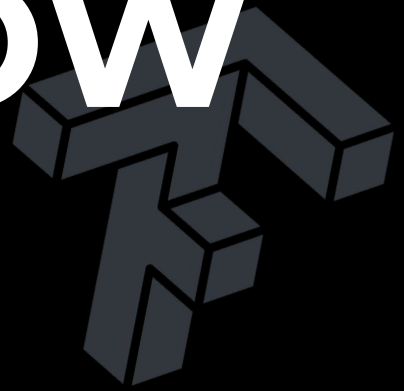
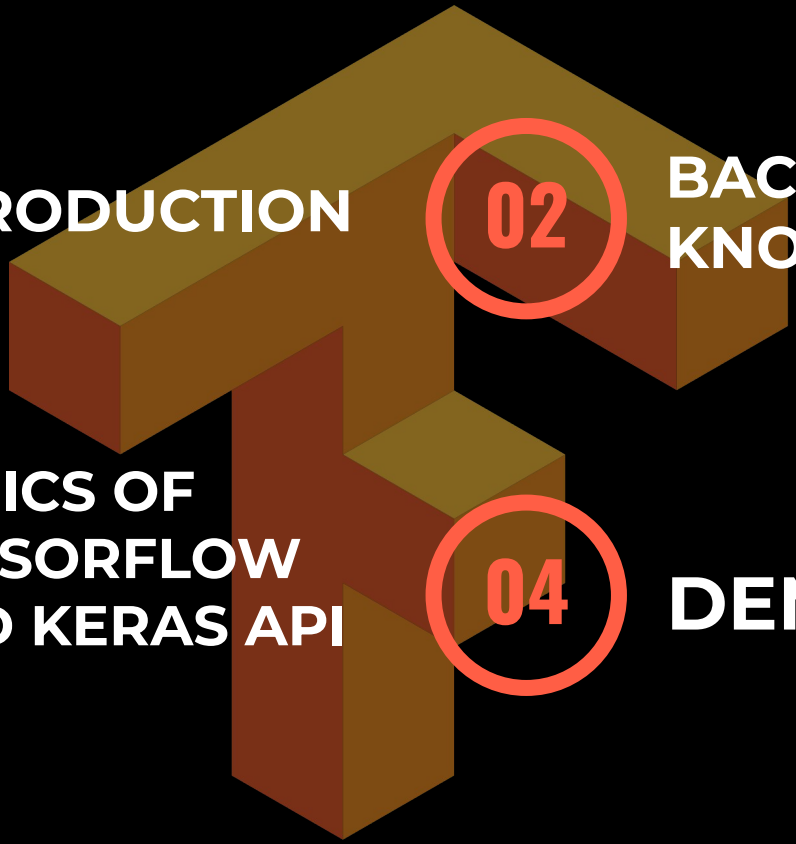
TENSORFLOW

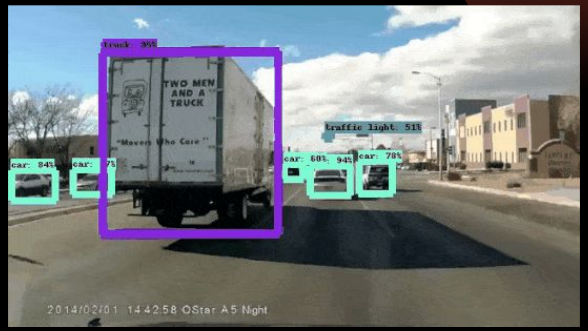
WORKSHOP



Satyen Sabnis & Shreya Gelli

TABLE OF CONTENTS





INTRODUCTION

Tensorflow is an open-source software library that was developed by researchers and engineers working on the Google Brain team. It is designed to expedite development of machine learning models, and is most commonly used for **deep neural networks**. Can be used for a wide variety of tasks from image and speech recognition to natural language processing.

WHY TENSORFLOW?

