Image classification of the fashion MNIST dataset by building a Neural network with Tensorflow

By
O.G.V.V Bandara

Department of Mathematics
University of Ruhuna
Matara.

2023

Acknowledgement

This project was done as a passion project. The main focus was to explore the functionalities of the TensorFlow environment and it's functionalities. Also to identify the different functions of the "keras" library in TensorFlow. Image classification is an essential functionality in many instances. One of the main instances being "search". Image classification is essential for search engines and the main objective of the project was to explore how to classify images using TensorFlow and further apply the functionality.

Abstract

The fashion MNIST dataset was used to train a neural network. The fashion MNIST data set was seperated into training set and the testing set. The neural network built has the ability to classify images. The model predicts the images with a 0.91 accuracy. The code has the ability to obtain an input value which indicates the object of the training set of the fashion MNIST dataset to be tested and then predict what the object is.

Contents

Chapter 1

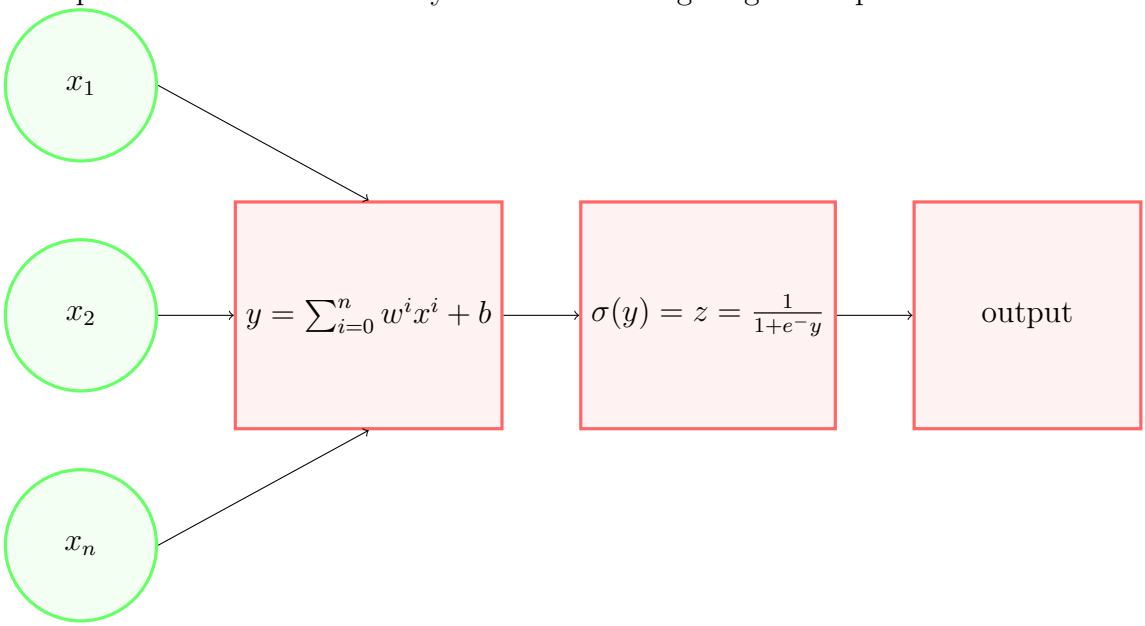
Introduction

1.1 Background of study

1.1.1 Neural networks

what is a neuron?

A neural network consists of multiple neurons. These neurons act as functions that take in an input and provide an output. The output of a neuron is used for the inputs of the neurons of the next layer of a neural network. The neural network consists of multiple neurons and neuron layers. The following diagram depicts a neuron.



