```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import keras
from keras.models import Sequential
from keras.layers import Conv2D,MaxPooling2D,Dense,Flatten,Dropout
from keras.optimizers import Adam
from keras.callbacks import TensorBoard
num_classes = 10
epochs = 20
```

```
train_df = pd.read_csv('/content/fashion-mnist_train.csv',sep=',')
```

```
test_df = pd.read_csv('/content/fashion-mnist_train.csv',sep=',')
```

```
train_df.head()
```

| | label | pixel1 | pixel2 | pixel3 | pixel4 | pixel5 | pixel6 | pixel7 | pixel8 | pixel9 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... |
| **1** | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... |
| **2** | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | ... |
| **3** | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 0 | 0 | 0 | ... |
| **4** | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... |

5 rows × 785 columns

```
test_df.head()
```

| | label | pixel1 | pixel2 | pixel3 | pixel4 | pixel5 | pixel6 | pixel7 | pixel8 | pixel9 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... |
| **1** | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... |
| **2** | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | ... |
| **3** | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 0 | 0 | 0 | ... |
| **4** | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... |

5 rows × 785 columns

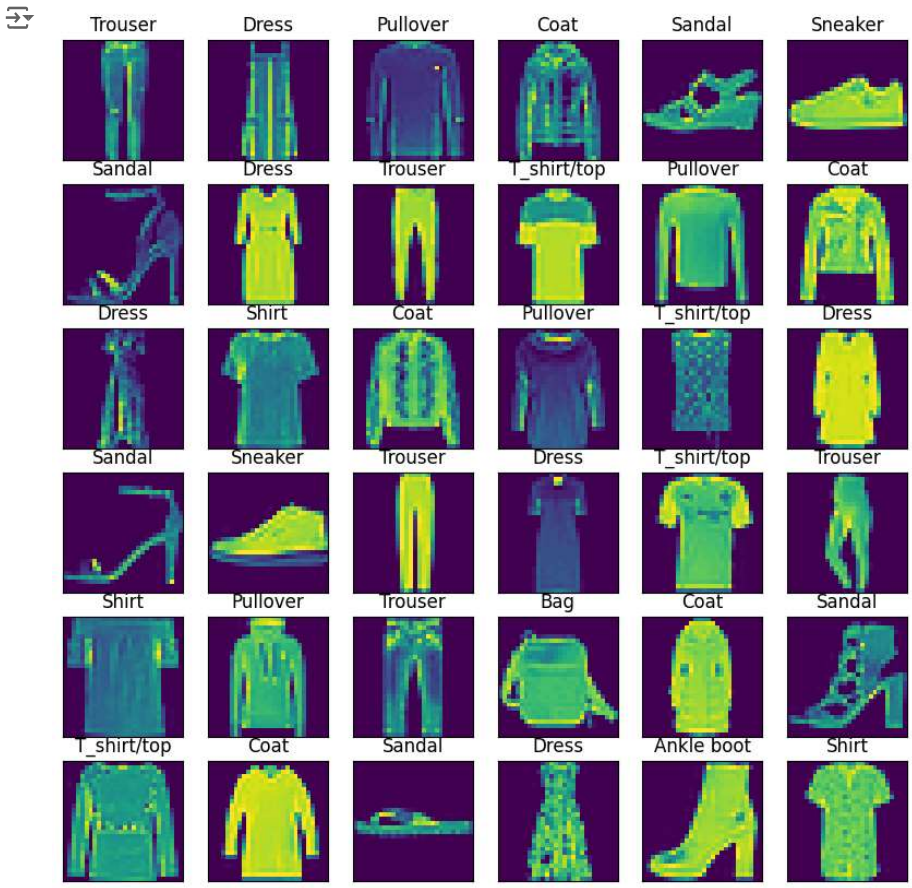
```
train_data = np.array(train_df, dtype = 'float32')
```

```
test_data = np.array(test_df, dtype='float32')
```

