



Advanced Microprocessors

CLASSIFICATION

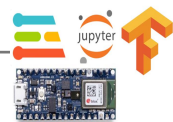
Dennis A. N. Gookyi

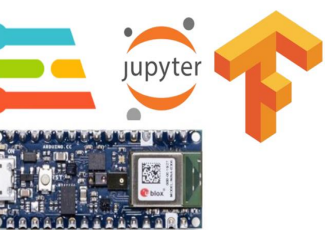




CONTENTS

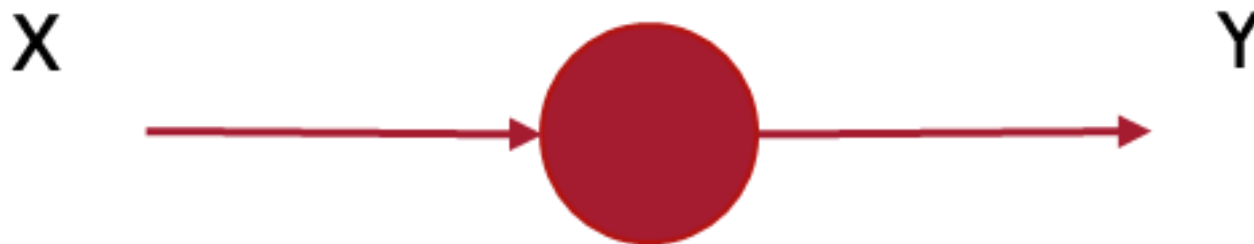
❖ Classification

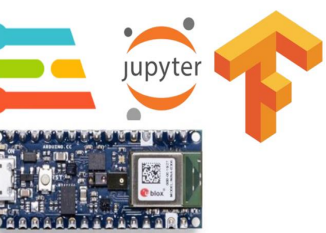




CLASSIFICATION

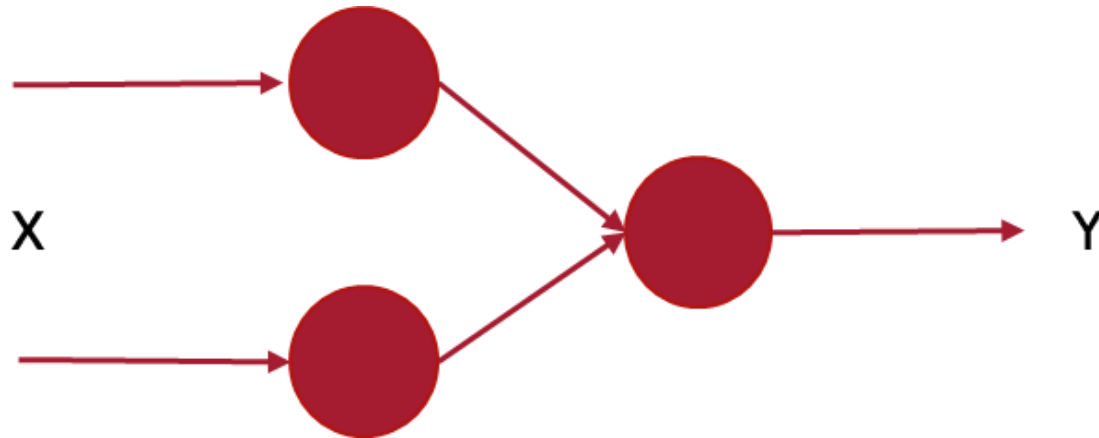
❖ Classification

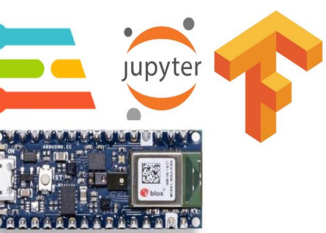




CLASSIFICATION

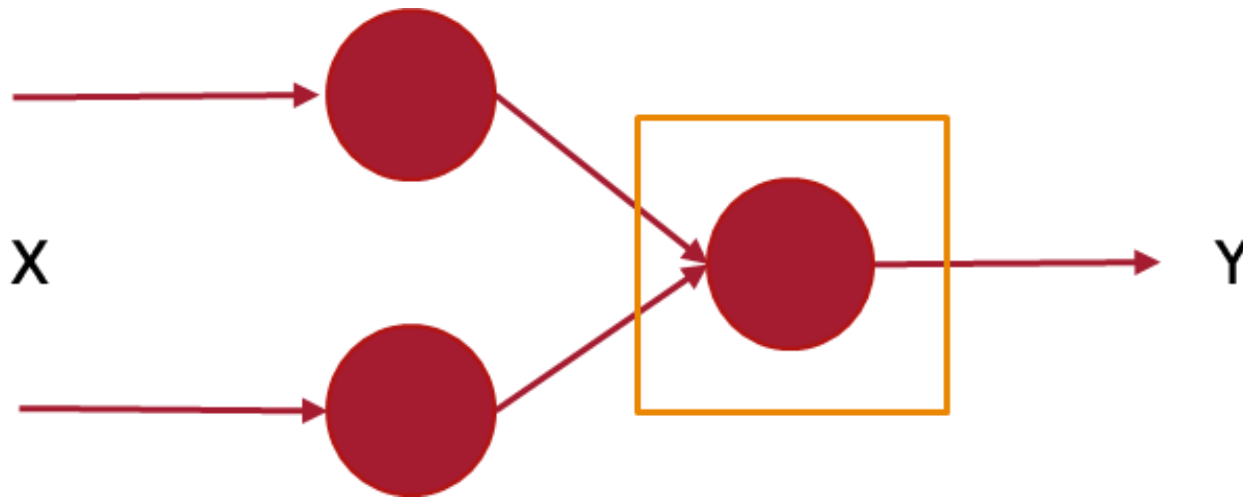
❖ Classification





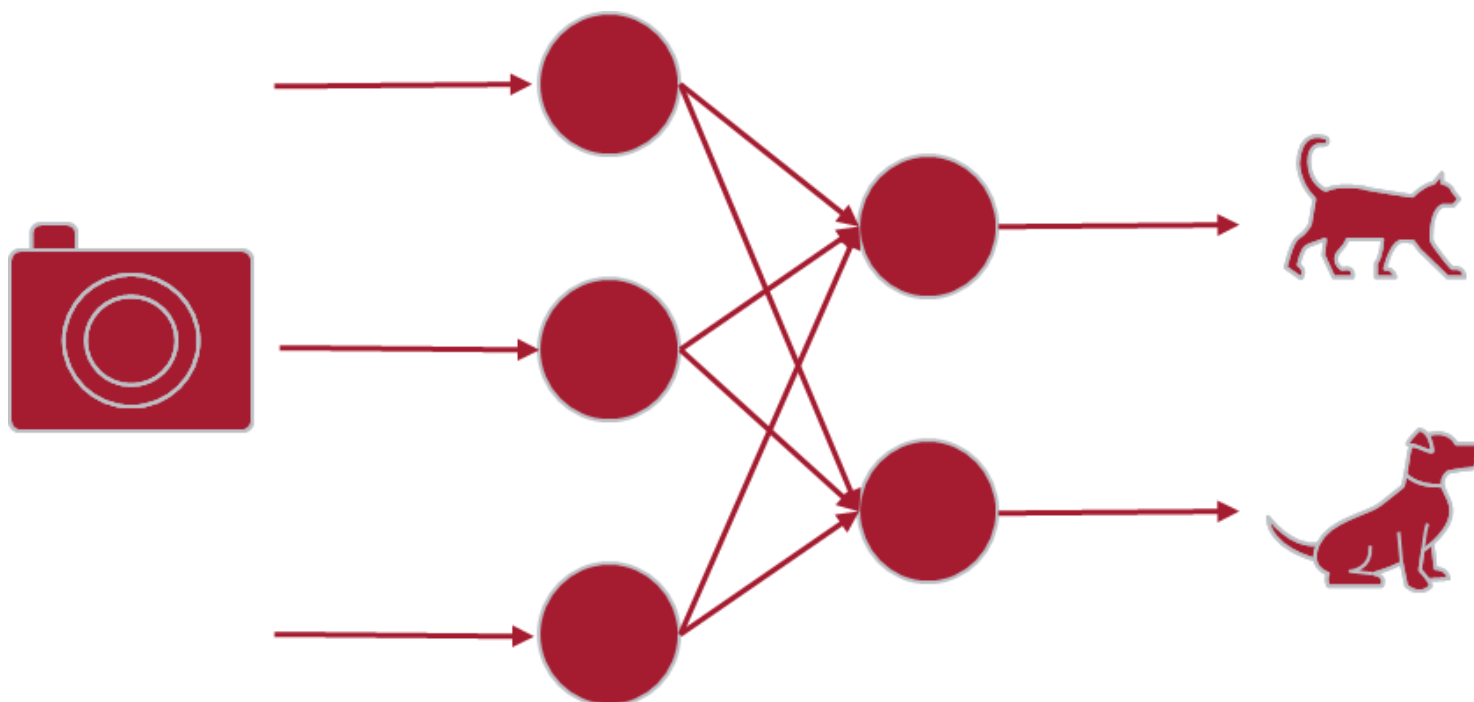
CLASSIFICATION

