Name: Thoomu Cyril Chowdary

Semo ID: S02059332

Semo email: cthoomu1s@semo.edu

CS505-01 Data Mining CRN-12050

Modify Parameters to The Models withe Neural Networks

Pease submit the screen dumps

Showing the results

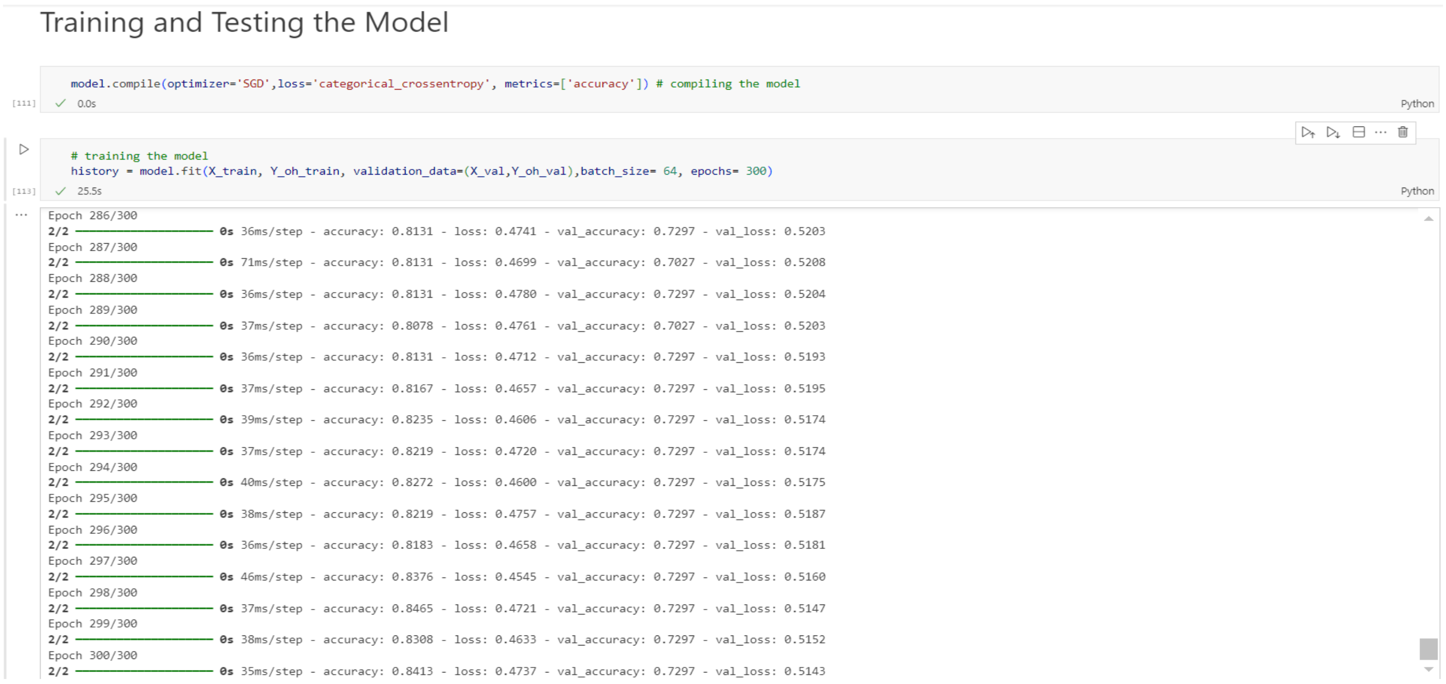
Make a Copy of the file:

tutorial05_different_approaches_to_define_neural_networks_keras.ipynb

1.- Do the following modifications:

model.compile(optimizer='SGD',loss='categorical_crossentropy', metrics=['accuracy']) # compiling the model

Train model and execute loss = model.evaluate(X_test, Y_oh_test, verbose=0) print screen dumps



Training and Testing the Model

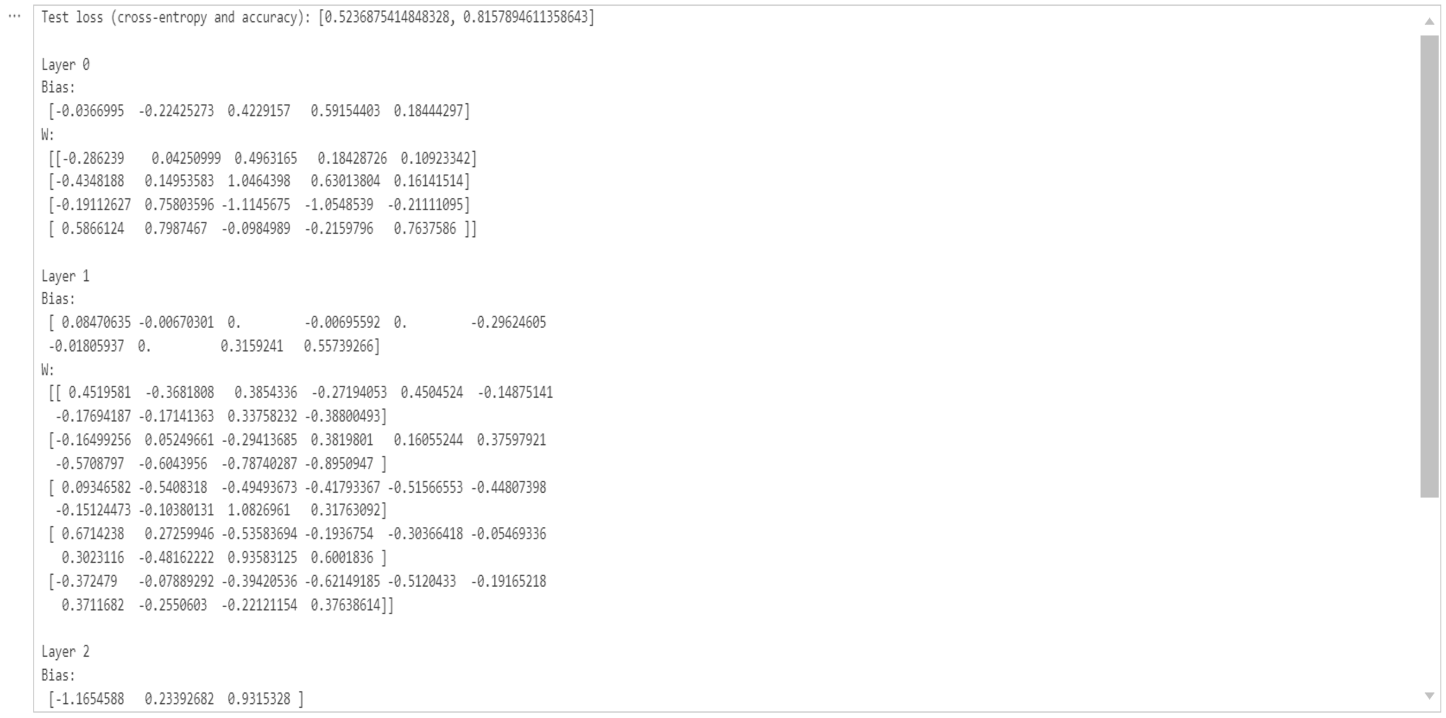
```
model.compile(optimizer='SGD',loss='categorical_crossentropy', metrics=['accuracy']) # compiling the model
```

