Fashion-MNIST

Exploring The Dataset

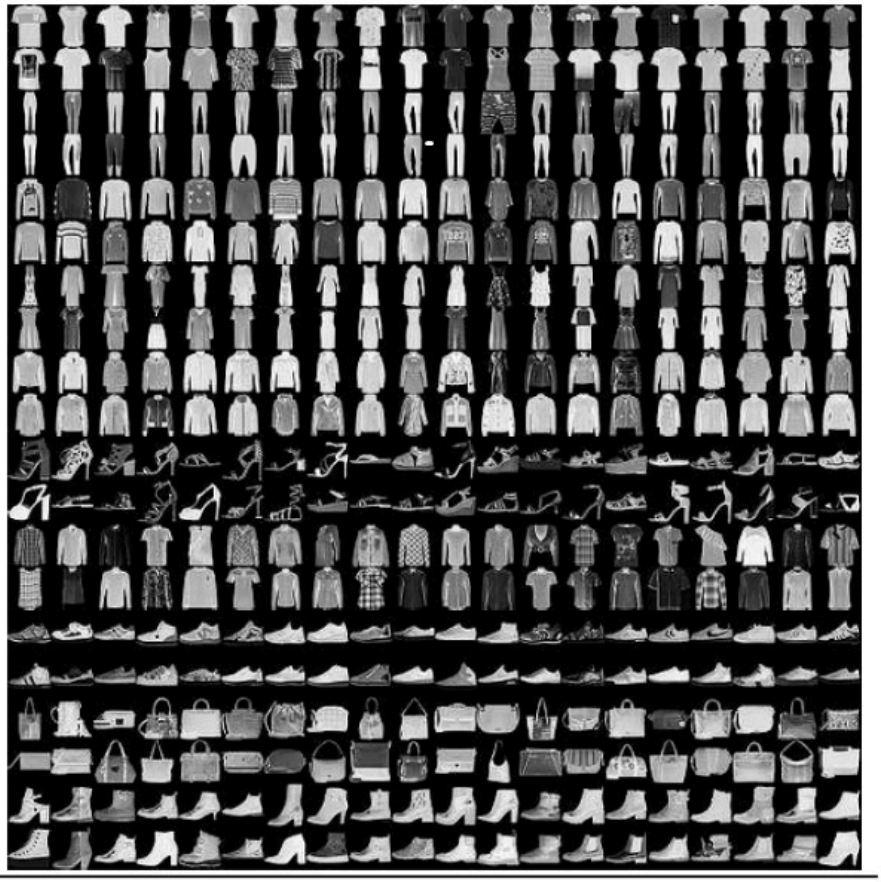
Import Libraries

In [1]:
```
import numpy as np
from keras.models import Sequential
from keras.layers import Dropout, Dense, Flatten
from tensorflow.keras.optimizers import SGD, Adam
from keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.utils import to_categorical
from keras.datasets import fashion_mnist
from matplotlib import pyplot as plt
import warnings
warnings.filterwarnings("ignore")
import seaborn as sns
```

