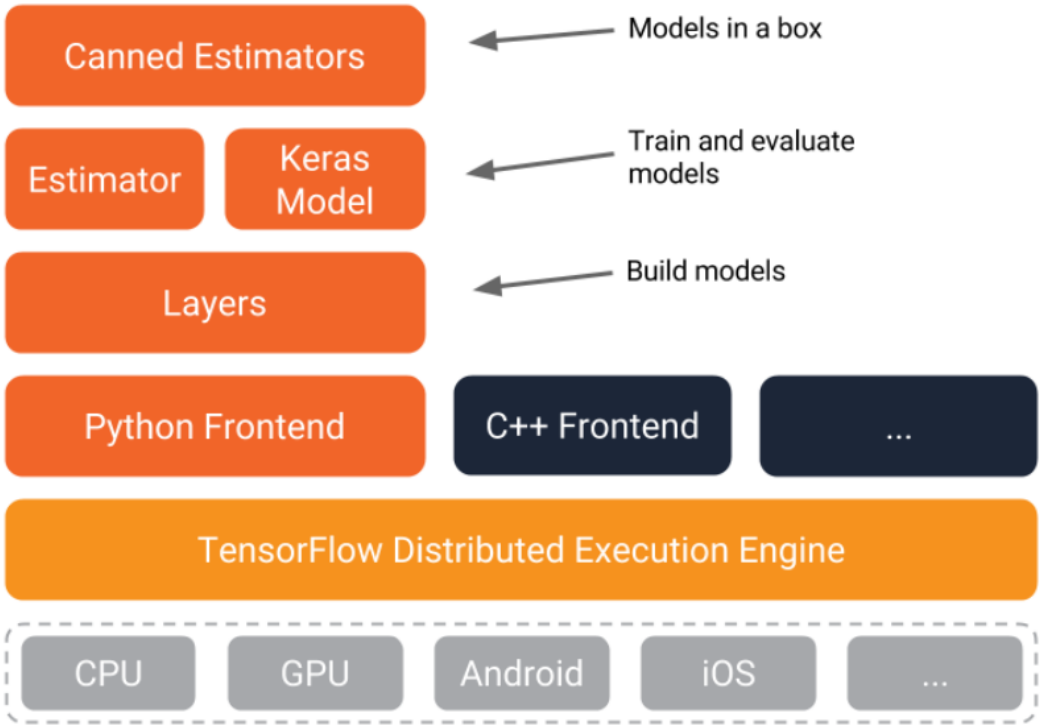
Dev Environment:

- Jupyter Notebooks in Python
- Google Colaboratory notebooks

Github Link:
https://github.com/iotlab1/Tensorflow-AI/blob/master/HelloWorld.ipynb

TENSORFLOW

Tensorflow is an open source software library for numerical computation using data-flow graphs from Google which supports lot of prominent algorithms.



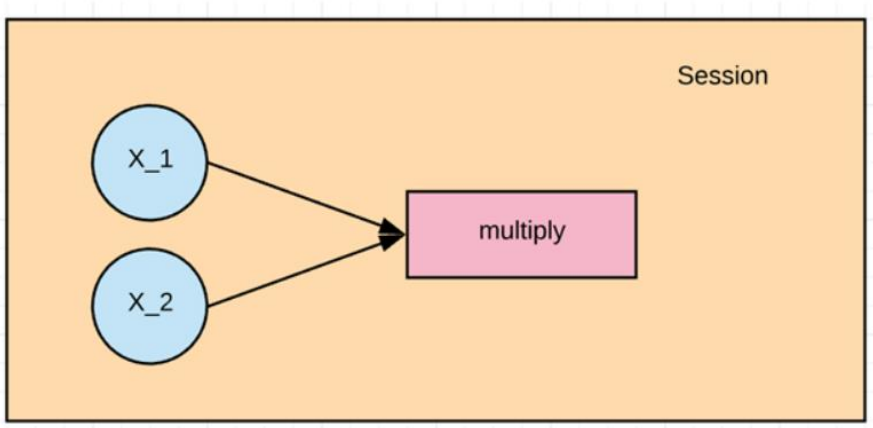
Tensorflow works in three parts:

- Preprocessing the data
- Build the model
- Train and estimate the model

Components of TensorFlow :

Tensor : A tensor is a vector or matrix of n-dimensions that represents all types of data. All values in a tensor hold identical data type with a known (or partially known) shape

Graphs: Graph is a set of computation that takes place successively. Each operation is called an op node and are connected to each other.