EASY MODEL BUILDING

Offers multiple levels of abstraction. Can **build** and **train** models **easy** using the high-level Keras API

ROBUST ML PRODUCTION

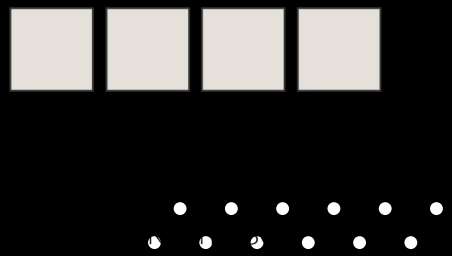
Whether on servers, edge devices, or on the web **deploying** your model is **easy**, regardless of platform or language

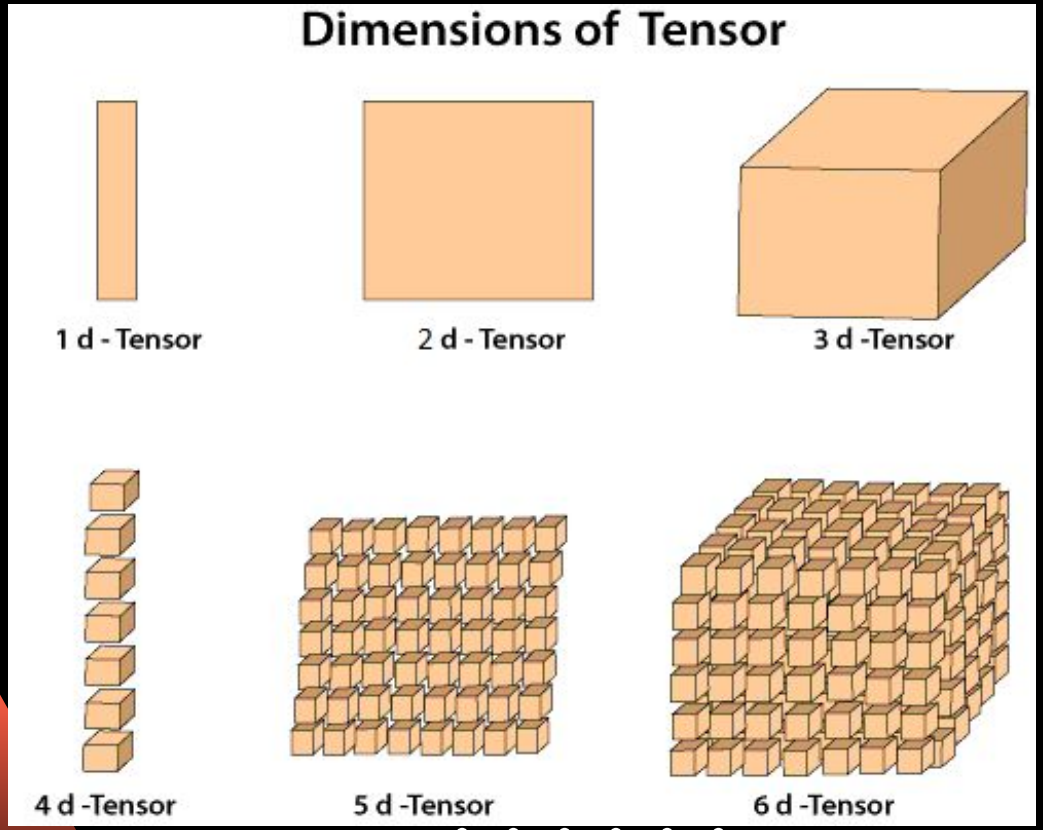
POWERFUL EXPERIMENTATION

Build and train complex models without sacrificing **speed** or **performance**

TENSORS

- Fundamental data structure in Tensorflow i.e the central data unit of how data is represented and manipulated in the framework
- Can be a scalar, vector, matrix, or multi-dimensional array dependent on your data
  - Ex: In image recognition, an RGB image would be represented as a 3D tensor (64,64,3)