$x_1, x_2, ..., x_n$= input pramaters/features

y= dependent variable

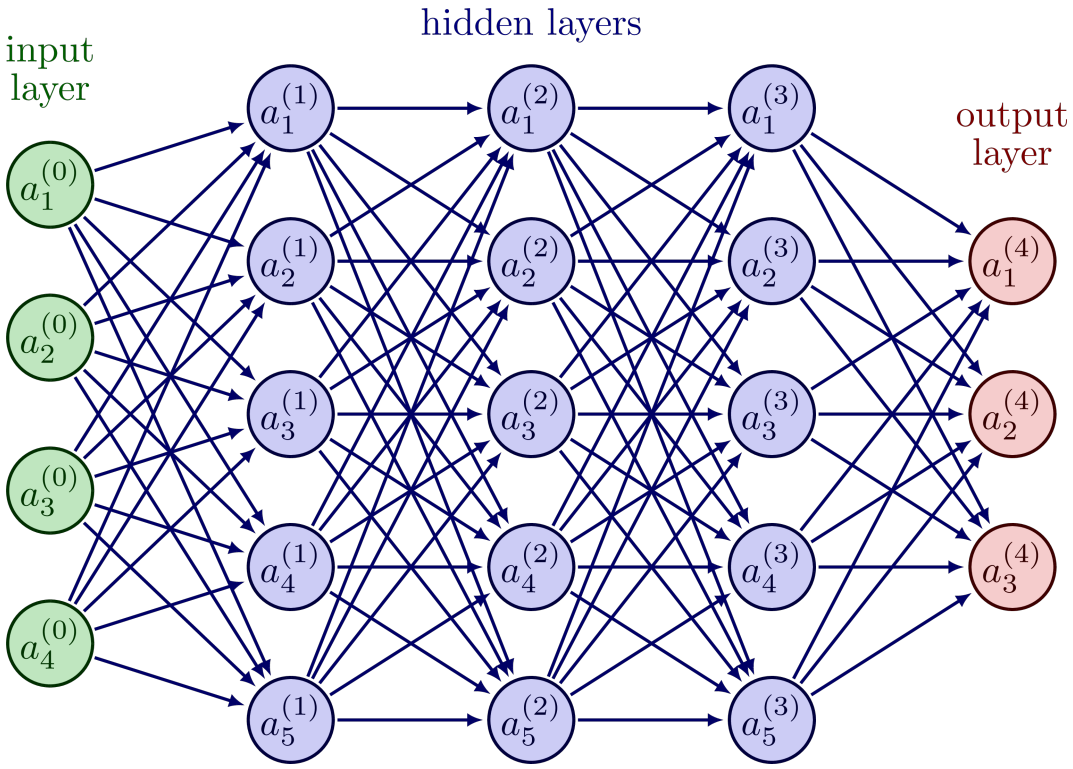
$\sigma(y)$=activation function

What is a neural network?

A neural network is a combination of multiple neurons. each of these neurons are stacked as layers and a neural network consists of multiple layers.

- input layer: features and labels
- hidden layers: intermediate neurons
- ouput layer: output values

The following figure depicts the stucture of a neural network.



1.2 Objectives

The major objective of the project is to perform an image classification using tensorflow. Some other objectives can be listed as,

- Object recognition: Image classification is widely used in object recognition systems, where it is used to identify and classify objects in images and videos.
- Medical diagnosis: Image classification is used in medical diagnosis to classify medical images such as X-rays, MRIs, and CT scans, to detect various diseases and abnormalities.
- Security and surveillance: Image classification is used in security and surveillance systems to classify and identify individuals, vehicles, and other objects in images and videos.
- Autonomous driving: Image classification is used in autonomous driving systems to identify and classify road signs, traffic lights, pedestrians, and other objects in the environment.
- Social media: Image classification is used in social media to classify images and videos, enabling content recommendation and personalized ads.

Chapter 2

Methodology

2.1 Nature of data

for this project we will be using the fashion MNIST datset that is inbuilt within tensorflow. The fashion MNIST dataset contains a variety of images. The datset can be divided into the training set and the test set where the training set contains 60,000 image sand the test set contains 10,000 images.

- The train_images and train _labels arrays are the training set—the data the model uses to learn.
- The model is tested against the test set, the test _images, and test _labels arrays

The total data is comprised of features and labels.

Features

features are the independent variables that act as inputs in machine learning algorithms. In a structured a single column can be considered as a feature. The features in the dataset vary according to the algorithm.

