# **Fashion-MNIST Clothing Classification**

# 

**A PROJECT REPORT**

**Submitted by**

##### **Ayush Bajpai (19BAI10167)**

##### **Pratham Kurele (19BAI10163)**

##### **Manan Mittal (19BAI10171)**

##### **Devansh Rathi (19BAI10173)**

*In partial fulfilment for the award of the degree*

*Of*

##### **BACHELOR OF TECHNOLOGY**

*In*

**COMPUTER SCIENCE AND ENGINEERING**



**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING**

**VIT BHOPAL UNIVERSITY**

**KOTHRIKALAN, SEHORE**

**MADHYA PRADESH - 466114**

**VIT BHOPAL UNIVERSITY, KOTHRIKALAN,**

**SEHORE**

**MADHYA PRADESH – 466114**

**BONAFIDE CERTIFICATE**

Certified that this project report titled **“Fashion-MNIST Clothing Classification” is** the Bonafide work of Manan Mittal (19BAI10171), Devansh Rathi (19BAI10173), Ayush Bajpai (19BAI10167), and Pratham Kurele (19BAI10163) who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported here does not form part of any other project / research work on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**PROGRAM CHAIR PROJECT GUIDE**

**Dr. S. Southranjan** Dr. Shakkeera

School of Computer Science and School of Computer Science and . Engineering Engineering

VIT BHOPAL UNIVERSITY VIT BHOPAL UNIVERSITY

The Project Exhibition I Examination is held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ACKNOWLEDGEMENT**

First and foremost, I would like to thank the Lord Almighty for His presence and immense blessings throughout the project work.

I wish to express my heartfelt gratitude to Dr. Manas Kumar Mishra, Head of the Department, School of Computing Science and Engineering for much of his valuable support encouragement in carrying out this work.

I would like to thank my internal guide Dr. Shakkeera, for continually guiding and actively participating in my project, giving valuable suggestions to complete the project work.

I would like to thank all the technical and teaching staff of the School of Computing Science and Engineering, who extended directly or indirectly all support.

Last, but not the least, I am deeply indebted to my parents who have been the greatest support while I worked day and night for the project to make it a success.

**LIST OF FIGURES (Table 1.1)**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **TITLE** | **PAGE NO.** |
| **1** | **INTRODUCTION** | **9** |
| **2** | **Literature Survey** | **11** |
| **3** | **System Survey** | **12** |
| **4** | **Work Done** | **14** |
| **5** | **Disadvantages** | **16** |
| **6** | **Reference** | **16** |
| **7** | **Appendix-1** | **17** |
| **8** | **Appendix-2** | **24** |

**ABSTRACT**

**The Fashion-MNIST clothing classification problem is a new standard dataset used in computer vision and deep learning.**

**Although the dataset is relatively simple, it can be used as the basis for learning and practicing how to develop, evaluate, and use deep convolutional neural networks for image classification from scratch. This includes how to develop a robust test harness for estimating the performance of the model, how to explore improvements to the model, and how to save the model and later load it to make predictions on new data.**

**We will discover how to develop a convolutional neural network for clothing classification**

**TABLE OF CONTENTS (Table1.2)**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | List of Figures  List of Tables  Abstract | iii  iv  v |
| 1 | **INTRODUCTION** **1.1** **Introduction**  1.2 Motivation for the work  1.3 About Introduction to the project  including techniques  1.4 Problem Statement  1.5 Objective of the work  1.6 Organization of the thesis  1.7 Summary | 9 |
| 2 | **LITERATURE SURVEY**  2.1 Introduction  2.2 <Core area of the project>  2.3 Existing Algorithms  2.4 Research issues from literature Survey  2.5 Summary | 12 |
| 3 | **SYSTEM ANALYSIS**  3.1  **Introduction**  3.2  **Proposed System**  3.3 **Summary** | 13 |
| 4 | **WORK DONE**  4.1 Introduction  4.2 Module 1  4.3 Module 2  4.4 Module 3  4.5 Module 4  4.6 Module 5  4.7 Module 6  4.8 Summary | 15 |
| 5 | **LIMITATIONS AND DISADVANTAGE** | 16 |
| 6 | **FINAL OUTCOME** | 16 |
| 7 | Appendix-1 | 17 |
| 8 | Appendix-2 | 24 |