[111] ✓ 0.0s

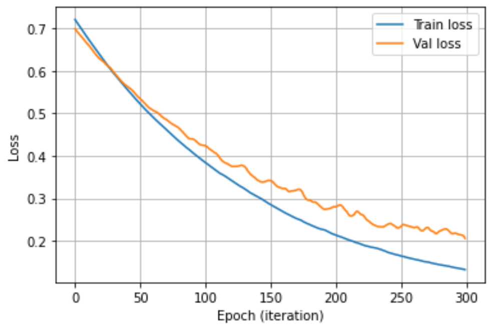
```
# training the model
history = model.fit(X_train, Y_oh_train, validation_data=(X_val,Y_oh_val),batch_size= 64, epochs= 300)
```

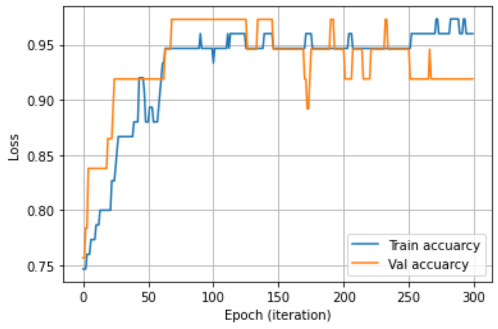
[113] ✓ 25.5s

```
Epoch 286/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - accuracy: 0.8131 - loss: 0.4741 - val_accuracy: 0.7297 - val_loss: 0.5203
Epoch 287/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 71ms/step - accuracy: 0.8131 - loss: 0.4699 - val_accuracy: 0.7027 - val_loss: 0.5208
Epoch 288/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - accuracy: 0.8131 - loss: 0.4780 - val_accuracy: 0.7297 - val_loss: 0.5204
Epoch 289/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - accuracy: 0.8078 - loss: 0.4761 - val_accuracy: 0.7027 - val_loss: 0.5203
Epoch 290/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - accuracy: 0.8131 - loss: 0.4712 - val_accuracy: 0.7297 - val_loss: 0.5193
Epoch 291/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - accuracy: 0.8167 - loss: 0.4657 - val_accuracy: 0.7297 - val_loss: 0.5195
Epoch 292/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step - accuracy: 0.8235 - loss: 0.4606 - val_accuracy: 0.7297 - val_loss: 0.5174
Epoch 293/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - accuracy: 0.8219 - loss: 0.4720 - val_accuracy: 0.7297 - val_loss: 0.5174
Epoch 294/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step - accuracy: 0.8272 - loss: 0.4600 - val_accuracy: 0.7297 - val_loss: 0.5175
Epoch 295/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - accuracy: 0.8219 - loss: 0.4757 - val_accuracy: 0.7297 - val_loss: 0.5187
Epoch 296/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - accuracy: 0.8183 - loss: 0.4658 - val_accuracy: 0.7297 - val_loss: 0.5181
Epoch 297/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step - accuracy: 0.8376 - loss: 0.4545 - val_accuracy: 0.7297 - val_loss: 0.5160
Epoch 298/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - accuracy: 0.8465 - loss: 0.4721 - val_accuracy: 0.7297 - val_loss: 0.5147
Epoch 299/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - accuracy: 0.8308 - loss: 0.4633 - val_accuracy: 0.7297 - val_loss: 0.5152
Epoch 300/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - accuracy: 0.8413 - loss: 0.4737 - val_accuracy: 0.7297 - val_loss: 0.5143
```



```
Test loss (cross-entropy and accuracy): [0.5236875414848328, 0.8157894611358643]

Layer 0
Bias:
 [-0.0366995  -0.22425273  0.4229157   0.59154403  0.18444297]
W:
 [[-0.286239    0.04250999  0.4963165   0.18428726  0.10923342]
 [-0.4348188   0.14953583  1.0464398   0.63013804  0.16141514]
 [-0.19112627  0.75803596 -1.1145675  -1.0548539  -0.21111095]
 [ 0.5866124   0.7987467  -0.0984989  -0.2159796   0.7637586 ]]

Layer 1
Bias:
 [ 0.08470635 -0.00670301  0.         -0.00695592  0.         -0.29624605
 -0.01805937  0.          0.3159241   0.55739266]
W:
 [[ 0.4519581  -0.3681808   0.3854336  -0.27194053  0.4504524  -0.14875141
  -0.17694187 -0.17141363  0.33758232 -0.38800493]
 [-0.16499256  0.05249661 -0.29413685  0.3819801   0.16055244  0.37597921
  -0.5708797  -0.6043956  -0.78740287 -0.8950947 ]
 [ 0.09346582 -0.5408318  -0.49493673 -0.41793367 -0.51566553 -0.44807398
  -0.15124473 -0.10380131  1.0826961   0.31763092]
 [ 0.6714238   0.27259946 -0.53583694 -0.1936754  -0.30366418 -0.05469336
   0.3023116  -0.48162222  0.93583125  0.6001836 ]
 [-0.372479   -0.07889292 -0.39420536 -0.62149185 -0.5120433  -0.19165218
   0.3711682  -0.2550603  -0.22121154  0.37638614]]

Layer 2
Bias:
 [-1.1654588   0.23392682  0.9315328 ]
```





model.compile(optimizer='RMSprop',loss='categorical_crossentropy', metrics=['accuracy']) # compiling the model

Train model and execute loss = model.evaluate(X_test, Y_oh_test, verbose=0) print screen dumps

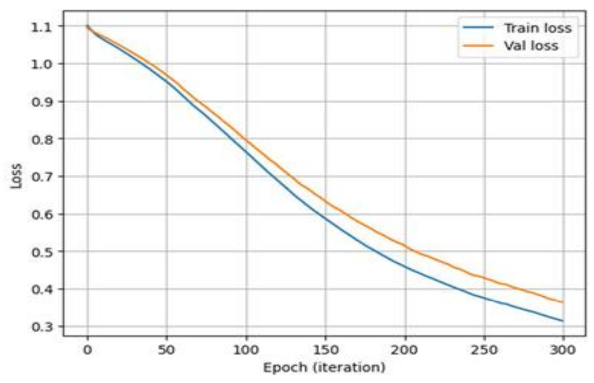
Training and Testing the Model

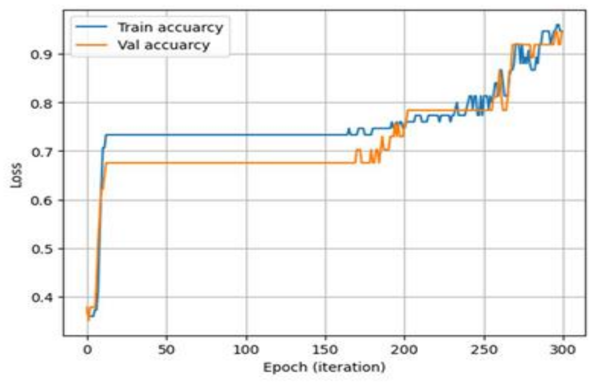
```python
model.compile(optimizer='RMSprop',loss='categorical_crossentropy', metrics=['accuracy']) # compiling the model
```
[116] ✓ 0.0s Python

```python
# training the model
history = model.fit(X_train, Y_oh_train, validation_data=(X_val,Y_oh_val),batch_size= 64, epochs= 300)
```
[117] ✓ 26.4s Python

```
Epoch 286/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - accuracy: 1.0000 - loss: 0.2416 - val_accuracy: 0.8919 - val_loss: 0.2930
Epoch 287/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - accuracy: 1.0000 - loss: 0.2416 - val_accuracy: 0.8919 - val_loss: 0.2950
Epoch 288/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - accuracy: 1.0000 - loss: 0.2439 - val_accuracy: 0.8919 - val_loss: 0.2978
Epoch 289/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 1.0000 - loss: 0.2377 - val_accuracy: 0.8919 - val_loss: 0.2964
Epoch 290/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - accuracy: 1.0000 - loss: 0.2414 - val_accuracy: 0.8919 - val_loss: 0.2880
Epoch 291/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 1.0000 - loss: 0.2355 - val_accuracy: 0.8919 - val_loss: 0.2906
Epoch 292/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 1.0000 - loss: 0.2396 - val_accuracy: 0.9189 - val_loss: 0.2828
Epoch 293/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - accuracy: 1.0000 - loss: 0.2392 - val_accuracy: 0.9189 - val_loss: 0.2822
Epoch 294/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 1.0000 - loss: 0.2338 - val_accuracy: 0.8919 - val_loss: 0.2859
Epoch 295/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - accuracy: 1.0000 - loss: 0.2341 - val_accuracy: 0.8919 - val_loss: 0.2883
Epoch 296/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - accuracy: 1.0000 - loss: 0.2347 - val_accuracy: 0.9189 - val_loss: 0.2815
Epoch 297/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - accuracy: 1.0000 - loss: 0.2286 - val_accuracy: 0.9189 - val_loss: 0.2749
Epoch 298/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - accuracy: 1.0000 - loss: 0.2331 - val_accuracy: 0.9189 - val_loss: 0.2726
Epoch 299/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - accuracy: 1.0000 - loss: 0.2332 - val_accuracy: 0.8919 - val_loss: 0.2811
Epoch 300/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - accuracy: 1.0000 - loss: 0.2280 - val_accuracy: 0.8919 - val_loss: 0.2878
```

