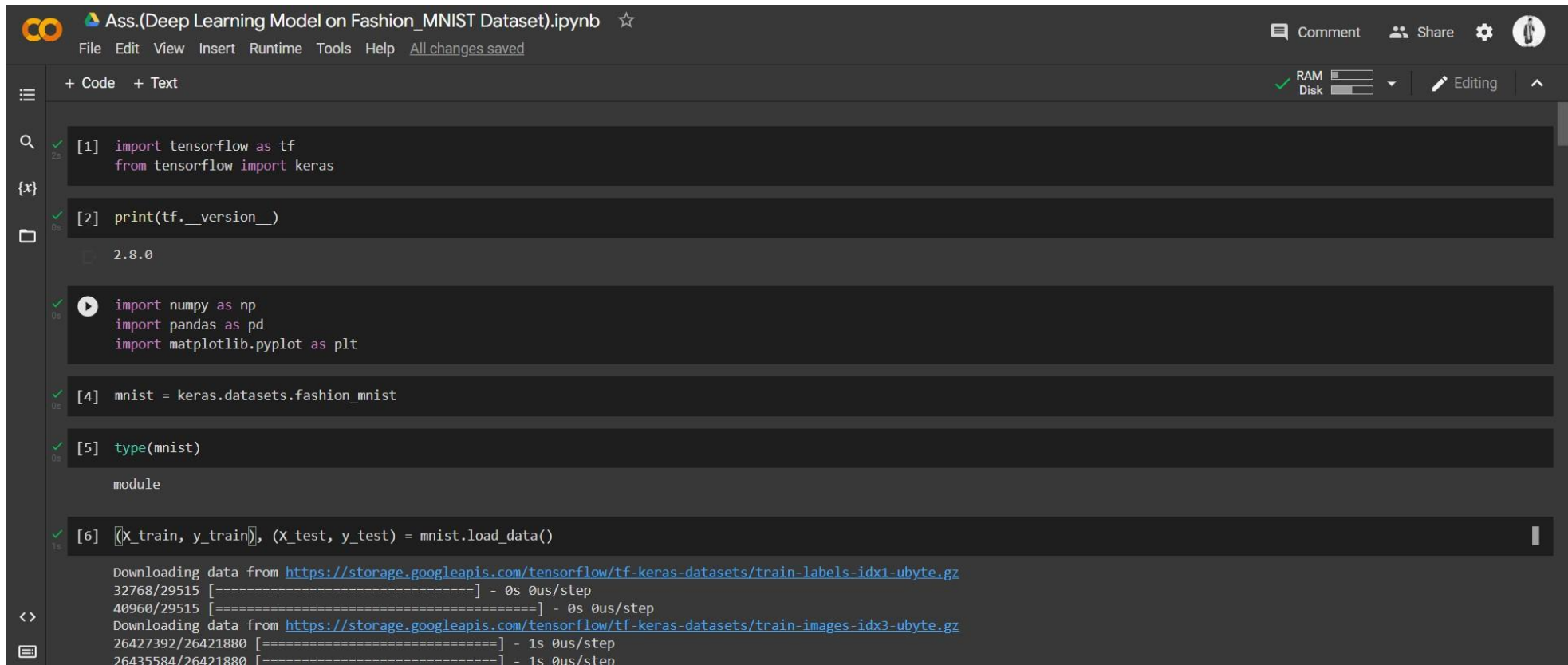


Deep Learning Model on Fashion_MNIST



The screenshot displays a Jupyter Notebook titled "Ass.(Deep Learning Model on Fashion_MNIST Dataset).ipynb". The interface includes a top menu bar with options like File, Edit, View, Insert, Runtime, Tools, and Help. On the right, there are buttons for Comment, Share, and a user profile icon, along with RAM and Disk usage indicators. The left sidebar shows a file explorer with a folder icon and a search icon. The main area contains six code cells, each with a green checkmark indicating successful execution. The code in the cells is as follows:

```
[1] import tensorflow as tf
    from tensorflow import keras

[2] print(tf.__version__)

2.8.0

[3] import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt

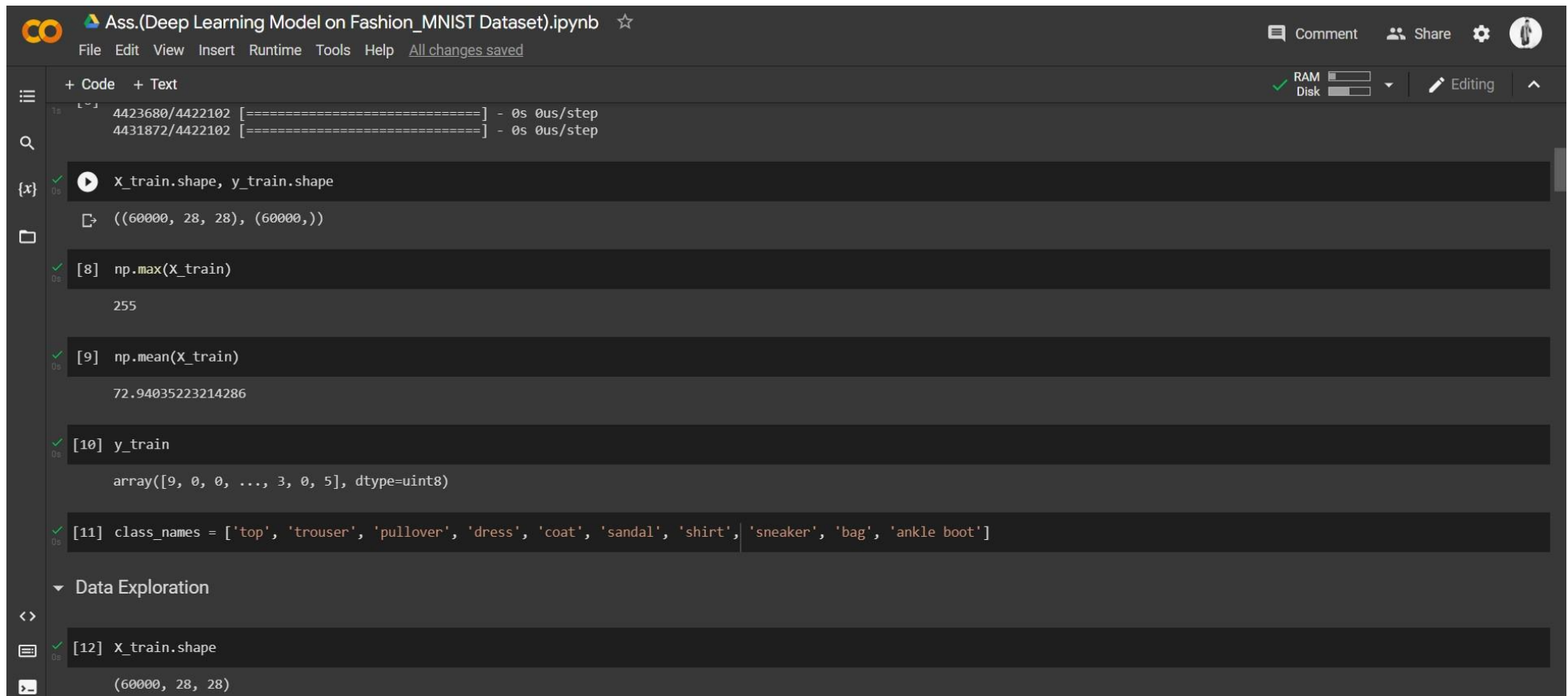
[4] mnist = keras.datasets.fashion_mnist

[5] type(mnist)

module

[6] (X_train, y_train), (X_test, y_test) = mnist.load_data()