Load Data

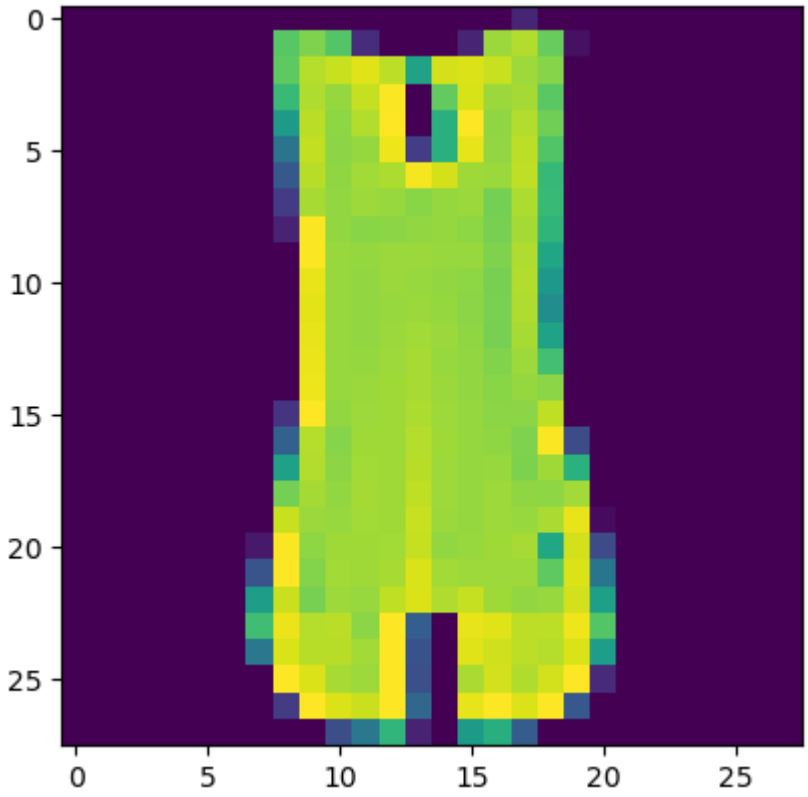
In [2]:
```
#Lets start by loading the Cifar10 data
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
print(x_train.shape, y_train.shape)
print(x_test.shape, y_test.shape)
```

```
(60000, 28, 28) (60000,)
(10000, 28, 28) (10000,)
```

Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

In [3]:
```
plt.imshow(x_train[4])
```

Out[3]:
```
<matplotlib.image.AxesImage at 0x205e7b9c460>
```



Process Data

In [4]:
```
x_train = x_train.astype('float32')/255.0
x_test =  x_test.astype('float32')/255.0
```

In [5]:
```
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```

In [6]:
```
print (y_train.shape)
```

```
(60000, 10)
```

Define Model

In [7]:
```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1), p
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))
model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))
```

Model Summary

In [8]:
```
model.summary()
```

```
Model: "sequential"
```

Layer (type)	Output Shape	Pa
conv2d (Conv2D)	(None, 28, 28, 32)	
conv2d_1 (Conv2D)	(None, 28, 28, 64)	1
max_pooling2d (MaxPooling2D)	(None, 14, 14, 64)	
dropout (Dropout)	(None, 14, 14, 64)	
flatten (Flatten)	(None, 12544)	
dense (Dense)	(None, 256)	3,21
dropout_1 (Dropout)	(None, 256)	
dense_1 (Dense)	(None, 10)	

Total params: 3,232,906 (12.33 MB)

Trainable params: 3,232,906 (12.33 MB)

Compile Model

In [9]:
```
model.compile(loss='categorical_crossentropy', optimizer=Adam(), metrics=['accuracy'])
```

Fit data to model

In [10]:
```
epochs = 20
batch_size = 128
hist = model.fit(x_train, y_train, batch_size=batch_size,
                 epochs=epochs,verbose=1,
                 validation_data=(x_test, y_test))
print("The model has successfully trained")
```