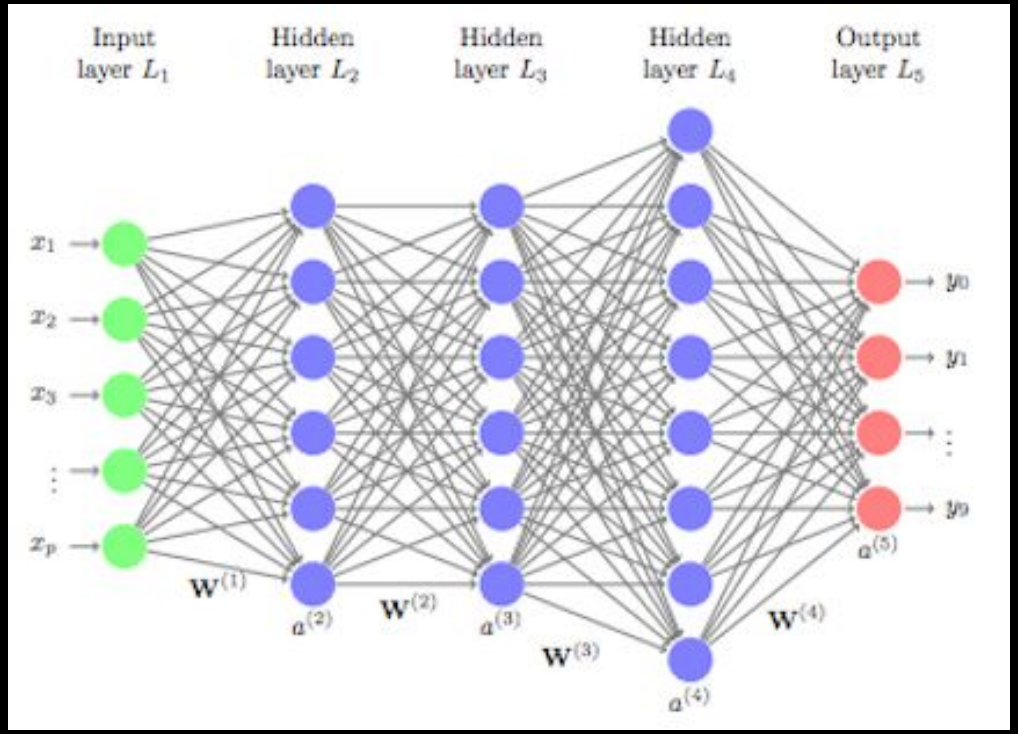

Dimensions of Tensor
1 d - Tensor
2 d - Tensor
3 d -Tensor
4 d -Tensor
5 d -Tensor
6 d -Tensor

NEURAL NETWORKS

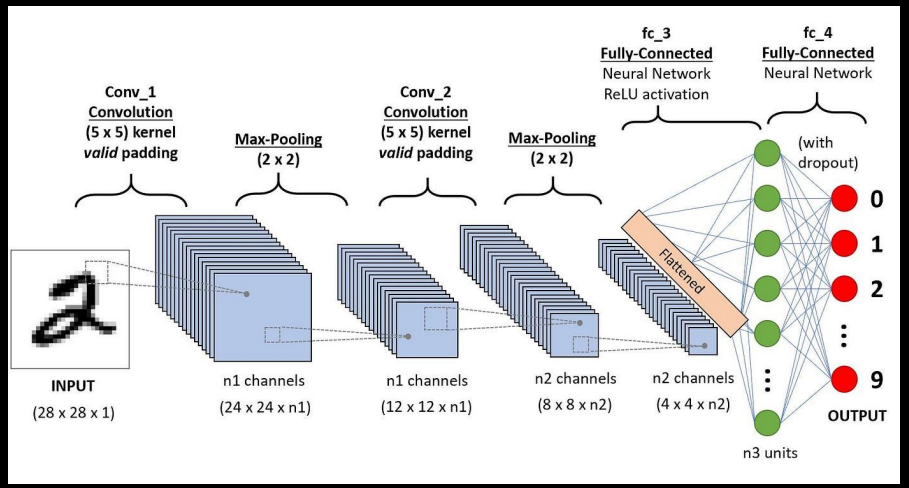
- ➔ A class of machine learning models that were inspired by the structure of the **human mind**

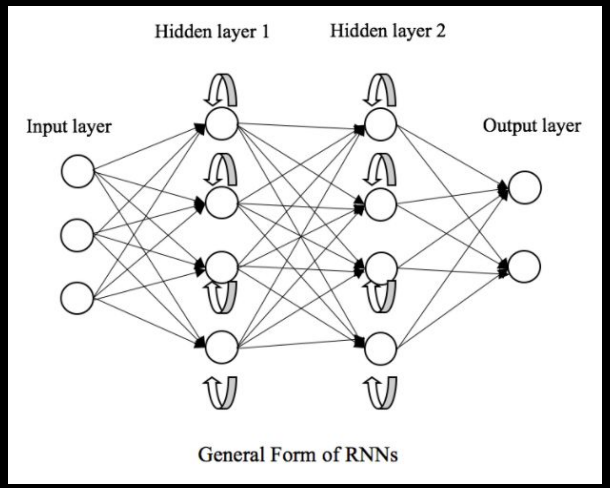
- ➔ They are composed of interconnected units, called **neurons**, that work to together to learn **complex** patterns and relationships within data
  - ◆ Each neuron will take input, process them with weights and biases, and then produce output
  - ◆ Organized into layers(input, hidden layers, output)
- ➔ Used for classification, regression, generations, reinforcement learning, natural language processing, etc.

