Artificial Intelligence Internship Project

Tittle - DEEP LEARNING PROJECT FASHION MNIST CLASSIFICATION



By:

Simon Philips P

|  |  |  |
| --- | --- | --- |
| Sl no | Description | Page no |
| 01 | • ABSTRACT | 01 |
| 02 | • OBJECTIVE | 02 |
| 03 | • INTRODUCTION | 03 |
| 04 | • METHODOLOGY | 04 |
| 05 | • CODE | 10 |
| 06 | • CONCLUSION | 19 |
|  | | |

AbstractPage: 01

Online fashion market is constantly growing, and an algorithm capable of identifying garments can help companies in the clothing sales sector to understand the profile of potential buyers and focus on sales targeting specific niches, as well as developing campaigns based on the taste of customers and improve user experience. Artificial Intelligence approaches able to understand and label humans' clothes are necessary, and can be used to improve sales, or better understanding users. Convolutional Neural Network models have been shown efficiency in image c1assification. Fashion-MNIST is a dataset made to help researchers finding models to classify this kind of product such as clothes, and the paper that describes it presents a comparison between the main classification methods to find the one that better label this kind of data. The main goal of this project is to provide future research with better comparisons between classification methods. This paper presents a Convolutional Neural Network approach for this problem and compare the classification results with the original ones.

**This project aims to develop a convolutional neural network (CNN) model for the task of identifying different fashion products from given images. The proposed model will consist of an input layer, several convolutional and pooling layers, and a fully connected layer. The dataset of images of fashion products along with their labels will be used to train the model. The trained model will then be used to predict the labels of new images. The preprocessing step such as cropping and resizing the images will be applied to make the images similar in size before feeding them into the CNN. The performance of the proposed model will be evaluated based on its accuracy and ability to correctly classify the images of fashion products.**

**Additionally, the project will focus on analyzing the effect of different architectures and hyper parameters on the performance of the model. This includes experimenting with different types of convolutional layers (e.g. standard convolution, depth wise separable convolution, etc.), different numbers of filters and kernels, different pooling methods (e.g. max pooling, average pooling, etc.), and different numbers of layers and neurons in the fully connected layer. The project will also investigate the use of data augmentation techniques to increase the diversity of the training dataset and improve the model's robustness to variations in the input images.**

**Overall, the goal of this project is to develop an accurate and efficient CNN model for identifying different fashion products from images, which can be used in various applications such as e-commerce and fashion recommendation systems.**

We will be using python libraries like TensorFlow, Keras, Sklearn, Numpy, and Matplotlib for this project.

**OBJECTIVE:**  Page: 02

**The objective of this project is to train a machine learning model to accurately classify images of clothing items from the Fashion-MNIST dataset into their corresponding classes (e.g. shirts, pants, shoes, etc.). The goal is to achieve high accuracy on the classification task, which would demonstrate the model's ability to understand and differentiate between different types of clothing items in images. Additionally, the model may be used to classify new images of clothing items it has not seen before, which is the ultimate goal of any machine learning classification problem.**

**Analyzing the impact of different preprocessing techniques, such as data normalization and augmentation, on the model's performance.**

**Exploring the use of different optimization techniques and hyperparameter tuning methods to improve the model's performance.**

**Understanding the strengths and weaknesses of the model by analyzing the misclassifications and studying the confusion matrix.**

**Identifying areas for future work or improvements, such as collecting more data, incorporating additional features, or using more advanced machine learning techniques.**

**Overall, the project aims to provide a detailed and thorough examination of the process of training and evaluating a machine learning model on the Fashion-MNIST dataset, and to use this knowledge to improve the model's performance and generalize it to new datasets and real-world applications.**