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
32768/29515 [=====] - 0s 0us/step
40960/29515 [=====] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26427392/26421880 [=====] - 1s 0us/step
26435584/26421880 [=====] - 1s 0us/step
```



The screenshot shows a Jupyter Notebook titled "Ass.(Deep Learning Model on Fashion_MNIST Dataset).ipynb". The interface includes a top bar with a file menu, a toolbar with icons for code, text, and runtime, and a status bar showing RAM and disk usage. The notebook content is as follows:

```
4423680/4422102 [=====] - 0s 0us/step
4431872/4422102 [=====] - 0s 0us/step
```

+ Code + Text

[x] 0s `X_train.shape, y_train.shape`

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

[8] 0s `np.max(X_train)`

`255`

[9] 0s `np.mean(X_train)`

`72.94035223214286`

[10] 0s `y_train`

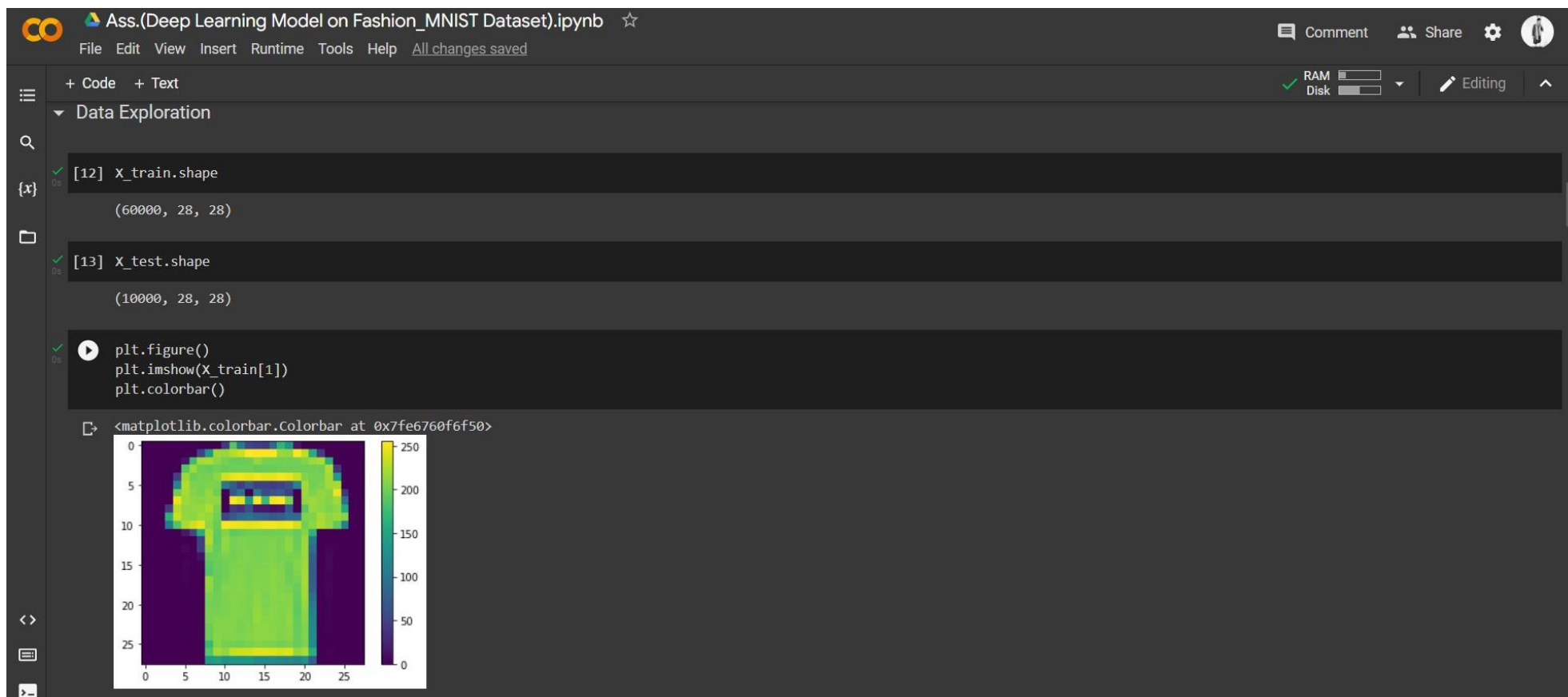
`array([9, 0, 0, ..., 3, 0, 5], dtype=uint8)`