```
Epoch 1/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 56s 117ms/step - accuracy: 0.7780 - loss: 0.6247 - val_a
ccuracy: 0.8886 - val_loss: 0.3040
Epoch 2/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 59s 126ms/step - accuracy: 0.8901 - loss: 0.3072 - val_a
ccuracy: 0.9010 - val_loss: 0.2646
Epoch 3/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 63s 134ms/step - accuracy: 0.9094 - loss: 0.2481 - val_a
ccuracy: 0.9095 - val_loss: 0.2378
Epoch 4/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 69s 146ms/step - accuracy: 0.9193 - loss: 0.2163 - val_a
ccuracy: 0.9083 - val_loss: 0.2488
Epoch 5/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 73s 155ms/step - accuracy: 0.9271 - loss: 0.1946 - val_a
ccuracy: 0.9166 - val_loss: 0.2222
Epoch 6/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 66s 140ms/step - accuracy: 0.9375 - loss: 0.1655 - val_a
ccuracy: 0.9242 - val_loss: 0.2082
Epoch 7/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 66s 140ms/step - accuracy: 0.9422 - loss: 0.1574 - val_a
ccuracy: 0.9245 - val_loss: 0.2128
Epoch 8/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 62s 133ms/step - accuracy: 0.9489 - loss: 0.1374 - val_a
ccuracy: 0.9283 - val_loss: 0.2076
Epoch 9/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 63s 135ms/step - accuracy: 0.9517 - loss: 0.1255 - val_a
ccuracy: 0.9295 - val_loss: 0.2075
Epoch 10/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 66s 140ms/step - accuracy: 0.9561 - loss: 0.1163 - val_a
ccuracy: 0.9343 - val_loss: 0.2020
Epoch 11/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 65s 138ms/step - accuracy: 0.9613 - loss: 0.1027 - val_a
ccuracy: 0.9327 - val_loss: 0.2145
Epoch 12/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 62s 132ms/step - accuracy: 0.9638 - loss: 0.0969 - val_a
ccuracy: 0.9282 - val_loss: 0.2274
Epoch 13/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 58s 123ms/step - accuracy: 0.9664 - loss: 0.0874 - val_a
ccuracy: 0.9323 - val_loss: 0.2220
Epoch 14/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 59s 126ms/step - accuracy: 0.9702 - loss: 0.0772 - val_a
ccuracy: 0.9280 - val_loss: 0.2378
Epoch 15/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 66s 142ms/step - accuracy: 0.9710 - loss: 0.0749 - val_a
ccuracy: 0.9355 - val_loss: 0.2297
Epoch 16/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 66s 141ms/step - accuracy: 0.9745 - loss: 0.0673 - val_a
ccuracy: 0.9328 - val_loss: 0.2449
Epoch 17/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 63s 135ms/step - accuracy: 0.9756 - loss: 0.0648 - val_a
ccuracy: 0.9331 - val_loss: 0.2521
Epoch 18/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 64s 136ms/step - accuracy: 0.9785 - loss: 0.0576 - val_a
ccuracy: 0.9327 - val_loss: 0.2535
Epoch 19/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 63s 133ms/step - accuracy: 0.9793 - loss: 0.0564 - val_a
ccuracy: 0.9335 - val_loss: 0.2568
Epoch 20/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 58s 124ms/step - accuracy: 0.9803 - loss: 0.0543 - val_a
```

```
ccuracy: 0.9376 - val_loss: 0.2810
The model has successfully trained
```

Save the Model to use for later

In [11]:
```
model.save_weights('FashionMNIST.weights.h5')
print("Saving the model as FashionMNIST.h5")
```

```
Saving the model as FashionMNIST.h5
```

Evaluate Model

In [12]:
```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