**INTRODUCTION**

**1.1** **Introduction: -**

Fashion mnist is used to classify unorganized data in simple form and help us to determine a particular data in a dataset. It helps us to search in an unorganized data in an effective way and decreases our effort. This project is based on **Image Classification.**

**1.2 Motivation for work: -**

There is only one way for our team to motivate our self to work hard: We don’t think about it as hard work. We think about it as part of making our self into who We want to be. Once we have made the choice to do something, We try not to think so much about how difficult or frustrating or impossible that might be; We just think about how good it must feel to be that, or how proud We might be to have done that. Make hard look easy.

**1.3** **Our project and techniques used: -**

This trains a neural network model to classify images of clothing, like sneakers and shirts. It's okay if you don't understand all the details; this is a fast-paced overview of a complete Tensor Flow program with the details explained.

**1.4** **Problem Statement: -**

It is very difficult to find the data in a very large unorganized dataset.

**1.5** **Objective of the Work: -**

The objective of the project is to use the Fashion-MNIST data set to identify the different fashion products from the given pictures using various best possible models (ML algorithms) and report the values of the performance measures for different models. Report the model that performs best, and fine-tune the same model using one of the model fine-tuning techniques, and report the best possible combination of hyper parameters for the selected model. Use the selected model to make final predictions and report the values of various performance measures for the same.

**1.7** **Summary: -**

Fashion MNIST is intended as a drop-in replacement for the classic MNIST dataset—often used as the "Hello, World" of machine learning programs for computer vision. The MNIST dataset contains images of handwritten digits (0, 1, 2, etc.) in a format identical to that of the articles of clothing you'll use here.

This guide uses Fashion MNIST for variety, and because it's a slightly more challenging problem than regular MNIST. Both datasets are relatively small and are used to verify that an algorithm works as expected. They're good starting points to test and debug code.

**LITERATURE SURVEY**

**2.1 Introduction**

This research is done by the MIT because of the research in Machine Learning and Deep Learning.

We obtain this training model and because of this model we can classify the images in to categories.

**2.2 Existing Algorithms**

**Step** **1**: Start with Python compiler

**Step** **2**: Import Tensor Flow

**Step 3**: Explore the data

**Step** **4**: Preprocess the data

**Step** **5**: Build the model

**Step** **6**: Train the model

**Step** **7**: Use the Trained model

**Step 8**: Stop

**2.3 Research issues/observations from**

There are many issue with the project because the Success ratios is only 80-90% sometimes it is above 90% because of the similar images.

* The Pixel of few images is very similar when it comes to the animals, plants and cloths so it is very difficult to distinguish between them.
* The success rate of this Almost 90% and sometimes it becomes very effective and give success rate 99%.

**SYSTEM ANALYSIS**

**3.1 Introduction**

**The** [Fashion-MNIST](https://github.com/zalandoresearch/fashion-mnist) **dataset is proposed as a more challenging replacement dataset for the MNIST dataset.**

**It is a dataset composed of 60,000 small square 28×28 pixel grayscale images of items of 10 types of clothing, such as shoes, t-shirts, dresses, and more. The mapping of all 0-9 integers to class labels is listed below.**

* **0: T-shirt/top**
* **1: Trouser**
* **2: Pullover**
* **3: Dress**
* **4: Coat**
* **5: Sandal**
* **6: Shirt**
* **7: Sneaker**
* **8: Bag**
* **9: Ankle boot**

**It is a more challenging classification problem than MNIST and top results are achieved by deep learning convolutional neural networks with a classification accuracy of about 90% to 95% on the hold out test dataset.**

**The example below loads the Fashion-MNIST dataset using the Keras API and creates a plot of the first nine images in the training dataset.**

**3.2 Proposed System**

The [Fashion MNIST](https://github.com/zalandoresearch/fashion-mnist) dataset was developed as a response to the wide use of the [MNIST dataset](http://yann.lecun.com/exdb/mnist/), that has been effectively “*solved*” given the use of modern convolutional neural networks.