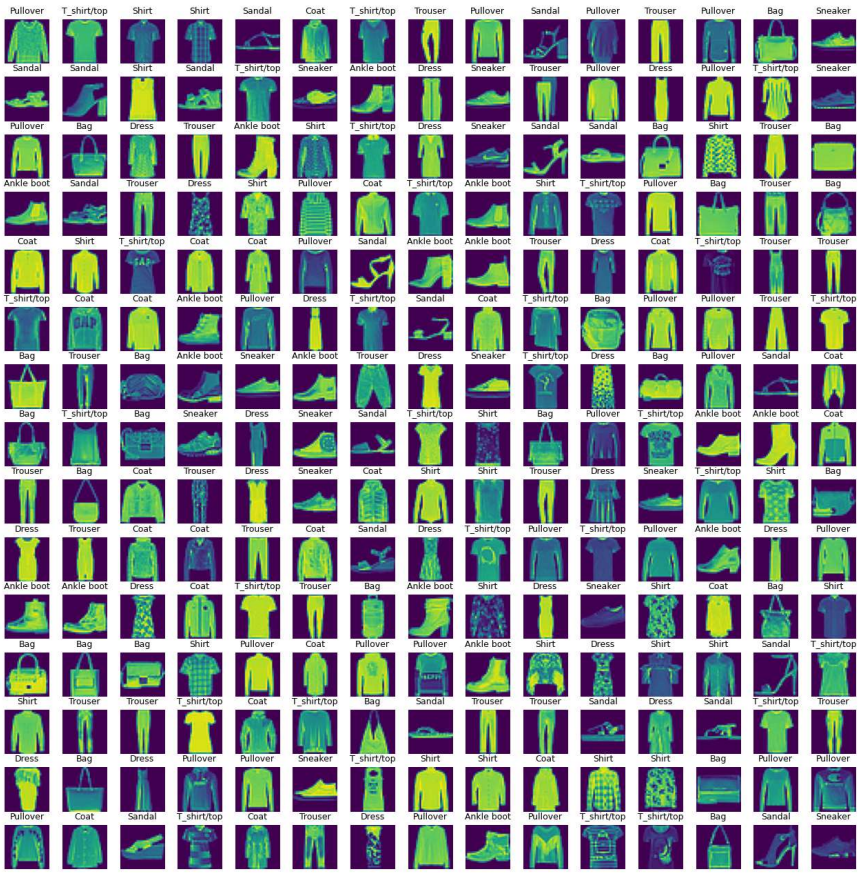
```
x_train = train_data[:,1:]/255

y_train = train_data[:,0]

x_test= test_data[:,1:]/255

y_test=test_data[:,0]
```

```
x_train,x_validate,y_train,y_validate = train_test_split(x_train,y_train,test_size = 0.2,random_state = 42)
```

```
class_names = ['T_shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
plt.figure(figsize=(10, 10))
for i in range(36):
    plt.subplot(6, 6, i + 1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(x_train[i].reshape((28,28)))
    label_index = int(y_train[i])
    plt.title(class_names[label_index])
plt.show()
```




```
W_grid = 15
L_grid = 15

fig, axes = plt.subplots(L_grid, W_grid, figsize = (16,16))
axes = axes.ravel() # flaten the 15 x 15 matrix into 225 array
n_train = len(train_data) # get the length of the train dataset

# Select a random number from 0 to n_train
for i in np.arange(0, W_grid * L_grid): # create evenly spaces variables

    # Select a random number
    index = np.random.randint(0, n_train)
    # read and display an image with the selected index
    axes[i].imshow( train_data[index,1:].reshape((28,28)) )
    labelindex = int(train_data[index,0])
    axes[i].set_title(class_names[labelindex], fontsize = 9)
    axes[i].axis('off')

plt.subplots_adjust(hspace=0.3)
```