❖ Classification



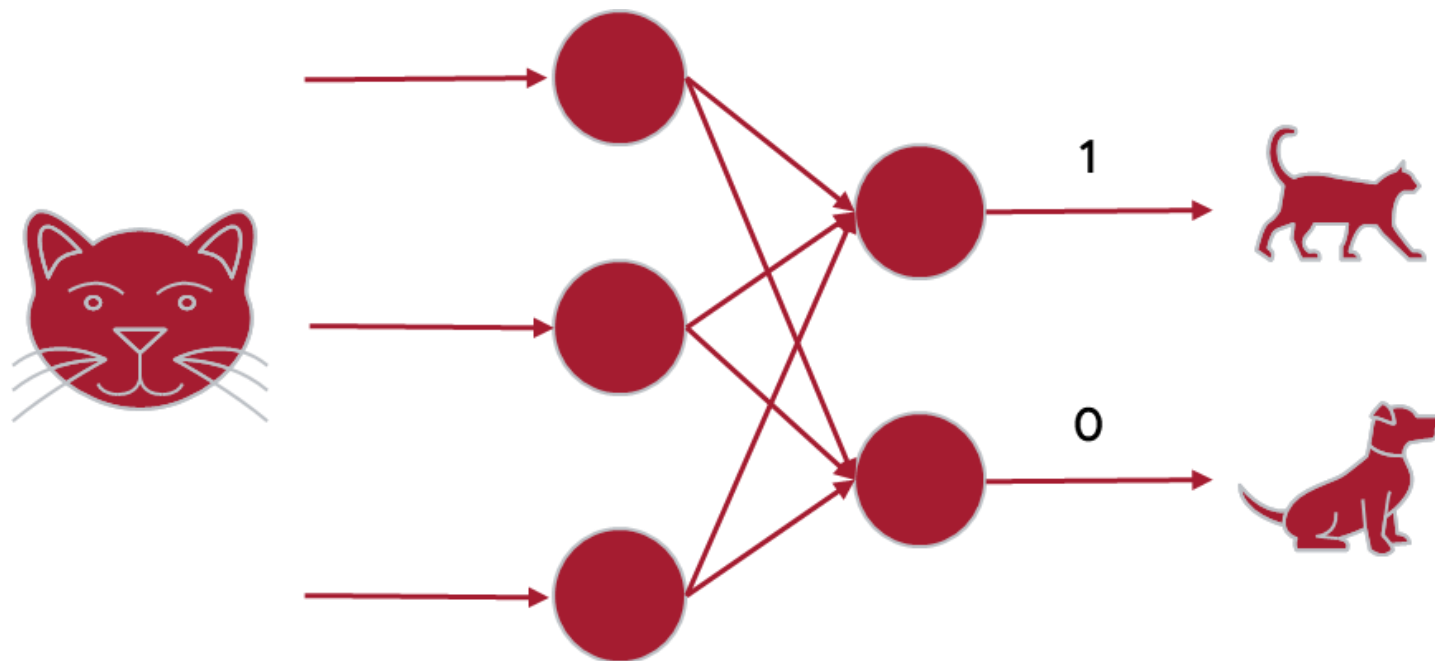
CLASSIFICATION

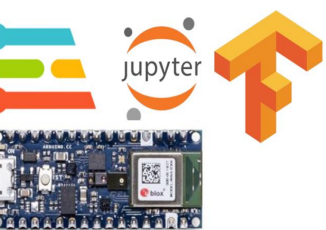
❖ Classification



CLASSIFICATION

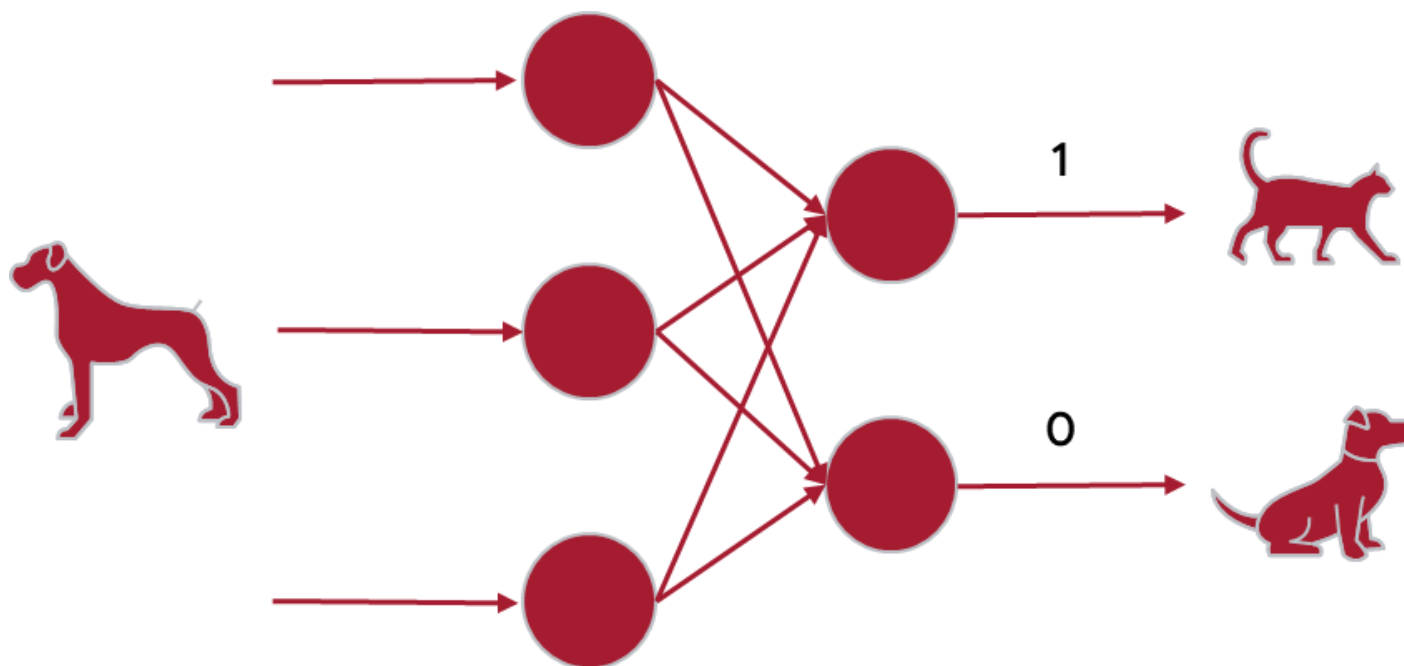
❖ Classification





CLASSIFICATION

❖ Classification





CLASSIFICATION

❖ Classification

Data

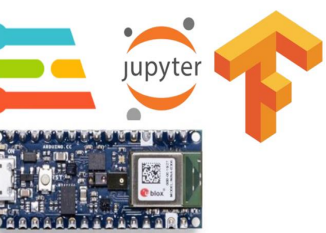
Label



$[1, 0]$



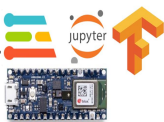
$[0, 1]$

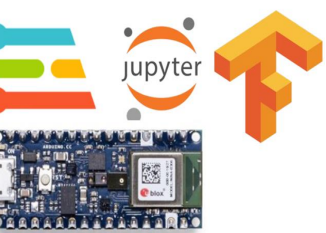


CLASSIFICATION

❖ Classification

0	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
1	[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
2	[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
3	[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
4	[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
5	[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
6	[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
7	[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
8	[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
9	[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]





CLASSIFICATION

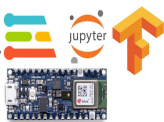
❖ Classification

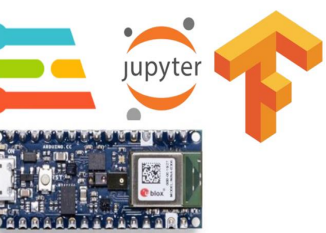
```
import tensorflow as tf

data = tf.keras.datasets.mnist
(training_images, training_labels), (val_images, val_labels) = data.load_data()

training_images = training_images / 255.0
val_images = val_images / 255.0

model = tf.keras.models.Sequential(
    [tf.keras.layers.Flatten(input_shape=(28,28)),
     tf.keras.layers.Dense(20, activation=tf.nn.relu),
     tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
```