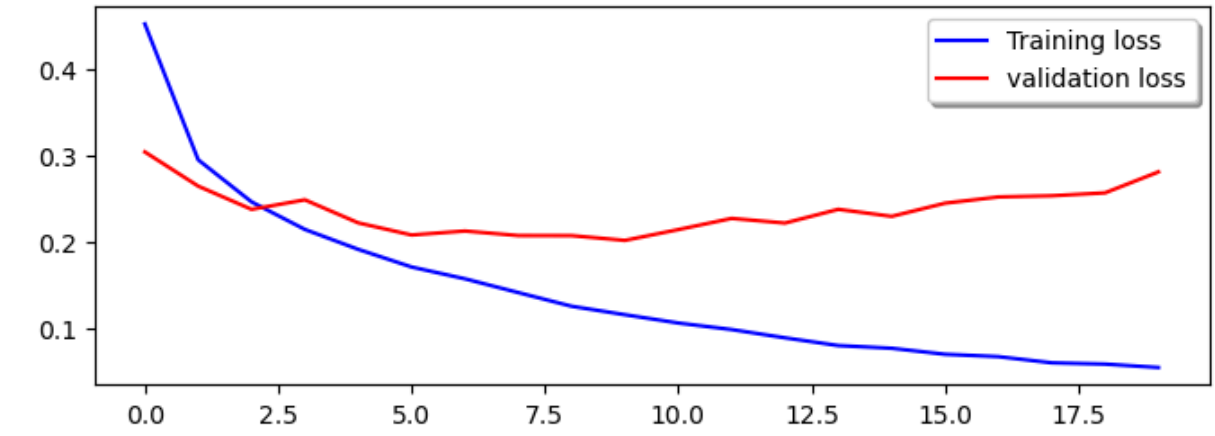
```
Test loss: 0.27781593799591064
Test accuracy: 0.9376000165939331
```

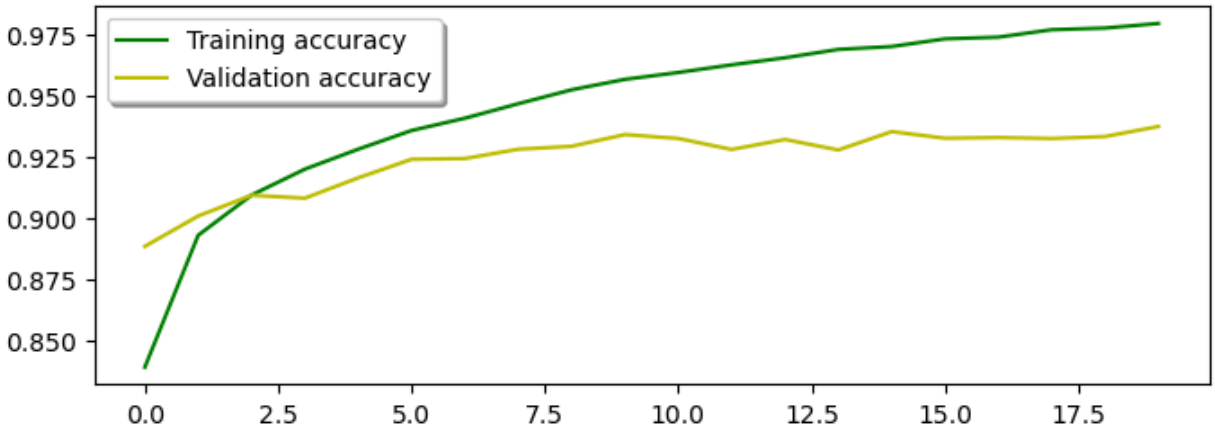
This model predicted 93.76% of Test Images correctly, which indicates that the model did pretty good job in generalizing the data.

Plotting the Training and Validation Curves

In [13]:
```
fig, ax = plt.subplots(2,1, figsize=(8, 6))
ax[0].plot(hist.history['loss'], color='b', label="Training loss")
ax[0].plot(hist.history['val_loss'], color='r',
           label="validation loss",axes =ax[0])
legend = ax[0].legend(loc='best', shadow=True)

ax[1].plot(hist.history['accuracy'], color='g', label="Training accuracy")
ax[1].plot(hist.history['val_accuracy'], color='y',
           label="Validation accuracy")
legend = ax[1].legend(loc='best', shadow=True)
```