Placeholder: A placeholder is simply a variable that we will assign data to at a later date. It allows us to create our operations and build our computation graph, without needing the data

Github Link: https://github.com/iotlab1/Tensorflow-AI/blob/master/Tensorflowexample.ipynb

Step 1: Define the variable

```
X_1 = tf.placeholder(tf.float32, name = "X_1")
X_2 = tf.placeholder(tf.float32, name = "X_2")
```

Step 2: Define the computation

```
multiply = tf.multiply(X_1, X_2, name = "multiply")
```

Step 3: Execute the operation

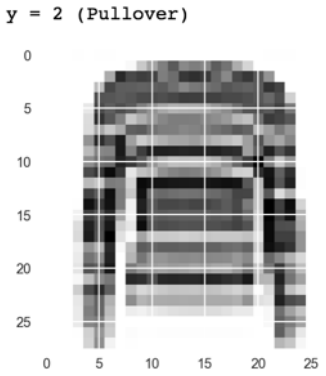
```
X_1 = tf.placeholder(tf.float32, name = "X_1")
X_2 = tf.placeholder(tf.float32, name = "X_2")

multiply = tf.multiply(X_1, X_2, name = "multiply")

with tf.Session() as session:
    result = session.run(multiply, feed_dict={X_1:[1,2,3], X_2:[4,5,6]})
    print(result)
```

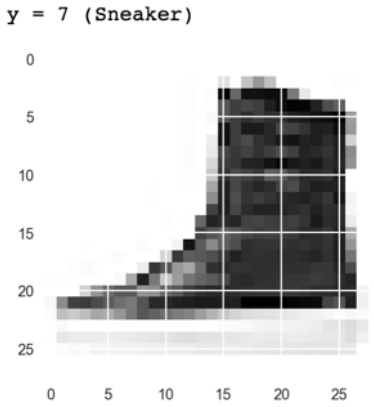
INTRODUCTION TO COMPUTER VISION

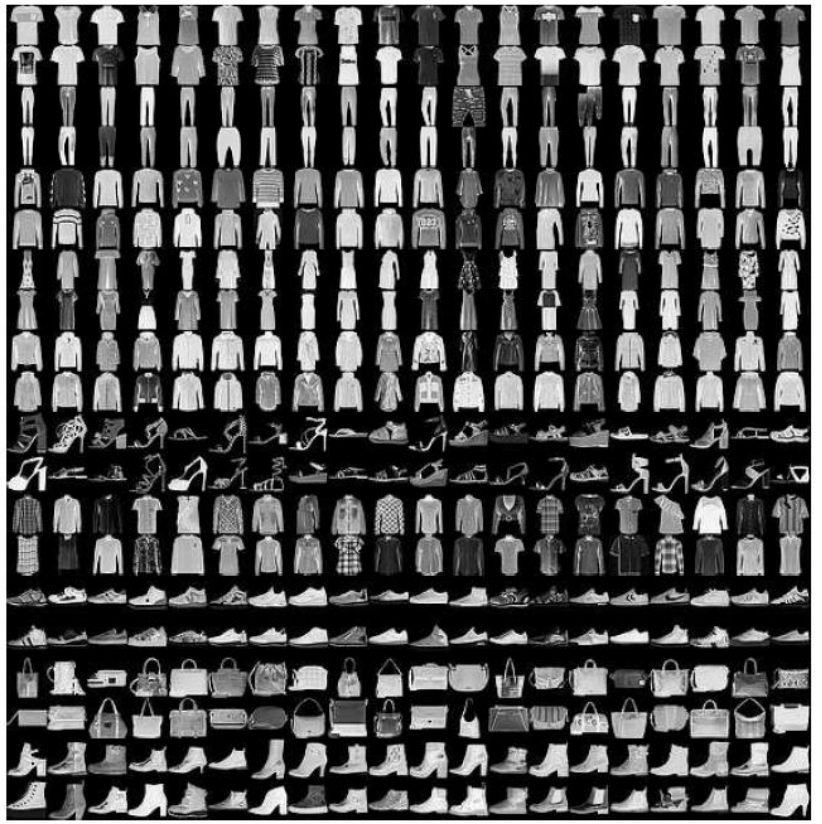
- Computer vision is a field of having a computer understand and label what is present in an image.
- It is like, use a lot of pictures say that of clothing and tell the computer what is that picture of and the computer figure out the pattern and give you the difference between shoe , a shirt and handbag.

CLASSIFICATION WITH FASHION MNIST DATASET

- 70k Images(Training: Test -> 6:1)
- 10 Categories
- Images are 28x28
- Images are in Grey scale

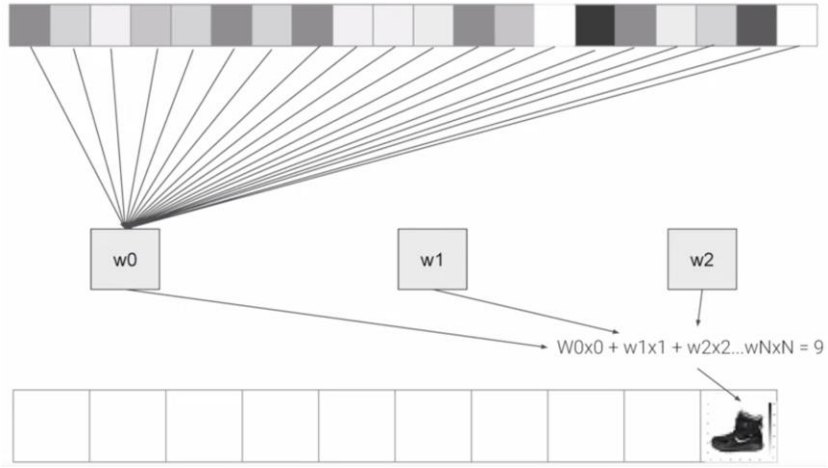




Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

Steps to be followed:

