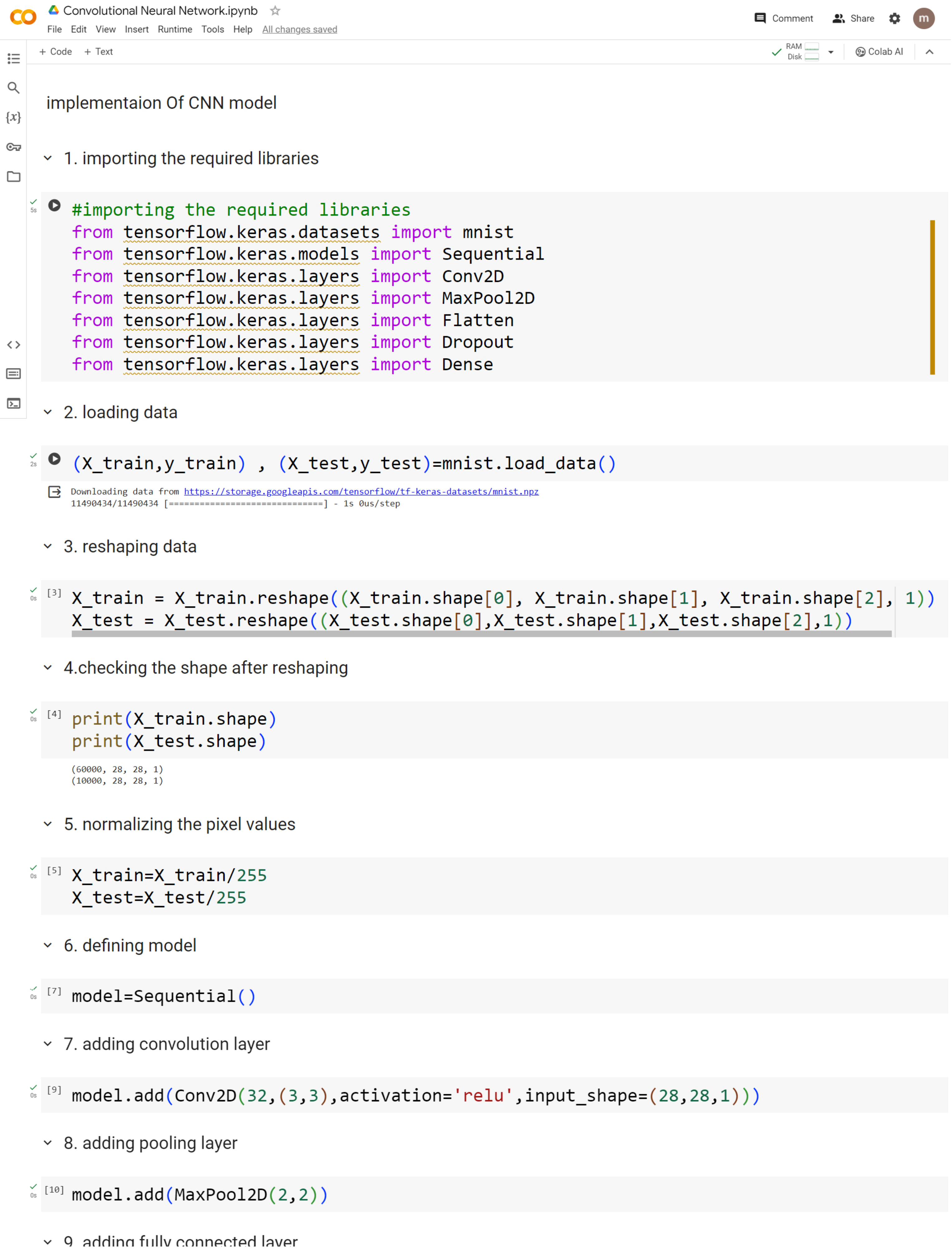

Convolutional Neural Network.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
m
+ Code + Text
RAM
Disk
Colab AI

implementaion Of CNN model

1. importing the required libraries

```
#importing the required libraries
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPool2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Dense
```

2. loading data

```
(X_train,y_train) , (X_test,y_test)=mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 1s 0us/step
```

3. reshaping data

```
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], X_train.shape[2], 1))
X_test = X_test.reshape((X_test.shape[0],X_test.shape[1],X_test.shape[2],1))
```

4.checking the shape after reshaping

```
print(X_train.shape)
print(X_test.shape)
```

```
(60000, 28, 28, 1)
(10000, 28, 28, 1)
```

