Machine Learning and Optimization Methods - IT3071

Assignment 1 - Practical test

IT NUMBER: IT22295538
NAME: I.L.M. BALASOORIYA
BATCH: DS.WE.0201

1.introduction

In order to address a multi-class classification problem, I created and built a deep learning model in this project utilizing TensorFlow's Keras API. The objective was to precisely categorize the data using 60 features into three categories: 'L', 'M', and 'H'. The model's performance was optimized by experimenting with different architectures and hyperparameters.

2. Data Preparation

I started by preprocessing the 480 rows and 17 columns that made up the dataset:

I started by looking over the data frame to see if there were any columns that were missing or had null values.

Encoding categorical variables: For categorical columns with more than two unique values per column, I employed one-hot encoding. used the label encoding, which had two distinct values, after that. Because of the classification, I specifically utilized label encoding for the Class column.

Columns dropping, I let go of the dummy One Way to Prevent Multicollinearity

Numerical feature scaling: To make sure that features had the same scale, MinMax scaling was used.

20% of the dataset was set aside for testing and the remaining 80% for training.

IT22295538.ipynb

```
[21] data2.sample(5)
```

edhands	VisITedResources	AnnouncementsView	Discussion	ParentAnsweringSurvey	ParentschoolSatisfaction	StudentAbsenceDays	...	Topic_Chemistry
28	60	19	50	1	1	0	...	False
70	92	50	7	1	1	0	...	False
72	80	58	66	0	0	0	...	False
80	80	51	59	1	1	0	...	False
72	51	42	24	1	0	1	...	False

```
[22] data2.shape
```

```
(480, 67)
```

```
[24] for col in data2 :
       print(f'{col}\t\t{data2[col].unique()}') #use this loop to check, is there any non numeric values are still in the dataframe
```

```
StudentAbsenceDays              [0 1]
Class           [0 1 2]
NationalITy_Egypt               [False  True]
NationalITy_Iran                [False  True]
NationalITy_Iraq                [False  True]
NationalITy_Jordan              [False  True]
NationalITy_KW          [ True False]
NationalITy_Lybia               [False  True]
NationalITy_Morocco             [False  True]
```

IT22295538.ipynb

```
data.isnull().sum() #check whethere there are any null values
```

	0
gender	0
NationalITy	0
PlaceofBirth	0
StageID	0
GradeID	0
SectionID	0
Topic	0
Semester	0
Relation	0
raisedhands	0
VisITedResources	0
AnnouncementsView	0
Discussion	0
ParentAnsweringSurvey	0
ParentschoolSatisfaction	0
StudentAbsenceDays	0

1m 34s completed at 6:34 PM

Welcome To Colab - Colab | IT22295538.ipynb - Colab

colab.research.google.com/drive/1lSoGlw6wdYAFoPY4l0DFHbLJPfk387yZ#scrollTo=KWhjrO78Cz1s

