Course: Data Mining

Assignment

Modify Parameters to The Models with Neural Networks

Submitting by: Manish Chaulagai

Do the following modifications:

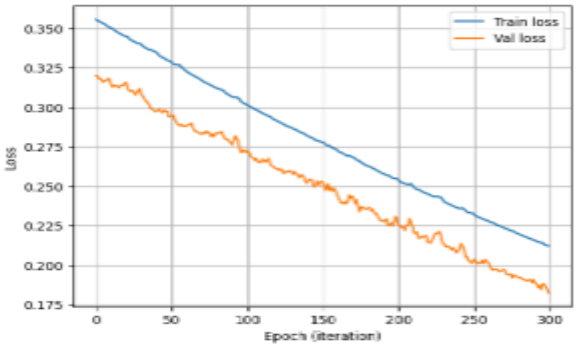
1. model.compile(optimizer='SGD',loss='categorical_crossentropy', metrics=['accuracy']) # compiling the model.Train model and execute loss = model.evaluate(X_test, Y_oh_test, verbose=0)

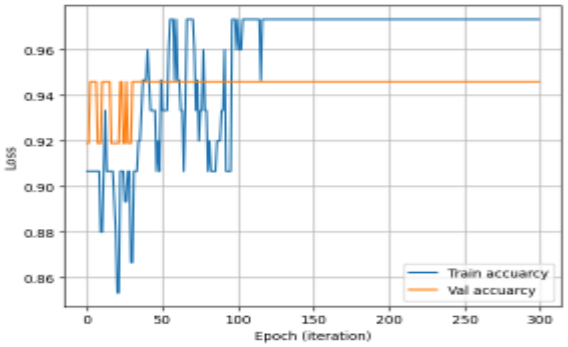
```
Test loss (cross-entropy and accuracy): [0.1978459587226944, 0.9473684430122375]

Layer 0
Bias:
 [ 0.          0.         -0.09133741  0.98257315 -0.12405898]
W:
 [[-0.47510503 -0.7265717   0.41301653  1.060562    0.8513031 ]
 [-0.22424817 -0.090747   -0.56575084  0.36271587 -0.2752011 ]
 [ 0.310562   -0.0703612   0.19455622 -0.62020165  0.869183  ]
 [-0.70329857 -0.20342726 -0.38557157 -0.87289757  1.0015596 ]]

Layer 1
Bias:
 [-0.1497416   0.         -0.00089244  0.37545776 -0.09731438  0.33233562
 -0.11424197  0.32853356  0.60353297 -0.01590705]
W:
 [[ 0.48191875 -0.5193738  -0.14982516  0.3031872  -0.12904894 -0.22757837
   0.6303268  -0.3693826   0.28981793 -0.16607645]
 [ 0.3319252  -0.4587778  -0.5763796   0.4351248  -0.46770436 -0.30676016
  -0.34243417 -0.23009068  0.5032564   0.16841567]
 [-0.34030637  0.52148445 -0.33044973  0.29421344  0.06290595  0.07198477
  -0.61382467 -0.25046167 -0.24695861  0.2497934 ]
 [-0.23324859 -0.07622695  0.60530543  0.73728424 -0.0877004   1.0245292
  -0.07041913  1.0514976   1.2015029   0.46457484]
 [ 0.9041595  -0.54274416 -0.43301144 -0.9092969   1.2013642  -0.94545627
   1.0284847   0.26142582 -0.68446554 -0.33468075]]

Layer 2
Bias:
 [-0.17604806  0.38689885 -0.2567573 ]
W:
 [[-1.1512209  -0.397598    0.95958734]
 [-0.5137823  -0.32509917 -0.25868243]
 [ 0.33452087 -0.7188645  -0.38298842]
 [ 0.41270632 -0.8819828  -0.49721763]
 [-0.4898258   0.03082809  0.6427525 ]
 [ 1.002806   -0.40341175 -0.4941628 ]
 [-1.2457329  -0.4622121   0.40918377]
 [-0.03737662  0.8792185  -0.6609233 ]
 [ 0.9865588   0.51808596 -0.7910855 ]
 [ 0.76147497 -0.73322356  0.16213806]]
```