FEEDFORWARD NEURAL NETWORKS



MANY DIFFERENT TYPES....





Convolutional Neural Networks (CNN)

Recurrent Neural Networks (RNN)

KERAS API AND TENSORFLOW

➔ Core data structures of Keras are **layers** and **models**

tf.keras.layers.Layer - is the fundamental abstraction of Keras and encapsulates a state(weights) and some computation

➔ Model is an object that groups layers together and that can be trained on data

- The simplest model is the Sequential model(tf.keras.models.Sequential) which is a linear stack of layers. But other complex architectures can be built

tf.keras.layers.Model - features built-in training and evaluation methods

tf.keras.Model.fit: trains the model
tf.keras.Model.predict: generates output predictions for input samples
tf.keras.Model.evaluate: returns the loss and metrics value for the model

https://www.tensorflow.org/guide/keras

KERAS API LAYERS

tf.Layers.Dense - Regularly connected NN layers, neurons are connected

tf.Layers.Conv2D - 2D convolutional layer(i.e finds features in the data)
tf.Layers.MaxPooling2D - Max pooling layer for 2D inputs (i.e dimensionality reduction)
tf.Layers.Flatten - flattens the input

Many more...

tf.Layers.RandomContrast - pre-processing layer which randomly adjusts contrast during training
tf.Layers.TextVectorization - pre-processing layer which maps text features to integer sequences

https://www.tensorflow.org/api_docs/python/tf/keras/layers

KERAS API: LOSS FUNCTIONS

- A loss function compares the target and predicted output values and measures how well the neural network models the training data.

- **MeanSquaredError:**
  - Regression
  - Computes the mean of squares of errors between labels and predictions
  - Ex:

```
>>> y_true = [[0., 1.], [0., 0.]]
>>> y_pred = [[1., 1.], [1., 0.]]
>>> # Using 'auto'/'sum_over_batch_size' reduction type.
>>> mse = tf.keras.losses.MeanSquaredError()
>>> mse(y_true, y_pred).numpy()
0.5
```

- **Binary Cross Entropy:**
  - Binary classification
  - Computes the cross entropy loss between true and predicted labels
  - Ex:

```
>>> # Example 1: (batch_size = 1, number of samples = 4)
>>> y_true = [0, 1, 0, 0]
>>> y_pred = [-18.6, 0.51, 2.94, -12.8]
>>> bce = tf.keras.losses.BinaryCrossentropy(from_logits=True)
>>> bce(y_true, y_pred).numpy()
0.865
```

KERAS API: LOSS FUNCTIONS CONT.

- **Categorical Cross Entropy:**
  - Multi-class
  - Computes the cross entropy between true and predicted labels, but is used when there are 2 or more label classes
  - Ex:

```
tf.keras.losses.CategoricalCrossentropy(
    from_logits=False,
    label_smoothing=0.0,
    axis=-1,
    reduction="auto",
    name="categorical_crossentropy",
)
```

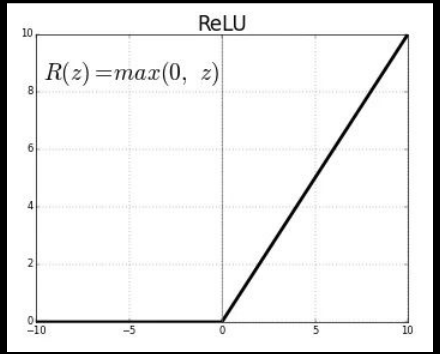
```
>>> y_true = [[0, 1, 0], [0, 0, 1]]
>>> y_pred = [[0.05, 0.95, 0], [0.1, 0.8, 0.1]]
>>> # Using 'auto'/'sum_over_batch_size' reduction type.
>>> cce = tf.keras.losses.CategoricalCrossentropy()
>>> cce(y_true, y_pred).numpy()
1.177
```

KERAS API: ACTIVATION FUNCTIONS

- In terms of neural networks, an activation function compares the input value with a threshold value and determines if the neuron should be activated or not.