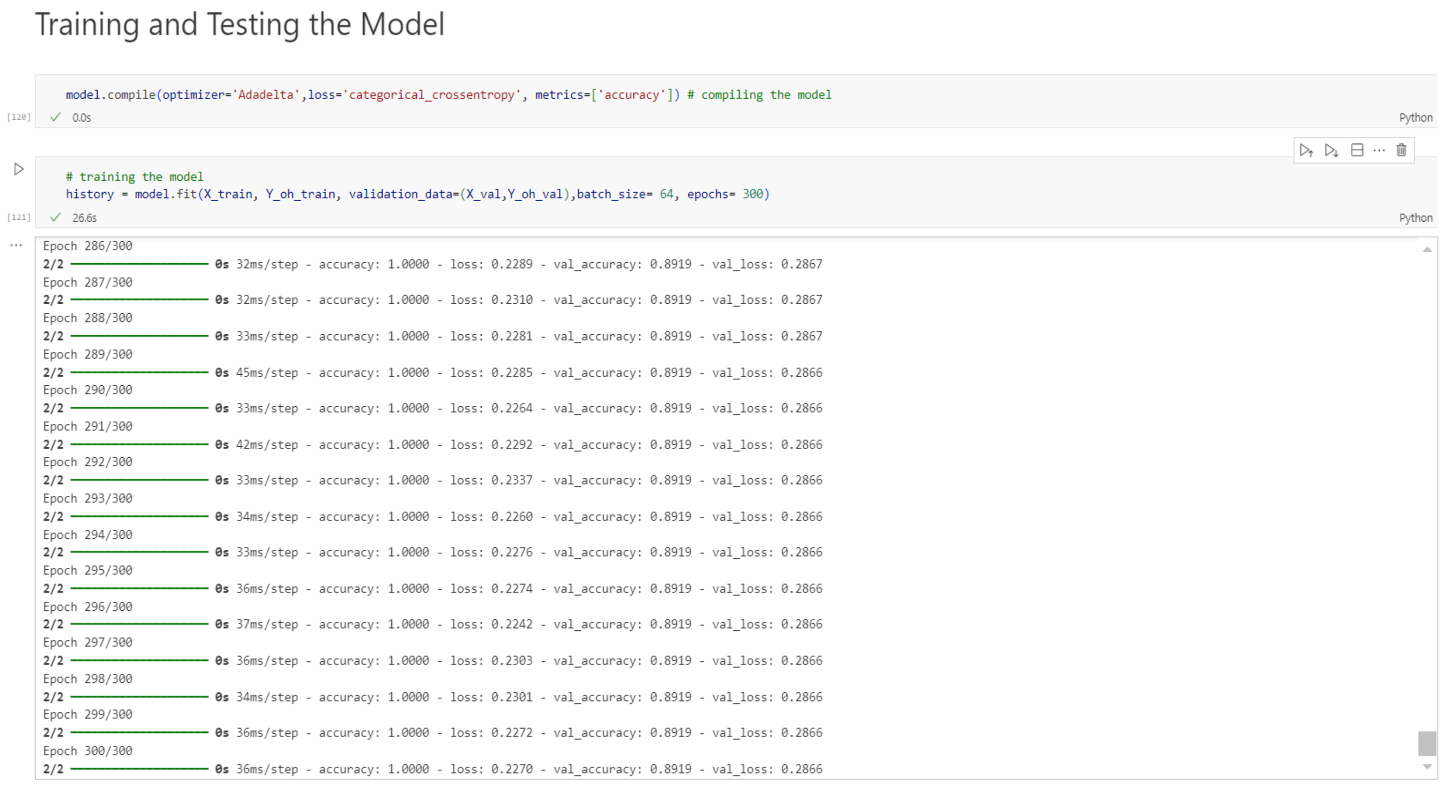
```
Test loss (cross-entropy and accuracy): [0.25814089179039, 1.0]

Layer 0
Bias:
 [-0.13498312 -0.35740823  0.6129222   0.81119657  0.36762995]
W:
 [[-0.14200217  0.13711931  0.38353345  0.10588858  0.16255262]
 [-0.2939697   0.23388055  1.0977377   0.70702153 -0.01464017]
 [-0.06110045  0.8323914  -1.2042543  -1.1110417  -0.13144428]
 [ 0.77725685  0.94617397 -0.24966045 -0.35755625  0.74486965]]

Layer 1
Bias:
 [ 0.03121604 -0.00670301  0.         -0.00879402  0.         -0.31457743
 -0.10699973  0.          0.47994897  0.7501659 ]
W:
 [[ 0.20713966 -0.3681808   0.3854336  -0.27194053  0.4504524  -0.14875141
  -0.17694187 -0.17141363  0.33705753 -0.74605685]
 [-0.5307094   0.05249661 -0.29413685  0.37977687  0.16055244  0.35762557
  -0.5780685  -0.6043956  -0.9208475  -1.0565699 ]
 [ 0.53140587 -0.5408318  -0.49493673 -0.41793367 -0.51566553 -0.44807398
  -0.23979351 -0.10380131  1.3729163   0.6028423 ]
 [ 1.086525    0.27259946 -0.53583694 -0.1936754  -0.30366418 -0.05469336
   0.21388255 -0.48162222  1.4059473   1.0973687 ]
 [-0.53980386 -0.07889292 -0.39420536 -0.6230955  -0.5120433  -0.21000785
   0.29474616 -0.2550603  -0.21372405  0.36679882]]

Layer 2
Bias:
 [-1.7486603   0.16195932  1.4106075 ]
```





model.compile(optimizer='Adadelta',loss='categorical_crossentropy', metrics=['accuracy']) # compiling the model

Train model and execute loss = model.evaluate(X_test, Y_oh_test, verbose=0) print screen dumps



Training and Testing the Model

```
model.compile(optimizer='Adadelta',loss='categorical_crossentropy', metrics=['accuracy']) # compiling the model
```

