

CS564 Foundations of Machine Learning

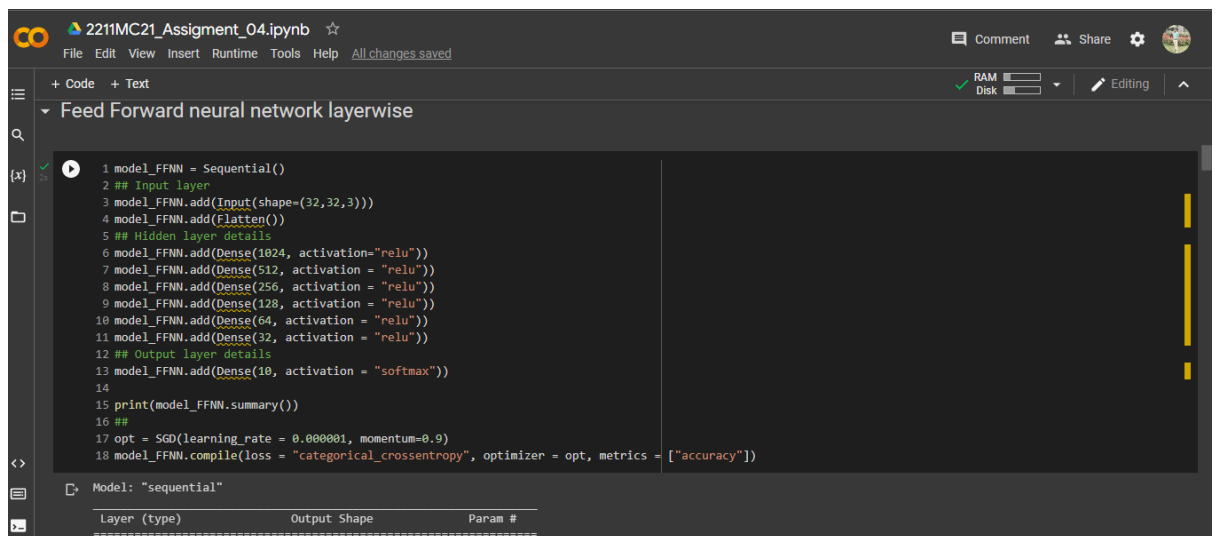
ASSIGNMENT 4

Nikunj Pansari

2211MC21

Problem Statement:

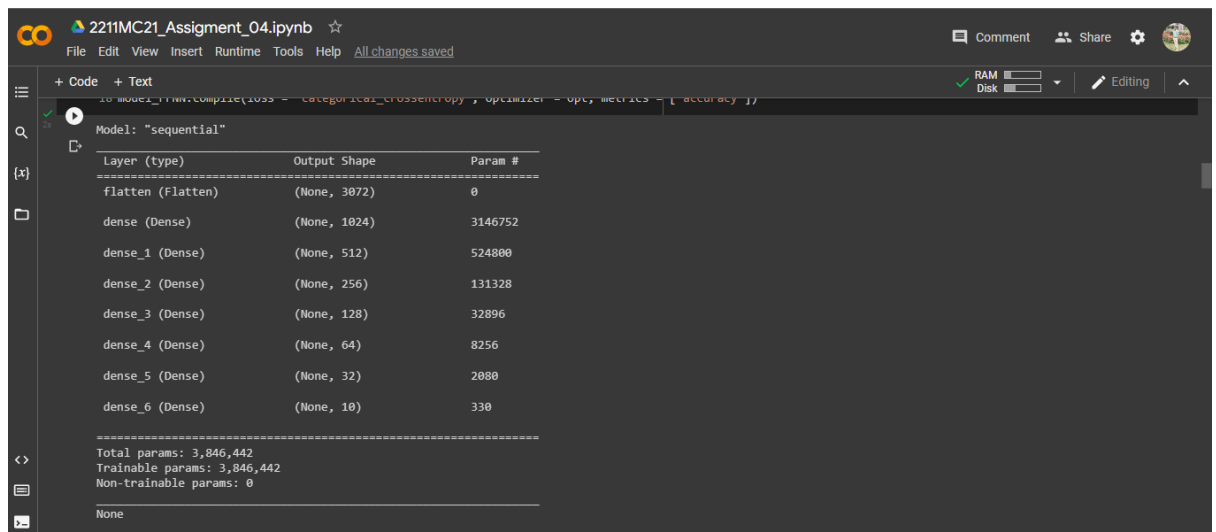
- Design and implement a Feed Forward Neural Network (FFNN) and a Recurrent Neural Network (RNN) for the task of image classification on the CIFAR-10 dataset.
- **Implementation:**
 - Feed-forward neural network for CIFAR-10 dataset.



```
1 model_FFNN = Sequential()
2 ## Input layer
3 model_FFNN.add(Input(shape=(32,32,3)))
4 model_FFNN.add(Flatten())
5 ## Hidden layer details
6 model_FFNN.add(Dense(1024, activation="relu"))
7 model_FFNN.add(Dense(512, activation = "relu"))
8 model_FFNN.add(Dense(256, activation = "relu"))
9 model_FFNN.add(Dense(128, activation = "relu"))
10 model_FFNN.add(Dense(64, activation = "relu"))
11 model_FFNN.add(Dense(32, activation = "relu"))
12 ## Output layer details
13 model_FFNN.add(Dense(10, activation = "softmax"))
14
15 print(model_FFNN.summary())
16 ##
17 opt = SGD(learning_rate = 0.000001, momentum=0.9)
18 model_FFNN.compile(loss = "categorical_crossentropy", optimizer = opt, metrics = ["accuracy"])
```