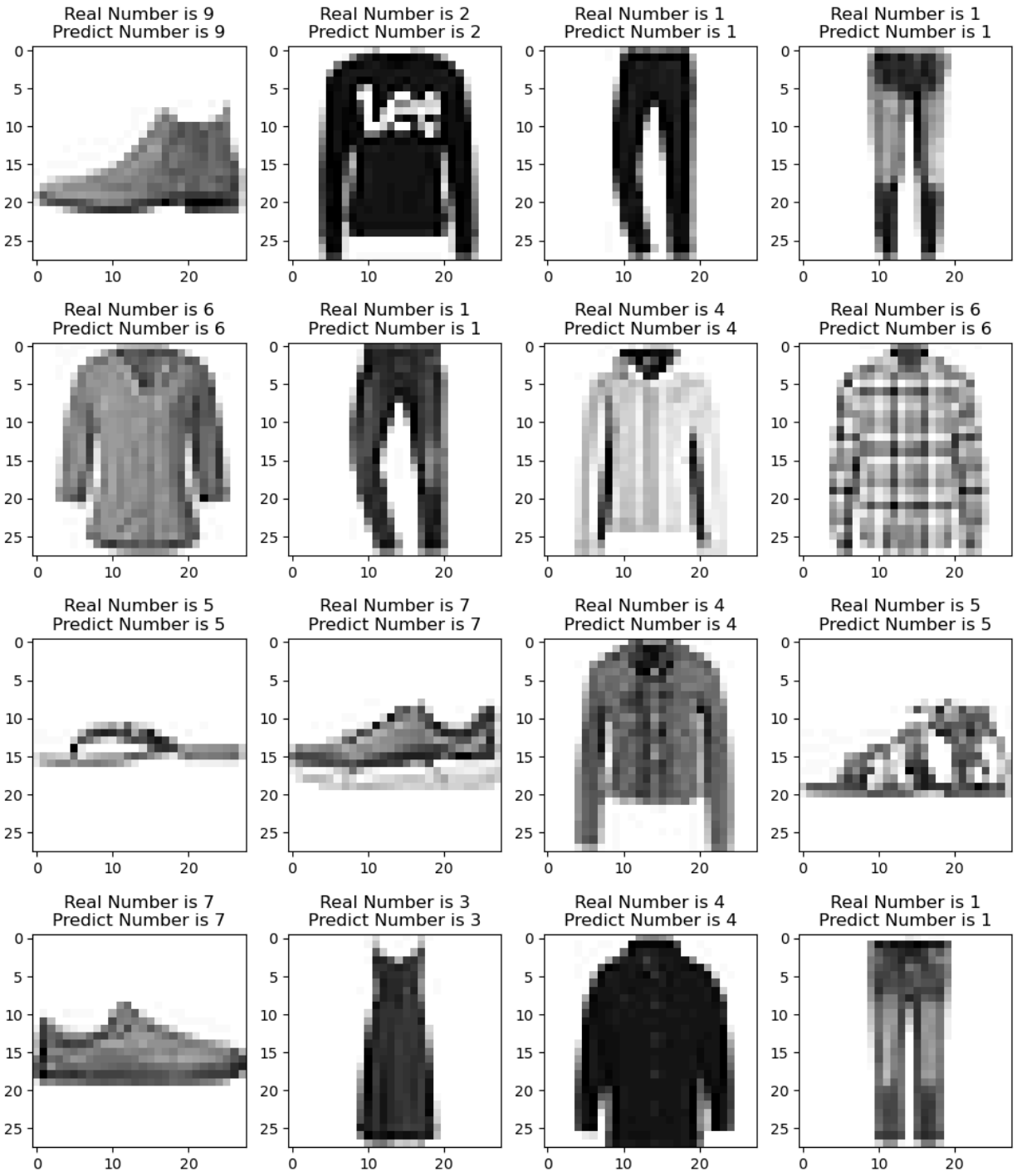


Make Predictions

In [14]:
```
y_pred = model.predict(x_test)
X_test__ = x_test.reshape(x_test.shape[0], 28, 28)

fig, axis = plt.subplots(4, 4, figsize=(12, 14))
for i, ax in enumerate(axis.flat):
    ax.imshow(X_test__[i], cmap='binary')
    ax.set(title = f"Real Number is {y_test[i].argmax()}\nPredict Number is {y_pred[i]
```