IT22295538.ipynb

File Edit View Insert Runtime Tools Help

Files

sample_data

academic_dataset.csv

Disk 74.87 GB available

```
[6]  ParentschoolSatisfaction  0
     StudentAbsenceDays        0
     Class                     0

dtype: int64
```

```python
for col in data :
  print(f'{col}\t\t{data[col].unique()}')  # check and print unique values per column
```

```
gender          ['M' 'F']
NationalITy             ['KW' 'lebanon' 'Egypt' 'SaudiArabia' 'USA' 'Jordan' 'venzuela' 'Iran'
 'Tunis' 'Morocco' 'Syria' 'Palestine' 'Iraq' 'Lybia']
PlaceofBirth            ['KuwaIT' 'lebanon' 'Egypt' 'SaudiArabia' 'USA' 'Jordan' 'venzuela' 'Iran'
 'Tunis' 'Morocco' 'Syria' 'Iraq' 'Palestine' 'Lybia']
StageID         ['lowerlevel' 'MiddleSchool' 'HighSchool']
GradeID         ['G-04' 'G-07' 'G-08' 'G-06' 'G-05' 'G-09' 'G-12' 'G-11' 'G-10' 'G-02']
SectionID               ['A' 'B' 'C']
Topic           ['IT' 'Math' 'Arabic' 'Science' 'English' 'Quran' 'Spanish' 'French'
 'History' 'Biology' 'Chemistry' 'Geology']
Semester                ['F' 'S']
Relation                ['Father' 'Mum']
raisedhands             [ 15  20  10  30  40  42  35  50  12  70  19   5  62  36  55  69  60   2
   0   8  25  75   4  45  14  33   7  13  29  39  49  16  28  27  21  80
  17  65  22  11   1   3 100   6  90  77  24  66  23  82  72  51  85  87
  95  81  53  92  83  67  96  57  73   9  32  52  59  61  79  18  74  97
  41  71  98  78  89  88  86  76  99  84]
VisITedResources                [16 20  7 25 50 30 12 10 21 80 88  6  1 14 70 40 13 15 60  0  2 19 85 90
  5 22 11 54 35 33  4 39 75 69  3  8 89 44 92 26 27 29 98  9 42 65 79 55
 63 91 51 58 68 82 72 52 62 71 66 43 95 31 41 81 61 83 84 17 94 48 86 74
 76 97 87 99 34 64 28 38 36 24 59 57 77 18 93 96 78]
AnnouncementsView               [ 2  3  0  5 12 13 15 16 25 30 19 44 22 20 35 36 40 33  4 52 50 10  9  8
```

✓ 1m 34s completed at 6:34 PM

Welcome To Colab - Colab | IT22295538.ipynb - Colab

colab.research.google.com/drive/1lSoGlw6wdYAFoPY4l0DFHbLJPfk387yZ#scrollTo=KWhjrO78Cz1s

IT22295538.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

sample_data

academic_dataset.csv

Disk 74.87 GB available

```
 79 21 31 28 38 48 97 98 63 72 82 71 45 68 92 58 57 62]
ParentAnsweringSurvey           ['Yes' 'No']
ParentschoolSatisfaction                ['Good' 'Bad']
StudentAbsenceDays              ['Under-7' 'Above-7']
Class           ['M' 'L' 'H']
```

```python
[12] #Label encoding

data.gender.replace({'F':0, 'M':1}, inplace=True)
data.Semester.replace({'F':0, 'S':1}, inplace=True)
data.Relation.replace({'Father':0, 'Mum':1}, inplace=True)
data.ParentAnsweringSurvey.replace({'No':0, 'Yes':1}, inplace=True)
data.ParentschoolSatisfaction.replace({'Bad':0, 'Good':1}, inplace=True)
data.StudentAbsenceDays.replace({'Under-7':0, 'Above-7':1}, inplace=True)
data.Class.replace({'M':0,'L':1, 'H':2}, inplace=True)


#from sklearn.preprocessing import LabelEncoder
#le = LabelEncoder()
#data['Class'] = le.fit_transform(data['Class'])
#data['Semester']=le.fit_transform(data['Semester'])
#data['gender']=le.fit_transform(data['gender'])
#data['Relation']=le.fit_transform(data['Relation'])
#data['ParentAnsweringSurvey']=le.fit_transform(data['ParentAnsweringSurvey'])
#data['ParentschoolSatisfaction']=le.fit_transform(data['ParentschoolSatisfaction'])
#data['StudentAbsenceDays']=le.fit_transform(data['StudentAbsenceDays'])
```

```python
[13] data.sample(5)
```

```
gender NationalITy PlaceofBirth StageID GradeID SectionID Topic Semester Relation raisedhands VisITedResources AnnouncementsV
```