Labels

The data is raw and unprocessed. therefore labels are added to the raw data for easier identification and to be easily used in prediction models. Labels are also known as tags, which are used to give an identification to a piece of data and tell some information about that element.Labels are sometimes the final outputs of prediction. The model is fit with the training data set which comprises of labels and the testing datasets are used for label predicition.

The features and labels of our example the fashion _MNIST data-set can be considered as follows.

Label	Class
0	Tshirt/Top
1	Trouser
2	Pullover
3	dress
4	Coat
5	Sandal
6	shirt
7	Sneaker
8	Bag
9	Anckle boot

2.2 Gradient Descent

A key concept used in neural networks is the concept of gradient descent. The figure below shows the gradient descent in 2 dimensions.
The goal of an algorithm is to predict the function that would be used as a classifier or predictor. The function uses two variables which are the independent variables and the dependent variables. The gradient descent is calculated using the "cost function" or also known as the mean squared error.

$$M.S.E = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_{pred})^2 \tag{2.1}$$

y_i= The actual dependent variable values

y_{pred}= predicted dependent variable values

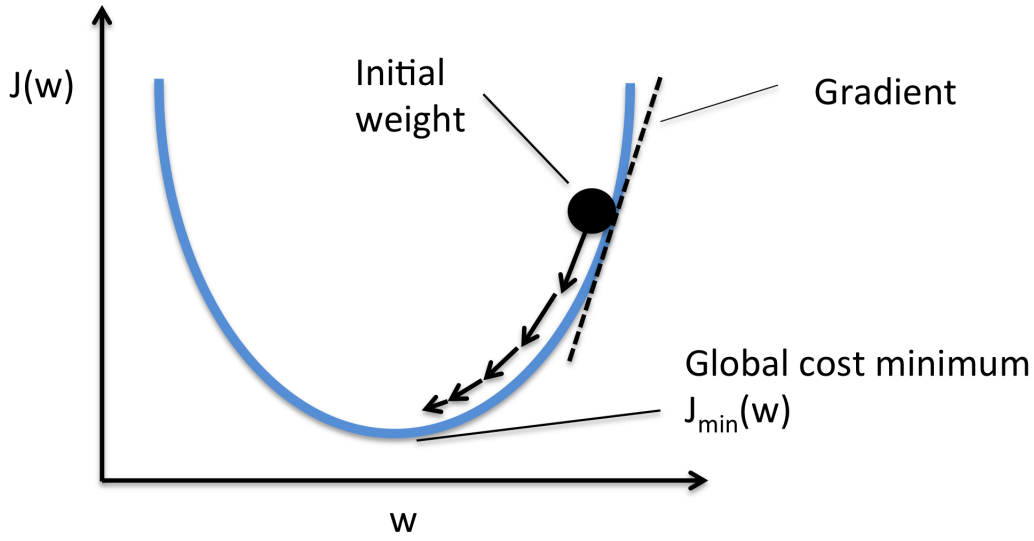
M.S.E = mean squared error/cost function

n=number of values

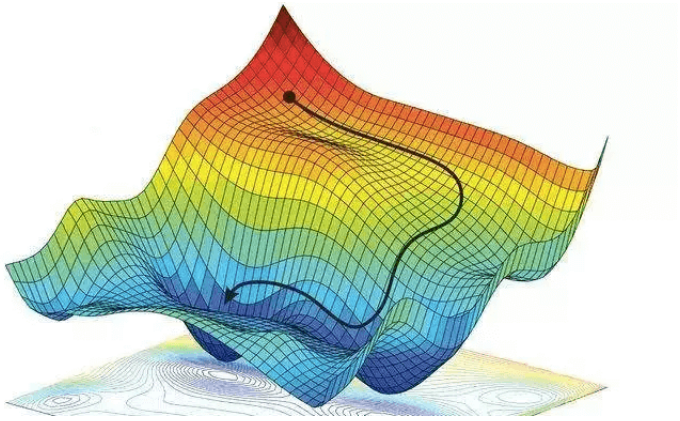
y_pred is replaced by $mx_i + b$. then

$$M.S.E = \frac{1}{n}\sum_{i=1}^{n}(y_i - (mx_i + b))^2 \tag{2.2}$$

The objective of gradient descent is to estimate the values of "m" and "b", hence find the best fit line for the training set. The follwing figure shows the gradient descent in 3 dimensions where
$j(w)$= the cost function
The objective pf the gradient descent algorithm is to find $j_{min}(w)$



The following figure shows the gradient descent in 3 dimensions. The y axis is the cost function and the other two axes represent the gradient and intecept also referred to as the coefficient and intercept. The objective of a machine learning algorithm is to predict values by reducing the cost function as much as possible. In other words mean to find the minima of the cost function.