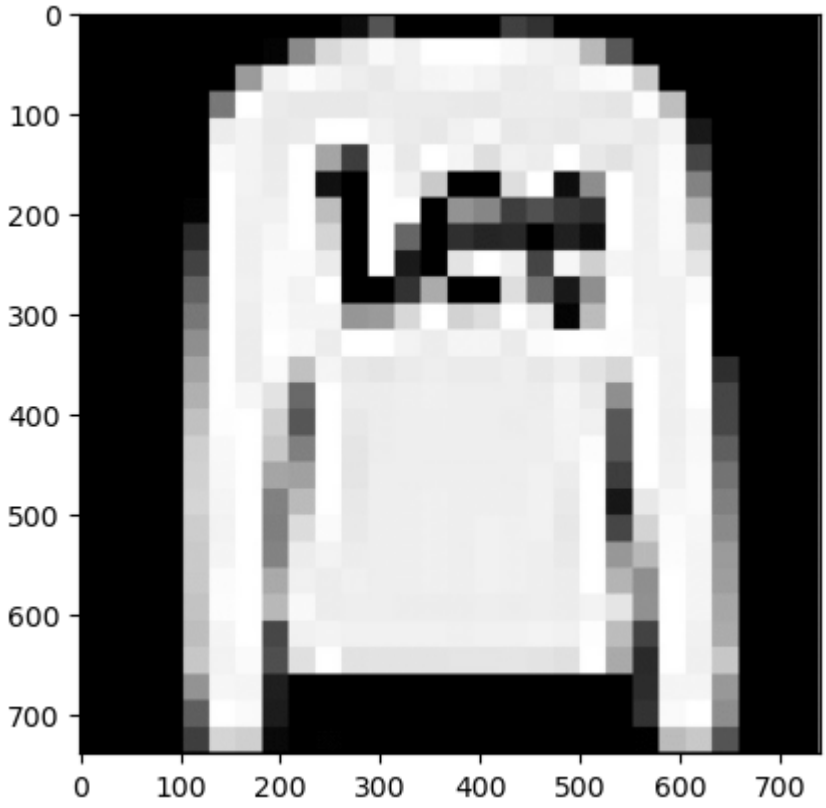
```
313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 9ms/step
```



Testing Model with unknown image

In [15]:
```
import tensorflow as tf
import cv2  as cv
from matplotlib import pyplot as plt
img=cv.imread('sample_image.jpg')
plt.imshow(img)
```

Out[15]:
```
<matplotlib.image.AxesImage at 0x205802764a0>
```



In [16]:
```
img.shape
```

Out[16]:
```
(740, 740, 3)
```

In [17]:
```
img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
img = cv.resize(img, (28, 28))
```

In [18]:
```
img.shape
```

Out[18]:
```
(28, 28)
```

In [19]:
```
from tensorflow.keras.utils import img_to_array
from tensorflow import keras
model = keras.models.load_model('FashionMNIST.h5')
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be buil
t. `model.compile_metrics` will be empty until you train or evaluate the model.
```