- **relu**
  - Rectified linear unit activation function
  - Most used function, but does not map negative values well because it makes any negative value zero therefore decreasing the ability for a model to fit properly
  - Ex:

```
>>> foo = tf.constant([-10, -5, 0.0, 5, 10], dtype = tf.float32)
>>> tf.keras.activations.relu(foo).numpy()
array([ 0.,  0.,  0.,  5., 10.], dtype=float32)
>>> tf.keras.activations.relu(foo, alpha=0.5).numpy()
array([-5. , -2.5,  0. ,  5. , 10. ], dtype=float32)
>>> tf.keras.activations.relu(foo, max_value=5.).numpy()
array([0., 0., 0., 5., 5.], dtype=float32)
>>> tf.keras.activations.relu(foo, threshold=5.).numpy()
array([-0., -0.,  0.,  0., 10.], dtype=float32)
```



  - Takes in an input tensor, a float for slope for values that fall below the threshold, a float that sets the saturation threshold, and a float of the activation function's threshold where values below it will be set to zero. It returns the input tensor transformed after the function is applied.

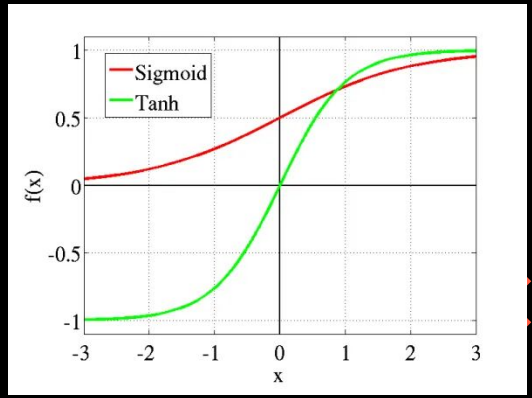
KERAS API: ACTIVATION FUNCTIONS CONT.

- **tanh**
  - Hyperbolic tangent activation function
  - Advantage of this function is that negative values will be mapped as negative and zeros will be mapped near zero
  - Range is from (-1, 1)
  - Classification between two classes
  - Ex:

```
>>> a = tf.constant([-3.0, -1.0, 0.0, 1.0, 3.0], dtype = tf.float32)
>>> b = tf.keras.activations.tanh(a)
>>> b.numpy()
array([-0.9950547, -0.7615942,  0.,  0.7615942,  0.9950547], dtype=float32)
```

- **sigmoid**
  - Used for models that need to predict the probability (0 or 1)
  - sigmoid(x) = 1 / 1 (1 + exp(-x))
  - Ex:

```
>>> a = tf.constant([-20, -1.0, 0.0, 1.0, 20], dtype = tf.float32)
>>> b = tf.keras.activations.sigmoid(a)
>>> b.numpy()
array([2.0611537e-09, 2.6894143e-01, 5.0000000e-01, 7.3105860e-01,
1.0000000e+00], dtype=float32)
```



KERAS API: COMMON OPTIMIZER

- **tf.keras.optimizers.Adam()**
- Stochastic gradient descent based optimization
- Uses momentum and adaptive learning rates to converge to the cost function's minimum
- During training, the function uses the first and second moment to change the learning rate
- First moment is the mean of the gradients and the second moment is the variance of the gradients.
- Maintains an exponentially decaying average of the past and squared gradients through each iteration
- Includes a bias correction mechanism to make sure that initial estimates of moments are near zero.
- Advantages: suitable for large datasets and fast convergence
- Parameters:
  - Learning rate: step size at each iteration during gradient descent
  - Beta 1: exponential decay rate of moment 1 estimates
  - Beta 2: exponential decay rate of moment 2 estimates
  - Epsilon: small constant in denominator to avoid division by zero, 1e-8
  - Decay: learning decay rate over time, 0 means constant

INSTALLATION

- TensorFlow is compatible with the following...
  - Python 3.8-3.11
  - Ubuntu 16.04 or later
  - Windows 7 or later
  - macOS 10.12.6 (Sierra) or later
  - WSL2 via Windows 10 19044 or higher
- Make sure to have the latest pip
  - pip install --upgrade pip
  - **pip install tensorflow**
- To run a TensorFlow container...
  - docker pull tensorflow/tensorflow:latest # Download latest stable image
  - docker run -it -p 8888:8888 tensorflow/tensorflow:latest-jupyter # Start Jupyter server