5. normalizing the pixel values

```
X_train=X_train/255
X_test=X_test/255
```

6. defining model

```
model=Sequential()
```

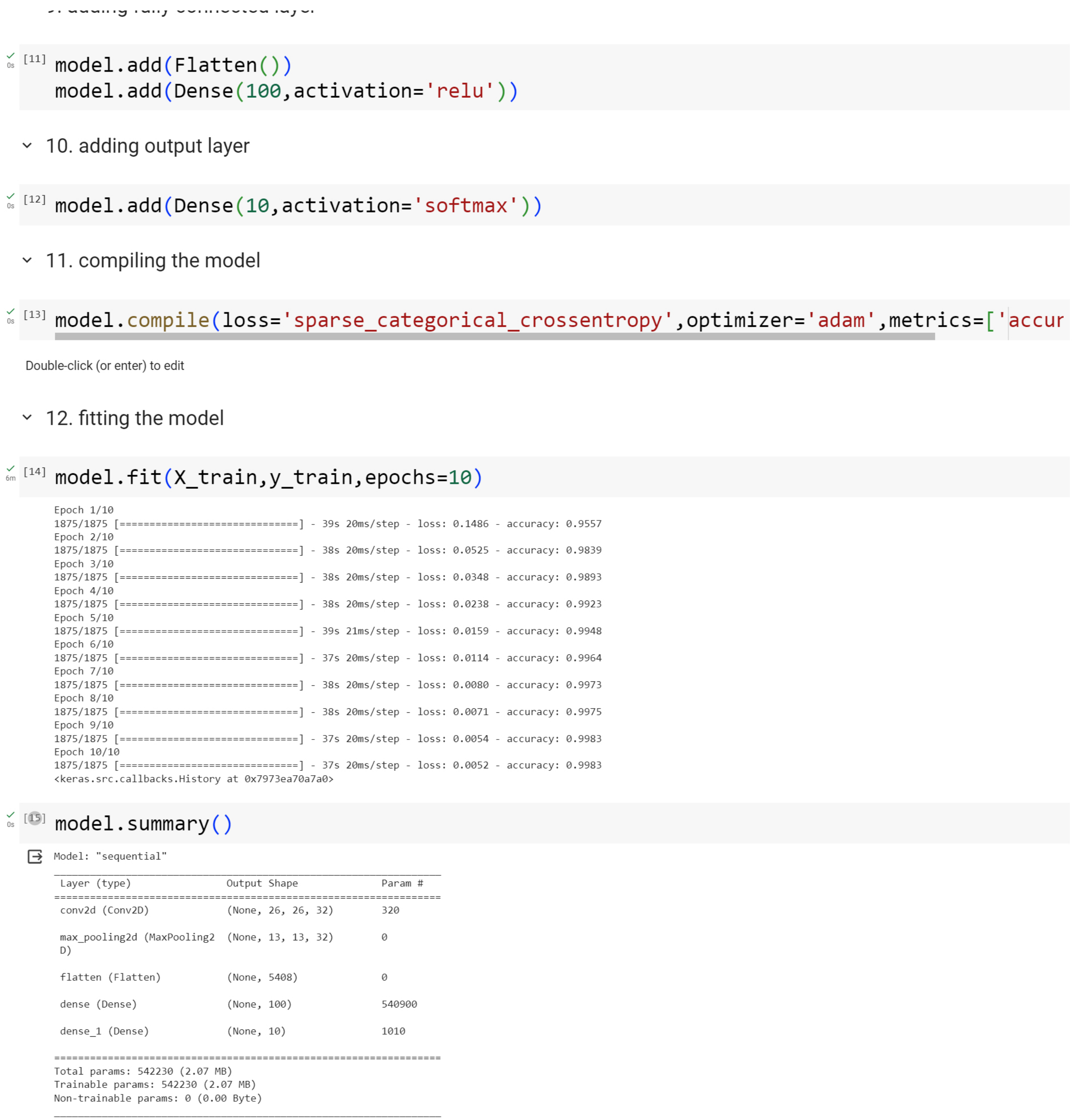
7. adding convolution layer

```
model.add(Conv2D(32,(3,3),activation='relu',input_shape=(28,28,1)))
```

8. adding pooling layer

```
model.add(MaxPool2D(2,2))
```

9. adding fully connected layer



9. adding fully connected layer

```
[11] model.add(Flatten())
     model.add(Dense(100,activation='relu'))
```

10. adding output layer

```
[12] model.add(Dense(10,activation='softmax'))
```

11. compiling the model

```
[13] model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['accur
```