In [20]:
```
model.input_shape
```

Out[20]:
```
(None, 28, 28, 1)
```

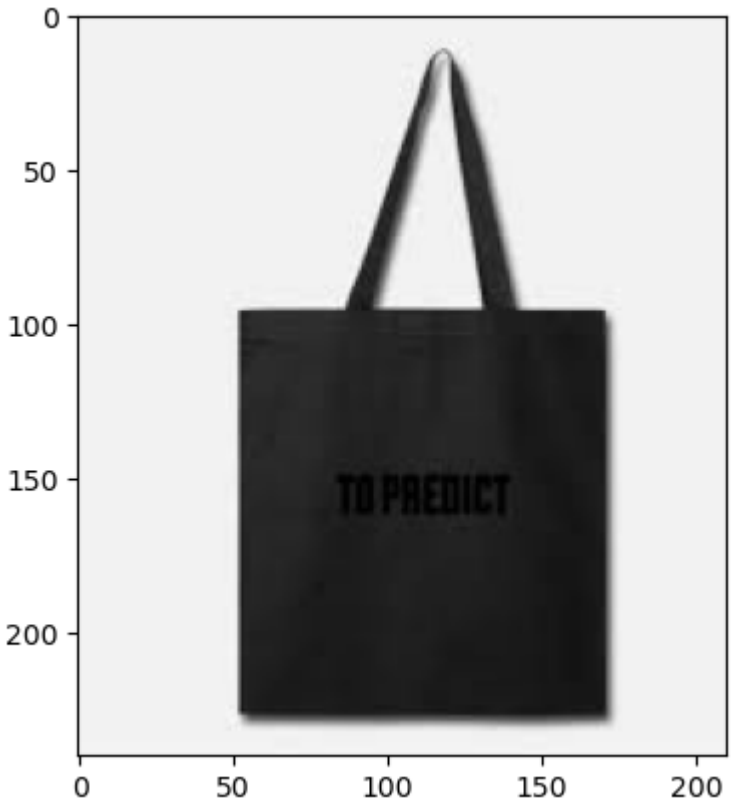
In [21]:
```
import numpy as np
from tensorflow.keras.utils import img_to_array
x = img_to_array(img)
x = np.expand_dims(x, axis=0)
preds = np.argmax(model.predict(x))
print(preds)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 219ms/step
2
```

Incorrectly Predicted Class

In [22]:
```
img2=cv.imread('sample_image2.jpg')
plt.imshow(img2)
```

Out[22]:
```
<matplotlib.image.AxesImage at 0x205e76ae320>
```



In [23]:
```
img2.shape
```

Out[23]:
```
(240, 210, 3)
```

In [24]:
```
img2 = cv.cvtColor(img2, cv.COLOR_BGR2GRAY)
img2 = cv.resize(img2, (28, 28))
```

In [25]:
```
img2.shape
```

Out[25]:
```
(28, 28)
```

In [26]:
```
import numpy as np
from tensorflow.keras.utils import img_to_array
x = img_to_array(img2)
x = np.expand_dims(x, axis=0)
preds = np.argmax(model.predict(x))
print(preds)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
2
```

Classification Report

In [27]:
```
from sklearn.metrics import confusion_matrix, classification_report
y_pred_classes = np.argmax(y_pred,axis = 1)
y_true = np.argmax(y_test,axis = 1)
classes = ['T-shirt/Top','Trouser','Pullover','Dress','Coat','Sandal','Shirt',
           'Sneaker','Bag','Ankle Boot']
print(classification_report(y_true, y_pred_classes, target_names = classes))
```

```
              precision    recall  f1-score   support

 T-shirt/Top       0.91      0.89      0.90      1000
     Trouser       1.00      0.98      0.99      1000
    Pullover       0.90      0.89      0.90      1000
       Dress       0.94      0.94      0.94      1000
        Coat       0.89      0.91      0.90      1000
      Sandal       0.99      0.99      0.99      1000
       Shirt       0.81      0.82      0.82      1000
     Sneaker       0.97      0.98      0.97      1000
         Bag       0.99      0.99      0.99      1000
  Ankle Boot       0.98      0.97      0.98      1000

    accuracy                           0.94     10000
   macro avg       0.94      0.94      0.94     10000
weighted avg       0.94      0.94      0.94     10000
```

Confusion Matrix

In [28]:
```
confusion_matrix(y_true, y_pred_classes)
sns.heatmap(confusion_matrix(y_true, y_pred_classes),annot=True,cmap='Set3',
            linewidths=0.3, linecolor='black',fmt='.0f')
```

Out[28]:
```
<Axes: >
```