CLASSIFICATION

❖ Classification

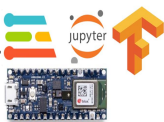
```
import tensorflow as tf

data = tf.keras.datasets.mnist
(training_images, training_labels), (val_images, val_labels) = data.load_data()

training_images = training_images / 255.0
val_images = val_images / 255.0

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28,28)),
    tf.keras.layers.Dense(20, activation=tf.nn.relu),
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
```

Collect
Data





CLASSIFICATION

❖ Classification



60,000 Labelled Training Examples
10,000 Labelled Validation Examples



CLASSIFICATION

❖ Classification

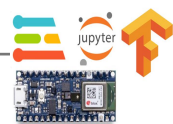
```
import tensorflow as tf

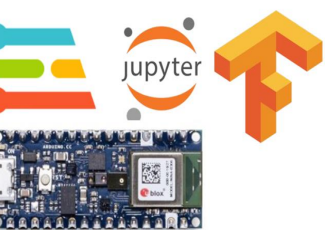
data = tf.keras.datasets.mnist
(training_images, training_labels), (val_images, val_labels) = data.load_data()

training_images = training_images / 255.0
val_images = val_images / 255.0

model = tf.keras.models.Sequential(
    [tf.keras.layers.Flatten(input_shape=(28,28)),
     tf.keras.layers.Dense(20, activation=tf.nn.relu),
     tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
```

Preprocess
Data





CLASSIFICATION

❖ Classification

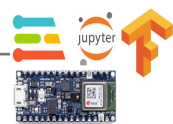
```
import tensorflow as tf

data = tf.keras.datasets.mnist
(training_images, training_labels), (val_images, val_labels) = data.load_data()

training_images = training_images / 255.0
val_images = val_images / 255.0

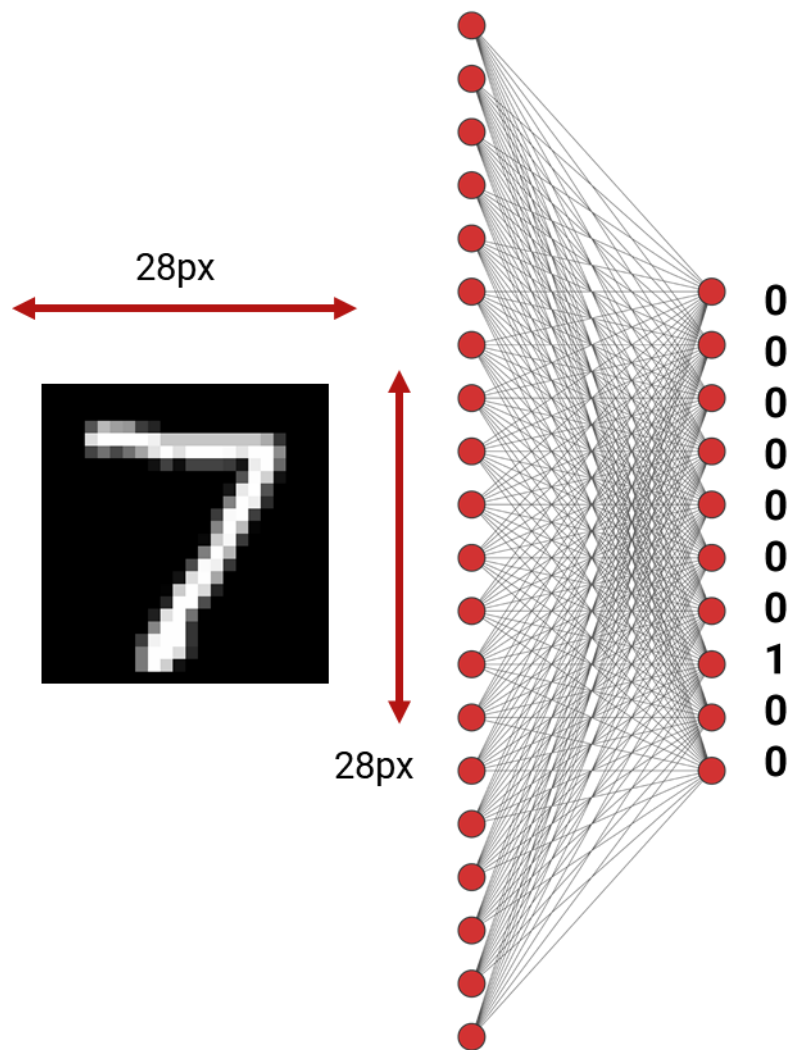
model = tf.keras.models.Sequential(
    [tf.keras.layers.Flatten(input_shape=(28,28)),
     tf.keras.layers.Dense(20, activation=tf.nn.relu),
     tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
```