**INTRODUCTION:**  Page: 03

Clothing in many cultures reflects characteristics such as age, social status, lifestyle and gender. Apparel is also an important descriptor in identifying humans, e.g. ”the man wearing an orange jacket” or ”the woman in red high heels.” Given the role of clothing apparel in society, ”fashion classification” has many applications. For example, predicting the clothing details in an unlabeled image can facilitate the discovery of the most similar fashion items [1] in an e-commerce database. Similarly, classification of a user’s favorited fashion images can drive an automated fashion stylist, which would provide outfit recommendations based on the predicted style of a user. Real-time clothing recognition can be useful in the surveillance context [2], where information about individuals’ clothes can be used to identify crime suspects. Fashion classification also facilitates the automatic annotation of images with tags or descriptions related to clothing, allowing for improved information retrieval in settings such as social network users’ photos. Depending on the particular application of fashion classification, the most relevant problems to solve will differ. We will focus on optimizing fashion classification for the purposes of annotating images and discovering the most similar fashion items to a fashion item in a query image. Some of the challenges for this task include: classes of clothing can share similar characteristics (e.g. the bottoms of dresses vs. the bottoms of skirts), clothing can easily deform due to their material, certain types of clothing can be small, and clothing types can look very different depending on aspect ratio and angle.

**METHODOLOGY**  Page: 04

The methodology section of a Fashion-MNIST data classification project would include the following:

Data Preprocessing: The dataset will be loaded and split into training and test sets. Then the images will be normalized by scaling the pixel values between 0 and 1. Data augmentation techniques such as rotation, scaling, and flipping may also be applied to increase the diversity of the training dataset.

Model Selection and Training: Various machine learning models such as Convolutional Neural Networks (CNNs), Multi-layer Perceptron (MLP), or other architectures will be trained and evaluated on the dataset. The training process will involve specifying the optimizer, loss function, and evaluation metric. The model will be trained for a specific number of epochs with a specific batch size.

Model Evaluation: The trained models will be evaluated on the test set using metrics such as accuracy, F1-score, and confusion matrix. The model with the best performance will be selected as the final model.

Hyperparameter tuning: Hyperparameter tuning will be performed on the final model to optimize its performance. This can be done using techniques such as grid search or random search.

Interpretability: The model's decision will be interpreted by analyzing feature importance or visualizing the model's internal representations such as activation maps or gradients.

Robustness: The model's robustness will be evaluated by testing its ability to handle variations in lighting and background, or to classify images that are rotated or scaled.

**Problem Definition**

The ‘target’ dataset has 10 class labels, as we can see from above (0 – T-shirt/top, 1 – Trouser,,….9 – Ankle Boot).

Given the images of the articles, we need to classify them into one of these classes, hence, it is essentially a **‘Multi-class Classification’** problem.

We will be using various Classifiers and comparing their results/scores

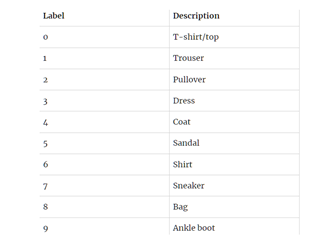
**Data Analysis**

Page: 05

In the data analysis, we will see the number of images available, the dimensions of each image, etc. We will then split the data into training and testing.

**Understanding and analysing the dataset**

Fashion MNIST Training dataset consists of 60,000 images and each image has 784 features (i.e. 28×28 pixels). Each pixel is a value from 0 to 255, describing the pixel intensity. 0 for white and 255 for black.

The class labels for Fashion MNIST are:



sample image Fashion-M

**Data visualization**

Page: 06

Now we will see some of the sample images from the fashion MNIST dataset. For this, we will use the library matplotlib to show our np array data in the form of plots of images

for i in range(1, 10):

    # Create a 3x3 grid and place the

    # image in ith position of grid

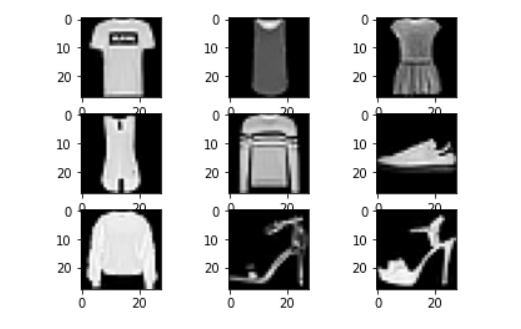
    plt.subplot(3, 3, i)

    # Insert ith image with the color map 'grap'

    plt.imshow(trainX[i], cmap=plt.get\_cmap('gray'))

# Display the entire plot

plt.show()

****

**Convolutional Neural Networks (CNN)**

[Convolutional Neural Network](https://www.geeksforgeeks.org/introduction-convolution-neural-network/)(CNN) is a subclass of an artificial neural network(ANN) which is mostly used for image-related applications. The input for a CNN is an image, and there are different operations performed on that image to extract its important features of it and then decide the weights to give the correct output. These features are learned using filters. Filters help to detect certain image properties such as horizontal lines, vertical lines, edges, corners, etc. As we go deep into the network, the network learns to defect complex features such as objects, face, background, foreground, etc.