✓ 1m 34s completed at 6:34 PM

Welcome To Colab - Colab | IT22295538.ipynb - Colab

colab.research.google.com/drive/1lSoGlw6wdYAFoPY4l0DFHbLJPfk387yZ#scrollTo=KWhjrO78Cz1s

IT22295538.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

Files

sample_data

academic_dataset.csv

Disk 74.87 GB available

+ Code + Text

[13]

86	1	SaudiArabia	SaudiArabia	lowerlevel	G-02	B	IT	0	0	70	12

[20]
```
#one-hot encoding
data2 = pd.get_dummies(data, columns=['NationalITy', 'PlaceofBirth','StageID','GradeID','SectionID','Topic'])
```

```
data2.sample(5)
```

edhands	VisITedResources	AnnouncementsView	Discussion	ParentAnsweringSurvey	ParentschoolSatisfaction	StudentAbsenceDays	...	Topic_Chemistry
28	60	19	50	1	1	0	...	False
70	92	50	7	1	1	0	...	False
72	80	58	66	0	0	0	...	False
80	80	51	59	1	1	0	...	False
72	51	42	24	1	0	1	...	False

[22]
```
data2.shape
```
```
(480, 67)
```

[24]
```
for col in data2 :
  print(f'{col}\t\t{data2[col].unique()}') #use this loop to check, is there any non numeric values are still in the dataframe
```

✓ 1m 34s completed at 6:34 PM

Welcome To Colab - Colab | IT22295538.ipynb - Colab

colab.research.google.com/drive/1lSoGlw6wdYAFoPY4l0DFHbLJPfk387yZ#scrollTo=KWhjrO78Cz1s

IT22295538.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

Files

sample_data

academic_dataset.csv

+ Code + Text

[24]
```
Topic_History          [False  True]
Topic_IT               [ True False]
Topic_Math             [False  True]
Topic_Quran            [False  True]
Topic_Science          [False  True]
Topic_Spanish          [False  True]
```

[25]
```
#Drop One Dummy Variable to Avoid Multicollinearity

data2.drop(['NationalITy_Egypt','PlaceofBirth_Iran','StageID_MiddleSchool','GradeID_G-08','SectionID_A','Topic_History'], axis='columns', inplac
```

```
data2.sample(5)
```

	gender	Semester	Relation	raisedhands	VisITedResources	AnnouncementsView	Discussion	ParentAnsweringSurvey	ParentschoolSatisfaction	S
79	0	0	1	80	90	49	55	1	0	
358	0	0	1	72	98	52	15	1	1	
126	0	0	0	2	9	7	55	1	1	
432	1	0	0	95	87	62	81	0	0	
338	0	0	0	78	98	10	11	0	1	

5 rows × 61 columns

[27]
```
data2.shape
```
```
(480, 61)
```

Disk 74.87 GB available

✓ 1m 34s completed at 6:34 PM

```
[27] data2.shape
```

```
(480, 61)
```

```
[29] scaller = MinMaxScaler()
```

```
[30] col_to_scale = ['raisedhands','VisITedResources','AnnouncementsView','Discussion']
```

```
#Min-max scaling transforms the data to a fixed range
data2[col_to_scale] = scaller.fit_transform(data2[col_to_scale])
```

```
[32] data2.sample(5)
```

	gender	Semester	Relation	raisedhands	VisITedResources	AnnouncementsView	Discussion	ParentAnsweringSurvey	ParentschoolSatisfaction	S
416	1	0	0	0.98	0.909091	0.877551	0.714286	1	1	
432	1	0	0	0.95	0.878788	0.632653	0.816327	0	0	
260	1	1	0	0.10	0.171717	0.122449	0.132653	0	0	
30	0	0	0	0.35	0.808081	0.510204	0.704082	1	1	
233	0	1	1	0.32	0.808081	0.591837	0.459184	1	1	