Model: "sequential"

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 3072)	0
dense (Dense)	(None, 1024)	3146752
dense_1 (Dense)	(None, 512)	524800
dense_2 (Dense)	(None, 256)	131328
dense_3 (Dense)	(None, 128)	32896
dense_4 (Dense)	(None, 64)	8256
dense_5 (Dense)	(None, 32)	2080
dense_6 (Dense)	(None, 10)	330



```
10 model_FFNN.compile(loss = "categorical_crossentropy", optimizer = opt, metrics = ["accuracy"])
```

Model: "sequential"

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 3072)	0
dense (Dense)	(None, 1024)	3146752
dense_1 (Dense)	(None, 512)	524800
dense_2 (Dense)	(None, 256)	131328
dense_3 (Dense)	(None, 128)	32896
dense_4 (Dense)	(None, 64)	8256
dense_5 (Dense)	(None, 32)	2080
dense_6 (Dense)	(None, 10)	330

Total params: 3,846,442
Trainable params: 3,846,442
Non-trainable params: 0

```
2211MC21_Assignment_04.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
1 model_FFNN.fit(X_train, y_cat_train, validation_data = (X_test, y_cat_test), epochs = 50, batch_size = 128)

Epoch 11/50
391/391 [=====] - 2s 5ms/step - loss: 2.3128 - accuracy: 0.1034 - val_loss: 2.3196 - val_accuracy: 0.1014
Epoch 12/50
391/391 [=====] - 2s 5ms/step - loss: 2.3115 - accuracy: 0.1033 - val_loss: 2.3186 - val_accuracy: 0.1013
Epoch 13/50
391/391 [=====] - 2s 5ms/step - loss: 2.3103 - accuracy: 0.1032 - val_loss: 2.3177 - val_accuracy: 0.1012
Epoch 14/50
391/391 [=====] - 2s 5ms/step - loss: 2.3094 - accuracy: 0.1034 - val_loss: 2.3170 - val_accuracy: 0.1011
Epoch 15/50
391/391 [=====] - 2s 5ms/step - loss: 2.3085 - accuracy: 0.1035 - val_loss: 2.3163 - val_accuracy: 0.1009
Epoch 16/50
391/391 [=====] - 2s 5ms/step - loss: 2.3077 - accuracy: 0.1033 - val_loss: 2.3157 - val_accuracy: 0.1012
Epoch 17/50
391/391 [=====] - 2s 5ms/step - loss: 2.3070 - accuracy: 0.1034 - val_loss: 2.3152 - val_accuracy: 0.1013
Epoch 18/50
391/391 [=====] - 2s 5ms/step - loss: 2.3064 - accuracy: 0.1033 - val_loss: 2.3148 - val_accuracy: 0.1016
Epoch 19/50
391/391 [=====] - 2s 5ms/step - loss: 2.3058 - accuracy: 0.1033 - val_loss: 2.3144 - val_accuracy: 0.1012
Epoch 20/50
391/391 [=====] - 2s 5ms/step - loss: 2.3053 - accuracy: 0.1032 - val_loss: 2.3140 - val_accuracy: 0.1012
Epoch 21/50
391/391 [=====] - 2s 5ms/step - loss: 2.3048 - accuracy: 0.1034 - val_loss: 2.3137 - val_accuracy: 0.1013
Epoch 22/50
391/391 [=====] - 2s 5ms/step - loss: 2.3044 - accuracy: 0.1035 - val_loss: 2.3134 - val_accuracy: 0.1011
Epoch 23/50
391/391 [=====] - 2s 5ms/step - loss: 2.3040 - accuracy: 0.1036 - val_loss: 2.3131 - val_accuracy: 0.1013
```