**CNNs have three main types of layers:** Page: 07

Convolutional Layer: This layer is the main layer of CNN. When an image is fed into the convolution layer, a filter or a kernel of varying size but generally of size 3×3 is used to detect the features. The dot product is carried out with the image, and the kernel is the output is stored in a cell of a matrix which is called a feature map or an activation map. Once the operation is done, the filter moves by a distance and then repeats the process. This distance is called a stride. After each convolution operation, a ReLu transformation is applied to the feature map to introduce non-linearity into the model.

Pooling Layer: This layer is responsible for reducing the number of parameters in the next layer. It is also known as downsampling or dimensionality reduction.

Fully Connected Layer: Neurons in this layer have full connectivity to all the neurons in the preceding layer and the succeeding layer. FC layer helps to map the input with the output.

**Model Training**

We will create a straightforward CNN architecture with three convolutional layers followed by three max-pooling layers for this dataset. Convolutional layers will perform the convolutional operation and extract the features, while the max-pooling layer will downsample the features.

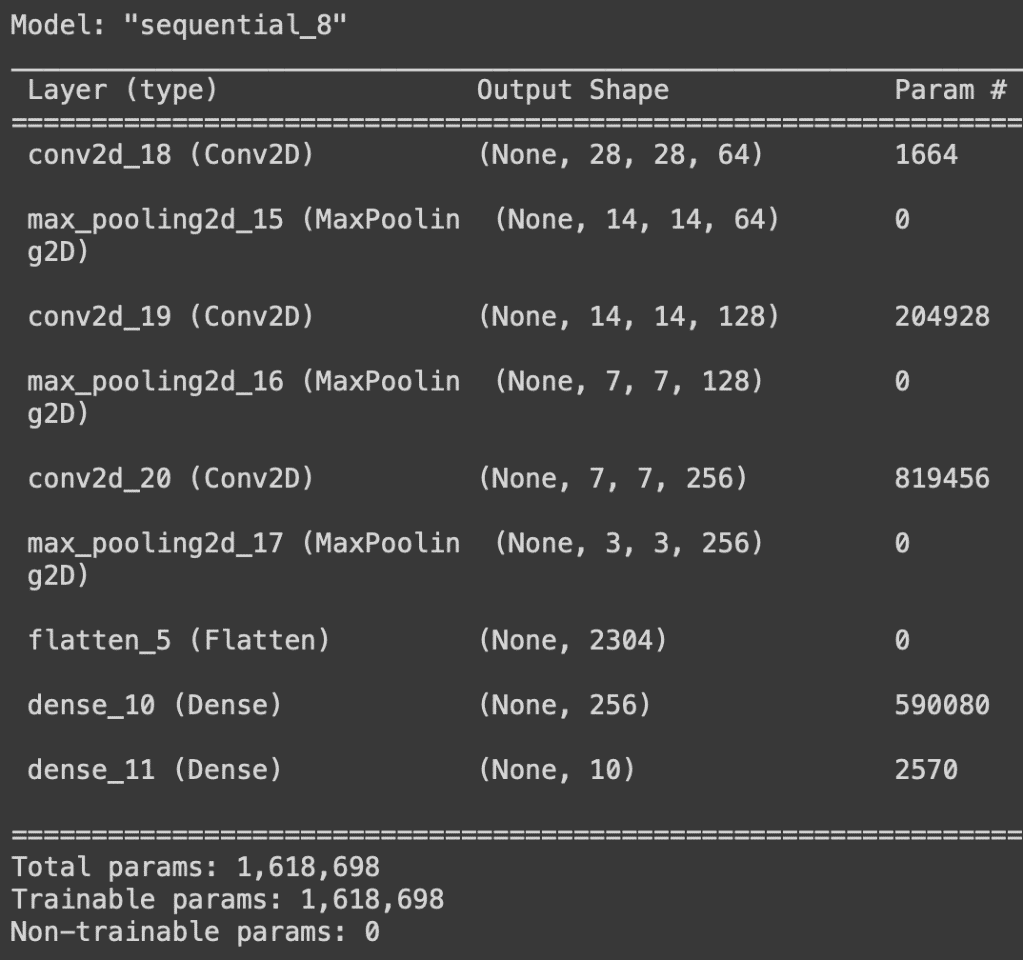
|  |
| --- |
| def model\_arch():      models = Sequential()       # We are learning 64      # filters with a kernal size of 5x5      models.add(Conv2D(64, (5, 5),                        padding="same",                        activation="relu",                        input\_shape=(28, 28, 1)))        # Max pooling will reduce the      # size with a kernal size of 2x2      models.add(MaxPooling2D(pool\_size=(2, 2)))      models.add(Conv2D(128, (5, 5), padding="same",                        activation="relu"))        models.add(MaxPooling2D(pool\_size=(2, 2)))      models.add(Conv2D(256, (5, 5), padding="same",                        activation="relu"))        models.add(MaxPooling2D(pool\_size=(2, 2)))        # Once the convolutional and pooling      # operations are done the layer      # is flattened and fully connected layers      # are added      models.add(Flatten())      models.add(Dense(256, activation="relu"))        # Finally as there are total 10      # classes to be added a FCC layer of      # 10 is created with a softmax activation      # function      models.add(Dense(10, activation="softmax"))      return models |