5 rows × 61 columns

✓ 1m 34s completed at 6:34 PM

```
[326] #Split the dataset into training and validation sets
      from sklearn.model_selection import train_test_split
```

```
[327] x = data2.drop('Class',axis='columns')
      y = data2.Class
```

```
x.sample(5)
```

	gender	Semester	Relation	raisedhands	VisITedResources	AnnouncementsView	Discussion	ParentAnsweringSurvey	ParentschoolSatisfaction	S
330	1	0	0	0.40	0.070707	0.510204	0.408163	0	1	
150	1	1	0	0.80	0.808081	0.520408	0.591837	1	1	
191	1	1	0	0.15	0.252525	0.377551	0.122449	1	1	
302	0	0	1	0.11	0.202020	0.214286	0.224490	0	0	
362	1	0	0	0.90	0.989899	0.418367	0.377551	1	1	

5 rows × 60 columns

Double-click (or enter) to edit

```
[329] x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2,random_state=47)
```

```
[330] x_train.shape
```

✓ 1m 34s completed at 6:34 PM

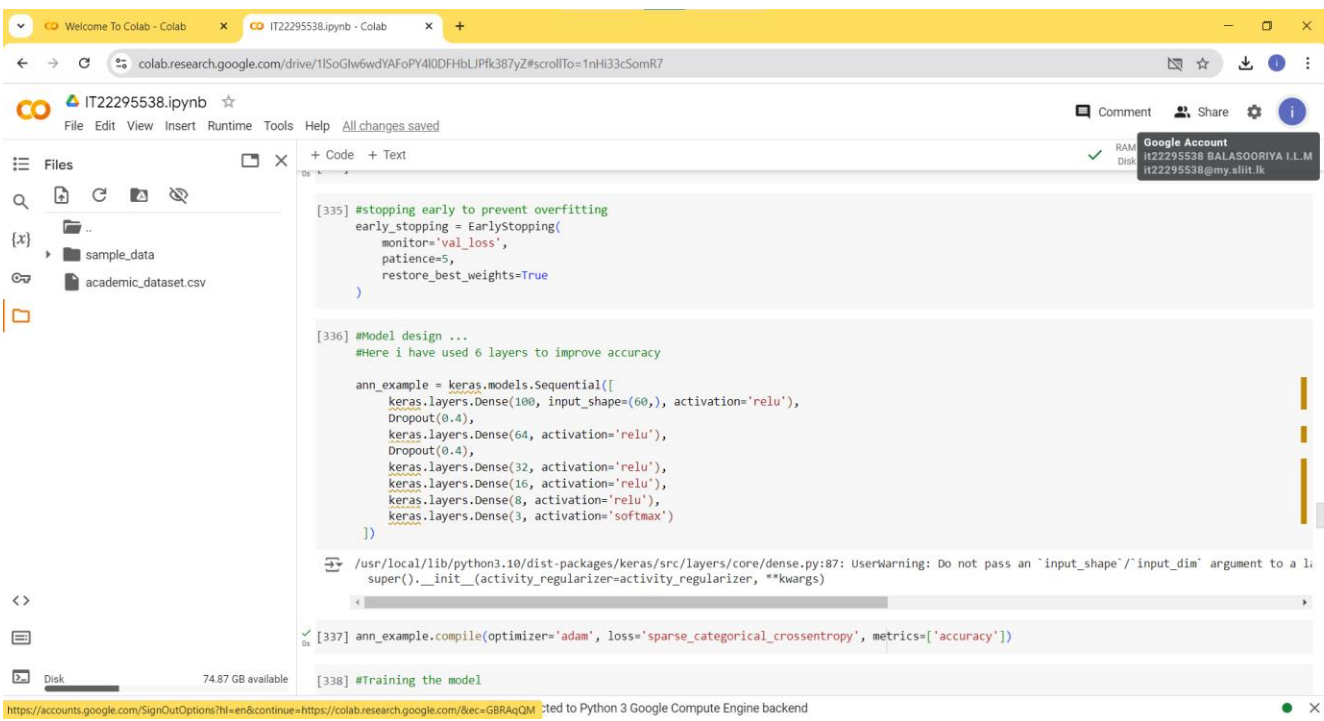
3. Model designing

I experimented with different model architectures. The final architecture consisted of:

Dense Layers: 5 layers with 'relu' activations and a 'Softmax' output layer for classification into 3 classes.