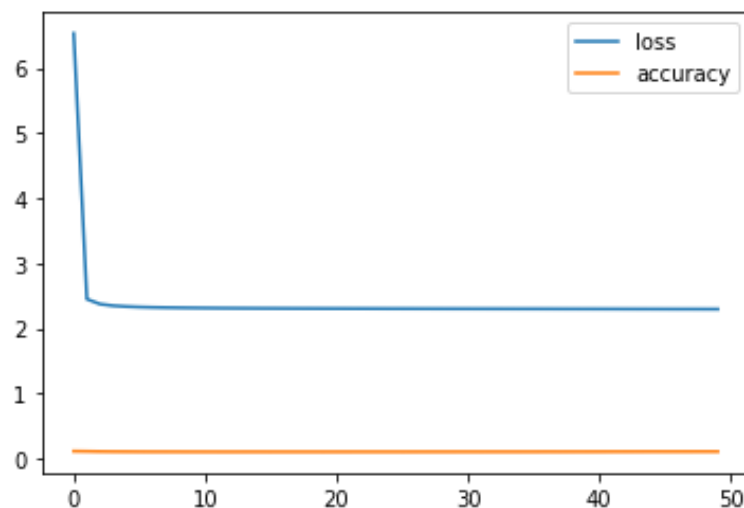
```
2211MC21_Assignment_04.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
391/391 [=====] - 2s 5ms/step - loss: 2.2999 - accuracy: 0.1074 - val_loss: 2.3075 - val_accuracy: 0.1057
<keras.callbacks.History at 0x7fabb446bd0>

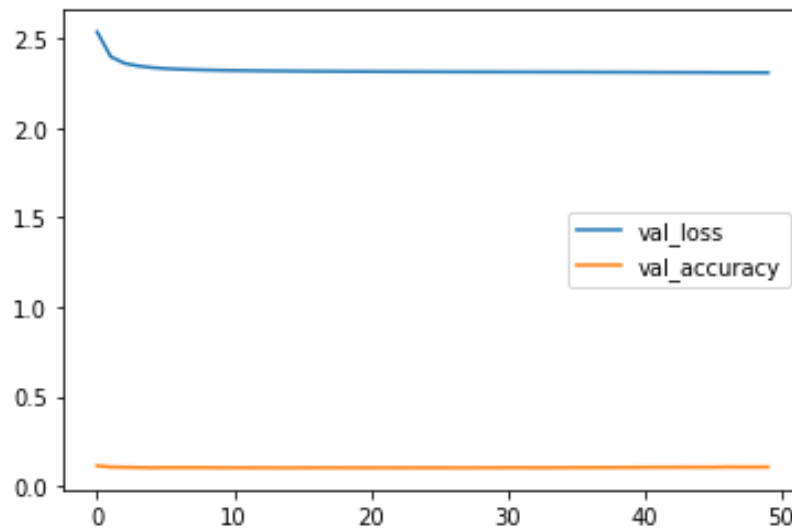
1 metrics_FFNN = pd.DataFrame(model_FFNN.history.history)
2 metrics_FFNN

   loss  accuracy  val_loss  val_accuracy
0  6.532239   0.11330   2.535487   0.1121
1  2.450019   0.11094   2.396581   0.1054
2  2.373081   0.10790   2.359746   0.1040
3  2.347943   0.10644   2.344705   0.1030
4  2.335569   0.10568   2.335771   0.1016
5  2.327861   0.10480   2.330216   0.1023
6  2.322760   0.10426   2.326642   0.1024
7  2.319282   0.10416   2.324337   0.1024
8  2.316667   0.10390   2.322378   0.1022
9  2.314572   0.10354   2.320843   0.1017
10 2.312843   0.10338   2.319582   0.1014
```

➤ Plot shows the Loss vs Accuracy for Feed forward Neural network (FFNN).