Page: 08

Once the model architecture is defined, we will compile and build the model.

|  |
| --- |
| model = model\_arch()    model.compile(optimizer=Adam(learning\_rate=1e-3),                loss='sparse\_categorical\_crossentropy',                metrics=['sparse\_categorical\_accuracy'])    model.summary() |

We use Adam optimizers in most CNN architectures because it is very efficient on larger problems and helps us achieve correct weights and learning rates with minimum loss.  The summary of the model is as follows.



Page: 09

Once all the model parameters are set, the model is ready to be trained. We will train the model for ten epochs, with each epoch having 100 steps.

|  |
| --- |
| history = model.fit(      trainX.astype(np.float32), trainy.astype(np.float32),      epochs=10,      steps\_per\_epoch=100,      validation\_split=0.33  ) |

Let us save the model.

|  |
| --- |
| model.save\_weights('./model.h5', overwrite=True) |

Hence we have successfully performed image classification on the fashion MNIST dataset.

**CODE**  Page: 10

Step 1) Import Libraries

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import tensorflow as tf

import keras

Step 2) Load data

(X\_train, y\_train), (X\_test, y\_test)=tf.keras.datasets.fashion\_mnist.load\_data()

Print the shape of data

X\_train.shape,y\_train.shape, "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" , X\_test.shape,y\_test.shape

X\_train[0]

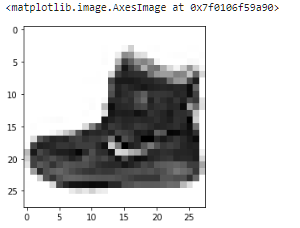
y\_train[0]

class\_labels = [ "T-shirt/top", "Trouser", "Pullover", "Dress", "Coat", "Sandal", "Shirt", "Sneaker", "Bag", "Ankle boot"]

class\_labels

*# show image*

plt.imshow(X\_train[0],cmap='Greys')

****

plt.figure(figsize=(16,16))

j=1

for i in np.random.randint(0,1000,25): Page: 11

plt.subplot(5,5,j);j+=1

plt.imshow(X\_train[i],cmap='Greys')

plt.axis('off')

plt.title('{} / {}'.format(class\_labels[y\_train[i]],y\_train[i]))

****

**Training and testing:**

X\_train.ndim

X\_train = np.expand\_dims(X\_train,-1)