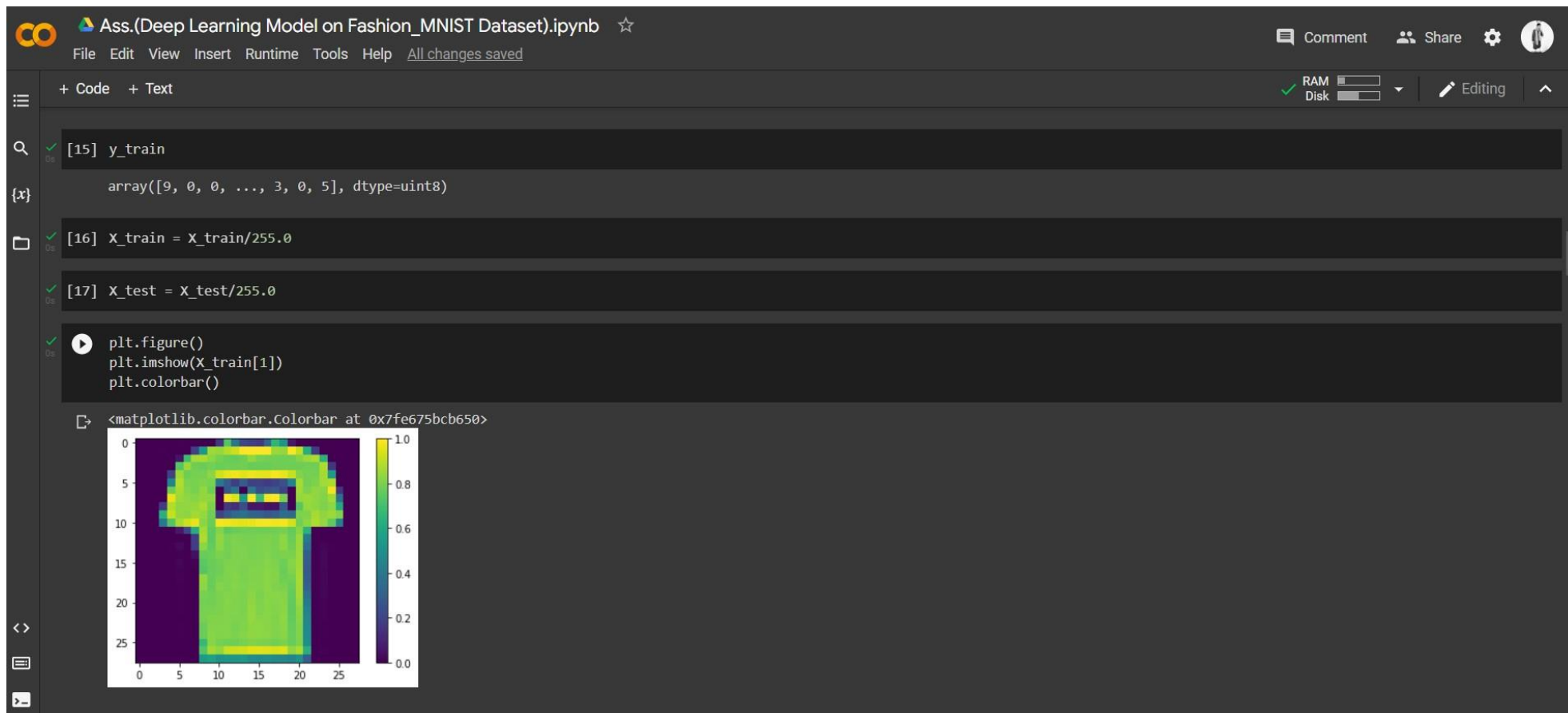
[11] 0s `class_names = ['top', 'trouser', 'pullover', 'dress', 'coat', 'sandal', 'shirt', 'sneaker', 'bag', 'ankle boot']`

▼ Data Exploration

[12] 0s `X_train.shape`

`(60000, 28, 28)`





Ass.(Deep Learning Model on Fashion_MNIST Dataset).ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk

Editing

Build the model with TF 2.0

[19] from tensorflow.keras import Sequential
from tensorflow.keras.layers import Flatten, Dense