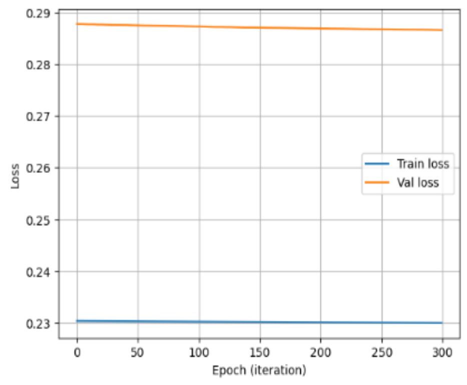
[120] ✓ 0.0s Python

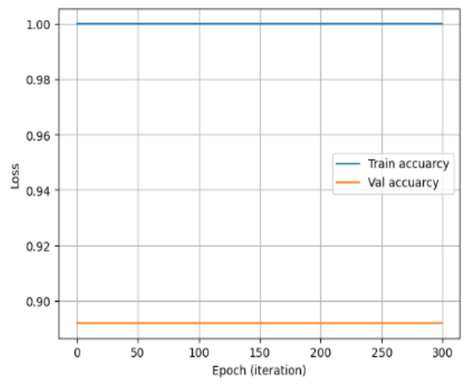
```
# training the model
history = model.fit(X_train, Y_oh_train, validation_data=(X_val,Y_oh_val),batch_size= 64, epochs= 300)
```

[121] ✓ 26.6s Python

```
Epoch 286/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 1.0000 - loss: 0.2289 - val_accuracy: 0.8919 - val_loss: 0.2867
Epoch 287/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 1.0000 - loss: 0.2310 - val_accuracy: 0.8919 - val_loss: 0.2867
Epoch 288/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 1.0000 - loss: 0.2281 - val_accuracy: 0.8919 - val_loss: 0.2867
Epoch 289/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - accuracy: 1.0000 - loss: 0.2285 - val_accuracy: 0.8919 - val_loss: 0.2866
Epoch 290/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 1.0000 - loss: 0.2264 - val_accuracy: 0.8919 - val_loss: 0.2866
Epoch 291/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step - accuracy: 1.0000 - loss: 0.2292 - val_accuracy: 0.8919 - val_loss: 0.2866
Epoch 292/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 1.0000 - loss: 0.2337 - val_accuracy: 0.8919 - val_loss: 0.2866
Epoch 293/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 1.0000 - loss: 0.2260 - val_accuracy: 0.8919 - val_loss: 0.2866
Epoch 294/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 1.0000 - loss: 0.2276 - val_accuracy: 0.8919 - val_loss: 0.2866
Epoch 295/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - accuracy: 1.0000 - loss: 0.2274 - val_accuracy: 0.8919 - val_loss: 0.2866
Epoch 296/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - accuracy: 1.0000 - loss: 0.2242 - val_accuracy: 0.8919 - val_loss: 0.2866
Epoch 297/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - accuracy: 1.0000 - loss: 0.2303 - val_accuracy: 0.8919 - val_loss: 0.2866
Epoch 298/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 1.0000 - loss: 0.2301 - val_accuracy: 0.8919 - val_loss: 0.2866
Epoch 299/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - accuracy: 1.0000 - loss: 0.2272 - val_accuracy: 0.8919 - val_loss: 0.2866
Epoch 300/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - accuracy: 1.0000 - loss: 0.2270 - val_accuracy: 0.8919 - val_loss: 0.2866
```

```
Test loss (cross-entropy and accuracy): [0.2577400505542755, 1.0]

Layer 0
Bias:
 [-0.13543561 -0.35772446  0.6132437   0.8115504   0.36791676]
W:
 [[-0.14224991  0.13696897  0.38362917  0.10602062  0.16289961]
 [-0.2942179   0.2337426   1.0979195   0.7072452  -0.01466968]
 [-0.06137059  0.83223677 -1.2041472  -1.1109128  -0.13108042]
 [ 0.77709633  0.94608325 -0.24961124 -0.35749102  0.7451701 ]]

Layer 1
Bias:
 [ 0.03126279 -0.00670301  0.         -0.00879402  0.         -0.31457743
 -0.10699973  0.          0.4801719   0.7505655 ]
W:
 [[ 0.20699993 -0.3681808   0.3854336  -0.27194053  0.4504524  -0.14875141
  -0.17694187 -0.17141363  0.33697477 -0.7463477 ]
 [-0.53096145  0.05249661 -0.29413685  0.37977687  0.16055244  0.35762557
  -0.5780685  -0.6043956  -0.92086846 -1.0564437 ]
 [ 0.5320796  -0.5408318  -0.49493673 -0.41793367 -0.51566553 -0.44807398
  -0.23979351 -0.10380131  1.3733208   0.6033168 ]
 [ 1.0871395   0.27259946 -0.53583694 -0.1936754  -0.30366418 -0.05469336
   0.21388255 -0.48162222  1.4066122   1.0980369 ]
 [-0.53987586 -0.07889292 -0.39420536 -0.6230955  -0.5120433  -0.21000785
   0.29474616 -0.2550603  -0.2136348   0.36708045]]

Layer 2
Bias:
 [-1.7496338  0.1616593  1.4113752]
```

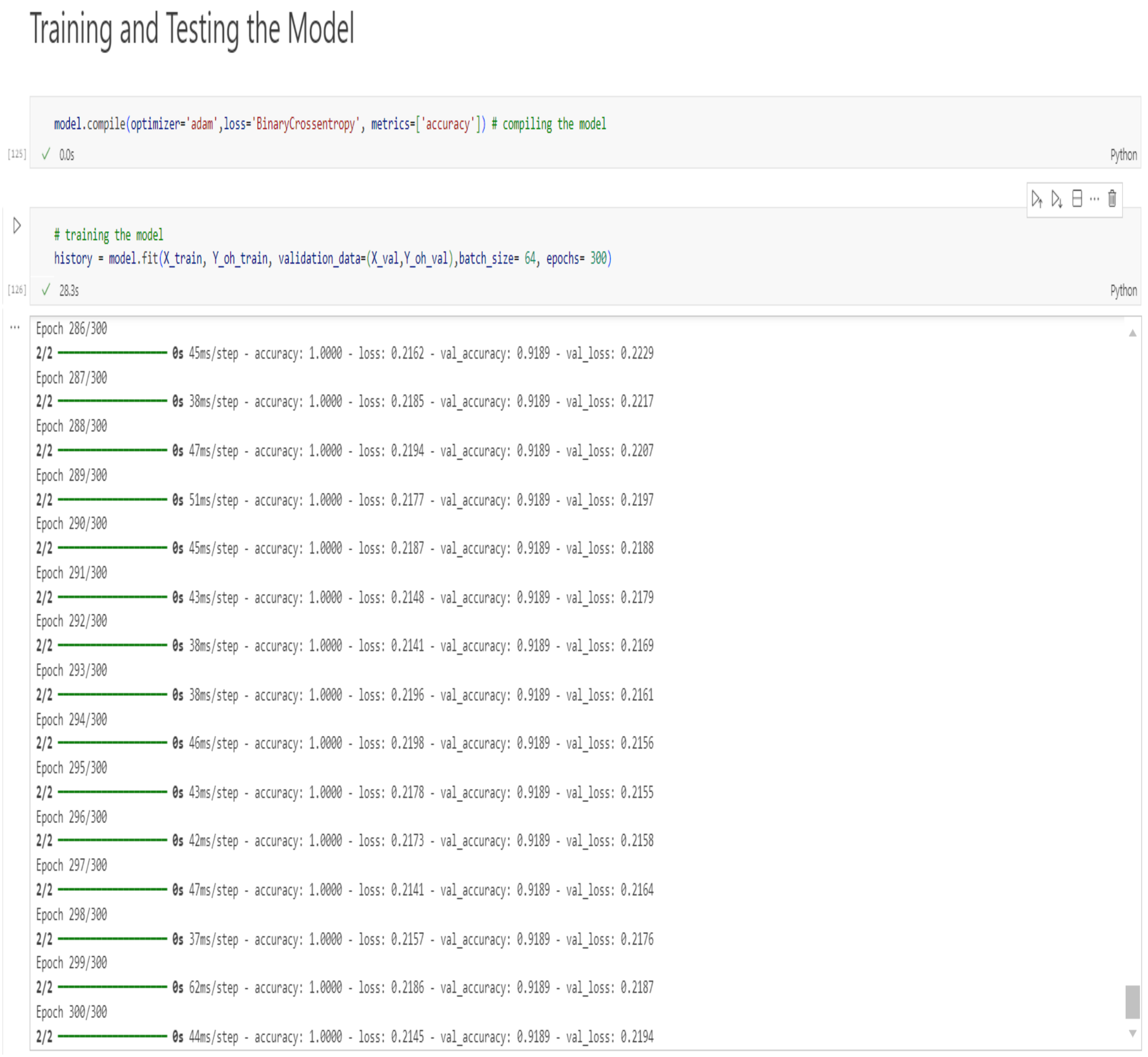




2. - Do the following Modifications

model.compile(optimizer='adam',loss='BinaryCrossentropy', metrics=['accuracy']) # compiling the model

