Machine Learning

CS-697AB

Final Project Report

Name:FNU Aishwarya Ajay Venkatesha

ID: E425P738

Procedure:

Step 1 : Importing required libraries.

```
#Importing Libraries
import tensorflow as tf
from tensorflow import keras
from keras.layers import Dense,Input, Flatten, Bidirectional, BatchNormalization , Dropout, Concatenate, Conv2D, MaxPooling2D
from keras.models import Sequential
tf.__version__
keras.__version__

'2.7.0'
```

Step 2 : Load the dataset Fashion_MNIST from keras and divide the test and train dataset from it.

```
[23] #Loading the FASHION_MNIST dataset
     fashion_mnist = keras.datasets.fashion_mnist
     (X_train_all, y_train_all), (X_test, y_test) = fashion_mnist.load_data()
```

Step 3 :

We know that the image data in x_train_all is from 0 to 255 , performing rescaling from 0 to 1.To achieve rescaling , the x_train_all is divided by 255 .

Note that the training set and the testing set should be preprocessed and reshaped in the same way. Mismatch in shape between training and test data leads to errors.

```
#Rescaling image data in x_train_all from (0 to 255) to (0 to 1)
X_train_all , X_test = X_train_all / 255.0 , X_test / 255.0

#Populating x_valid ,x_train, y_valid, y_train
X_valid, X_train = X_train_all[:5000], X_train_all[5000:]
y_valid, y_train = y_train_all[:5000], y_train_all[5000:]
```

Step 4 :

Building a neural network model 1 with below architecture

Layer	Type	No. of Filters	Activation shape	Kernel size
1	Conv2D	16	(28,28,1)	3X3
2	Conv2D	16	(28,28,1)	3X3
3	MaxPooling	-		
4	Conv2D	32	(28,28,1)	3X3
5	Conv2D	32	(28,28,1)	3X3
6	MaxPooling	-		
7	Flatten	-		
8	Dense	-	(90,1)	
9	softmax	-	(10,1)	

Model 1

```
model = Sequential()
model.add(Conv2D(16, kernel_size=(3, 3), activation='relu', input_shape=[28, 28,1]))
model.add(Conv2D(16, kernel_size=(3, 3), activation='relu', input_shape=[28, 28,1]))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2, ))
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=[28, 28,1]))
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=[28, 28,1]))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(90, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(loss="sparse_categorical_crossentropy", optimizer="sgd", metrics=["accuracy"])

model.summary()
```

```
model.summary()
```

```
Model: "sequential_31"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_68 (Conv2D)          (None, 26, 26, 16)        160

 conv2d_69 (Conv2D)          (None, 24, 24, 16)        2320

 max_pooling2d_39 (MaxPoolin  (None, 12, 12, 16)       0
 g2D)

 conv2d_70 (Conv2D)          (None, 10, 10, 32)        4640

 conv2d_71 (Conv2D)          (None, 8, 8, 32)          9248

 max_pooling2d_40 (MaxPoolin  (None, 4, 4, 32)         0
 g2D)

 flatten_28 (Flatten)        (None, 512)               0

 dense_65 (Dense)            (None, 90)                46170

 dense_66 (Dense)            (None, 10)                910

=================================================================
Total params: 63,448
Trainable params: 63,448
Non-trainable params: 0
_________________________________________________________________
```

Step 5 :

Compile the model

The model is configured before training. While configuring we use the following attributes to compile the model.

Loss function = "sparse_categorical_crossentropy" ; Measured the accuracy of the model while training.

Optimizer = “sgd”; Updates based on data it perceives and loss function

Metrics = [“accuracy”]; Monitors the training and testing steps.

```
model.compile(loss="sparse_categorical_crossentropy", optimizer="sgd", metrics=["accuracy"])
```

Step 6 :

Fitting the model with the data.

```
[ ] history = model.fit(X_train, y_train, epochs=15, validation_data=(X_valid, y_valid))
```

```
Epoch 1/15
1719/1719 [==============================] - 64s 37ms/step - loss: 0.9601 - accuracy: 0.6536 - val_loss: 0.5239 - val_accuracy: 0.8080
Epoch 2/15
1719/1719 [==============================] - 63s 37ms/step - loss: 0.5140 - accuracy: 0.8101 - val_loss: 0.4495 - val_accuracy: 0.8438
Epoch 3/15
1719/1719 [==============================] - 63s 37ms/step - loss: 0.4503 - accuracy: 0.8379 - val_loss: 0.5274 - val_accuracy: 0.8162
Epoch 4/15
1719/1719 [==============================] - 63s 37ms/step - loss: 0.4081 - accuracy: 0.8530 - val_loss: 0.3941 - val_accuracy: 0.8610
Epoch 5/15
1719/1719 [==============================] - 63s 37ms/step - loss: 0.3786 - accuracy: 0.8625 - val_loss: 0.3679 - val_accuracy: 0.8730
```

Step 7:

Evaluating the model and predicting the accuracy.

```
#Evaluating model and deriving the accuracy results
score = model.evaluate(X_test,y_test, verbose=0)
print('Test Loss: {:.4f}%'.format(score[0]*100))
print('Test Accuracy : {:.4f}%'.format(score[1]*100))
```

```
Test Loss: 32.1506%
Test Accuracy : 88.5500%
```

Step 8 :

Building a neural network model 2 with below architecture

Layer	Type	No. of Filters	Activation shape	Kernel size
1	Conv2D	4	(28,28,1)	3X3
2	Conv2D	8	(28,28,1)	3X3
3	MaxPooling	-		
4	Dropout	-		
5	Flatten	-		
6	Dense	-	(50,1)	
7	Dropout	-		
8	Softmax	-	(20,1)	

Model 2

```
[32]
model2 = Sequential()
model2.add(Conv2D(4, kernel_size=(3, 3), activation='relu', input_shape=[28, 28,1]))
model2.add(Conv2D(8, (3, 3), activation='relu'))
model2.add(MaxPooling2D(pool_size=(2, 2)))
model2.add(Dropout(0.25))
model2.add(Flatten())
model2.add(Dense(50, activation='sigmoid'))
model2.add(Dropout(0.5))
model2.add(Dense(20, activation='softmax'))

model2.compile(loss="sparse_categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
```

```
model2.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_4 (Conv2D)           (None, 26, 26, 4)         40        
                                                                 
 conv2d_5 (Conv2D)           (None, 24, 24, 8)         296       
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 12, 12, 8)        0         
 2D)                                                             
                                                                 
 dropout_4 (Dropout)         (None, 12, 12, 8)         0         
                                                                 
 flatten_3 (Flatten)         (None, 1152)              0         
                                                                 
 dense_10 (Dense)            (None, 50)                57650     
                                                                 
 dropout_5 (Dropout)         (None, 50)                0         
                                                                 
 dense_11 (Dense)            (None, 20)                1020      
                                                                 
=================================================================
Total params: 59,006
Trainable params: 59,006
Non-trainable params: 0
_________________________________________________________________
```

Step 9:

Compile the model

The model is configured before training. While configuring we use the following attributes to compile the model.

Loss function = "sparse_categorical_crossentropy" ; Measured the accuracy of the model while training.

Optimizer = “adam”; Updates based on data it perceives and loss function

Metrics = ["accuracy"]; Monitors the training and testing steps.

```
model2.compile(loss="sparse_categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
```

Step 10 :

Fitting the model with the data.

```
history = model2.fit(X_train, y_train, epochs=15, validation_data=(X_valid, y_valid))
```

```
Epoch 1/15
1719/1719 [==============================] - 33s 19ms/step - loss: 0.8576 - accuracy: 0.7102 - val_loss: 0.4748 - val_accuracy: 0.8228
Epoch 2/15
1719/1719 [==============================] - 32s 19ms/step - loss: 0.5727 - accuracy: 0.7952 - val_loss: 0.3990 - val_accuracy: 0.8488
Epoch 3/15
1719/1719 [==============================] - 33s 19ms/step - loss: 0.5159 - accuracy: 0.8192 - val_loss: 0.3642 - val_accuracy: 0.8744
Epoch 4/15
1719/1719 [==============================] - 32s 19ms/step - loss: 0.4753 - accuracy: 0.8296 - val_loss: 0.3454 - val_accuracy: 0.8750
Epoch 5/15
1719/1719 [==============================] - 32s 19ms/step - loss: 0.4459 - accuracy: 0.8414 - val_loss: 0.3340 - val_accuracy: 0.8792
Epoch 6/15
1719/1719 [==============================] - 32s 19ms/step - loss: 0.4282 - accuracy: 0.8471 - val_loss: 0.3143 - val_accuracy: 0.8834
```

Step 11:

Evaluating the model and predicting the accuracy.

```
[34] #Evaluating model and deriving the accuracy results
     score2 = model2.evaluate(X_test,y_test, verbose=0)
     print('Test Loss: {:.4f}%'.format(score2[0]*100))
     print('Test Accuracy : {:.4f}%'.format(score2[1]*100))
```

```
Test Loss: 29.0910%
Test Accuracy : 89.6200%
```

Step 12 :

Building a neural network model 3 with below architecture

Layer	Type	No. of Filters	Activation shape	Kernel size
1	Conv2D	4	(28,28,1)	3X3
2	Conv2D	8	(28,28,1)	3X3
3	MaxPooling	-		
4	Dropout	-		
5	Flatten	-		
6	Dense	-		
7	Dropout	-		
8	Softmax	-		