2.2.1 Batch gradient descent

Consider the machine learning model to contain "n" number of variables. And let the independent variables be $x_1, x_2, ..., x_n$ respectively. The dependent variable be "y". And also let the weights be $w_1, w_2, ..., w_n$ respectively. The bias is "b".
The relationships between dependent variable and independent variables can be written as

$$y = \sum_{i=1}^{n} w^i x^i + b \tag{2.3}$$

If the training set has "p" number of samples. The model will run through all the samples. initially the weights and biases are set as
$w_1, w_2, ..., w_n$=1
b=1 y=the actual value of the dependent variable of the training set

first sample

$$\hat{y_1} = 1 * x_1 + 1 * x_2 + ... + 1 * x_n + b \tag{2.4}$$

$$error1 = (y_1 - \hat{y_1})^2 \tag{2.5}$$

second sample

$$\hat{y_2} = 1 * x_1 + 1 * x_2 + ... + 1 * x_n + b \tag{2.6}$$

$$error2 = (y_2 - \hat{y_2})^2 \tag{2.7}$$

p sample

$$\hat{y_p} = 1 * x_1 + 1 * x_2 + ... + 1 * x_n + b \tag{2.8}$$

$$error(p) = (y_p - \hat{y_p})^2 \tag{2.9}$$

when all the samples are evaluated and the error is calculated it is known as an ***e***poch at the end of the epoch,
total error=error1+error2+.....+error(p)

mean squared error=$\frac{totalerror}{p}$ The weights and biases are adjusted after the first epoch

$$w_1 = w_1 - learning\ rate * \frac{\partial(M.S.E)}{\partial w_1} \tag{2.10}$$

$$. \tag{2.11}$$

$$. \tag{2.12}$$

$$. \tag{2.13}$$

$$w_n = w_n - learning\ rate * \frac{\partial(M.S.E)}{\partial w_n} \tag{2.14}$$

$$b = b - learning\ rate * \frac{\partial(M.S.E)}{\partial b} \tag{2.15}$$

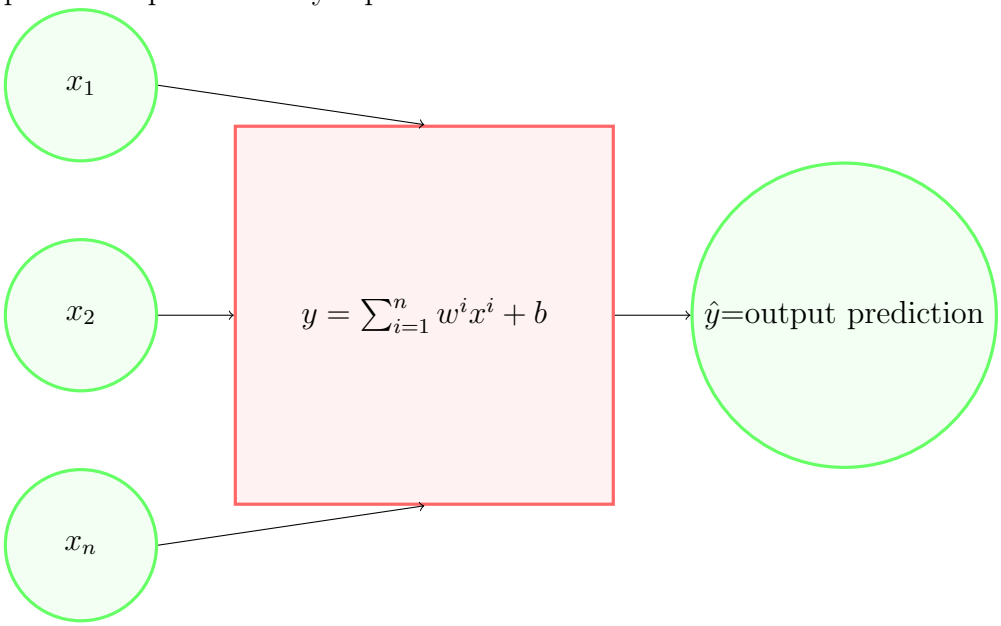
Now the values of $w_1, w_2, ..., w_n$ have changed and the next ***epoch*** will begin with the new weights. The epoch is again repeated for each of the samples of the "p" samples. The objective is to find the minimum of the loss function.