Fashion-MNIST was proposed to be a replacement for MNIST, and although it has not been solved, it is possible to routinely achieve error rates of 10% or less. Like MNIST, it can be a useful starting point for developing and practicing a methodology for solving image classification using convolutional neural networks.

Instead of reviewing the literature on well-performing models on the dataset, we can develop a new model from scratch.

The dataset already has a well-defined train and test dataset that we can use.

In order to estimate the performance of a model for a given training run, we can further split the training set into a train and validation dataset. Performance on the train and validation dataset over each run can then be plotted to provide learning curves and insight into how well a model is learning the problem.

The Keras API supports this by specifying the “*validation data*” argument to the *model. Fit()* function when training the model, that will, in turn, return an object that describes model performance for the chosen loss and metrics on each training epoch.

.

**3.3 Summary**

It can be a useful starting point for developing and practicing a methodology for solving image classification using convolutional neural networks with the help of MNIST.

**WORK DONE**

**4.1 Module 1: - Import the Fashion MNIST dataset**

We will import the dataset from the following

fashion\_mnist = tf.keras.datasets.fashion\_mnist

**4.2 Module 2: Explore the data**

Let's explore the format of the dataset before training the model. The following shows there are 60,000 images in the training set, with each image represented as 28 x 28 pixels:

By:-

train\_images.shape

len(train\_labels)

train\_labels

test\_images.shap

len(test\_labels)

**4.3 Module 3: - Preprocess the data**

The data must be preprocessed before training the network. If you inspect the first image in the training set, you will see that the pixel values fall in the range 0 to 255**.**

**4.4 Module 4: - Build the model**

1. Set up the layers

Layers extract representations from the data fed into them.

Building the neural network requires configuring the layers of the model, then compiling the model.

The first layer in this network, tf.keras.layers.Flatten, transforms the format of the images from a two-dimensional array (of 28 by 28 pixels) to a one-dimensional array (of 28 \* 28 = 784 pixels).

The second (and last) layer returns a logits array with length of 10. Each node contains a score that indicates the current image belongs to one of the 10 classes.

1. Compile the model

Before the model is ready for training, it needs a few more settings. These are added during the model's compile step:

Loss function —measures accuracy of model

Optimizer —this is how the model is updated based on the data it sees and its loss function.

Metrics —used to monitor the training and testing steps.

**4.5 Module 5: - Train the model**

Training the neural network model requires the following steps:

Feed the training data to the model. In this example, the training data is in the **train\_images** and **train\_labels** arrays.

The model learns to associate images and labels.

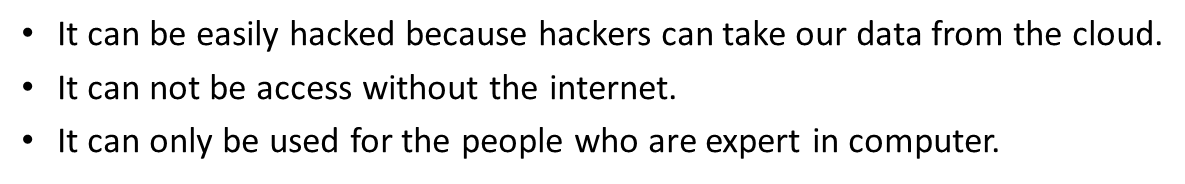
You ask the model to make predictions about a test set—in this example, the **test images** array.