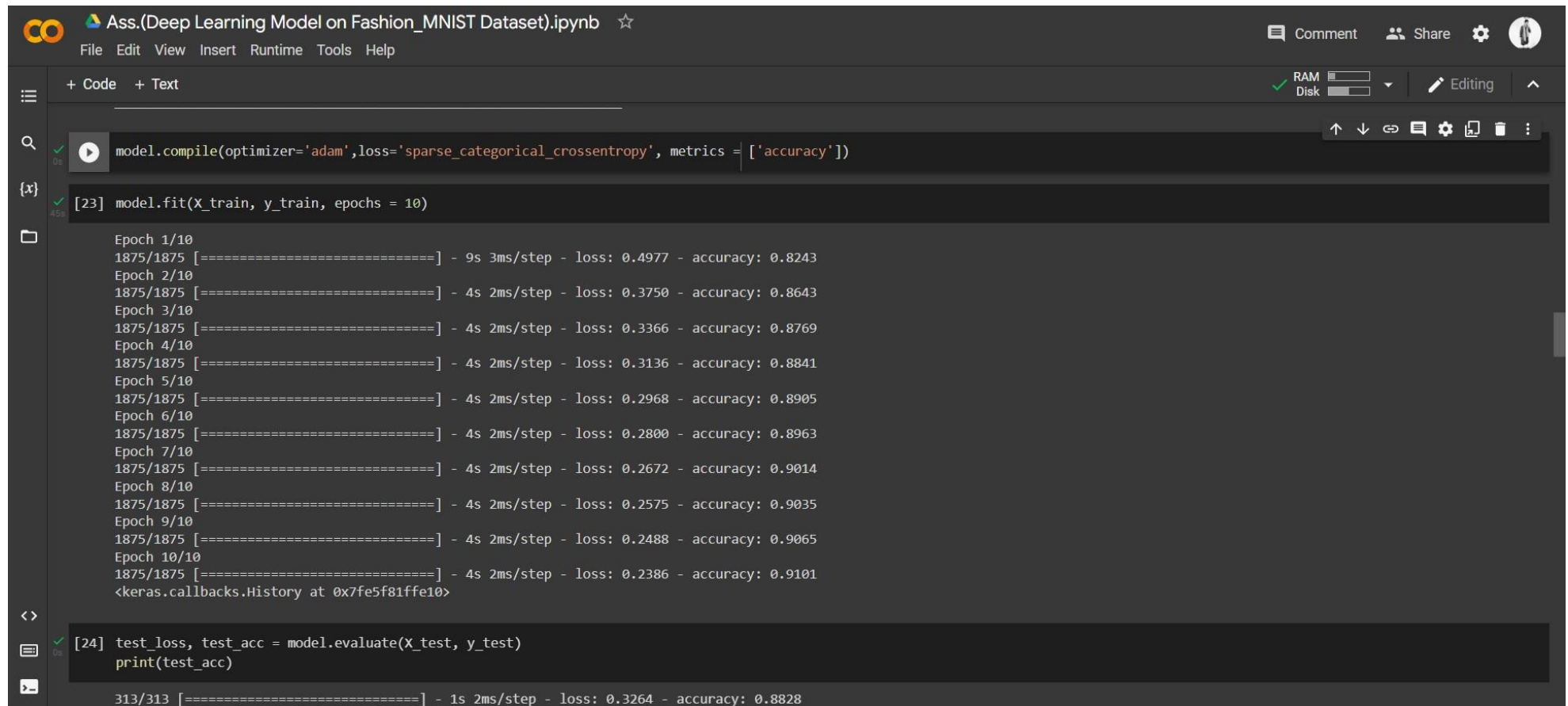
[20] model = Sequential()
model.add(Flatten(input_shape = (28, 28)))
model.add(Dense(128, activation = 'relu'))
model.add(Dense(10, activation = 'softmax'))

model.summary()

Model: "sequential"

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 784)	0
dense (Dense)	(None, 128)	100480
dense_1 (Dense)	(None, 10)	1290

Total params: 101,770
Trainable params: 101,770
Non-trainable params: 0



The screenshot displays a Jupyter Notebook titled "Ass.(Deep Learning Model on Fashion_MNIST Dataset).ipynb". The interface includes a top menu bar with options like File, Edit, View, Insert, Runtime, Tools, and Help. On the right, there are buttons for Comment, Share, and a user profile icon. Below the menu, a toolbar shows RAM and Disk usage, an Editing mode indicator, and navigation icons. The notebook contains two code cells. The first cell, labeled [23], defines a Keras model with the Adam optimizer and sparse categorical crossentropy loss, and compiles it with accuracy as a metric. The second cell, labeled [24], trains the model for 10 epochs using model.fit(X_train, y_train, epochs = 10). The output of the training process is displayed in the console, showing progress for each epoch from 1 to 10. Each epoch's output includes the number of samples processed (1875/1875), a progress bar, time per step, loss, and accuracy. The accuracy increases from 0.8243 in epoch 1 to 0.9101 in epoch 10. After the training loop, the test loss and accuracy are evaluated using model.evaluate(X_test, y_test), resulting in a test accuracy of 0.8828.