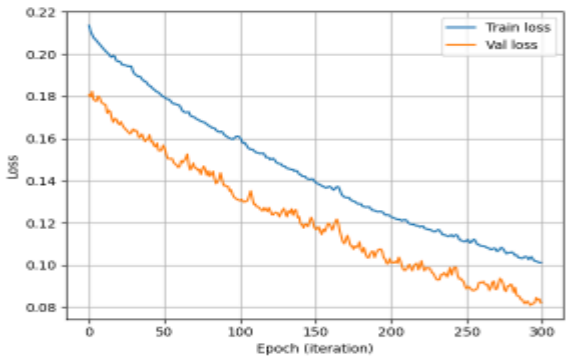
2. model.compile(optimizer='RMSprop'Links to an external site.",loss='categorical_crossentropy', metrics=['accuracy']) # compiling the model Train model and execute loss = model.evaluate(X_test, Y_oh_test, verbose=0)

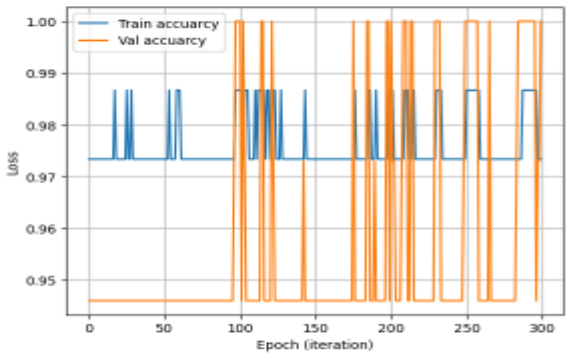
```
Test loss (cross-entropy and accuracy): [0.09588704258203506, 0.9473684430122375]

Layer 0
Bias:
 [ 0.          0.         -0.07357359  1.1021179  -0.20165585]
W:
 [[-0.47510603 -0.7265717   0.44357073  1.0334648   0.9032040 ]
 [-0.22424817 -0.090747   -0.46640283  0.34733945 -0.23209861]
 [ 0.310562   -0.0703612   0.16184211 -0.6647818   0.945294561
 [-0.70329857 -0.20342726 -0.42142448 -1.009364    1.2513111 ]]

Layer 1
Bias:
 [-0.23882133  0.          -0.00465887  0.33365855 -0.17338602  0.25073442
 -0.18917955  0.45297626  0.73710966 -0.00387678]
W:
 [[ 0.48191875 -0.5193738  -0.14982516  0.3031872  -0.12904894 -0.22757837
   0.6303268  -0.3693826   0.28981793 -0.16607645]
 [ 0.3319252  -0.4587778  -0.5763796   0.4351248  -0.46778436 -0.30576016
  -0.34243417 -0.23909868  0.5032564   0.16841567]
 [-0.5445584   0.52148445 -0.33844973  0.29421344 -0.06577943  0.06548416
  -0.77274866 -0.02873689 -0.02679571  0.2492229 ]
 [-0.46149716 -0.07622695  0.8450582   0.94702476 -0.3056044   1.2225888
  -0.2867276   1.3035774   1.4624895   0.7081905 ]
 [ 1.0226593  -0.54274416 -0.7058199  -1.2110809   1.3283045  -1.3064066
   1.1558638   0.17358154 -0.6776954  -0.5914603 ]]

Layer 2
Bias:
 [-0.57011694  0.517905   -0.36707887]
W:
 [[-1.6485107  -0.588319    1.1685238 ]
 [-0.5137823  -0.32589917 -0.25868243]
 [ 0.78709024 -1.1619341  -0.41381893]
 [ 0.89220774 -1.3533934  -0.61096185]
 [-1.0877096  -0.09100587  0.787669  ]
 [ 1.4818642  -0.83962935 -0.7193211 ]
 [-1.7594641  -0.59194106  0.56142503]
 [-0.48172374  1.0362512  -0.80358536]
 [ 0.956319    0.97684234 -1.252547  ]
 [ 1.1931193  -1.1636469   0.14611137]]
```





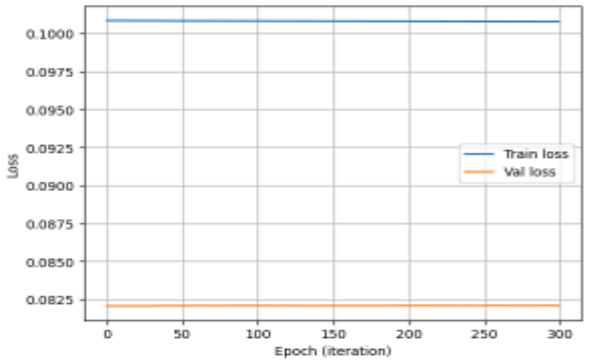
3. model.compile(optimizer='AdadeltaLinks to an external site.',loss='categorical_crossentropy', metrics=['accuracy']) # compiling the model Train model and execute loss = model.evaluate(X_test, Y_oh_test, verbose=0)