Train model and execute loss = model.evaluate(X_test, Y_oh_test, verbose=0) print screen dumps



Training and Testing the Model

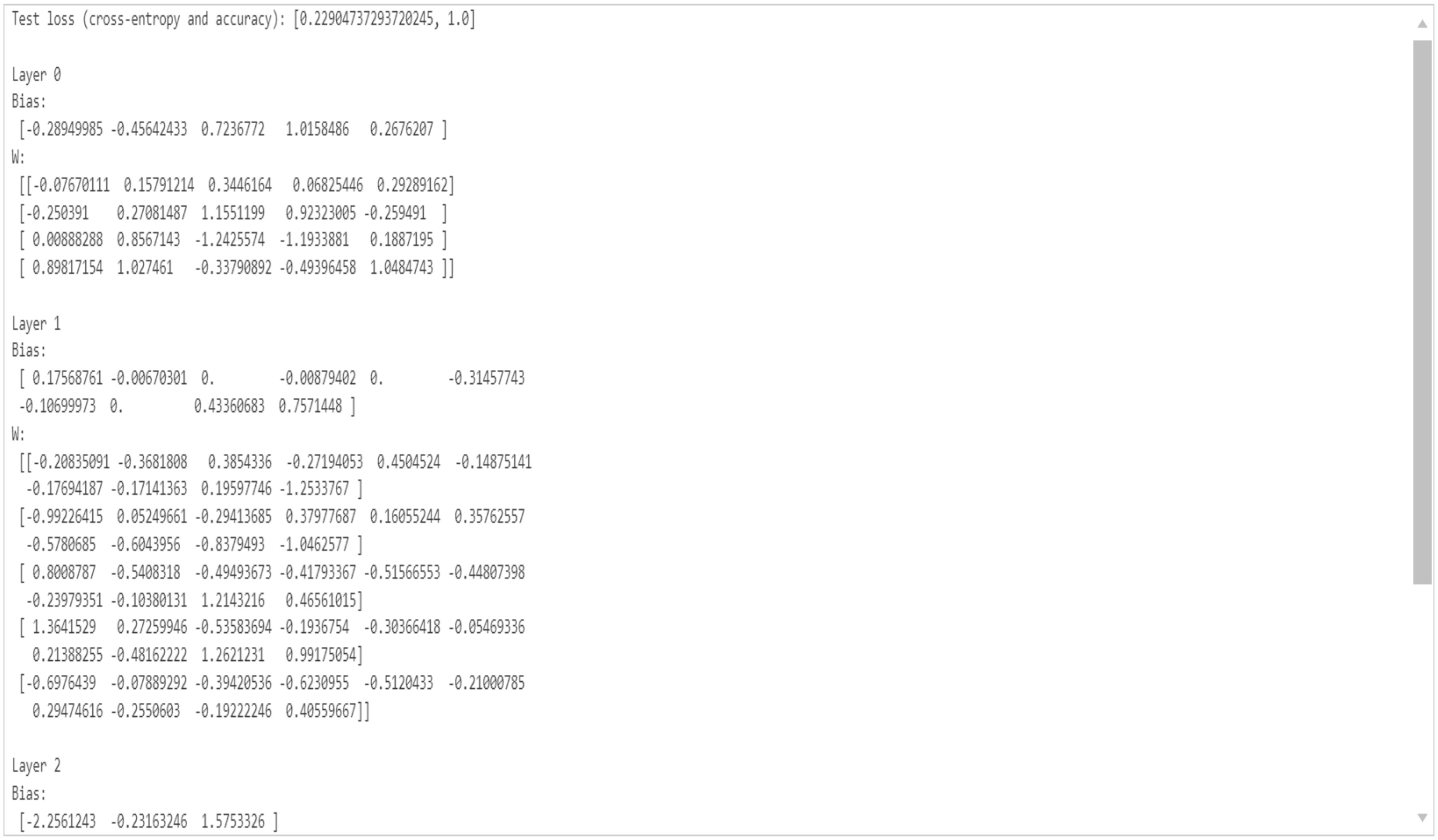
```
model.compile(optimizer='adam',loss='BinaryCrossentropy', metrics=['accuracy']) # compiling the model
```

[125] ✓ 0.0s Python

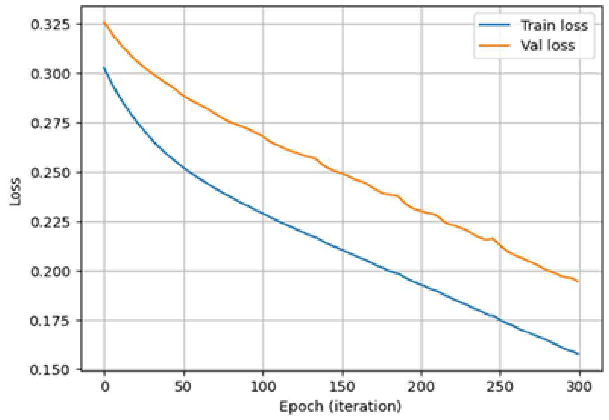
```
# training the model
history = model.fit(X_train, Y_oh_train, validation_data=(X_val,Y_oh_val),batch_size= 64, epochs= 300)
```

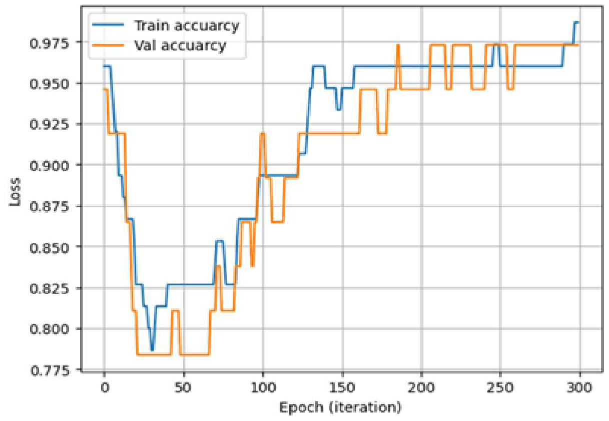
[126] ✓ 28.3s Python

```
Epoch 286/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - accuracy: 1.0000 - loss: 0.2162 - val_accuracy: 0.9189 - val_loss: 0.2229
Epoch 287/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - accuracy: 1.0000 - loss: 0.2185 - val_accuracy: 0.9189 - val_loss: 0.2217
Epoch 288/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - accuracy: 1.0000 - loss: 0.2194 - val_accuracy: 0.9189 - val_loss: 0.2207
Epoch 289/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step - accuracy: 1.0000 - loss: 0.2177 - val_accuracy: 0.9189 - val_loss: 0.2197
Epoch 290/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - accuracy: 1.0000 - loss: 0.2187 - val_accuracy: 0.9189 - val_loss: 0.2188
Epoch 291/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step - accuracy: 1.0000 - loss: 0.2148 - val_accuracy: 0.9189 - val_loss: 0.2179
Epoch 292/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - accuracy: 1.0000 - loss: 0.2141 - val_accuracy: 0.9189 - val_loss: 0.2169
Epoch 293/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - accuracy: 1.0000 - loss: 0.2196 - val_accuracy: 0.9189 - val_loss: 0.2161
Epoch 294/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step - accuracy: 1.0000 - loss: 0.2198 - val_accuracy: 0.9189 - val_loss: 0.2156
Epoch 295/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step - accuracy: 1.0000 - loss: 0.2178 - val_accuracy: 0.9189 - val_loss: 0.2155
Epoch 296/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step - accuracy: 1.0000 - loss: 0.2173 - val_accuracy: 0.9189 - val_loss: 0.2158
Epoch 297/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - accuracy: 1.0000 - loss: 0.2141 - val_accuracy: 0.9189 - val_loss: 0.2164
Epoch 298/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - accuracy: 1.0000 - loss: 0.2157 - val_accuracy: 0.9189 - val_loss: 0.2176
Epoch 299/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - accuracy: 1.0000 - loss: 0.2186 - val_accuracy: 0.9189 - val_loss: 0.2187
Epoch 300/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - accuracy: 1.0000 - loss: 0.2145 - val_accuracy: 0.9189 - val_loss: 0.2194
```



