CNN Implementation on fashion_mnist dataset from keras.datasets

```
from keras.datasets import fashion_mnist
(train_X,train_Y), (test_X,test_Y) = fashion_mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-u
29515/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-u
26421880/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ub
5148/5148 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ub
4422102/4422102 [==============================] - 0s 0us/step
```

```
#Analyzing the dat
import numpy as np
from keras.utils import to_categorical
import matplotlib.pyplot as plt
%matplotlib inline

print('Training data shape : ', train_X.shape, train_Y.shape)

print('Testing data shape : ', test_X.shape, test_Y.shape)
```

```
Training data shape :  (60000, 28, 28) (60000,)
Testing data shape :  (10000, 28, 28) (10000,)
```

Data Preprocessing

```
#The current data is in an int8 format, therefore to feed it into teh network
#the conversion to float32 is needed
train_X = train_X.astype('float32')
test_X = test_X.astype('float32')
train_X = train_X / 255   #rescaling the pixel kvalues in range 0 - 1
test_X = test_X / 255


# Change the labels from categorical to one-hot encoding
train_Y_one_hot = to_categorical(train_Y)
test_Y_one_hot = to_categorical(test_Y)

# Display the change for category label using one-hot encoding
print('Original label:', train_Y[0])
print('After conversion to one-hot:', train_Y_one_hot[0])
```

```
Original label: 9
After conversion to one-hot: [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
```

```
from sklearn.model_selection import train_test_split
train_X,valid_X,train_label,valid_label = train_test_split(train_X, train_Y_one_hot, test_size=0.2, random
```

Modelling of the data

```
#importing libraries
import keras
from keras.models import Sequential, Model
from keras.layers import Dense, Dropout, Flatten,Input
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import BatchNormalization
from keras.layers import LeakyReLU


batch_size = 64
epochs = 20
num_classes = 10


fashion_model = Sequential()
fashion_model.add(Conv2D(32, kernel_size=(3, 3),activation='linear',input_shape=(28,28,1),padding='same'))
fashion_model.add(LeakyReLU(alpha=0.1))
fashion_model.add(MaxPooling2D((2, 2),padding='same'))
fashion_model.add(Conv2D(64, (3, 3), activation='linear',padding='same'))
fashion_model.add(LeakyReLU(alpha=0.1))
fashion_model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
fashion_model.add(Conv2D(128, (3, 3), activation='linear',padding='same'))
fashion_model.add(LeakyReLU(alpha=0.1))
fashion_model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
fashion_model.add(Flatten())
fashion_model.add(Dense(128, activation='linear'))
fashion_model.add(LeakyReLU(alpha=0.1))
fashion_model.add(Dense(num_classes, activation='softmax'))
```

Compile the model

```
fashion_model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.Adam(),metric


fashion_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 32)        320

 leaky_re_lu (LeakyReLU)     (None, 28, 28, 32)        0

 max_pooling2d (MaxPooling2D  (None, 14, 14, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 14, 14, 64)        18496

 leaky_re_lu_1 (LeakyReLU)   (None, 14, 14, 64)        0
```

```
 max_pooling2d_1 (MaxPooling  (None, 7, 7, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 7, 7, 128)         73856

 leaky_re_lu_2 (LeakyReLU)   (None, 7, 7, 128)         0

 max_pooling2d_2 (MaxPooling  (None, 4, 4, 128)        0
 2D)

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 128)               262272

 leaky_re_lu_3 (LeakyReLU)   (None, 128)               0

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 356,234
Trainable params: 356,234
Non-trainable params: 0
_________________________________________________________________
```

Train the Model

```
fashion_train = fashion_model.fit(train_X, train_label, batch_size=batch_size,epochs=epochs,verbose=1,vali
```

```
Epoch 1/20
750/750 [==============================] - 69s 90ms/step - loss: 0.4649 - accuracy: 0.8292 - val_loss
Epoch 2/20
750/750 [==============================] - 66s 88ms/step - loss: 0.2924 - accuracy: 0.8931 - val_loss
Epoch 3/20
750/750 [==============================] - 69s 92ms/step - loss: 0.2445 - accuracy: 0.9099 - val_loss
Epoch 4/20
750/750 [==============================] - 63s 84ms/step - loss: 0.2119 - accuracy: 0.9223 - val_loss
Epoch 5/20
750/750 [==============================] - 62s 83ms/step - loss: 0.1841 - accuracy: 0.9315 - val_loss
Epoch 6/20
750/750 [==============================] - 65s 87ms/step - loss: 0.1604 - accuracy: 0.9396 - val_loss
Epoch 7/20
750/750 [==============================] - 67s 89ms/step - loss: 0.1403 - accuracy: 0.9476 - val_loss
Epoch 8/20
750/750 [==============================] - 65s 87ms/step - loss: 0.1207 - accuracy: 0.9555 - val_loss
Epoch 9/20
750/750 [==============================] - 68s 91ms/step - loss: 0.1023 - accuracy: 0.9619 - val_loss
Epoch 10/20
750/750 [==============================] - 63s 83ms/step - loss: 0.0871 - accuracy: 0.9671 - val_loss
Epoch 11/20
750/750 [==============================] - 63s 84ms/step - loss: 0.0752 - accuracy: 0.9721 - val_loss
Epoch 12/20
750/750 [==============================] - 62s 82ms/step - loss: 0.0668 - accuracy: 0.9758 - val_loss
Epoch 13/20
750/750 [==============================] - 62s 83ms/step - loss: 0.0548 - accuracy: 0.9796 - val_loss
Epoch 14/20
750/750 [==============================] - 62s 82ms/step - loss: 0.0525 - accuracy: 0.9803 - val_loss
Epoch 15/20
750/750 [==============================] - 62s 82ms/step - loss: 0.0393 - accuracy: 0.9851 - val_loss
Epoch 16/20
750/750 [==============================] - 62s 82ms/step - loss: 0.0427 - accuracy: 0.9844 - val_loss
```

```
Epoch 17/20
750/750 [==============================] - 62s 82ms/step - loss: 0.0353 - accuracy: 0.9869 - val_loss
Epoch 18/20
750/750 [==============================] - 62s 82ms/step - loss: 0.0320 - accuracy: 0.9884 - val_loss
Epoch 19/20
750/750 [==============================] - 63s 85ms/step - loss: 0.0302 - accuracy: 0.9887 - val_loss
Epoch 20/20
750/750 [==============================] - 62s 82ms/step - loss: 0.0323 - accuracy: 0.9881 - val_loss
```

The epochs value can be changed to increase the accuracy.

Model Evaluation on the Test Set

```
test_eval = fashion_model.evaluate(test_X, test_Y_one_hot, verbose=0)
```

```
print('Test loss:', test_eval[0])
print('Test accuracy:', test_eval[1])
```

```
Test loss: 0.4725803732872009
Test accuracy: 0.9205999970436096
```

Predict Labels

```
predicted_classes = fashion_model.predict(test_X)
```

```
predicted_classes = np.argmax(np.round(predicted_classes),axis=1)
```