Design a
Model



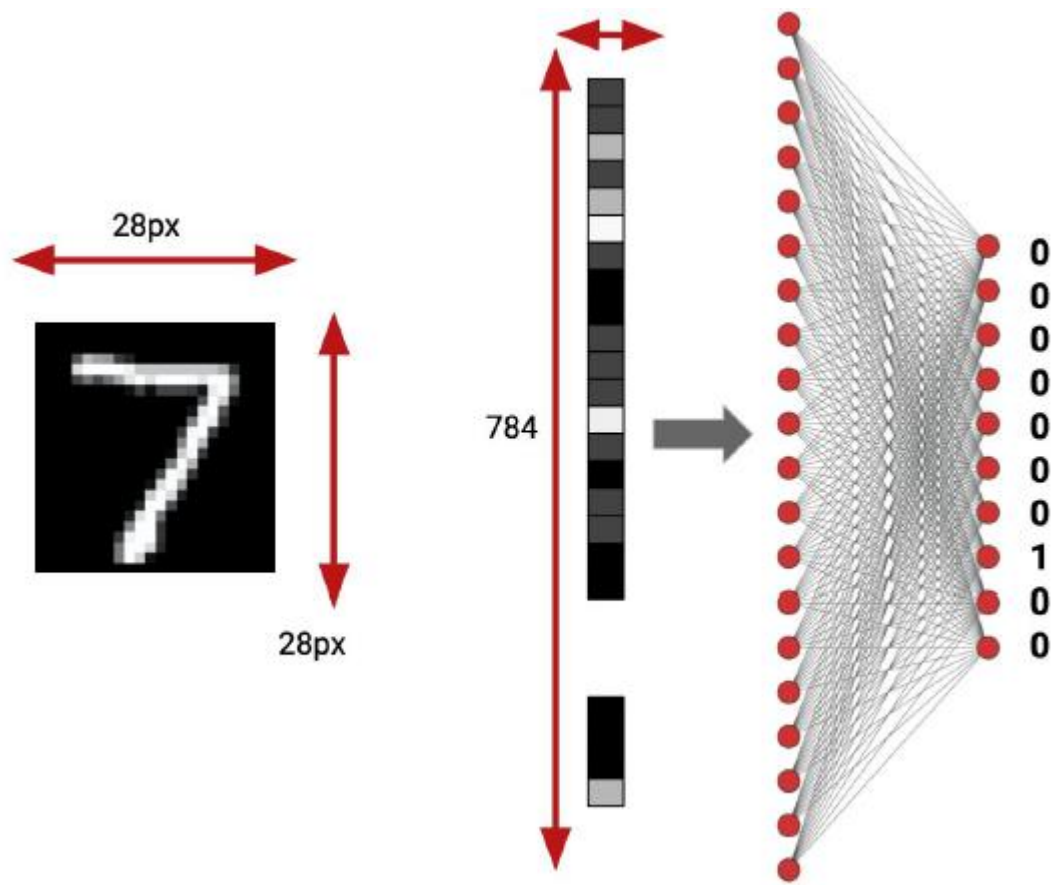
CLASSIFICATION

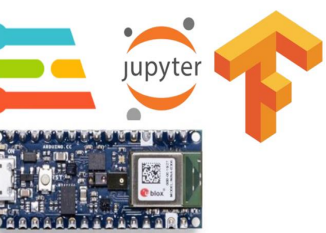
❖ Classification



CLASSIFICATION

❖ Classification





CLASSIFICATION

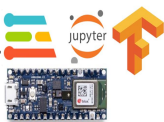
❖ Classification

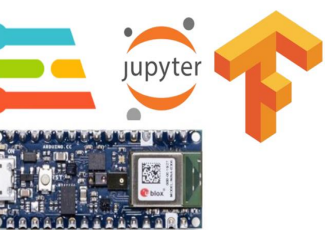
```
import tensorflow as tf

data = tf.keras.datasets.mnist
(training_images, training_labels), (val_images, val_labels) = data.load_data()

training_images = training_images / 255.0
val_images = val_images / 255.0

model = tf.keras.models.Sequential(
    [tf.keras.layers.Flatten(input_shape=(28,28)),
     tf.keras.layers.Dense(20, activation=tf.nn.relu),
     tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
```