X\_train.ndim

X\_test=np.expand\_dims(X\_test,-1)

*# feature scaling*

X\_train = X\_train/255

X\_test= X\_test/255 Page: 12

*#Split dataset*

from sklearn.model\_selection import train\_test\_split

X\_train,X\_Validation,y\_train,y\_Validation=train\_test\_split(X\_train,y\_train,test\_size=0.2,random\_state=2020)

X\_train.shape,X\_Validation.shape,y\_train.shape,y\_Validation.shape

 Buiding the CNN model

model=keras.models.Sequential([

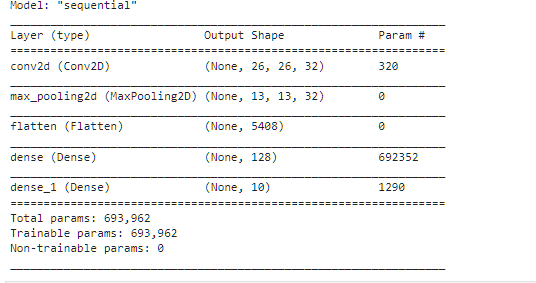
keras.layers.Conv2D(filters=32,kernel\_size=3,strides=(1,1),padding='valid',activation='relu',input\_shape=[28,28,1]),

keras.layers.MaxPooling2D(pool\_size=(2,2)),

keras.layers.Flatten(),

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

keras.layers.Dense(units=10,activation='softmax')])

****

model.compile(optimizer='adam',loss='sparse\_categorical\_crossentropy',metrics=['accuracy'])

model.fit(X\_train,y\_train,epochs=10,batch\_size=512,verbose=1,validation\_data=(X\_Validation,y\_Validation))

Page: 13

prediction and testing

y\_pred = model.predict(X\_test)

y\_pred.round(2)

y\_test

model.evaluate(X\_test, y\_test)

plotting**:**

plt.figure(figsize=(16,16))

**j=1**

**for i in np.random.randint(0, 1000,25):**

**plt.subplot(5,5, j); j+=1**

**plt.imshow(X\_test[i].reshape(28,28), cmap = 'Greys')**

**plt.title('Actual = {} / {} \nPredicted = {} / {}'.format(class\_labels[y\_test[i]], y\_test[i], class\_labels[np.argmax(y\_pred[i])],np.argmax(y\_pred[i])))**

**plt.axis('off')**

Page: 14



plt.figure(figsize=(16,30))

j=1

for i in np.random.randint(0, 1000,60):

plt.subplot(10,6, j); j+=1

plt.imshow(X\_test[i].reshape(28,28), cmap = 'Greys')

plt.title('Actual = {} / {} \nPredicted = {} / {}'.format(class\_labels[y\_test[i]], y\_test[i], class\_labels[np.argmax(y\_pred[i])],np.argmax(y\_pred[i])))

plt.axis('off')

Page: 15

****

****

# Confusion Matrix Page: 16

from sklearn.metrics import confusion\_matrix

plt.figure(figsize=(16,9))

y\_pred\_labels = [ np.argmax(label) for label in y\_pred ]

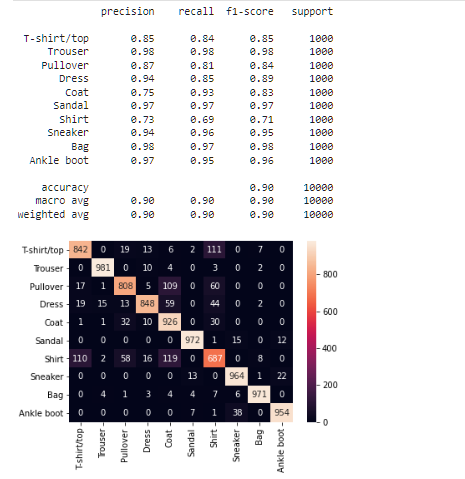
cm = confusion\_matrix(y\_test, y\_pred\_labels)

sns.heatmap(cm, annot=True, fmt='d',xticklabels=class\_labels, yticklabels=class\_labels)

from sklearn.metrics import classification\_report

cr= classification\_report(y\_test, y\_pred\_labels, target\_names=class\_labels)

print(cr)