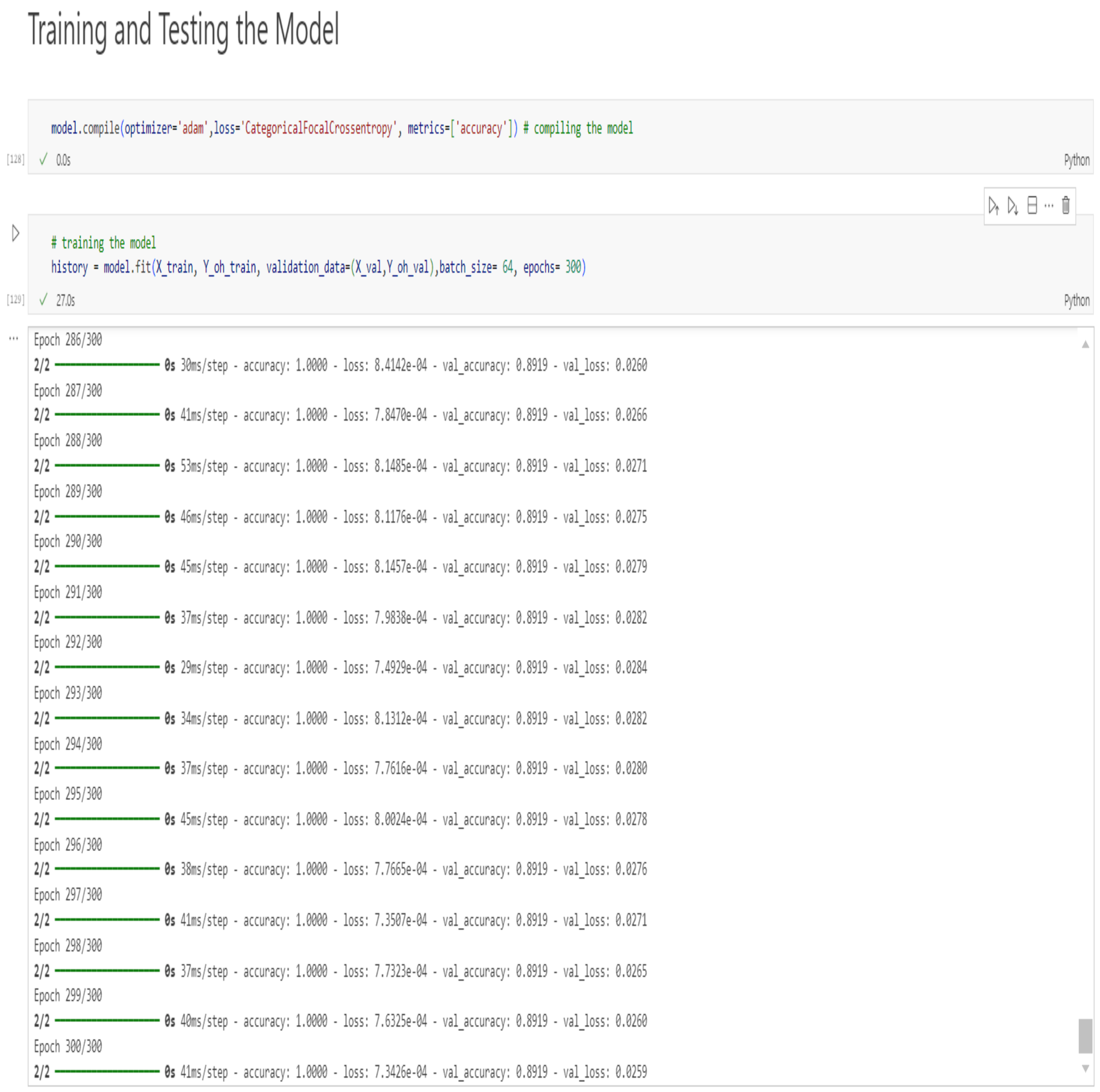
```
Test loss (cross-entropy and accuracy): [0.22904737293720245, 1.0]

Layer 0
Bias:
 [-0.28949985 -0.45642433  0.7236772   1.0158486   0.2676207 ]
W:
 [[-0.07670111  0.15791214  0.3446164   0.06825446  0.29289162]
 [-0.250391    0.27081487  1.1551199   0.92323005 -0.259491  ]
 [ 0.00888288  0.8567143  -1.2425574  -1.1933881   0.1887195 ]
 [ 0.89817154  1.027461   -0.33790892 -0.49396458  1.0484743 ]]

Layer 1
Bias:
 [ 0.17568761 -0.00670301  0.         -0.00879402  0.         -0.31457743
 -0.10699973  0.          0.43360683  0.7571448 ]
W:
 [[-0.20835091 -0.3681808   0.3854336  -0.27194053  0.4504524  -0.14875141
  -0.17694187 -0.17141363  0.19597746 -1.2533767 ]
 [-0.99226415  0.05249661 -0.29413685  0.37977687  0.16055244  0.35762557
  -0.5780685  -0.6043956  -0.8379493  -1.0462577 ]
 [ 0.8008787  -0.5408318  -0.49493673 -0.41793367 -0.51566553 -0.44807398
  -0.23979351 -0.10380131  1.2143216   0.46561015]
 [ 1.3641529   0.27259946 -0.53583694 -0.1936754  -0.30366418 -0.05469336
   0.21388255 -0.48162222  1.2621231   0.99175054]
 [-0.6976439  -0.07889292 -0.39420536 -0.6230955  -0.5120433  -0.21000785
   0.29474616 -0.2550603  -0.19222246  0.40559667]]

Layer 2
Bias:
 [-2.2561243  -0.23163246  1.5753326 ]
```





model.compile(optimizer='adam', loss='CategoricalFocalCrossentropy', metrics=['accuracy']) # compiling the model

Train model and execute loss = model.evaluate(X_test, Y_oh_test, verbose=0) print screen dumps