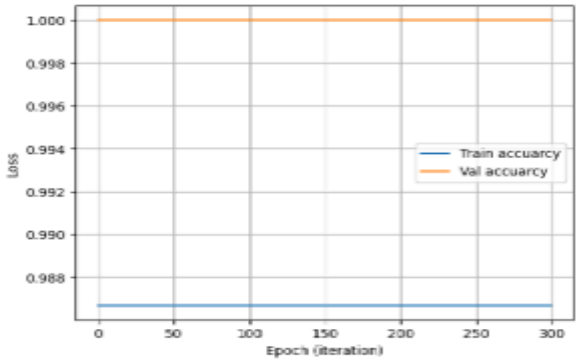
```
Test loss (cross-entropy and accuracy): [0.09585529565811157, 0.9473684430122375]

Layer 0
Bias:
 [ 0.          0.         -0.07365297  1.1021334  -0.28165831]
W:
 [[-0.47510683 -0.7265717   0.44358103  1.0333765   0.98329885]
 [-0.22424817 -0.050747   -0.4663583   0.34727132 -0.23202437]
 [ 0.310562   -0.0783612   0.16174015 -0.66488093  0.9454822 ]
 [-0.70329857 -0.20342726 -0.42152667 -1.0041151   1.2514701 ]]

Layer 1
Bias:
 [-0.23882563  0.         -0.00467225  0.3336498  -0.17338575  0.25064594
 -0.18917955  0.45299223  0.73713386 -0.00390381]
W:
 [[ 0.48191875 -0.5193738  -0.14982516  0.3031872  -0.12984894 -0.22757837
   0.6383268  -0.3693826   0.28981793 -0.16607645]
 [ 0.3319252  -0.4587778  -0.5763796   0.4351248  -0.46778436 -0.30676016
  -0.34243417 -0.23909068  0.5032564   0.16841567]
 [-0.5446106   0.52148445 -0.33044973  0.29421344 -0.06581234  0.06548416
  -0.7727883  -0.02866871 -0.02671725  0.2492229 ]
 [-0.4616435  -0.07622695  0.84514046  0.9471207  -0.30574584  1.2226248
  -0.28686035  1.3037335   1.4626504   0.70826584]
 [ 1.0228826  -0.54274416 -0.70592713 -1.2111111   1.3284503  -1.3866068
   1.1560096   0.17336634 -0.6778208  -0.5915845 ]]

Layer 2
Bias:
 [-0.5783819   0.51792264 -0.3670895 ]
W:
 [[-1.648654   -0.58855504  1.1687658 ]
 [-0.5137823  -0.32500917 -0.25868243]
 [ 0.78718764 -1.1620215  -0.41381893]
 [ 0.89234185 -1.35353    -0.61096185]
 [-1.0079092  -0.09118191  0.7878511 ]
 [ 1.4828169  -0.8397809  -0.7193232 ]
 [-1.7597017  -0.592126    0.56161517]
 [-0.40193737  1.0363163  -0.80356574]
 [ 0.95625144  0.9773525  -1.2530549 ]
 [ 1.1931958  -1.1637231   0.14611083]]
```





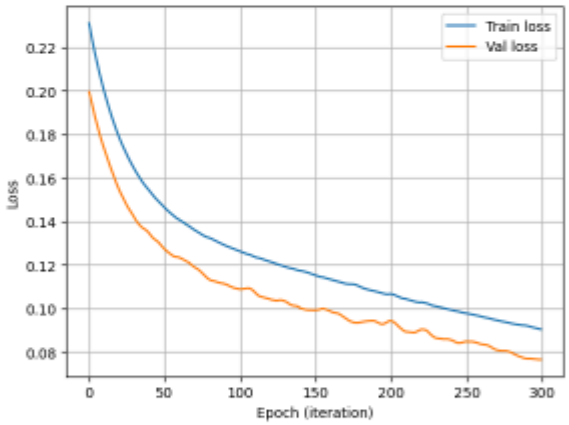
4. model.compile(optimizer='adam',loss='BinaryCrossentropyLinks to an external site.', metrics=['accuracy']) # compiling the model
   Train model and execute loss = model.evaluate(X_test, Y_oh_test, verbose=0)