CLASSIFICATION

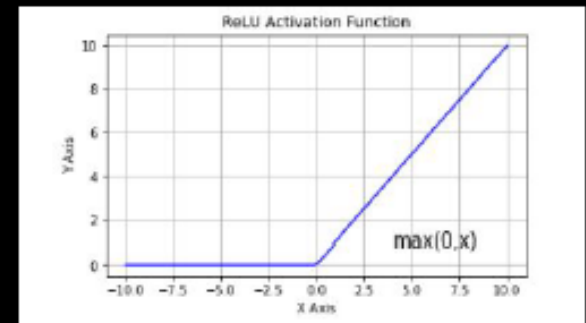
❖ Classification

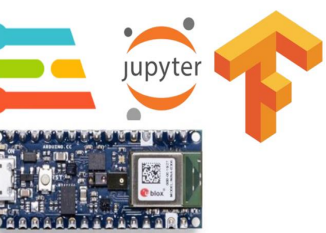
```
import tensorflow as tf

data = tf.keras.datasets.mnist
(training_images, training_labels), (val_images, val_labels) = data.load_data()

training_images = training_images / 255.0
val_images = val_images / 255.0

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28,28)),
    tf.keras.layers.Dense(20, activation=tf.nn.relu),
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
```





CLASSIFICATION

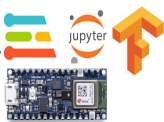
❖ Classification

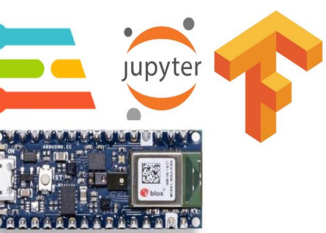
```
import tensorflow as tf

data = tf.keras.datasets.mnist
(training_images, training_labels), (val_images, val_labels) = data.load_data()

training_images = training_images / 255.0
val_images = val_images / 255.0

model = tf.keras.models.Sequential(
    [tf.keras.layers.Flatten(input_shape=(28,28)),
     tf.keras.layers.Dense(20, activation=tf.nn.relu),
     tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
```





CLASSIFICATION

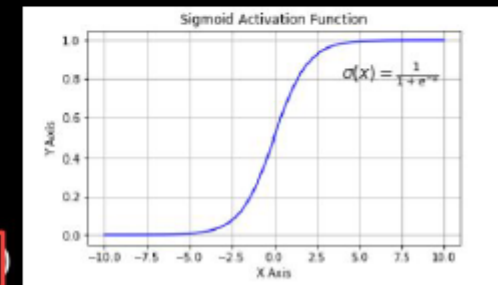
❖ Classification

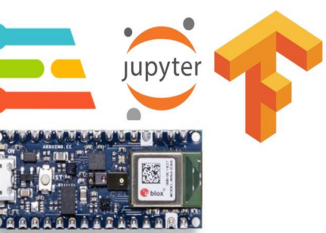
```
import tensorflow as tf

data = tf.keras.datasets.mnist
(training_images, training_labels), (val_images, val_labels) = data.load_data()

training_images = training_images / 255.0
val_images = val_images / 255.0

model = tf.keras.models.Sequential(
    [tf.keras.layers.Flatten(input_shape=(28,28)),
     tf.keras.layers.Dense(20, activation=tf.nn.relu),
     tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
```



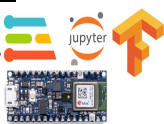


CLASSIFICATION

❖ Classification

```
model.compile(optimizer='adam',  
              loss='sparse_categorical_crossentropy',  
              metrics=['accuracy'])
```

Design a
Model



CLASSIFICATION

❖ Classification

```
model.compile(optimizer='adam',  
              loss='sparse_categorical_crossentropy',  
              metrics=['accuracy'])
```