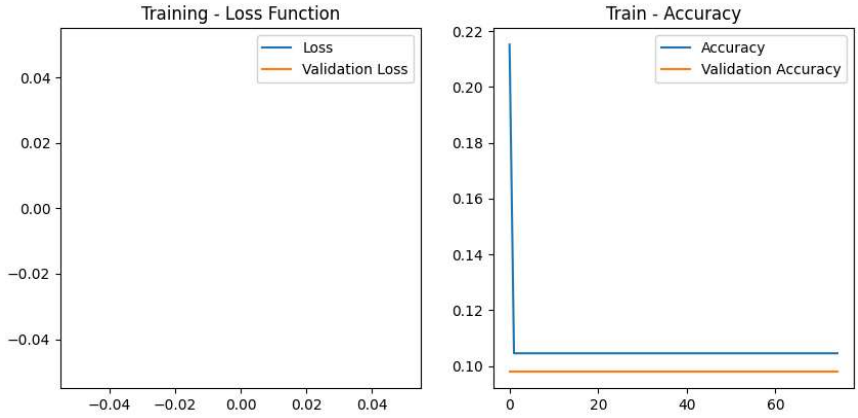

```
image_rows = 28
image_cols = 28
batch_size = 4096
image_shape = (image_rows,image_cols,1)


x_train = x_train.reshape(x_train.shape[0],*image_shape)
x_test = x_test.reshape(x_test.shape[0],*image_shape)
x_validate = x_validate.reshape(x_validate.shape[0],*image_shape)


cnn_model = Sequential([
    Conv2D(filters=32,kernel_size=3,activation='relu',input_shape = image_shape),
    MaxPooling2D(pool_size=2) ,# down sampling the output instead of 28*28 it is 14*14
    Dropout(0.2),
    Flatten(), # flatten out the layers
    Dense(32,activation='relu'),
    Dense(10,activation = 'softmax')

])
```

```
cnn_model.compile(loss ='sparse_categorical_crossentropy', optimizer='adam',metrics =['accuracy'])


history = cnn_model.fit(
    x_train,
    y_train,
    batch_size=4096,
    epochs=75,
    verbose=1,
    validation_data=(x_validate,y_validate),
)
```

```
Epoch 47/75
4/4 [==============================] - 7s 2s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 48/75
4/4 [==============================] - 9s 3s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 49/75
4/4 [==============================] - 7s 2s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 50/75
4/4 [==============================] - 9s 2s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 51/75
4/4 [==============================] - 7s 2s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 52/75
4/4 [==============================] - 9s 2s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 53/75
4/4 [==============================] - 8s 2s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 54/75
4/4 [==============================] - 9s 2s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 55/75
4/4 [==============================] - 9s 2s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 56/75
4/4 [==============================] - 7s 2s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 57/75
4/4 [==============================] - 9s 2s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 58/75
4/4 [==============================] - 7s 2s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 59/75
4/4 [==============================] - 9s 2s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 60/75
4/4 [==============================] - 7s 2s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 61/75
4/4 [==============================] - 9s 2s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 62/75
4/4 [==============================] - 8s 2s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 63/75
4/4 [==============================] - 8s 2s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 64/75
4/4 [==============================] - 9s 2s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 65/75
4/4 [==============================] - 7s 2s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 66/75
4/4 [==============================] - 9s 3s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 67/75
4/4 [==============================] - 7s 2s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 68/75
4/4 [==============================] - 9s 3s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 69/75
4/4 [==============================] - 7s 2s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 70/75
4/4 [==============================] - 9s 2s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 71/75
4/4 [==============================] - 8s 2s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 72/75
4/4 [==============================] - 9s 2s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 73/75
4/4 [==============================] - 18s 5s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 74/75
4/4 [==============================] - 8s 2s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
Epoch 75/75
4/4 [==============================] - 9s 2s/step - loss: nan - accuracy: 0.1045 - val_loss: nan - val_accuracy: 0.0979
```

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

plt.subplot(2, 2, 1)
plt.plot(history.history['loss'], label='Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.title('Training - Loss Function')