Model 3

```
[35] #Building a neural network with batch normalization and dropout layers
     input_data = Input(shape= X_train[0].shape)
     hidden_Layer_1 = Dense(30 , activation='relu')(input_data)
     model3 = BatchNormalization()(hidden_Layer_1)
     model3 = Dropout(0.4)(model3)
     hidden_Layer_2 = Dense(29, activation='relu')(model3)
     hidden_Layer_3 = Dense(28, activation='relu')(hidden_Layer_2)
     model3 = BatchNormalization()(hidden_Layer_3)
     model3 = Dropout(0.4)(model3)
     concat = Concatenate(axis=1)
     model3 = Flatten()(model3)
     output = Dense(10, activation='softmax')(model3)
     model3 = keras.models.Model(inputs = [input_data], outputs = [output])
```

```
model3.summary()
```

```
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_3 (InputLayer)        [(None, 28, 28)]          0         
                                                                 
 dense_12 (Dense)            (None, 28, 30)            870       
                                                                 
 batch_normalization_4 (Batc  (None, 28, 30)           120       
 hNormalization)                                                 
                                                                 
 dropout_6 (Dropout)         (None, 28, 30)            0         
                                                                 
 dense_13 (Dense)            (None, 28, 29)            899       
                                                                 
 dense_14 (Dense)            (None, 28, 28)            840       
                                                                 
 batch_normalization_5 (Batc  (None, 28, 28)           112       
 hNormalization)                                                 
                                                                 
 dropout_7 (Dropout)         (None, 28, 28)            0         
                                                                 
 flatten_4 (Flatten)         (None, 784)               0         
                                                                 
 dense_15 (Dense)            (None, 10)                7850      
                                                                 
=================================================================
Total params: 10,691
Trainable params: 10,575
Non-trainable params: 116
_________________________________________________________________
```

Step 13:

Compile the model

The model is configured before training. While configuring we use the following attributes to compile the model.

Loss function = "sparse_categorical_crossentropy" ; Measured the accuracy of the model while training.

Optimizer = "adam"; Updates based on data it perceives and loss function

Metrics = ["accuracy"]; Monitors the training and testing steps.

```
model3.compile(loss="sparse_categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
```

Step 14 :

Fitting the model with the data.

```
#Training Model with epochs = 15
history = model3.fit(X_train, y_train, epochs=15, validation_data=(X_valid, y_valid))
```

```
Epoch 1/15
1719/1719 [==============================] - 11s 6ms/step - loss: 0.6946 - accuracy: 0.7561 - val_loss: 0.4353 - val_accuracy: 0.8442
Epoch 2/15
1719/1719 [==============================] - 10s 6ms/step - loss: 0.4935 - accuracy: 0.8236 - val_loss: 0.3709 - val_accuracy: 0.8700
Epoch 3/15
1719/1719 [==============================] - 10s 6ms/step - loss: 0.4575 - accuracy: 0.8344 - val_loss: 0.3655 - val_accuracy: 0.8664
Epoch 4/15
1719/1719 [==============================] - 10s 6ms/step - loss: 0.4374 - accuracy: 0.8390 - val_loss: 0.3506 - val_accuracy: 0.8732
Epoch 5/15
1719/1719 [==============================] - 10s 6ms/step - loss: 0.4252 - accuracy: 0.8437 - val_loss: 0.3467 - val_accuracy: 0.8782
Epoch 6/15
1719/1719 [==============================] - 10s 6ms/step - loss: 0.4134 - accuracy: 0.8486 - val_loss: 0.3352 - val_accuracy: 0.8788
Epoch 7/15
1719/1719 [==============================] - 10s 6ms/step - loss: 0.4076 - accuracy: 0.8511 - val_loss: 0.3355 - val_accuracy: 0.8806
```

Step 15:

Evaluating the model and predicting the accuracy.

```
#Evaluating model and deriving the accuracy results
score3 = model3.evaluate(X_test,y_test, verbose=0)
print('Test Loss: {:.4f}%'.format(score3[0]*100))
print('Test Accuracy : {:.4f}%'.format(score3[1]*100))
```

```
Test Loss: 35.1427%
Test Accuracy : 87.3800%
```

Step 16:

Finding average of the accuracy of the three evaluated models.

```
#For determining accuracy of the committee of the 3 models.
average_accuracy = (score[1]*100+score2[1]*100+score3[1]*100)/3.0
print(average_accuracy)
```

```
88.69666655858357
```

Result Discussion: -

· The project utilises Fashion MNIST dataset sample for evaluation.

· I have created 3 convolutional neural networks , using different architecture as described above. The three CNN models have been compiled and fitted on the given Fashion_MNIST data. The models are therefore evaluated and their individual accuracy is captured.

· The table below presents the individual accuracy results for each model.

Model	Accuracy Rate	Epochs
Model 1	88.6600%	15
Model 2	89.6200%	15
Model 3	87.3800%	15

Table 1.4 : Represents the accuracy rate for the 3 models .

· Creating a committee of the 3 CNN models by averaging their outputs. Average accuracy of the ensemble is acquired by taking the average of the accuracy of the three neural networks for each test data. (Step 16).

· The averaged accuracy is the final accuracy which is 88.6966%. The overall accuracy of the committee is quite high suggesting that the committee can be used for further prediction and decision making. The CNN models can be improved to result in a better committee performance and decrease error rates.