- **Import Libraries:** tensorflow, keras, matplotlib
- **Load the MNIST Data:** Data is available directly in the tf.keras dataset API. Loading the data as separate training and testing sets. There will be images and labels in each sets. Labels are numbers from 0 to 9 for each of the 10 categories.
- **Normalize the data:** To transform features to be on a similar scale which performance and training stability of the model.
- **Define the neural network:** One input layer in the shape of the data, output layer in the shape of the classes and a hidden layer.
- **Compile the model:** Make a guess on the relationship between input and output data and measure the loss function and optimize the model accordingly.
- **Train the model with the data and evaluate**.



Github Link:

- https://github.com/zalandoresearch/fashion-mnist
- https://github.com/iotlab1/Tensorflow-AI/blob/master/FMNIST.ipynb

Neural Network Definition:

- **Sequential**: That defines a SEQUENCE of layers in the neural network
- **Flatten:** Remember earlier where our images were a square, when you printed them out? Flatten just takes that square and turns it into a 1 dimensional set.
- **Dense:** Adds a layer of neurons

  Each layer of neurons need an activation function to tell them what to do. There's lots of options, but just use these for now.
- **Relu:** effectively means "If X>0 return X, else return 0" -- so what it does is, it only passes values 0 or greater to the next layer in the network.
- **Softmax:** takes a set of values, and effectively picks the biggest one, so, for example, if the output of the last layer looks like [0.1, 0.1, 0.05, 0.1, 9.5, 0.1, 0.05, 0.05, 0.05], it saves you from fishing through it looking for the biggest value, and turns it into [0,0,0,0,1,0,0,0,0]

CALLBACK IN TENSORFLOW

```
import tensorflow as tf


class myCallback(tf.keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs={}):
    if(logs.get('acc')>0.6):
      print("\nReached 60% accuracy so cancelling training!")
      self.model.stop_training = True
```

on_epoch_end: logs include `acc` and `loss`, and
optionally include `val_loss`
(if validation is enabled in `fit`), and `val_acc`
(if validation and accuracy monitoring are enabled).

on_batch_begin: logs include `size`,
the number of samples in the current batch.

on_batch_end: logs include `loss`, and optionally `acc`
(if accuracy monitoring is enabled).

Github Link:

- https://github.com/zalandoresearch/fashion-mnist

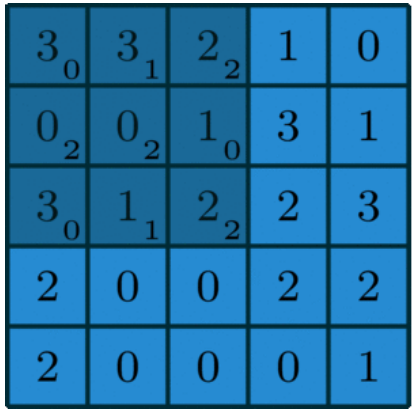
Tensforflow Documentation :