Verify that the predictions match the labels from the **test labels** array.

**4.6 Module 6: - Used the trained model**

With the model trained, you can use it to make predictions about some images.

**DISADVANTAGES/LIMITATIONS IN THE SYSTEM**



**REFERENCES**

* machinelearningmastery.com
* tensorflow.google.cn
* coursera.org
* en.wikipedia.org

**APPENDIX -1(Coding)**

**1)** This guide uses [tf.keras](https://tensorflow.google.cn/guide/keras), a high-level API to build and train models in Tensor Flow

# TensorFlow and tf.keras

import tensorflow as tf

# Helper libraries

import numpy as np

import matplotlib.pyplot as ppt

print(tf.\_\_version\_\_)

**2**) Import the Fashion MNIST dataset:-

* fashion\_mnist = tf.keras.datasets.fashion\_mnist

(train\_images, train\_labels), (test\_images, test\_labels) = fashion\_mnist.load\_data()

* class\_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',

**3)** Explore the data

train\_images.shape

len(train\_labels)

Train\_labels

test\_images.shape

len(test\_labels)

len(test\_labels)

**4)** Preprocess the data

plt.figure()

plt.imshow(train\_images[0])

plt.colorbar()

plt.grid(False)

plt.show()

train\_images = train\_images / 255.0

test\_images = test\_images / 255.0

plt.figure(figsize=(10,10))

for i in range(25):

plt.subplot(5,5,i+1)

plt.xticks([])

plt.yticks([])

plt.grid(False)

plt.imshow(train\_images[i], cmap=plt.cm.binary)

plt.xlabel(class\_names[train\_labels[i]])

plt.show()

**5)** Build the model

model = tf.keras.Sequential([

tf.keras.layers.Flatten(input\_shape=(28, 28)),

tf.keras.layers.Dense(128, activation='relu'),

tf.keras.layers.Dense(10)

])

model.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True),

metrics=['accuracy'])

**6)** Train the model

model.fit(train\_images, train\_labels, epochs=10)

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels, verbose=2)

print('\nTest accuracy:', test\_acc)

probability\_model = tf.keras.Sequential([model,

tf.keras.layers.Softmax()])

predictions = probability\_model.predict(test\_images)

predictions[0]

np.argmax(predictions[0])

test\_labels[0]

def plot\_image(i, predictions\_array, true\_label, img):

true\_label, img = true\_label[i], img[i]

plt.grid(False)

plt.xticks([])

plt.yticks([])

plt.imshow(img, cmap=plt.cm.binary)

predicted\_label = np.argmax(predictions\_array)

if predicted\_label == true\_label:

color = 'blue'

else:

color = 'red'

plt.xlabel("{} {:2.0f}% ({})".format(class\_names[predicted\_label],

100\*np.max(predictions\_array),

class\_names[true\_label]),

color=color)

def plot\_value\_array(i, predictions\_array, true\_label):

true\_label = true\_label[i]

plt.grid(False)

plt.xticks(range(10))

plt.yticks([])

thisplot = plt.bar(range(10), predictions\_array, color="#777777")

plt.ylim([0, 1])

predicted\_label = np.argmax(predictions\_array)

thisplot[predicted\_label].set\_color('red')

thisplot[true\_label].set\_color('blue')

**7)** Verify prediction

i = 0

plt.figure(figsize=(6,3))

plt.subplot(1,2,1)

plot\_image(i, predictions[i], test\_labels, test\_images)

plt.subplot(1,2,2)

plot\_value\_array(i, predictions[i], test\_labels)

plt.show()

i = 12

plt.figure(figsize=(6,3))

plt.subplot(1,2,1)

plot\_image(i, predictions[i], test\_labels, test\_images)

plt.subplot(1,2,2)

plot\_value\_array(i, predictions[i], test\_labels)

plt.show()