TensorFlow

DEMO

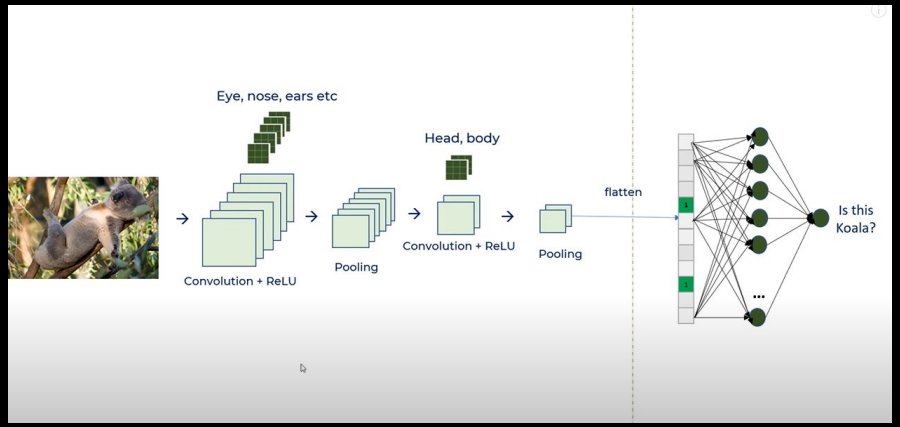
1) Binary Classification
2) Image Classification

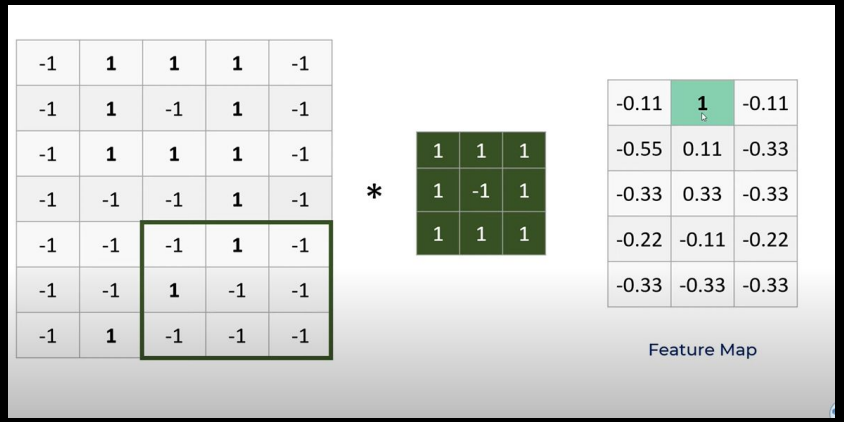
```
Epoch 19/20
21686/21686 [==============================] - 46s 2ms/step - loss: 0.5356 - accuracy: 0.7304 - val_loss: 0.5331 - val_accur
acy: 0.7326
Epoch 20/20
21686/21686 [==============================] - 44s 2ms/step - loss: 0.5353 - accuracy: 0.7304 - val_loss: 0.5343 - val_accur
acy: 0.7324
```

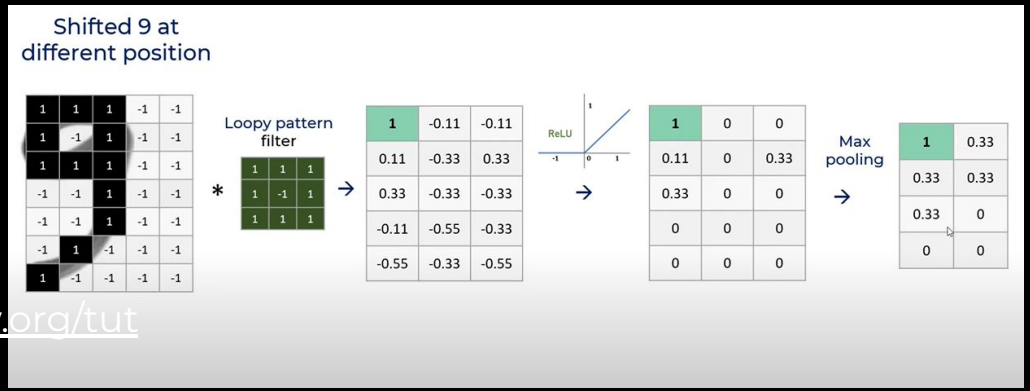
```
model.evaluate(X_test, y_test)
```

```
4647/4647 [==============================] - 7s 1ms/step - loss: 0.5357 - accuracy: 0.7313
9]: [0.5356993675231934, 0.7312746047973633]
```