- Plot shows the val_loss vs val_accuracy for Feed forward Neural network (FFNN).



- Recurrent Neural Network (RNN) for CIFAR 10 dataset.

```
1 model = Sequential()
2 model.add(Input(shape=(32,32,3)))
3 model.add(Flatten())
4 model.add(Reshape((1, 3072)))
5 model.add(SimpleRNN(1024, activation="relu"))
6 model.add(Reshape((1, 1024)))
7 model.add(SimpleRNN(1024, activation = "relu"))
8 model.add(Reshape((1, 1024)))
9 model.add(SimpleRNN(1024, activation = "relu"))
10 model.add(Dense(10, activation = "softmax"))
11
12 print(model.summary())
13 ##
14 opt = SGD(learning_rate = 0.000001, momentum=0.9)
15 model.compile(loss = "categorical_crossentropy", optimizer = opt, metrics = ["accuracy"])
```

Layer (type)	Output Shape	Param #
(None, 3072)		0

Saved successfully!

0s completed at 00:28

2211MC21_Assignment_04.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Non-trainable params: 0
None

```
1 model.fit(X_train, y_cat_train, validation_data = (X_test, y_cat_test), epochs = 50, batch_size = 128)
```

Epoch	391/391	Time	loss	accuracy	val_loss	val_accuracy
Epoch 1/50	391/391	6s 11ms/step	20.3938	0.2211	13.1852	0.2512
Epoch 2/50	391/391	4s 10ms/step	11.7402	0.2709	10.4716	0.2833
Epoch 3/50	391/391	4s 10ms/step	9.7873	0.2895	9.2023	0.2891
Epoch 4/50	391/391	4s 10ms/step	8.5399	0.3007	8.2649	0.3020
Epoch 5/50	391/391	4s 10ms/step	7.6817	0.3080	7.6888	0.3089
Epoch 6/50	391/391	4s 10ms/step	7.0534	0.3140	7.1573	0.3119
Epoch 7/50	391/391	4s 10ms/step	6.5126	0.3213	6.6656	0.3181
Epoch 8/50	391/391	4s 10ms/step	6.0973	0.3253	6.3049	0.3155
Epoch 9/50	391/391	4s 10ms/step	5.7527	0.3298	6.0040	0.3247
Epoch 10/50	391/391	4s 10ms/step	5.4501	0.3325	5.7097	0.3210
Epoch 11/50	391/391	4s 9ms/step	5.1831	0.3360	5.5366	0.3240

0s completed at 00:28

2211MC21_Assignment_04.ipynb

File Edit View Insert Runtime Tools Help All changes saved

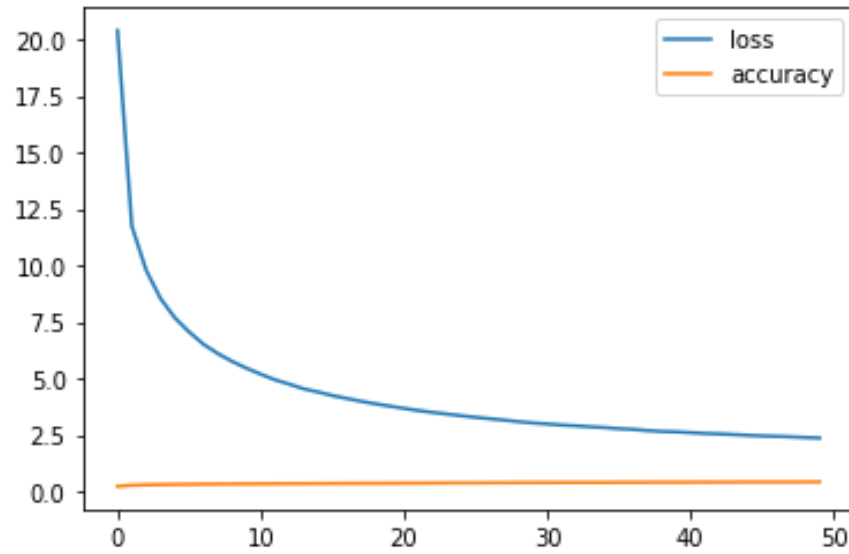
Non-trainable params: 0
None

```
1 model.fit(X_train, y_cat_train, validation_data = (X_test, y_cat_test), epochs = 50, batch_size = 128)
```