- https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/Callback

CONVOLUTION OPERATION

Image size: 224*224*3 = 100352

CNN: 1 x 1 x 1000



3	3	2	1	0
0	0	1	3	1
3	0	2	2	3
2	1	0	2	2
2	0	0	0	1

✸

0	1	2
2	2	0
3	1	2

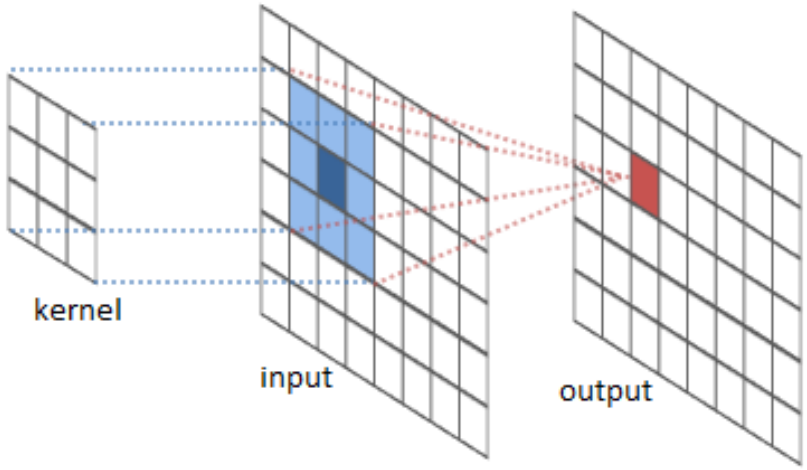
→

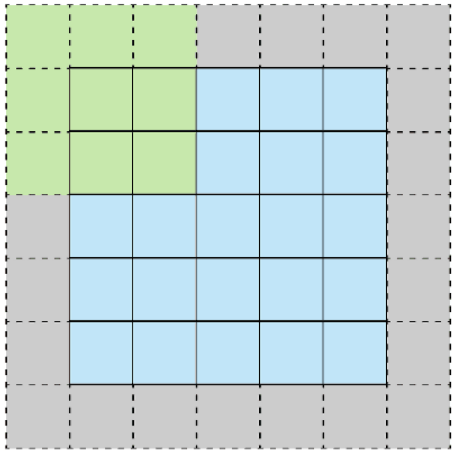
1. (3x0)+(3x1)+(2x2)+(0x2)+(0x2)+(1x0)+(3x0)+(1x1)+(2x2)=12
2. (3x0)+(2x1)+(1x2)+(0x2)+(1x2)+(3x0)+(1x0)+(2x1)+(2x2)=12
3. (2x0)+(1x1)+(0x2)+(1x2)+(3x2)+(1x0)+(2x0)+(2x1)+(3x2)=17
4. (0x0)+(0x1)+(1x2)+(3x2)+(1x2)+(2x0)+(2x0)+(0x1)+(0x2)=10
5. (0x0)+(1x1)+(3x2)+(1x2)+(2x2)+(2x0)+(0x0)+(0x1)+(2x2)=17
6. (1x0)+(3x1)+(1x2)+(2x2)+(2x2)+(3x0)+(0x0)+(2x1)+(2x2)=19
7. (3x0)+(1x1)+(2x2)+(2x2)+(0x2)+(0x0)+(2x0)+(0x1)+(0x2)=9
8. (1x0)+(2x1)+(2x2)+(0x2)+(0x2)+(2x0)+(0x0)+(0x1)+(0x2)=6
9. (2x0)+(2x1)+(3x2)+(0x2)+(2x2)+(2x0)+(0x0)+(0x1)+(1x2)=14

→

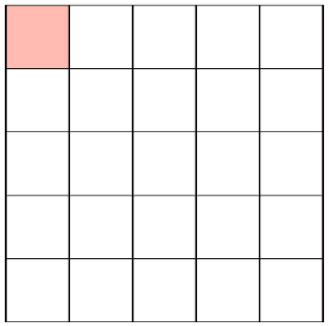
12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

CONVOLUTION OPERATION





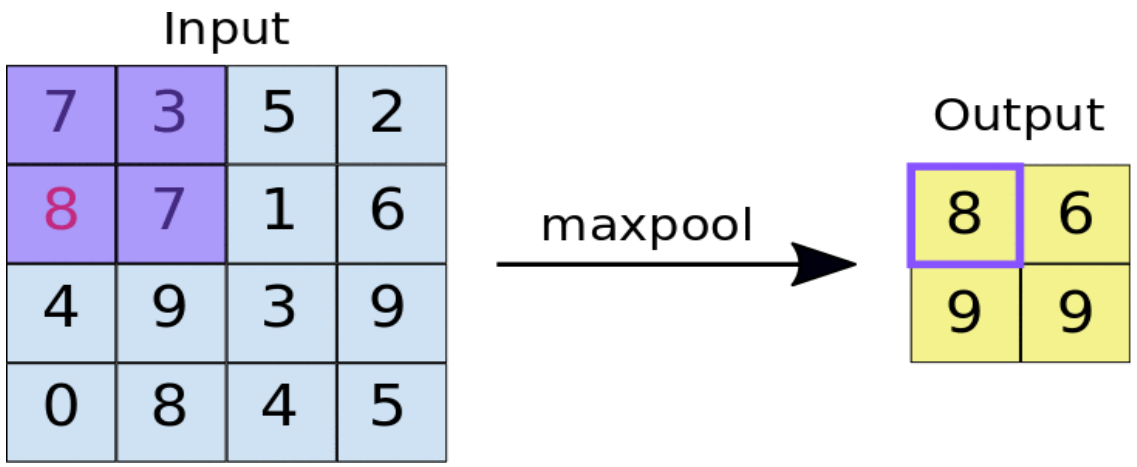
Stride 1 with Padding



Feature Map

- A kernel, which is a simple matrix of weights slides over a input data performing an elementwise multiplication with the part of the input it is currently on, and then summing up the results into a single output pixel.
- The kernel repeats this process for every location it slides over, converting a 2D matrix of features into yet another 2D matrix of features.

MAX POOLING



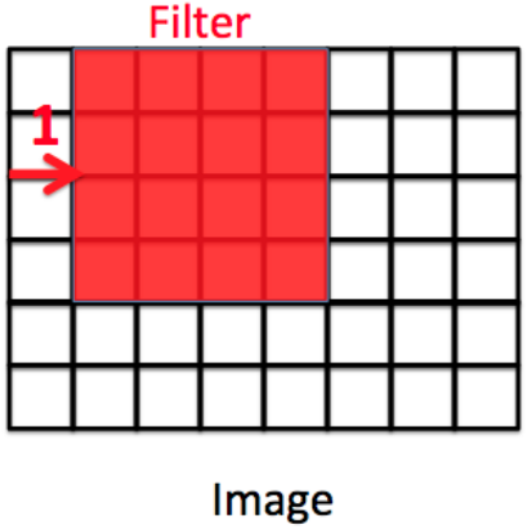
- It picks maximum values from the convoluted feature maps
- Max pooling extracts the most important features like edges.
- Max pooling is better for extracting the extreme features.