Convolutional Neural Networks (CNN)







https://www.tensorflow.org/tutorials/images/cnn

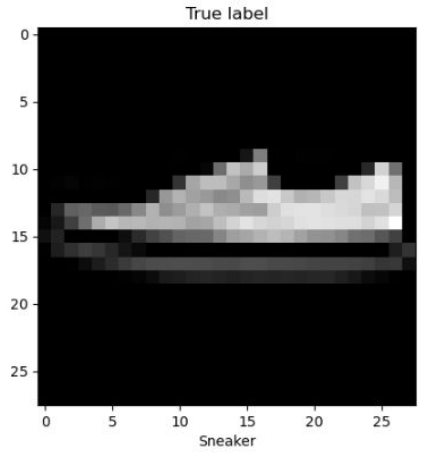
```
Epoch 9/10
1875/1875 [==============================] - 41s 22ms/step - loss: 0.1957 - accuracy: 0.9287 - val_loss: 0.3270 - val_accura
cy: 0.8950
Epoch 10/10
1875/1875 [==============================] - 44s 23ms/step - loss: 0.1879 - accuracy: 0.9297 - val_loss: 0.3379 - val_accura
cy: 0.8990
```

```
cnn.evaluate(X_test, y_test)
```

```
157/157 [==============================] - 1s 5ms/step - loss: 0.3223 - accuracy: 0.9016

[0.3223351240158081, 0.9016000032424927]
```

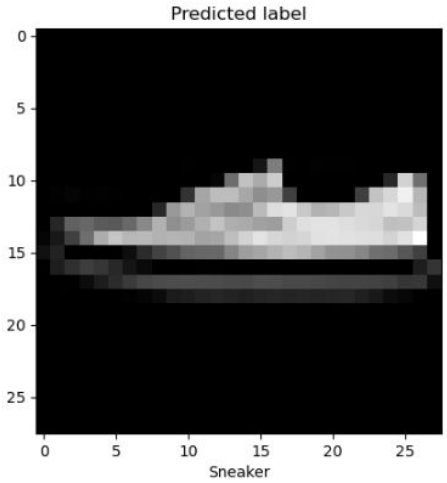
```
plt.title("True label")
plt.imshow(X_test[100],cmap='gray')
plt.xlabel(classes[y_test[100]]) #Show image and the corres
```

5]: Text(0.5, 0, 'Sneaker')




```
plt.title("Predicted label")
plt.imshow(X_test[100],cmap='gray')
plt.xlabel(classes[y_pred_label[100]]) #Show image and th
```

Text(0.5, 0, 'Sneaker')

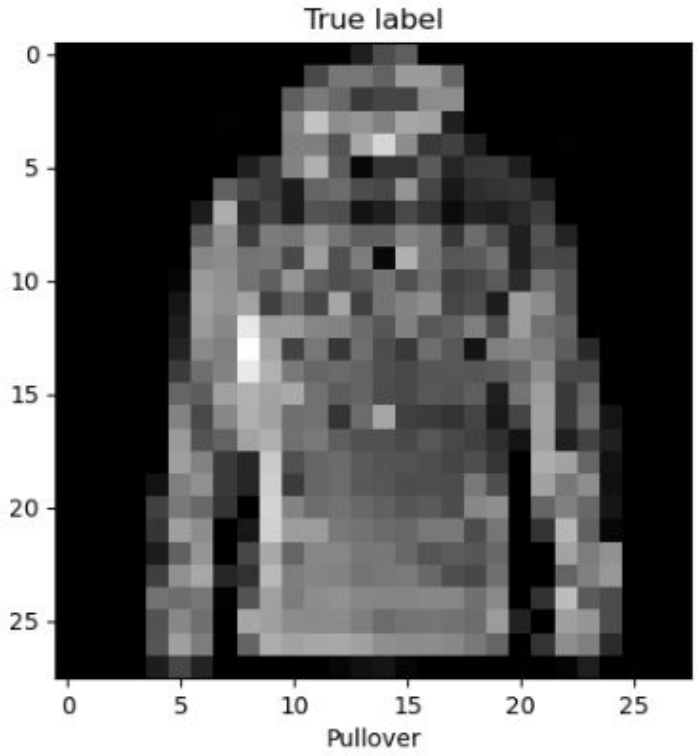



```
y_pred = cnn.predict(X_test) #use model to predict labels of in

print(y_pred.shape) #y_pred has 10 columns which each represent

y_pred_label = [np.argmax(element)  for element in y_pred] #fir


print(y_test[:10])
print(y_pred_label[:10])
```

```
157/157 [==============================] - 1s 5ms/step
(5000, 10)
[0 0 2 1 2 8 8 6 3 5]
[0, 0, 4, 1, 2, 8, 8, 6, 3, 5]
```