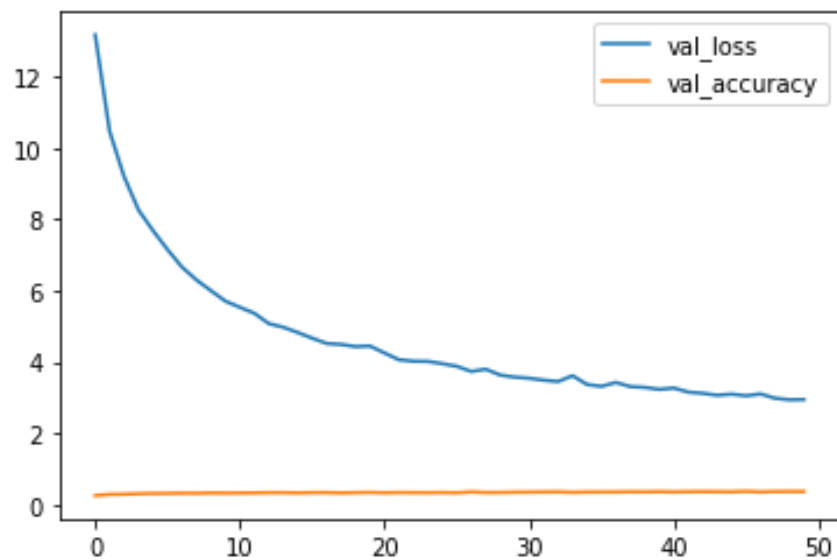
Epoch	391/391	Time	loss	accuracy	val_loss	val_accuracy
Epoch 1/50	391/391	6s 11ms/step	20.3938	0.2211	13.1852	0.2512
Epoch 2/50	391/391	4s 10ms/step	11.7402	0.2709	10.4716	0.2833
Epoch 3/50	391/391	4s 10ms/step	9.7873	0.2895	9.2023	0.2891
Epoch 4/50	391/391	4s 10ms/step	8.5399	0.3007	8.2649	0.3020
Epoch 5/50	391/391	4s 10ms/step	7.6817	0.3080	7.6888	0.3089
Epoch 6/50	391/391	4s 10ms/step	7.0534	0.3140	7.1573	0.3119
Epoch 7/50	391/391	4s 10ms/step	6.5126	0.3213	6.6656	0.3181
Epoch 8/50	391/391	4s 10ms/step	6.0973	0.3253	6.3049	0.3155
Epoch 9/50	391/391	4s 10ms/step	5.7527	0.3298	6.0040	0.3247
Epoch 10/50	391/391	4s 10ms/step	5.4501	0.3325	5.7097	0.3210
Epoch 11/50	391/391	4s 9ms/step	5.1831	0.3360	5.5366	0.3240

0s completed at 00:28

➤ Plot shows the Loss vs Accuracy for Feed forward Neural network (FFNN).



- Plot shows the val_loss vs val_accuracy for Feed forward Neural network (FFNN).



Inference from Results:

- For the Feed forward neural Network (FFNN), training the model for 50 epoch produces very less accuracy and slightly high error for the CIFAR 10 dataset.
- While for the Recurrent Neural Network (RNN), Loss Vs accuracy plot shows sufficiently better accuracy and less error for CIFAR 10 dataset.
- RNN works better in FFNN in general because of the high representational power of RNN than FFNN, as RNN contains more no of neurons for computations as compared to FFNN. This is also the reason that the accuracy for RNN is much higher than in case of FFNN.