Double-click (or enter) to edit

12. fitting the model

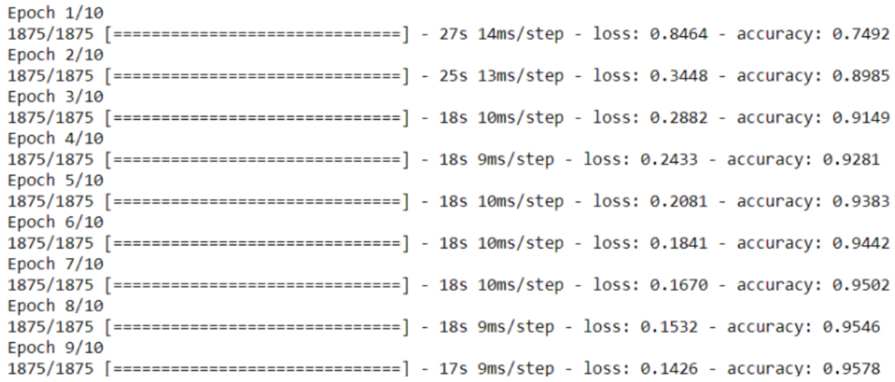
```
[14] model.fit(X_train,y_train,epochs=10)
```

```
Epoch 1/10
1875/1875 [==============================] - 39s 20ms/step - loss: 0.1486 - accuracy: 0.9557
Epoch 2/10
1875/1875 [==============================] - 38s 20ms/step - loss: 0.0525 - accuracy: 0.9839
Epoch 3/10
1875/1875 [==============================] - 38s 20ms/step - loss: 0.0348 - accuracy: 0.9893
Epoch 4/10
1875/1875 [==============================] - 38s 20ms/step - loss: 0.0238 - accuracy: 0.9923
Epoch 5/10
1875/1875 [==============================] - 39s 21ms/step - loss: 0.0159 - accuracy: 0.9948
Epoch 6/10
1875/1875 [==============================] - 37s 20ms/step - loss: 0.0114 - accuracy: 0.9964
Epoch 7/10
1875/1875 [==============================] - 38s 20ms/step - loss: 0.0080 - accuracy: 0.9973
Epoch 8/10
1875/1875 [==============================] - 38s 20ms/step - loss: 0.0071 - accuracy: 0.9975
Epoch 9/10
1875/1875 [==============================] - 37s 20ms/step - loss: 0.0054 - accuracy: 0.9983
Epoch 10/10
1875/1875 [==============================] - 37s 20ms/step - loss: 0.0052 - accuracy: 0.9983
<keras.src.callbacks.History at 0x7973ea70a7a0>
```

```
[15] model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2  (None, 13, 13, 32)        0
 D)

 flatten (Flatten)           (None, 5408)              0

 dense (Dense)               (None, 100)               540900

 dense_1 (Dense)             (None, 10)                1010

=================================================================
Total params: 542230 (2.07 MB)
Trainable params: 542230 (2.07 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Output:



```
Epoch 1/10
1875/1875 [==============================] - 27s 14ms/step - loss: 0.8464 - accuracy: 0.7492
Epoch 2/10
1875/1875 [==============================] - 25s 13ms/step - loss: 0.3448 - accuracy: 0.8985
Epoch 3/10
1875/1875 [==============================] - 18s 10ms/step - loss: 0.2882 - accuracy: 0.9149
Epoch 4/10
1875/1875 [==============================] - 18s 9ms/step - loss: 0.2433 - accuracy: 0.9281
Epoch 5/10
1875/1875 [==============================] - 18s 10ms/step - loss: 0.2081 - accuracy: 0.9383
Epoch 6/10
1875/1875 [==============================] - 18s 10ms/step - loss: 0.1841 - accuracy: 0.9442
Epoch 7/10
1875/1875 [==============================] - 18s 10ms/step - loss: 0.1670 - accuracy: 0.9502
Epoch 8/10
1875/1875 [==============================] - 18s 9ms/step - loss: 0.1532 - accuracy: 0.9546
Epoch 9/10
1875/1875 [==============================] - 17s 9ms/step - loss: 0.1426 - accuracy: 0.9578
```

```
Epoch 10/10
1875/1875 [==============================] - 18s 10ms/step - loss: 0.1329 - accuracy: 0.9600
```

13. evaluting the model

```
model.evaluate(X_test,y_test)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.1399 - accuracy: 0.9587

[0.13990521430969238, 0.9587000012397766]
```

Q.1 loss function ="categorical_crossentropy"

optimizer ="stocastic gradient descent (sgd)"

metric = "accuracy"