PADDING AND STRIDES

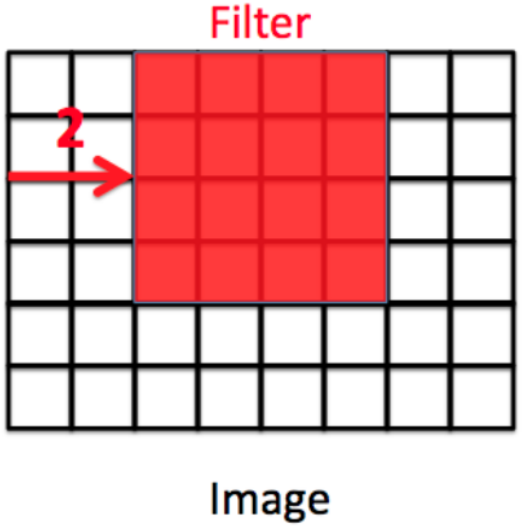
Padding: add extra pixels outside the image. And zero padding means every pixel value that you add is zero.

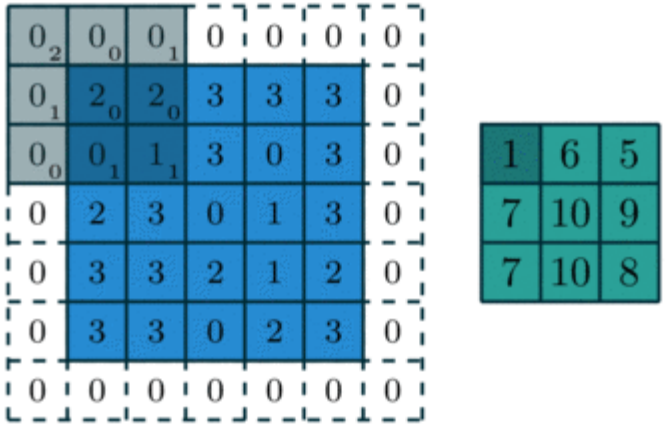
Stride: Stride is how far the filter moves in every step along one direction.

Stride with value 1



Stride with value 2



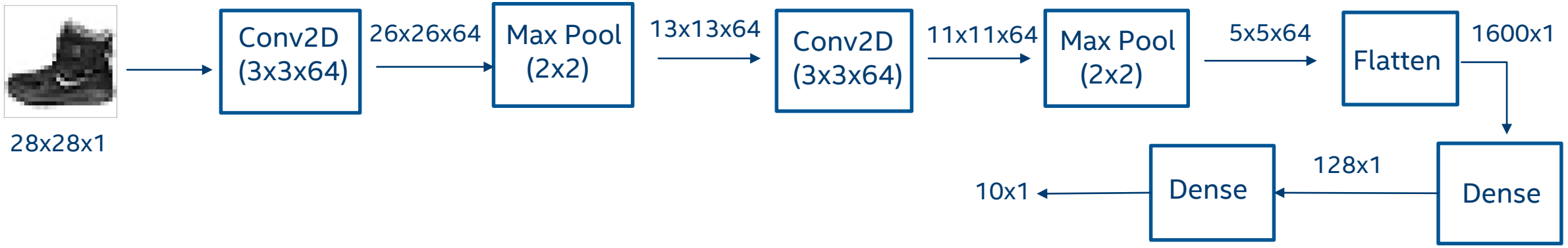


3*3 kernel applied to a 5*5 input padded with a 1*1 border of zeros using 2*2 strides

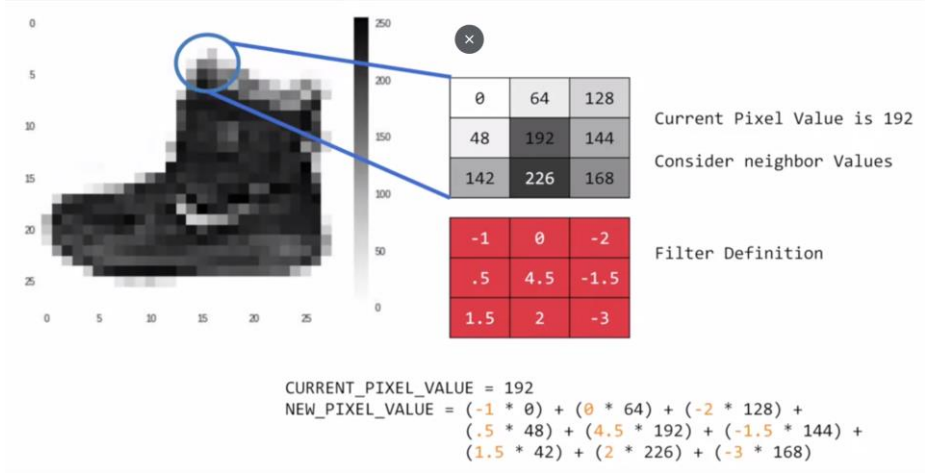
WHY CNN?