Mean Squared Error

$$MSE = \frac{1}{N} \sum (t_i - s_i)^2$$

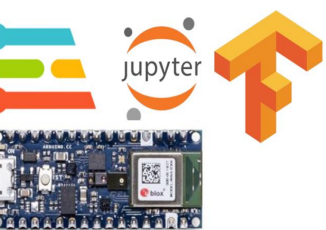
Prediction s_i
Ground Truth t_i

Cross Entropy Loss

$$CE = - \sum_i^C t_i \log(s_i)$$

Classes C
Prediction s_i
Ground Truth $\{0,1\}$ t_i

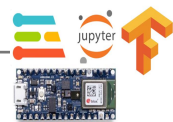
Design a
Model



CLASSIFICATION

❖ Classification

```
model.compile(optimizer='adam',  
              loss='sparse_categorical_crossentropy',  
              metrics=['accuracy'])  
  
model.fit(training_images, training_labels, epochs=20)
```

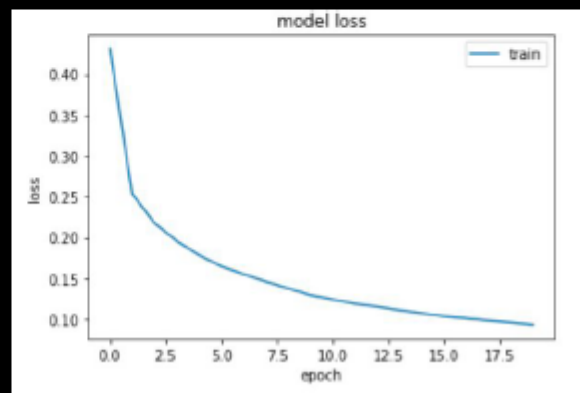
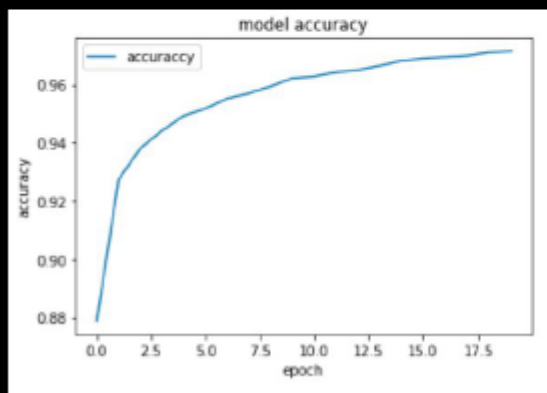




CLASSIFICATION

❖ Classification

```
model.compile(optimizer='adam',  
              loss='sparse_categorical_crossentropy',  
              metrics=['accuracy'])  
  
model.fit(training_images, training_labels, epochs=20)
```





CLASSIFICATION

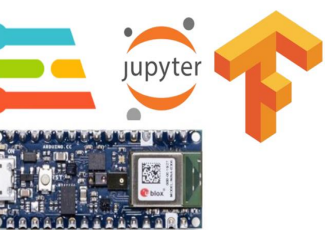
❖ Classification

```
classifications = model.predict(val_images)
print(classifications[0])
print(test_labels[0])
```

```
[2.4921512e-09 1.3765138e-10 8.8281205e-08
1.0477231e-03 2.8455029e-12 4.0820678e-06
2.0070659e-16 9.9894780e-01 1.0296049e-07
2.9972372e-07]
```

7





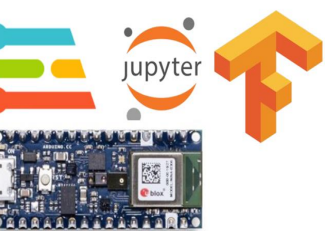
CLASSIFICATION

❖ Classification

Digits Classification using DNN with TF2
Code Time!

TF_MNIST_Classification.ipynb





CLASSIFICATION

❖ Going deeper with Deep learning

- Initializing neural networks
 - <https://www.deeplearning.ai/ai-notes/initialization/>
- Neural networks – PlayList - 3Blue1Brown
 - https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
- An introductory lecture for MIT course 6.S094 by Prof. Lex Fridman
 - <https://youtu.be/O5xeyoRL95U>
- A Complete Machine Learning Package by Jean de Dieu Nyandwi
 - https://github.com/Nyandwi/machine_learning_complete