2.2.2 stochastic gradient descent

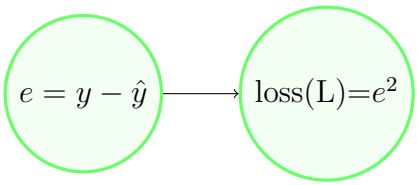
When the training set is very large and with many features, it requires heavy computation to perform "batch gradient descent". Therefore the process is performed by choosing random samples from the training set. The process is perform such that the minimum of the loss function is achieved.

2.3 Chain rule for neural networks

The input variables and output variables can be shown in a diagram which shows the process of prediction by input into a function.



The loss function is obtained from the predicted $\hat{y}$ value and the y value



we need to find the change in the weight with respctive to the loss to be able to make a perfect model with minimum loss
step1:

$$\frac{\partial L}{\partial e} = 2e \tag{2.16}$$

step2:

$$\frac{\partial e}{\partial \hat{y}} = -1 \tag{2.17}$$

step3:

$$\frac{\partial \hat{y}}{\partial w_1} = \hat{x_1} \tag{2.18}$$

$$\frac{\partial \hat{y}}{\partial w_2} = \hat{x_2} \tag{2.19}$$

$$... \tag{2.20}$$

$$\frac{\partial \hat{y}}{\partial w_n} = \hat{x_n} \tag{2.21}$$

Therefore using the chain rule

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial e} * \frac{\partial e}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial w_1} \tag{2.22}$$

similarly for all n

$$\frac{\partial L}{\partial w_n} = \frac{\partial L}{\partial e} * \frac{\partial e}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial w_n} \tag{2.23}$$

therefore the rate of change of w_1 with respective to w_1 is

$$\frac{\partial L}{\partial w_1} = -2ex_1 \tag{2.24}$$

2.4 backpropogation

During backpropogation the nature of how the weights and biases of a training set should be tweeked is found. using the chain rule the weights$(w_1, w_2, ..., w_n)$ are adjusted such that the cost function is minimum.starting from the output layer the weights and biases are adjusted backwards along each hidden layer such that the input layer weight and biases are adjusted. By minimizing the cost function the accuracy of the prediction $\hat{y}$ can be increased.

2.5 Layers

2.5.1 Flatten layer

The "Flatten" function is used to convert multidimensioanl input data into 1 dimensional input data. Usually used as the first layer in inputs which are mutidimensional such as images.

```
tf.keras.layers.Flatten(data_format=None, **kwargs)
```

2.5.2 Dense layer

Fully connected layer in tensorflow Keras API. Creates a layer of neurons where every neron is connected to every neuron in the previous layer.

```
tf.keras.layers.Dense(
    units,
    activation=None,
    use_bias=True,
    kernel_initializer="glorot_uniform",
    bias_initializer="zeros",
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    **kwargs
)
```

The arguments used for the code are

- units: Positive integer, dimensionality of the output space.
- activation: Activation function to use. If you don't specify anything, no activation is applied (ie. "linear" activation: a(x) = x).
- use_bias: Boolean, whether the layer uses a bias vector.
- kernel _initializer: Initializer for the kernel weights matrix.
- bias _initializer: Initializer for the bias vector.
- kernel _regularizer: Regularizer function applied to the kernel weights matrix.
- bias _regularizer: Regularizer function applied to the bias vector.
- activity _regularizer: Regularizer function applied to the output of the layer (its "activation").
- kernel _constraint: Constraint function applied to the kernel weights matrix.
- bias _constraint: Constraint function applied to the bias vector.