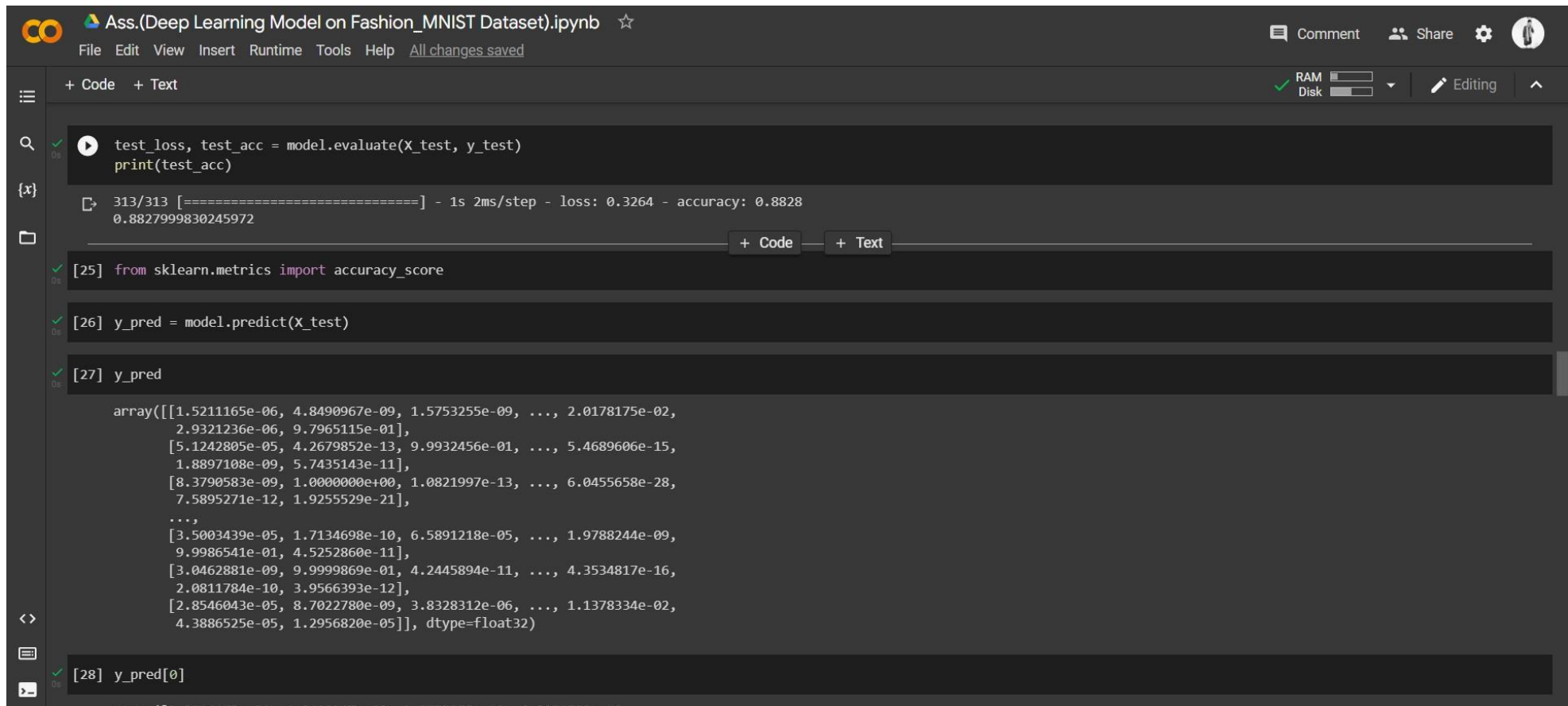
```
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy', metrics = ['accuracy'])
```

```
[23] model.fit(X_train, y_train, epochs = 10)
```

```
Epoch 1/10
1875/1875 [=====] - 9s 3ms/step - loss: 0.4977 - accuracy: 0.8243
Epoch 2/10
1875/1875 [=====] - 4s 2ms/step - loss: 0.3750 - accuracy: 0.8643
Epoch 3/10
1875/1875 [=====] - 4s 2ms/step - loss: 0.3366 - accuracy: 0.8769
Epoch 4/10
1875/1875 [=====] - 4s 2ms/step - loss: 0.3136 - accuracy: 0.8841
Epoch 5/10
1875/1875 [=====] - 4s 2ms/step - loss: 0.2968 - accuracy: 0.8905
Epoch 6/10
1875/1875 [=====] - 4s 2ms/step - loss: 0.2800 - accuracy: 0.8963
Epoch 7/10
1875/1875 [=====] - 4s 2ms/step - loss: 0.2672 - accuracy: 0.9014
Epoch 8/10
1875/1875 [=====] - 4s 2ms/step - loss: 0.2575 - accuracy: 0.9035
Epoch 9/10
1875/1875 [=====] - 4s 2ms/step - loss: 0.2488 - accuracy: 0.9065
Epoch 10/10
1875/1875 [=====] - 4s 2ms/step - loss: 0.2386 - accuracy: 0.9101
<keras.callbacks.History at 0x7fe5f81ffe10>
```

```
[24] test_loss, test_acc = model.evaluate(X_test, y_test)
print(test_acc)
```

```
313/313 [=====] - 1s 2ms/step - loss: 0.3264 - accuracy: 0.8828
```



The screenshot displays a Jupyter Notebook titled "Ass.(Deep Learning Model on Fashion_MNIST Dataset).ipynb". The interface includes a top menu bar with options like File, Edit, View, Insert, Runtime, Tools, and Help. Below the menu, there are tabs for "+ Code" and "+ Text". On the right side, there are icons for Comment, Share, and a user profile, along with RAM and Disk usage indicators.