# Plot the first X test images, their predicted labels, and the true labels.

# Color correct predictions in blue and incorrect predictions in red.

num\_rows = 5

num\_cols = 3

num\_images = num\_rows\*num\_cols

plt.figure(figsize=(2\*2\*num\_cols, 2\*num\_rows))

for i in range(num\_images):

plt.subplot(num\_rows, 2\*num\_cols, 2\*i+1)

plot\_image(i, predictions[i], test\_labels, test\_images)

plt.subplot(num\_rows, 2\*num\_cols, 2\*i+2)

plot\_value\_array(i, predictions[i], test\_labels)

plt.tight\_layout()

plt.show()

**8)** Use the trained model

# Grab an image from the test dataset.

img = test\_images[1]

print(img.shape)

# Add the image to a batch where it's the only member.

img = (np.expand\_dims(img,0))

print(img.shape)

predictions\_single = probability\_model.predict(img)

print(predictions\_single)

plot\_value\_array(1, predictions\_single[0], test\_labels)

\_ = plt.xticks(range(10), class\_names, rotation=45)

np.argmax(predictions\_single[0])

**Appendix 2(coding)**

**1)** **Import Tenserflow**

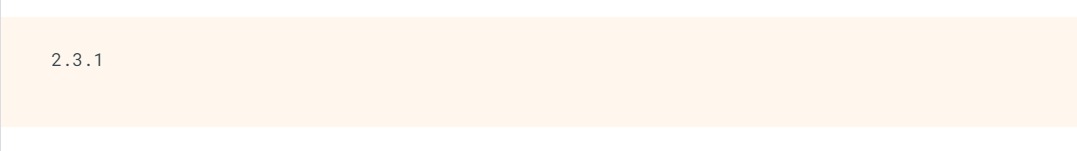


Fig.1

**2)** **Import the fashion MNIST dataset**

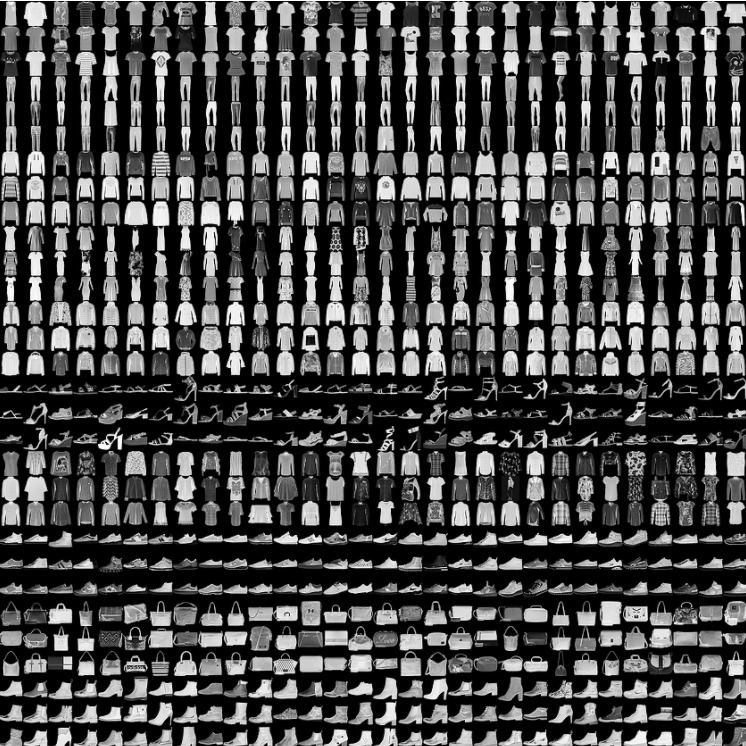
****

Fig. 2

**3) Explore the data**

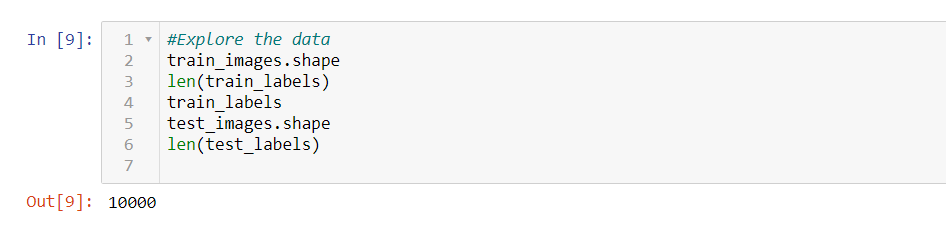
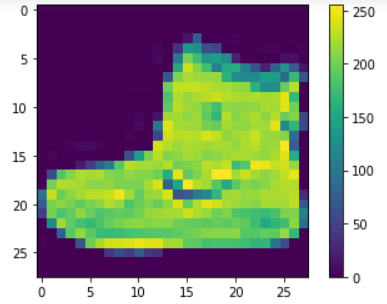
****

Fig. 3

**4) Preprocess the data**

**Fig.4**

****

**Fig.5**

**5) Train the model**

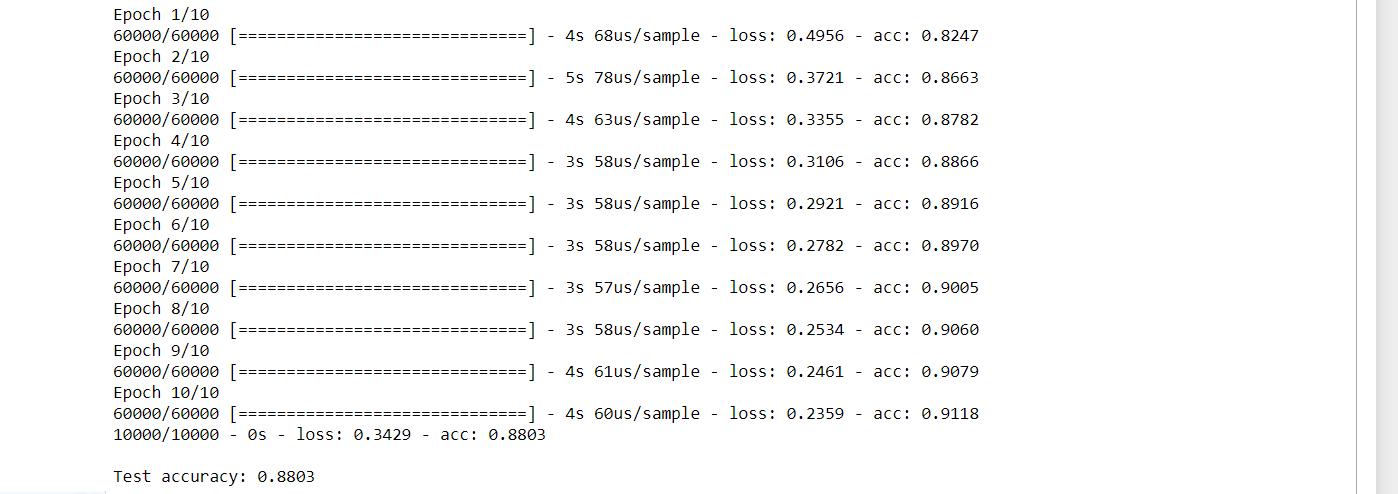
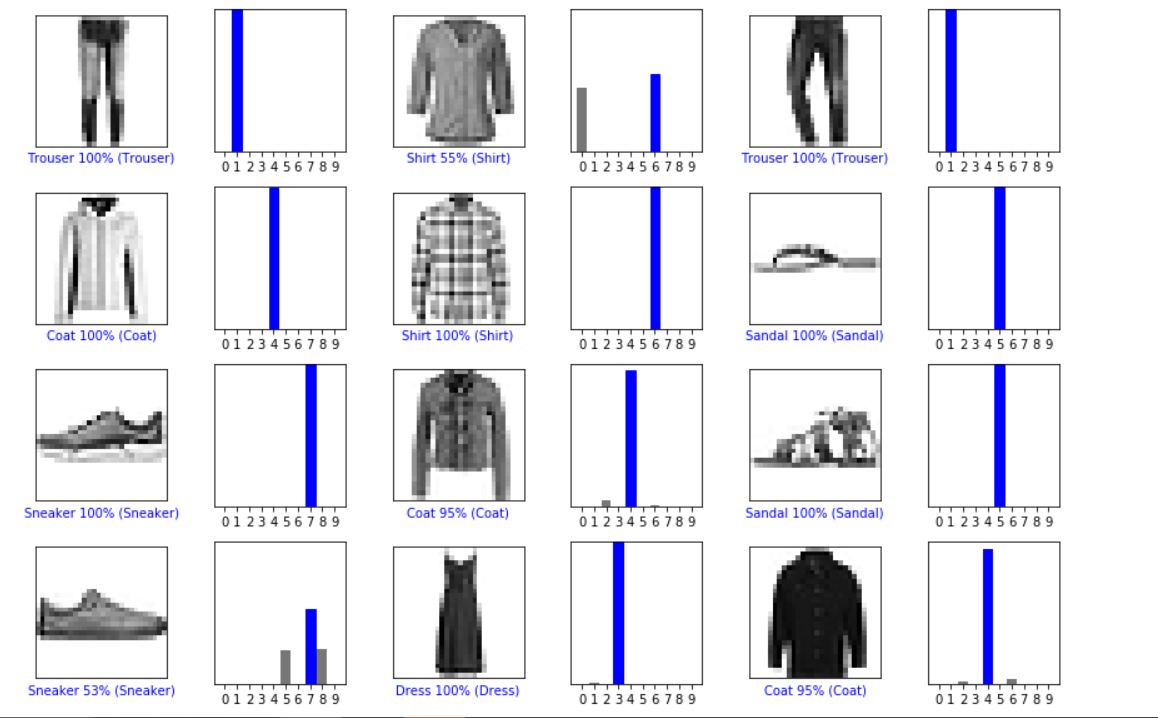
****