2.5.3 Sequential method

Is a high level API for building neural networks in tensorflow. The model can be defined as a sequence of layers instead of the requirement to write code for each layer seperately.

```
tf.keras.Sequential(
    layers=None, name=None
)
```

2.6 Compiling the model

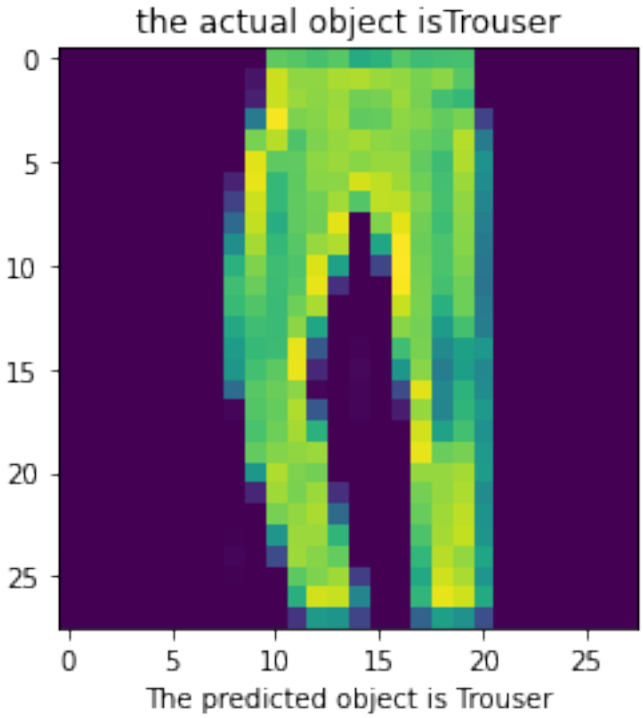
The model.compile() method in TensorFlow is used to configure the learning process of the model. It takes three important arguments:

- optimizer: The optimizer is a function that updates the model parameters during training based on the gradient of the loss function. It is responsible for minimizing the loss function and improving the accuracy of the model. Some commonly used optimizers are adam, sgd, rmsprop, etc.
- loss: The loss function is used to evaluate the performance of the model during training. It computes the difference between the predicted output and the true output for a given input. Some commonly used loss functions for different types of problems are binary _crossentropy, categorical _crossentropy, mean _squared _error, etc.
- metrics: Metrics are used to monitor the performance of the model during training and testing. It can be a list of multiple metrics to be evaluated during training. Some commonly used metrics are accuracy, precision, recall, etc

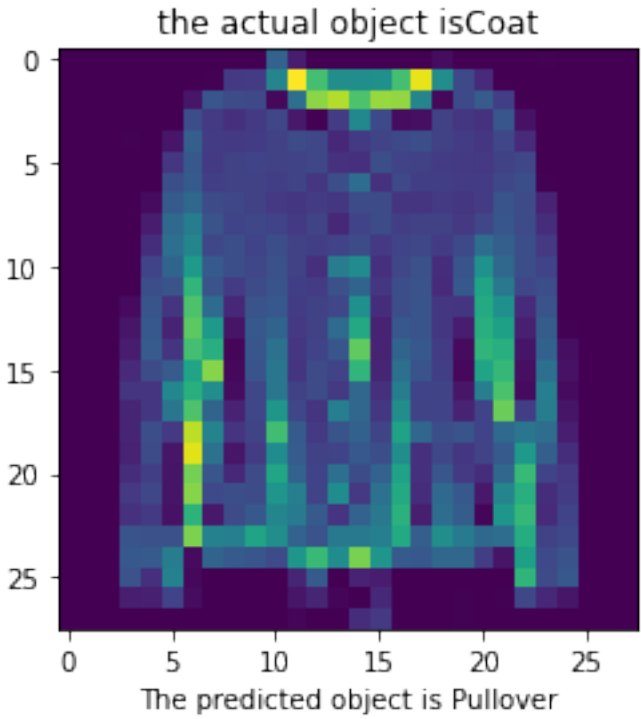
Chapter 3

Conclusion

The accuracy of the model was obtained to be 0.91. Most of the results appear to be accurate
Each of the images in the test set was tested to see if the predicted result would be same as the actual object.
The value a=5 was given. which indicates the 5 item of the testing images. The obtained result is