The notebook contains several code cells:

- Cell 1: A code cell with the following code:

```
test_loss, test_acc = model.evaluate(X_test, y_test)
print(test_acc)
```
- Cell 2: A code cell showing the output of the previous cell:

```
313/313 [=====] - 1s 2ms/step - loss: 0.3264 - accuracy: 0.8828
0.8827999830245972
```
- Cell 3: A code cell with the following code:

```
[25] from sklearn.metrics import accuracy_score
```
- Cell 4: A code cell with the following code:

```
[26] y_pred = model.predict(X_test)
```
- Cell 5: A code cell with the following code:

```
[27] y_pred
```
- Cell 6: A code cell showing the output of the previous cell:




```
array([[1.5211165e-06, 4.8490967e-09, 1.5753255e-09, ..., 2.0178175e-02,
        2.9321236e-06, 9.7965115e-01],
       [5.1242805e-05, 4.2679852e-13, 9.9932456e-01, ..., 5.4689606e-15,
        1.8897108e-09, 5.7435143e-11],
       [8.3790583e-09, 1.0000000e+00, 1.0821997e-13, ..., 6.0455658e-28,
        7.5895271e-12, 1.9255529e-21],
       ...,
       [3.5003439e-05, 1.7134698e-10, 6.5891218e-05, ..., 1.9788244e-09,
        9.9986541e-01, 4.5252860e-11],
       [3.0462881e-09, 9.9999869e-01, 4.2445894e-11, ..., 4.3534817e-16,
        2.0811784e-10, 3.9566393e-12],
       [2.8546043e-05, 8.7022780e-09, 3.8328312e-06, ..., 1.1378334e-02,
        4.3886525e-05, 1.2956820e-05]], dtype=float32)
```
- Cell 7: A code cell with the following code:

```
[28] y_pred[0]
```

Ass.(Deep Learning Model on Fashion_MNIST Dataset).ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM  Disk  Editing 

```
[28] y_pred[0]
array([1.5211165e-06, 4.8490967e-09, 1.5753255e-09, 6.3889520e-09,
       1.0603057e-09, 1.6549793e-04, 6.7691428e-07, 2.0178175e-02,
       2.9321236e-06, 9.7965115e-01], dtype=float32)
```

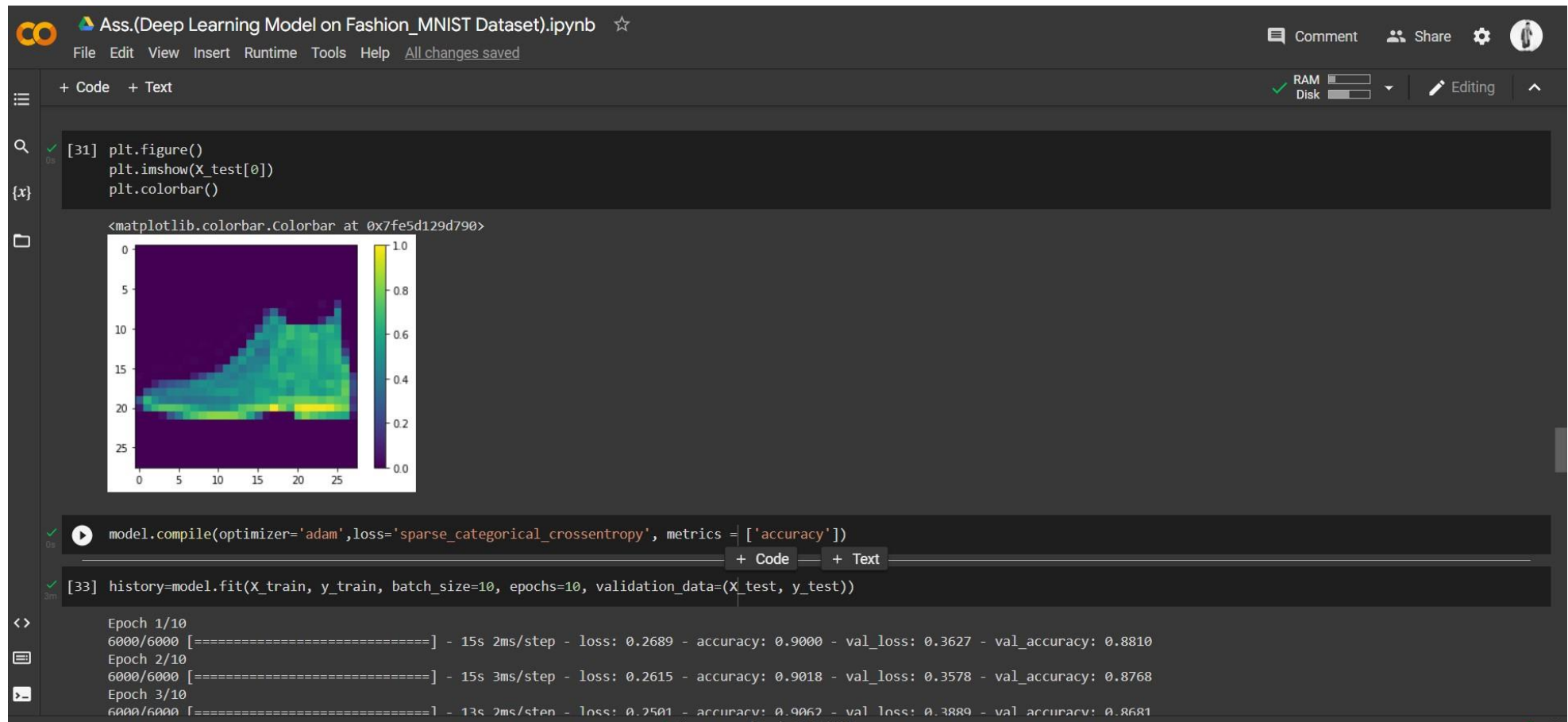
```
class_names
['top',
 'trouser',
 'pullover',
 'dress',
 'coat',
 'sandal',
 'shirt',
 'sneaker',
 'bag',
 'ankle boot']
```


```
[30] np.argmax(y_pred[0])
9
```

```
[31] plt.figure()
      plt.imshow(X_test[0])
      plt.colorbar()
```