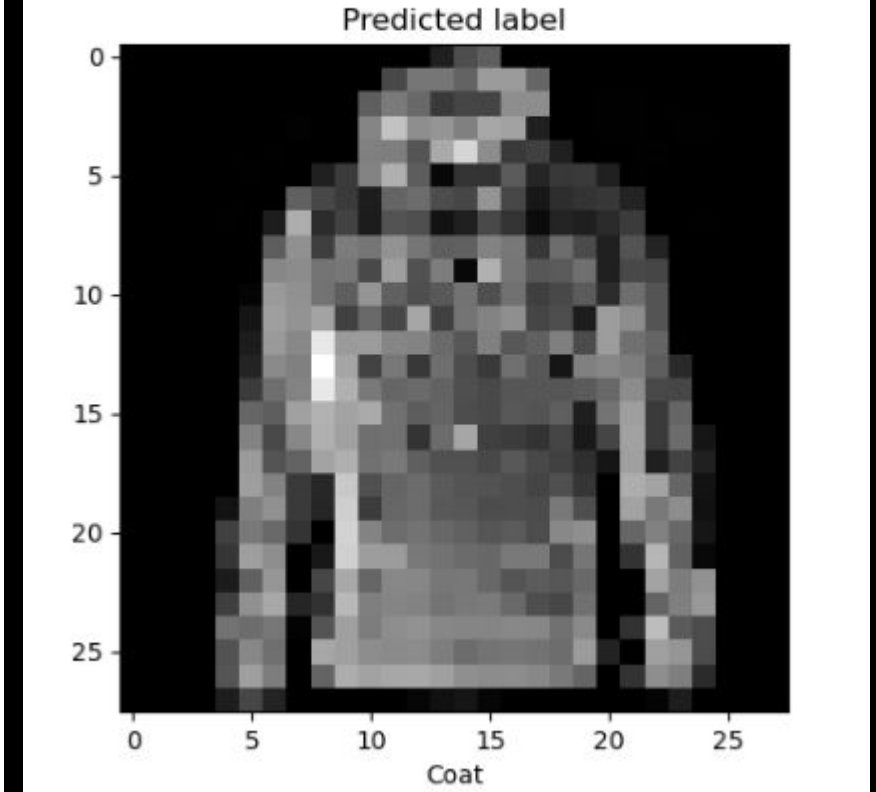
```
plt.title("True label")
plt.imshow(X_test[2],cmap='gray')
plt.xlabel(classes[y_test[2]]) #Show image and the corresponding lab
```

```
]: Text(0.5, 0, 'Pullover')
```




```
plt.title("Predicted label")
plt.imshow(X_test[2],cmap='gray')
plt.xlabel(classes[y_pred_label[2]]) #Show image and the co
```

```
: Text(0.5, 0, 'Coat')
```



QUESTIONS?

RESOURCES

- https://www.tensorflow.org/about
- https://slidesgo.com/theme/hazard-analysis-and-risk-assessment-consulting#search-Data&position-6&results-79
- https://www.youtube.com/watch?v=L35fFDpwIM4
- https://www.kaggle.com/datasets/sooyoungher/smoking-drinking-dataset
- https://www.youtube.com/watch?v=7HPwo4wnJeA
- https://www.youtube.com/watch?v=zfiSAzpy9NM
- https://www.tensorflow.org/tutorials/images/cnn
- https://www.tensorflow.org/install
- https://towardsdatascience.com/loss-functions-and-their-use-in-neural-networks-a470e703f1e9#:~:text=A%20loss%20function%20is%20a,the%20predicted%20and%20target%20outputs.
- https://keras.io/api/losses/
- https://www.v7labs.com/blog/neural-networks-activation-functions#:~:text=An%20Activation%20Function%20decides%20whether,prediction%20using%20simpler%20mathematical%20operations.
- https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6
- https://keras.io/api/layers/activations/
- https://spotintelligence.com/2023/03/01/adam-optimizer/#What_is_the_Adam_optimizer
- https://www.guru99.com/tensor-tensorflow.html
- https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414
- https://www.mygreatlearning.com/blog/types-of-neural-networks/