- CNN extracts the feature of image and convert it into lower dimension without loosing its characteristics.
- In the Fashion MNIST dataset each image is of size 28x28x1. If we are proceeding without convolution we will get 28x28=784 input neurons which is quite large to manage.
- By applying convolution layers, the dimension will reduce to **[(W-K+2P)/S]+1**

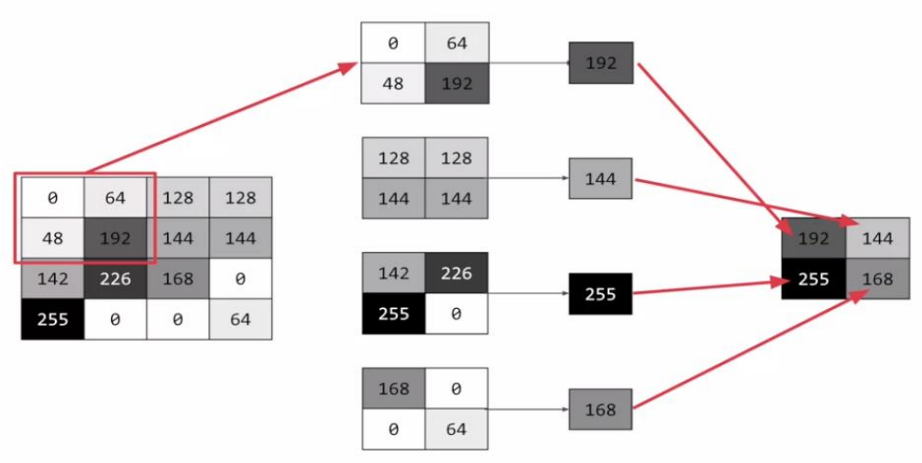
  Where, **W =** Size of Input **K** = Size of kernel/filter **P** = Padding size **S** = Stride
- By default padding is zero and stride is one for convolution and for max pooling default value of stride is equal to the pool size.



CONVOLUTIONS AND POOLING



Convolutions



Pooling

Convolution Visualization:

https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1
https://github.com/iotlab1/Tensorflow-AI/blob/master/CNN.ipynb