<matplotlib.colorbar.Colorbar at 0x7fe5d129d790>

0 10



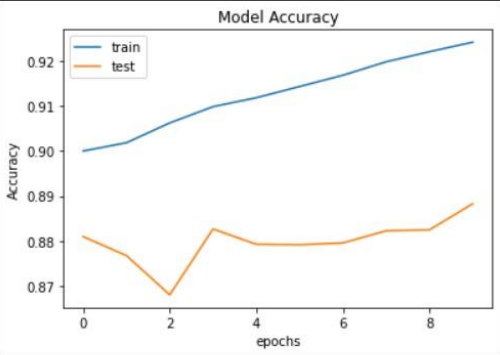
 Ass.(Deep Learning Model on Fashion_MNIST Dataset).ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

+ Code + Text

✓ 3m [33] 6000/6000 [=====] - 15s 2ms/step - loss: 0.2140 - accuracy: 0.9198 - val_loss: 0.3576 - val_accuracy: 0.8823
Epoch 9/10
6000/6000 [=====] - 15s 2ms/step - loss: 0.2107 - accuracy: 0.9220 - val_loss: 0.3631 - val_accuracy: 0.8825
Epoch 10/10
6000/6000 [=====] - 15s 2ms/step - loss: 0.2060 - accuracy: 0.9241 - val_loss: 0.3562 - val_accuracy: 0.8883


✓ 0s ▶ plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('epochs')
plt.legend(['train', 'test'], loc='upper left')
plt.show()



The plot shows training and validation accuracy over 10 epochs. Training accuracy (blue line) starts at approximately 0.90 and increases steadily to about 0.925. Validation accuracy (orange line) starts at approximately 0.88, dips to 0.87 at epoch 2, and then fluctuates between 0.88 and 0.89 for the remainder of the training process.



Epoch	train	test
0	0.900	0.882
1	0.902	0.878
2	0.905	0.870
3	0.910	0.883
4	0.912	0.880
5	0.915	0.880
6	0.918	0.880
7	0.920	0.883
8	0.922	0.883
9	0.924	0.885
10	0.925	0.888

✓ [35] plt.plot(history.history['loss'])

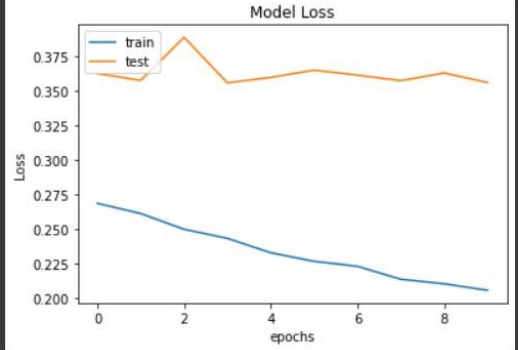
 Ass.(Deep Learning Model on Fashion_MNIST Dataset).ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

+ Code + Text

RAM  Disk  Editing

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('epochs')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