The value a=25 was given. which indicates the 25 item in the fashion _MNIST dataset testing images. The obtained result was


the actual object isCoat
0
5
10
15
20
25
0
5
10
15
20
25
The predicted object is Pullover

Chapter 4

Appendix

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt


#loading the fashion MNIST dataset
fashion_mnist=tf.keras.datasets.fashion_mnist
(train_images,train_labels),(test_images,test_labels)=fashion_mnist.load_data()


#exploring the training data
train_images.shape


#number of train labels
len(train_labels)

#the train labels
print(train_labels)


# # exploring testing data


test_images.shape

len(test_labels)


# # preprocessing data


#visualizing image using matplotlib
print("testing training image")
```

```
fig=plt.figure()
plt.imshow(train_images[0])
plt.colorbar()


# the values of color range from 0 to 255. they have to be adjusted
to the range 0 to 1

train_images=train_images/255
test_images=test_images/255


# # building the neural network

# setting up layers



model=tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28,28)), #shape of layer is 28 by 28
    tf.keras.layers.Dense(128, activation="relu"),#layer has 128 neurons
                                                    densly connected
    tf.keras.layers.Dense(10)#layer has 10 neurons densly connected
])


# # compiling the model


model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])


# # training the model


model.fit(train_images,train_labels,epochs=10)


# # making predictions

# converting linear outputs of model to probabilities


probability_model=tf.keras.Sequential([model,
                                       tf.keras.layers.Softmax()])
```

```
predictions=probability_model.predict(test_images)


# The labels for each image of the testing set has been tested.

#testing an image
a=input("enter the image to be tested\n")
prediction=predictions[int(a)] #an array of probabilites
max_value=np.argmax(prediction) #maximum values of array of probabilities
print(max_value)

#the array of class names
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

#making a plot to visualize prediciton
import matplotlib.pyplot as plt
import numpy as np

fig,ax=plt.subplots()
ax.imshow(test_images[int(a)])

#setting labels
label=class_names[max_value]
label2=class_names[test_labels[int(a)]]
xlabel=f"The predicted object is {label}"
title=f"the actual object is{label2}"
ax.set_xlabel(xlabel)
ax.set_title(title)
plt.show()
```

Bibliography

[1] Brown, R. (2021) Why machine learning model validation in important for AI models?, Medium. Medium. Available at: https://cogitotech.medium.com/why-machine-learning-model-validation-in-important-for-ai-models-1c55e0c97761 (Accessed: March 27, 2023).

[2] Data labelling in Machine Learning - Javatpoint (no date). Available at: https://www.javatpoint.com/data-labelling-in-machine-learning (Accessed: March 27, 2023).

[3] (2018) YouTube. YouTube. Available at: https://www.youtube.com/watch?v=vsWrXfO3wWw (Accessed: March 27, 2023).

[4] (2020) YouTube. YouTube. Available at: https://www.youtube.com/watch?v=5ogmEkujoqEamp;t= (Accessed: March 27, 2023).

[5] (2018) YouTube. YouTube. Available at: https://www.youtube.com/watch?v=0e0z28wAWfg (Accessed: March 27, 2023).

[6] (2020) YouTube. YouTube. Available at: https://www.youtube.com/watch v=IU5fuoYBTAMamp;t=1012s (Accessed: March 27, 2023).

[7] Keras - flatten layers (no date) Tutorials Point. Available at: https://www.tutorialspoint.com/keras/keras _flatten _layers.htm (Accessed: March 27, 2023).

[8] Team, K. (no date) Keras documentation: Dense layer, Keras. Available at: https://keras.io/api/layers/core _layers/dense/ (Accessed: March 27, 2023).