Fig.6

**6) Verify the predictions**

****

**Fig.7**

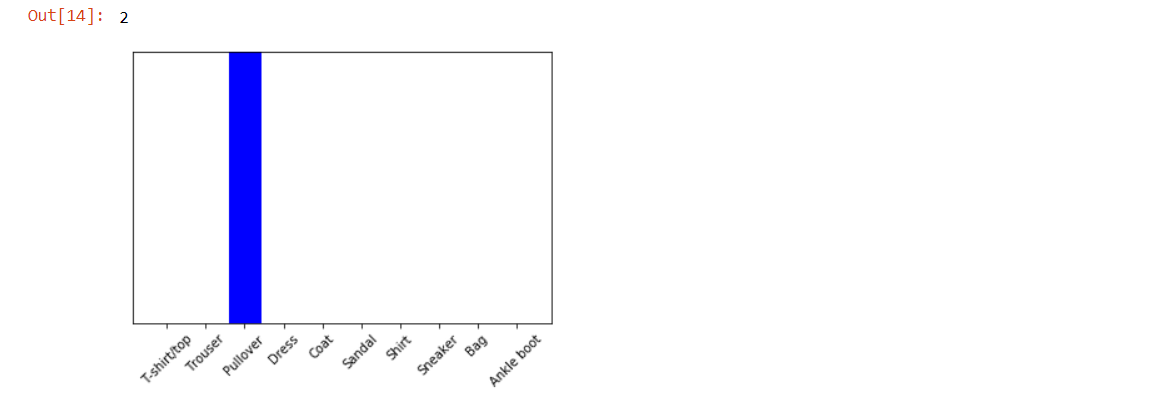
**7) Output**

Fig.8