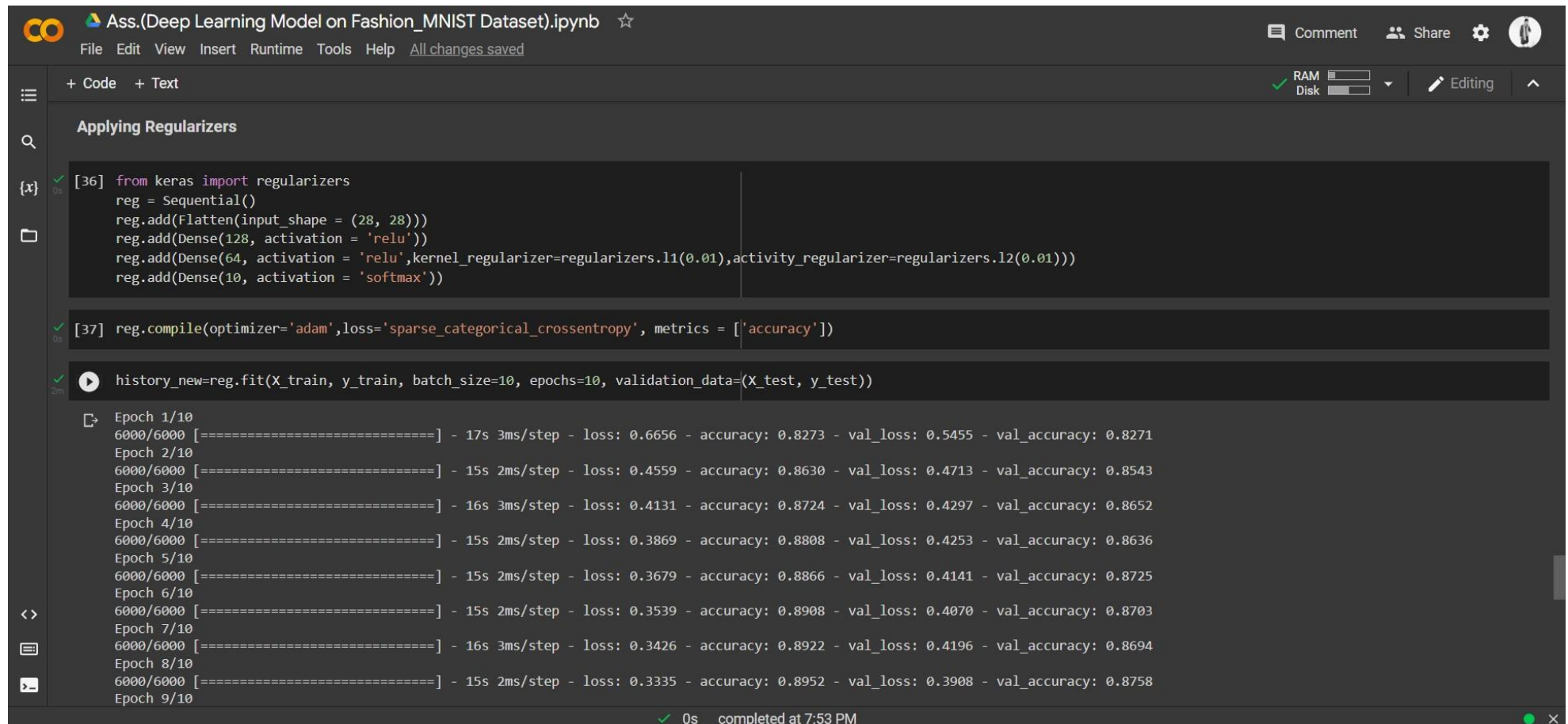


Epoch	Train Loss	Test Loss
0	0.270	0.360
1	0.260	0.355
2	0.250	0.390
3	0.240	0.355
4	0.230	0.360
5	0.225	0.365
6	0.220	0.360
7	0.215	0.355
8	0.210	0.365
9	0.205	0.360

Applying Regularizers

```
[36] from keras import regularizers
      reg = Sequential()
      reg.add(Flatten(input_shape = (28, 28)))
      reg.add(Dense(128, activation = 'relu'))
```

completed at 7:53 PM



The screenshot shows a Jupyter Notebook titled "Ass.(Deep Learning Model on Fashion_MNIST Dataset).ipynb". The interface includes a top menu bar with options like File, Edit, View, Insert, Runtime, Tools, and Help. Below the menu, there's a toolbar with icons for code, text, and other functions. The main area displays three code cells. The first cell defines a Keras Sequential model with a Flatten layer, two Dense layers with ReLU activation and L2 regularization, and a final Dense layer with Softmax activation. The second cell compiles the model using the Adam optimizer and sparse categorical crossentropy loss. The third cell fits the model for 10 epochs. The output of the third cell shows the training progress for each epoch, including loss, accuracy, and validation metrics.

```
[36] from keras import regularizers
reg = Sequential()
reg.add(Flatten(input_shape = (28, 28)))
reg.add(Dense(128, activation = 'relu'))
reg.add(Dense(64, activation = 'relu', kernel_regularizer=regularizers.l1(0.01), activity_regularizer=regularizers.l2(0.01)))
reg.add(Dense(10, activation = 'softmax'))

[37] reg.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics = ['accuracy'])

history_new=reg.fit(X_train, y_train, batch_size=10, epochs=10, validation_data=(X_test, y_test))

Epoch 1/10
6000/6000 [=====] - 17s 3ms/step - loss: 0.6656 - accuracy: 0.8273 - val_loss: 0.5455 - val_accuracy: 0.8271
Epoch 2/10
6000/6000 [=====] - 15s 2ms/step - loss: 0.4559 - accuracy: 0.8630 - val_loss: 0.4713 - val_accuracy: 0.8543
Epoch 3/10
6000/6000 [=====] - 16s 3ms/step - loss: 0.4131 - accuracy: 0.8724 - val_loss: 0.4297 - val_accuracy: 0.8652
Epoch 4/10
6000/6000 [=====] - 15s 2ms/step - loss: 0.3869 - accuracy: 0.8808 - val_loss: 0.4253 - val_accuracy: 0.8636
Epoch 5/10
6000/6000 [=====] - 15s 2ms/step - loss: 0.3679 - accuracy: 0.8866 - val_loss: 0.4141 - val_accuracy: 0.8725
Epoch 6/10
6000/6000 [=====] - 15s 2ms/step - loss: 0.3539 - accuracy: 0.8908 - val_loss: 0.4070 - val_accuracy: 0.8703
Epoch 7/10
6000/6000 [=====] - 16s 3ms/step - loss: 0.3426 - accuracy: 0.8922 - val_loss: 0.4196 - val_accuracy: 0.8694
Epoch 8/10
6000/6000 [=====] - 15s 2ms/step - loss: 0.3335 - accuracy: 0.8952 - val_loss: 0.3908 - val_accuracy: 0.8758
Epoch 9/10
```

completed at 7:53 PM