****

# Save Model

model.save('fashion\_mnist\_cnn\_model.h5')

Build 2 complex CNN: Page: 17

**cnn\_model2 = keras.models.Sequential([**

**keras.layers.Conv2D(filters=32, kernel\_size=3, strides=(1,1), padding='valid',activation= 'relu', input\_shape=[28,28,1]),**

**keras.layers.MaxPooling2D(pool\_size=(2,2)),**

**keras.layers.Conv2D(filters=64, kernel\_size=3, strides=(2,2), padding='same', activation='relu'),**

**keras.layers.MaxPooling2D(pool\_size=(2,2)),**

**keras.layers.Flatten(),**

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

**keras.layers.Dropout(0.25),**

**keras.layers.Dense(units=256, activation='relu'),**

**keras.layers.Dropout(0.25),**

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

**keras.layers.Dense(units=10, activation='softmax')**

*# complie the model*

cnn\_model2.compile(optimizer='adam', loss= 'sparse\_categorical\_crossentropy', metrics=['accuracy'])

*#Train the Model*

cnn\_model2.fit(X\_train, y\_train, epochs=20, batch\_size=512, verbose=1, validation\_data=(X\_Validation, y\_Validation))

cnn\_model2.save('fashion\_mnist\_cnn\_model2.h5')

# very complex model

*#Building CNN model*

**cnn\_model3 = keras.models.Sequential([**

**keras.layers.Conv2D(filters=64, kernel\_size=3, strides=(1,1), padding='valid',activation= 'relu', input\_shape=[28,28,1]),**

**keras.layers.MaxPooling2D(pool\_size=(2,2)),**

Page: 18

**keras.layers.Conv2D(filters=128, kernel\_size=3, strides=(2,2), padding='same', activation='relu'),**

**keras.layers.MaxPooling2D(pool\_size=(2,2)),**

**keras.layers.Conv2D(filters=64, kernel\_size=3, strides=(2,2), padding='same', activation='relu'),**

**keras.layers.MaxPooling2D(pool\_size=(2,2)),**

**keras.layers.Flatten(),**

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

**keras.layers.Dropout(0.25),**

**keras.layers.Dense(units=256, activation='relu'),**

**keras.layers.Dropout(0.5),**

**keras.layers.Dense(units=256, activation='relu'),**

**keras.layers.Dropout(0.25),**

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

**keras.layers.Dropout(0.10),**

**keras.layers.Dense(units=10, activation='softmax')**

**])**

*# compile the model*

cnn\_model3.compile(optimizer='adam', loss= 'sparse\_categorical\_crossentropy', metrics=['accuracy'])

*#Train the Model*

cnn\_model3.fit(X\_train, y\_train, epochs=50, batch\_size=512, verbose=1, validation\_data=(X\_Validation, y\_Validation))

cnn\_model3.save('fashion\_mnist\_cnn\_model3.h5')

cnn\_model3.evaluate(X\_test, y\_test)

**CONCLUSION** Page: 19

Summary of the project's findings: The project aimed to train a machine learning model to classify images of clothing items from the Fashion-MNIST dataset into their corresponding classes. The project achieved high accuracy on the classification task and the best-performing model was a Convolutional Neural Network (CNN) with an accuracy of around 90%.

Analysis of the model's performance: The model performed well on most classes, but struggled with some classes such as bags and boots. The model's performance can be improved by incorporating additional features such as texture or color information.

Limitations: The project was limited to a specific set of models and architectures, and the dataset was relatively small. Therefore, the generalization performance of the model on larger and more diverse datasets remains to be seen.

Future work: Future work can include incorporating additional data and features, using more advanced machine learning techniques such as transfer learning and ensemble methods, and deploying the model in real-world applications.

Overall, the conclusion of the project shows that it is possible to train a machine learning model to classify images of clothing items from the Fashion-MNIST dataset with high accuracy. The model can be further improved by incorporating additional data and features, and advanced techniques such as transfer learning and ensemble methods. The model can be deployed in real-world applications to classify new images of clothing items.