plt.subplot(2, 2, 2)
plt.plot(history.history['accuracy'], label='Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.title('Train - Accuracy')
```

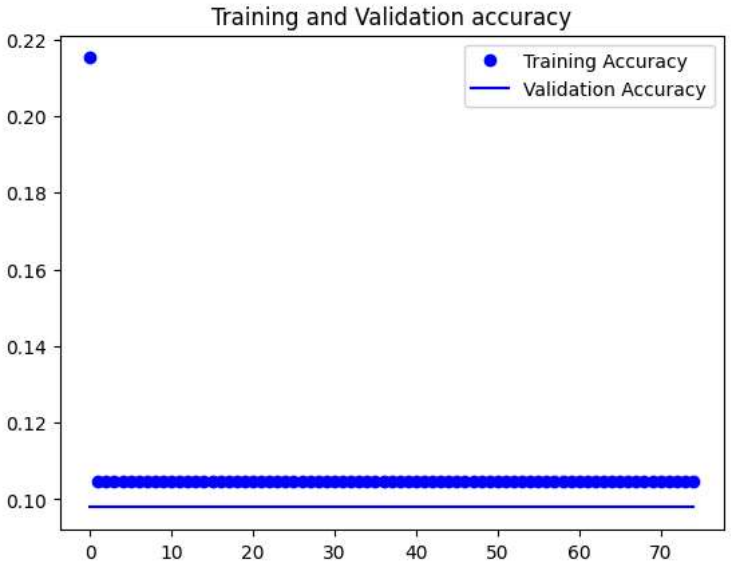
```
Text(0.5, 1.0, 'Train - Accuracy')
```

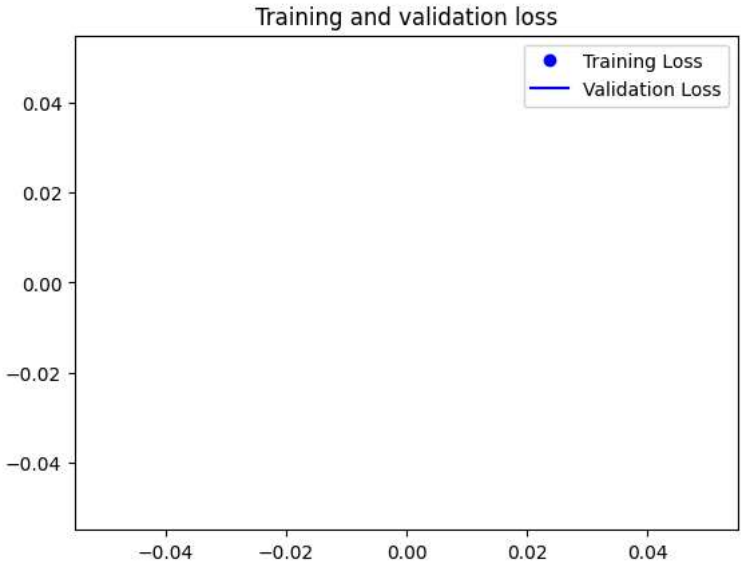



```
score = cnn_model.evaluate(x_test,y_test,verbose=0)
print('Test Loss : {:.4f}'.format(score[0]))
print('Test Accuracy : {:.4f}'.format(score[1]))
```

```
Test Loss : nan
Test Accuracy : 0.1025
```

```
import matplotlib.pyplot as plt
%matplotlib inline
accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(len(accuracy))
plt.plot(epochs, accuracy, 'bo', label='Training Accuracy')
plt.plot(epochs, val_accuracy, 'b', label='Validation Accuracy')
plt.title('Training and Validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training Loss')
plt.plot(epochs, val_loss, 'b', label='Validation Loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()
```




```
# Get the predictions for the test data
predictions = cnn_model.predict(x_test)

# Convert probabilities to class labels
predicted_classes = np.argmax(predictions, axis=1)

# Get the true labels
y_true = test_df.iloc[:, 0].values

# Find indices of correct and incorrect predictions
correct = np.nonzero(predicted_classes == y_true)[0]
incorrect = np.nonzero(predicted_classes != y_true)[0]

# Generate classification report
from sklearn.metrics import classification_report

target_names = ["Class {}".format(i) for i in range(num_classes)]
print(classification_report(y_true, predicted_classes, target_names=target_names))
```

```
651/651 [==============================] - 3s 5ms/step
                precision    recall  f1-score   support

       Class 0       0.10      1.00      0.19      2135
       Class 1       0.00      0.00      0.00      2100
       Class 2       0.00      0.00      0.00      2038
       Class 3       0.00      0.00      0.00      2111
       Class 4       0.00      0.00      0.00      2039
       Class 5       0.00      0.00      0.00      2063
       Class 6       0.00      0.00      0.00      2093
       Class 7       0.00      0.00      0.00      2134
       Class 8       0.00      0.00      0.00      2045
       Class 9       0.00      0.00      0.00      2063

      accuracy                           0.10     20821
     macro avg       0.01      0.10      0.02     20821
  weighted avg       0.01      0.10      0.02     20821
```

```
L = 4
W = 4
fig, axes = plt.subplots(L, W, figsize = (12,12))
axes = axes.ravel()

for i in np.arange(0, L * W):
    axes[i].imshow(x_test[i].reshape(28,28))
    axes[i].set_title(f"Prediction Class = {predicted_classes[i]:0.1f}\n Original Class = {y_test[i]:0.1f}")
    axes[i].axis('off')

plt.subplots_adjust(wspace=0.5)
```