WHAT IS HAPPENING IN CAT-DOG CLASSIFICATION

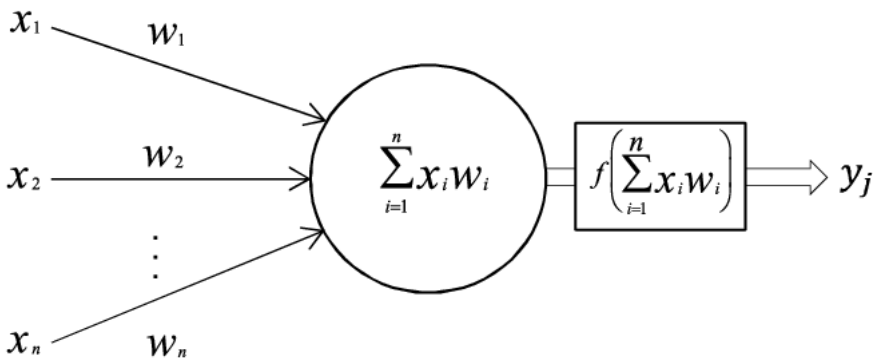


Fig : Single neuron operation

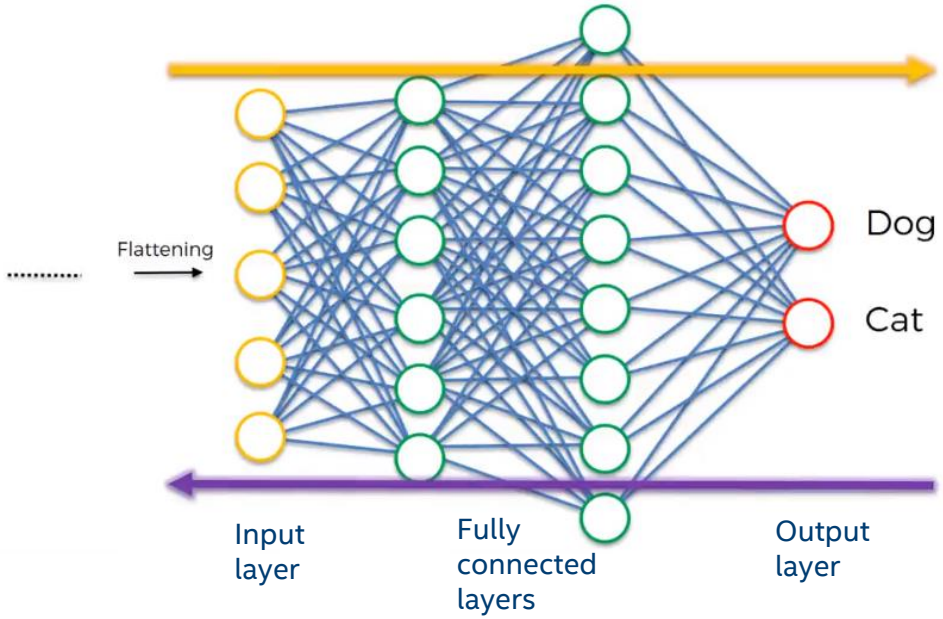
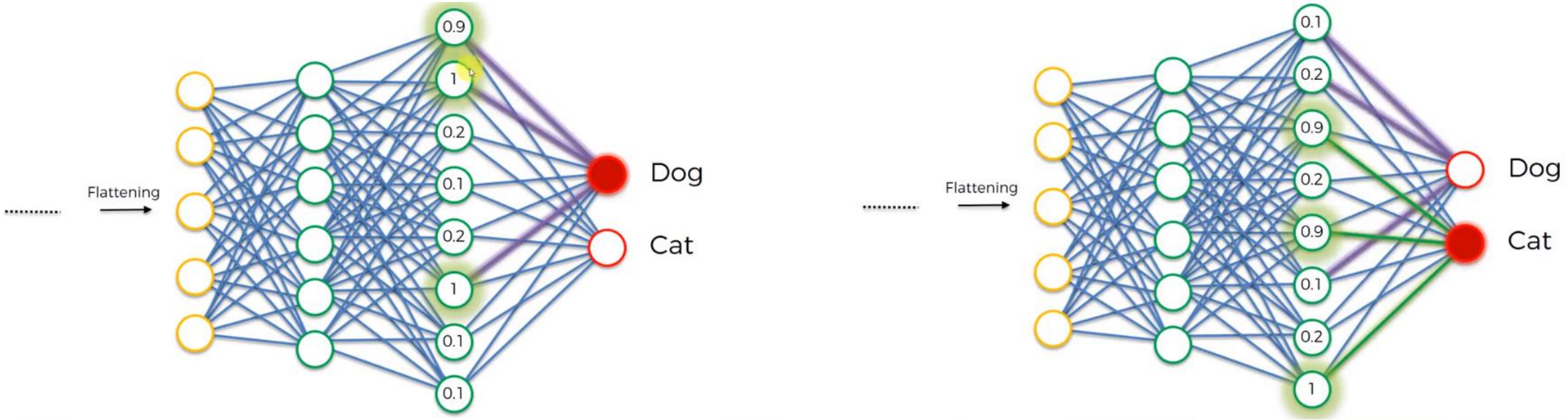


Fig : Full connection

- The data goes through the whole network, from the start to the end and the results are compared, error is calculated and it is backpropagate to update the weights of network.
- The above process going on and on and the network gets optimized.



- Lets say the numbers shown specifies each features like if it is 1, the neuron detecting the specific feature easily and if it is 0, that neuron is not important for that feature.
- In the case of dog, 1st, 2nd and 6th neurons are fired up, lets say those indicates eyebrow, big nose and floppy ears. So the dog neuron will understand that these neurons has more importance for it so that it should listen to these more.
- Similarly for cat, lets say 3rd , 5th and last neurons indicates whiskers, pointing triangle ear and cat eyes are lights up. So that the cat neuron paying more attention to it.

LOSS FUNCTIONS

- The function used to evaluate a candidate solution (i.e. a set of weights) is referred to as the objective function. We may seek to maximize or minimize the objective function.
- Typically, with neural networks, we seek to minimize the error. As such, the objective function is often referred to as a cost function or a loss function.

Mean Squared Error (MSE)

- Calculated as the average of the squared differences between the predicted and actual values.
- The result is always positive regardless of the sign of the predicted and actual values and a perfect value is 0.0.

Sparse Categorical Cross-Entropy

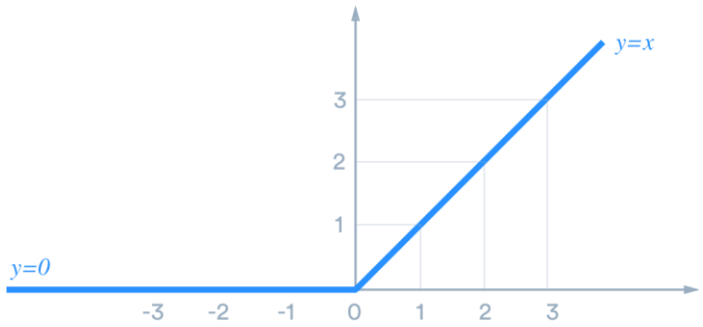
- Can be calculated for multiple-class classification
- Used when the targets are **integers**

ACTIVATION FUNCTION

- Activation function adds some kind of non-linear property to the function.
- Without the activation functions, the neural network could perform only linear mappings from inputs ***x*** to the outputs ***y***.