Training and Testing the Model

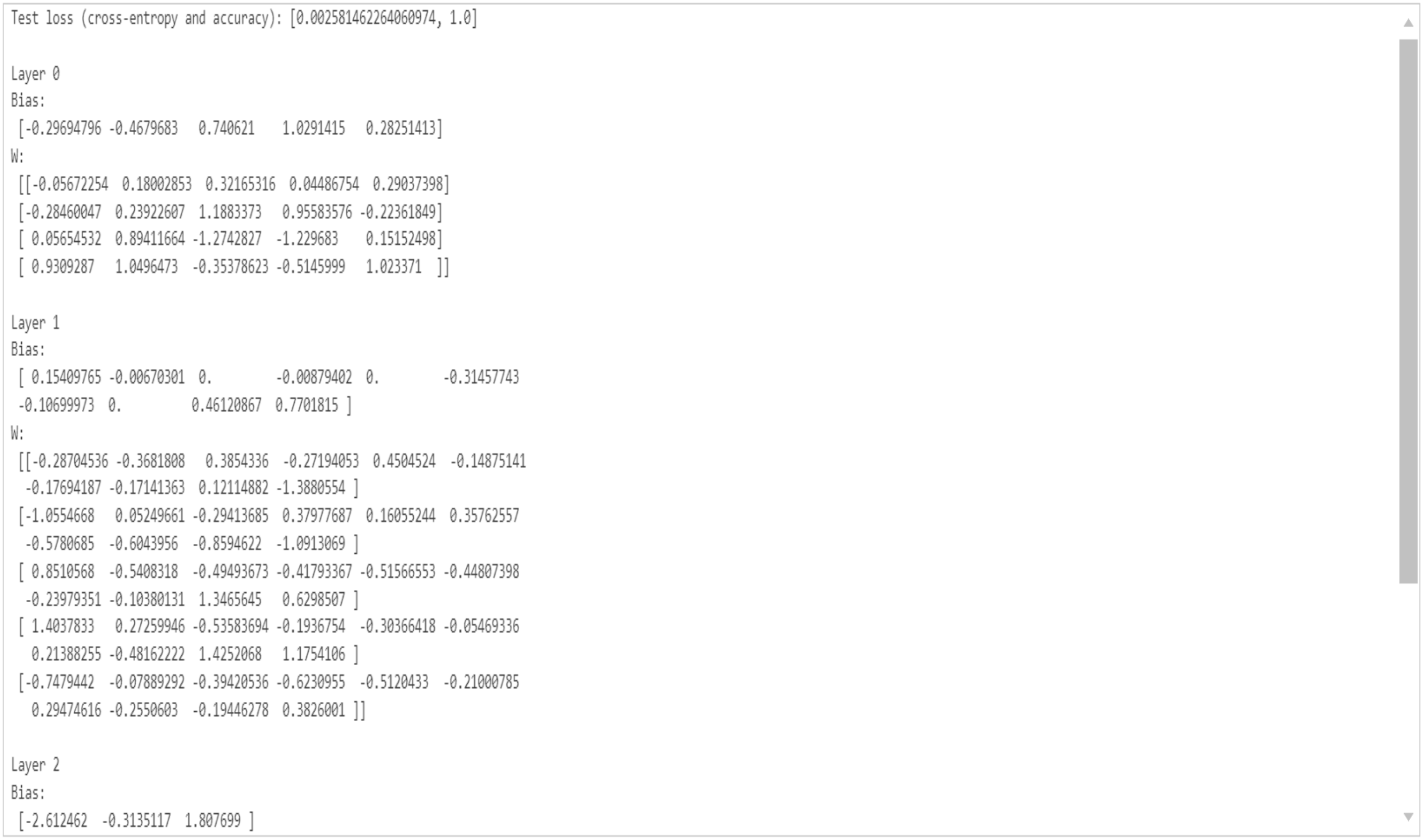
```
model.compile(optimizer='adam',loss='CategoricalFocalCrossentropy', metrics=['accuracy']) # compiling the model
```

[128] ✓ 0.0s Python

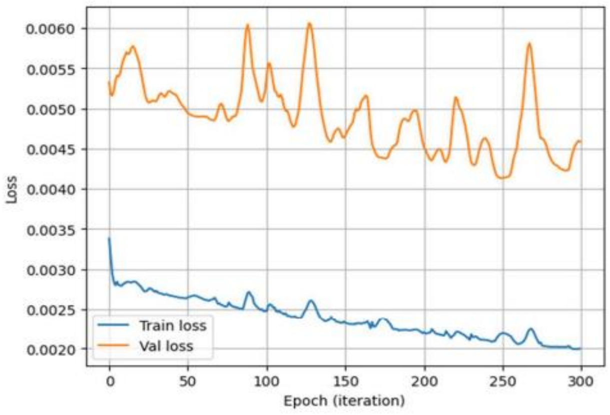
```
# training the model
history = model.fit(X_train, Y_oh_train, validation_data=(X_val,Y_oh_val),batch_size= 64, epochs= 300)
```

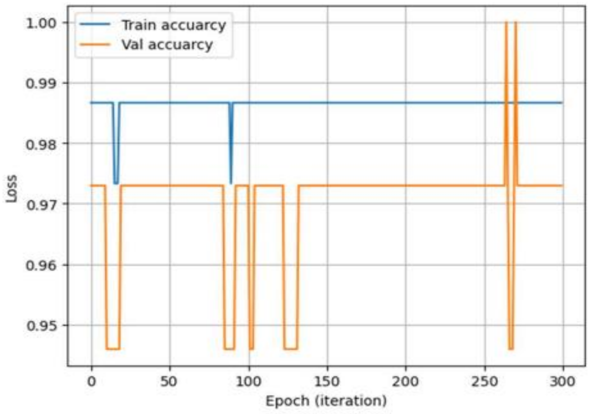
[129] ✓ 27.0s Python

```
Epoch 286/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - accuracy: 1.0000 - loss: 8.4142e-04 - val_accuracy: 0.8919 - val_loss: 0.0260
Epoch 287/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step - accuracy: 1.0000 - loss: 7.8470e-04 - val_accuracy: 0.8919 - val_loss: 0.0266
Epoch 288/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step - accuracy: 1.0000 - loss: 8.1485e-04 - val_accuracy: 0.8919 - val_loss: 0.0271
Epoch 289/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step - accuracy: 1.0000 - loss: 8.1176e-04 - val_accuracy: 0.8919 - val_loss: 0.0275
Epoch 290/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - accuracy: 1.0000 - loss: 8.1457e-04 - val_accuracy: 0.8919 - val_loss: 0.0279
Epoch 291/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - accuracy: 1.0000 - loss: 7.9838e-04 - val_accuracy: 0.8919 - val_loss: 0.0282
Epoch 292/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step - accuracy: 1.0000 - loss: 7.4929e-04 - val_accuracy: 0.8919 - val_loss: 0.0284
Epoch 293/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 1.0000 - loss: 8.1312e-04 - val_accuracy: 0.8919 - val_loss: 0.0282
Epoch 294/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - accuracy: 1.0000 - loss: 7.7616e-04 - val_accuracy: 0.8919 - val_loss: 0.0280
Epoch 295/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - accuracy: 1.0000 - loss: 8.0024e-04 - val_accuracy: 0.8919 - val_loss: 0.0278
Epoch 296/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - accuracy: 1.0000 - loss: 7.7665e-04 - val_accuracy: 0.8919 - val_loss: 0.0276
Epoch 297/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step - accuracy: 1.0000 - loss: 7.3507e-04 - val_accuracy: 0.8919 - val_loss: 0.0271
Epoch 298/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - accuracy: 1.0000 - loss: 7.7323e-04 - val_accuracy: 0.8919 - val_loss: 0.0265
Epoch 299/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step - accuracy: 1.0000 - loss: 7.6325e-04 - val_accuracy: 0.8919 - val_loss: 0.0260
Epoch 300/300
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step - accuracy: 1.0000 - loss: 7.3426e-04 - val_accuracy: 0.8919 - val_loss: 0.0259
```