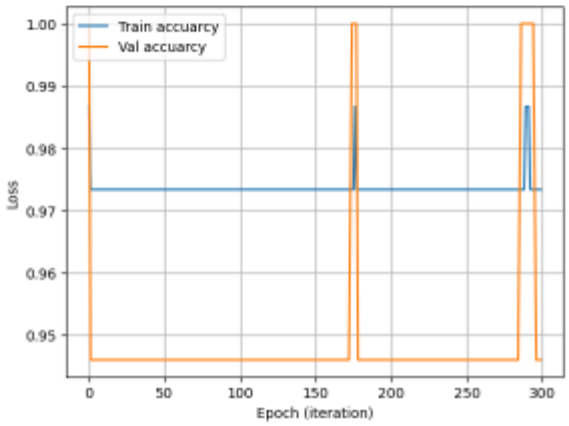
```
Test loss (cross-entropy and accuracy): [0.08294162899255753, 0.9473684430122375]

Layer 0
Bias:
 [ 0.          0.         -0.0359267   1.2498652  -0.29310703]
W:
 [[-0.47510603 -0.7265717   0.4760408   1.0065806   0.92936975]
 [-0.22424817 -0.098747   -0.46461573  0.39798826 -0.2811272 ]
 [ 0.318562   -0.8703612   0.11953554 -0.7216647   1.022096  ]
 [-0.70329857 -0.20342726 -0.4679155  -1.1518855   1.4216532 ]]

Layer 1
Bias:
 [-0.34790242  0.          0.1385873   0.47506088 -0.32786837  0.34931833
 -0.1683966   0.46887927  0.84374046  0.1131085 ]
W:
 [[ 0.48191875 -0.5193738  -0.14982516  0.3831872  -0.12984894 -0.22757837
   0.6383268  -0.3693826   0.28981793 -0.16607645]
 [ 0.3319252  -0.4587778  -0.5763796   0.4351248  -0.46778436 -0.30676016
  -0.34243417 -0.23909868  0.5832564   0.16841567]
 [-0.84211004  0.52148445 -0.33044973  0.29421344 -0.34710988  0.06548416
  -1.0828373   0.28129823  0.27834448  0.21012725]
 [-0.7474145  -0.87622695  1.0419022   1.1419454  -0.58193856  1.4831578
  -0.48850605  1.4135842   1.6631864   0.80445966]
 [ 1.1455418  -0.54274416 -0.08956025 -1.4957731   1.3452157  -1.6154879
   1.3512598   0.03892156 -0.71344864 -0.88815956]]

Layer 2
Bias:
 [-0.90344775  0.42823712 -0.57031053]
W:
 [[-1.943402   -0.84615153  1.227468  ]
 [-0.5137823  -0.32589917 -0.25868243]
 [ 0.693856   -1.4872511  -0.6654735 ]
 [ 0.84126794 -1.5915855  -0.85652554]
 [-1.3286352  -0.38714458  0.77973396]
 [ 1.4751499  -1.8739826  -0.9440627 ]
 [-2.1383835  -0.88998284  0.52163714]
 [-0.7378885   0.96953607 -1.8420508 ]
 [ 0.6169379   1.0510229  -1.637887  ]
 [ 1.1423732  -1.4118114  -0.88973102]]
```





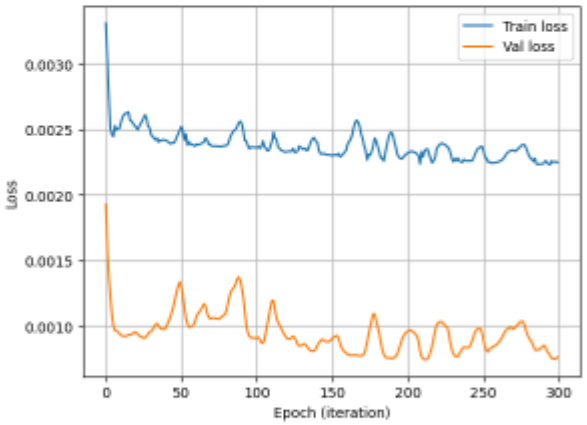
5. model.compile(optimizer='adam',loss='CategoricalFocalCrossentropy', metrics=['accuracy']) # compiling the model Train model and execute loss = model.evaluate(X_test, Y_oh_test, verbose=0)