ReLU

- Rectified Linear Unit.
- Defined as *y = max(0, x)*
- ReLU is linear (identity) for all positive values, and zero for all negative values.
- Most commonly used activation function in neural networks.



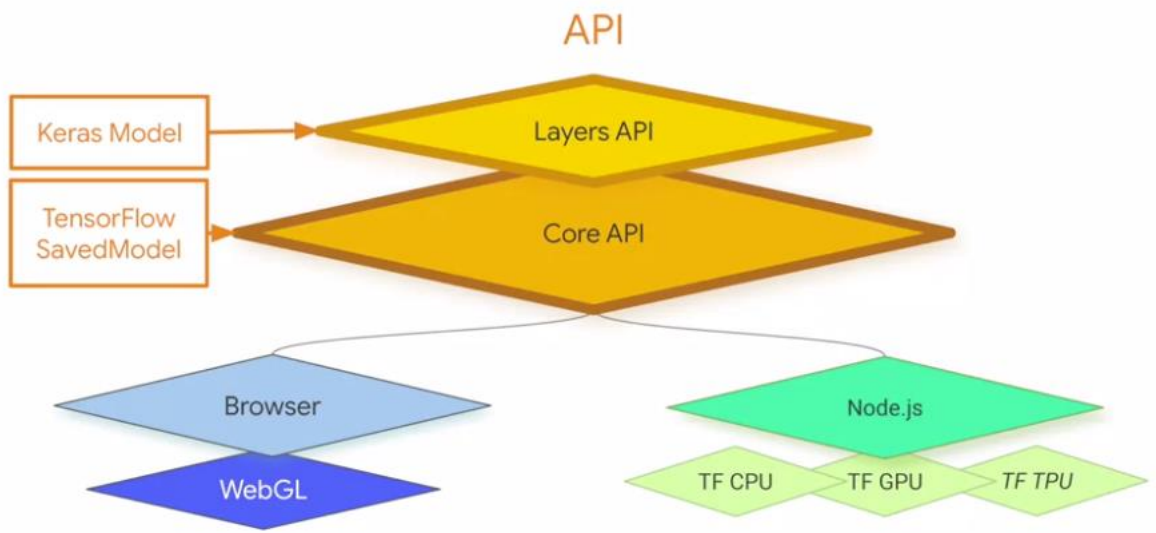
Softmax

- Able to handle multiple classes.
- Normalizes the outputs for each class between 0 and 1, and divides by their sum, giving the probability of the input value being in a specific class.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

TENSORFLOW.JS - ARCHITECTURE

- Layer API and TensorFlow.js looks and feels a lot like Keras.
- Syntax will be different as here java script is using instead of Python.
- Lower level APIs which are called as Core APIs designed to work with TensorFlow saved model formats.
- Core API works with Browser and WebGL for accelerated training and inference.
- WebGL is a JavaScript API for rendering interactive 2D and 3D graphics within any compatible web browser without the use of plug-ins.
- On Node.js we can build server side or terminal applications.
- Node.js can take advantages of CPUs, GPUs and TPUs based on the availability.



HELLO WORLD WITH TENSORFLOW.JS

Building model infers relationship between two numbers

-1	0	1	2	3	4
-3	-1	2	3	5	7

y=2x-1

Steps to be followed:

- **Create a simple webpage:**

```
<html>
<head></head>
<body>
    <h1>First HTML Page</h1>
</body>
</html>
```

- **Add the script tag below head and above body to load the TensorFlow.js file.**

```
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest"></script>
```

- **Define the model in one script tag above body.**

```
<script lang="js">
    const model = tf.sequential();
    model.add(tf.layers.dense({units: 1, inputShape: [1]}));
    model.compile({loss:'meanSquaredError',
                   optimizer:'sgd'});
    model.summary();
</script>
```

- **Add the data before closing the script tag.**

```
const xs = tf.tensor2d([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], [6, 1]);
const ys = tf.tensor2d([-3.0, -1.0, 2.0, 3.0, 5.0, 7.0], [6, 1]);
```

- **Include the asynchronous function for training in the script tag.**

```
async function doTraining(model){
    const history =
          await model.fit(xs, ys,
                { epochs: 500,
                  callbacks:{
                      onEpochEnd: async(epoch, logs) =>{
                          console.log("Epoch:" + epoch + " Loss:" + logs.loss);
                      }
                  }
                });
}
```

- **Train the model and predict.**

```
doTraining(model).then(() => {
    alert(model.predict(tf.tensor2d([10], [1,1])));
});
```

REFERENCES

- https://github.com/iotlab1/Tensorflow-AI
- https://colah.github.io/posts/2014-07-Understanding-Convolutions/
- https://www.tensorflow.org/api_docs/python/tf/keras/layers
- https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1
- https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/
- https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/
- https://www.coursera.org/learn/introduction-tensorflow/home/welcome

Thank You