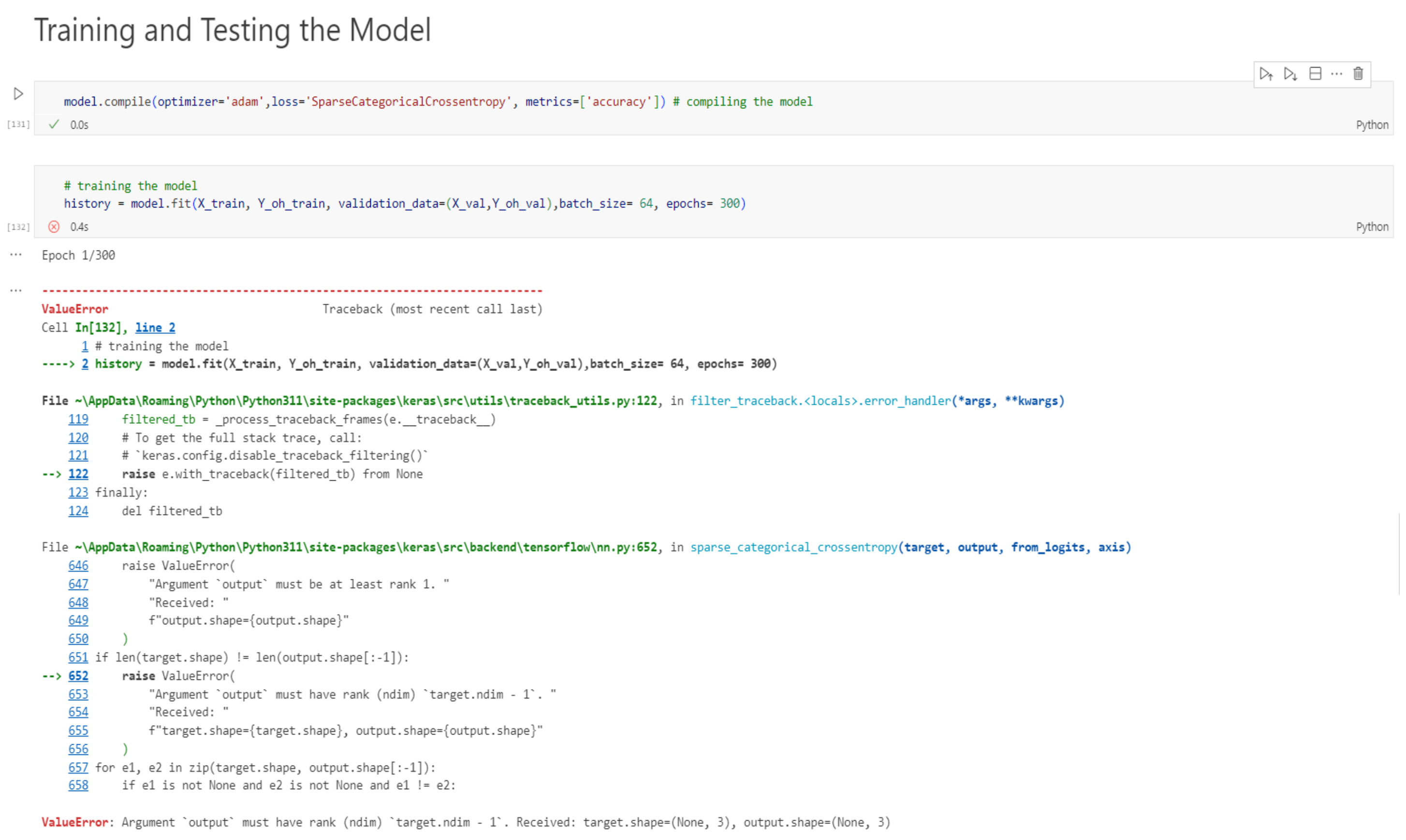
```
Test loss (cross-entropy and accuracy): [0.002581462264060974, 1.0]

Layer 0
Bias:
 [-0.29694796 -0.4679683   0.740621    1.0291415   0.28251413]
W:
 [[-0.05672254  0.18002853  0.32165316  0.04486754  0.29037398]
 [-0.28460047  0.23922607  1.1883373   0.95583576 -0.22361849]
 [ 0.05654532  0.89411664 -1.2742827  -1.229683    0.15152498]
 [ 0.9309287   1.0496473  -0.35378623 -0.5145999   1.023371  ]]

Layer 1
Bias:
 [ 0.15409765 -0.00670301  0.         -0.00879402  0.         -0.31457743
 -0.10699973  0.          0.46120867  0.7701815 ]
W:
 [[-0.28704536 -0.3681808   0.3854336  -0.27194053  0.4504524  -0.14875141
  -0.17694187 -0.17141363  0.12114882 -1.3880554 ]
 [-1.0554668   0.05249661 -0.29413685  0.37977687  0.16055244  0.35762557
  -0.5780685  -0.6043956  -0.8594622  -1.0913069 ]
 [ 0.8510568  -0.5408318  -0.49493673 -0.41793367 -0.51566553 -0.44807398
  -0.23979351 -0.10380131  1.3465645   0.6298507 ]
 [ 1.4037833   0.27259946 -0.53583694 -0.1936754  -0.30366418 -0.05469336
   0.21388255 -0.48162222  1.4252068   1.1754106 ]
 [-0.7479442  -0.07889292 -0.39420536 -0.6230955  -0.5120433  -0.21000785
   0.29474616 -0.2550603  -0.19446278  0.3826001 ]]

Layer 2
Bias:
 [-2.612462  -0.3135117  1.807699 ]
```





model.compile(optimizer='adam',loss='SparseCategoricalCrossentropy', metrics=['accuracy']) # compiling the model

Train model and execute loss = model.evaluate(X_test, Y_oh_test, verbose=0) print screen dumps



Training and Testing the Model

```
[131]  model.compile(optimizer='adam',loss='SparseCategoricalCrossentropy', metrics=['accuracy']) # compiling the model
       ✓ 0.0s                                                                                      Python
```

```
       # training the model
       history = model.fit(X_train, Y_oh_train, validation_data=(X_val,Y_oh_val),batch_size= 64, epochs= 300)
[132]  ⊗ 0.4s                                                                                      Python
```

```
Epoch 1/300

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[132], line 2
      1 # training the model
----> 2 history = model.fit(X_train, Y_oh_train, validation_data=(X_val,Y_oh_val),batch_size= 64, epochs= 300)

File ~\AppData\Roaming\Python\Python311\site-packages\keras\src\utils\traceback_utils.py:122, in filter_traceback.<locals>.error_handler(*args, **kwargs)
    119     filtered_tb = _process_traceback_frames(e.__traceback__)
    120     # To get the full stack trace, call:
    121     # `keras.config.disable_traceback_filtering()`
--> 122     raise e.with_traceback(filtered_tb) from None
    123 finally:
    124     del filtered_tb

File ~\AppData\Roaming\Python\Python311\site-packages\keras\src\backend\tensorflow\nn.py:652, in sparse_categorical_crossentropy(target, output, from_logits, axis)
    646     raise ValueError(
    647         "Argument `output` must be at least rank 1. "
    648         "Received: "
    649         f"output.shape={output.shape}"
    650     )
    651 if len(target.shape) != len(output.shape[:-1]):
--> 652     raise ValueError(
    653         "Argument `output` must have rank (ndim) `target.ndim - 1`. "
    654         "Received: "
    655         f"target.shape={target.shape}, output.shape={output.shape}"
    656     )
    657 for e1, e2 in zip(target.shape, output.shape[:-1]):
    658     if e1 is not None and e2 is not None and e1 != e2:

ValueError: Argument `output` must have rank (ndim) `target.ndim - 1`. Received: target.shape=(None, 3), output.shape=(None, 3)
```