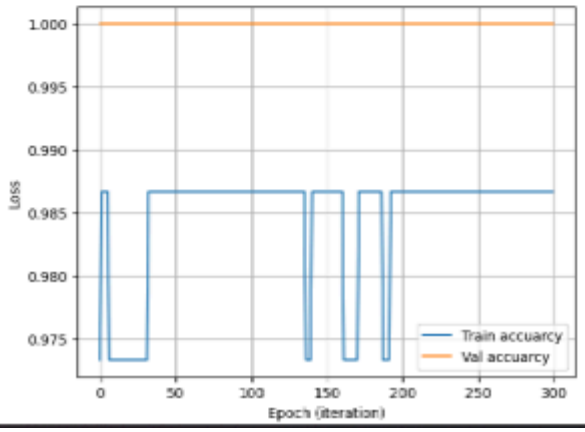
```
Test loss (cross-entropy and accuracy): [0.007724316325038671, 0.9473684430122375]

Layer 0
Bias:
 [ 0.          0.         -0.05163411  1.2749044  -0.31869906]
W:
 [[-0.47510603 -0.7265717   0.45888366  0.9842717   0.9513218 ]
 [-0.22424817 -0.090747   -0.35470486  0.36617234 -0.2406648 ]
 [ 0.310562   -0.0703612   0.10156424 -0.7084861   1.0084816 ]
 [-0.70329857 -0.20342726 -0.48519728 -1.1394151   1.4887424 ]]

Layer 1
Bias:
 [-0.37332448  0.          0.10754375  0.44198486 -0.3534104   0.31483456
 -0.19386835  0.4860519   0.8689849   0.07758402]
W:
 [[ 0.48191875 -0.5193738  -0.14982516  0.3831872  -0.12904894 -0.22757837
   0.6303268  -0.3603826   0.28981793 -0.16607645]
 [ 0.3319252  -0.4587778  -0.5763706   0.4351248  -0.46770436 -0.38676016
  -0.34243417 -0.23909068  0.5032564   0.16841567]
 [-0.86132425  0.52148445 -0.33044973  0.29421344 -0.36147562  0.06548416
  -1.0087126   0.30046135  0.20956254  0.21012725]
 [-0.7462633  -0.07622695  1.0164834   1.1137189  -0.58089566  1.3744626
  -0.4875092   1.4121155   1.6618862   0.86484295]
 [ 1.1424116  -0.54274416 -1.0152427  -1.524193    1.3419796  -1.6443777
   1.3480679   0.04181978 -0.7105397  -0.90997344]]

Layer 2
Bias:
 [-1.0841691   0.445536   -0.59558266]
W:
 [[-2.1216842  -0.8479843   1.229244  ]
 [-0.5137823  -0.32589917 -0.25868243]
 [ 0.6010303  -1.4042188  -0.6657018 ]
 [ 0.8379089  -1.5879918  -0.8567882 ]
 [-1.5263073  -0.38748677  0.7880085 ]
 [ 1.4723041  -1.0709255  -0.94428897]
 [-2.3463783  -0.808123    0.5197999 ]
 [-0.9432992   0.97658486 -1.0489943 ]
 [ 0.45772824  1.0723945  -1.6592294 ]
 [ 1.140735   -1.410057   -0.08985411]]
```





6. 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)

Training and Testing the Model

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

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

```
Epoch 1/300
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[350], 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 ~\anaconda3\Lib\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 ~\anaconda3\Lib\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)
```

```
loss = model.evaluate(X_test, Y_oh_test, verbose=0)
print('Test loss (cross-entropy and accuracy):',loss)
print()
W = model.get_weights()
for ii in range(len(W)//2):
    print("Layer %d" %ii)
    print('Bias:\n', W[2*ii + 1])
    print('W:\n', W[2*ii])
    print()

plt.plot(history.history['loss'], label = "Train loss")
plt.plot(history.history['val_loss'], label = "Val loss")
plt.xlabel("Epoch (iteration)")
plt.ylabel("Loss")
plt.legend()
plt.grid()
plt.show()


plt.plot(history.history['accuracy'], label = "Train accuarcy")
plt.plot(history.history['val_accuracy'], label = "Val accuarcy")
plt.xlabel("Epoch (iteration)")
plt.ylabel("Loss")
plt.legend()
plt.grid()
```