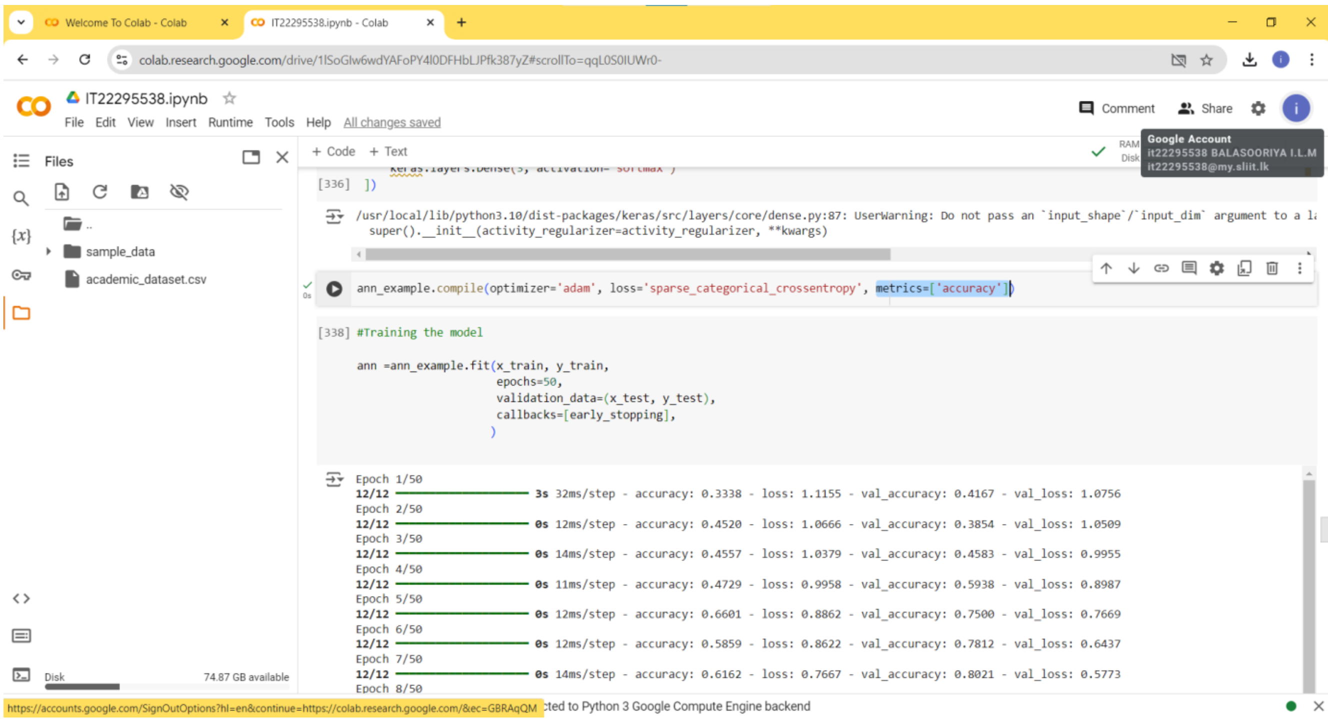
Dropout Layers: Dropout was introduced to prevent overfitting, with a rate of 0.4.

Optimizer: Adam was selected as the optimizer, and sparse categorical cross-entropy was used as the loss function and accuracy as metrics.



4. Training

Training loss and accuracy were logged during the training process, which was monitored for 50 epochs with early stopping to avoid overfitting.



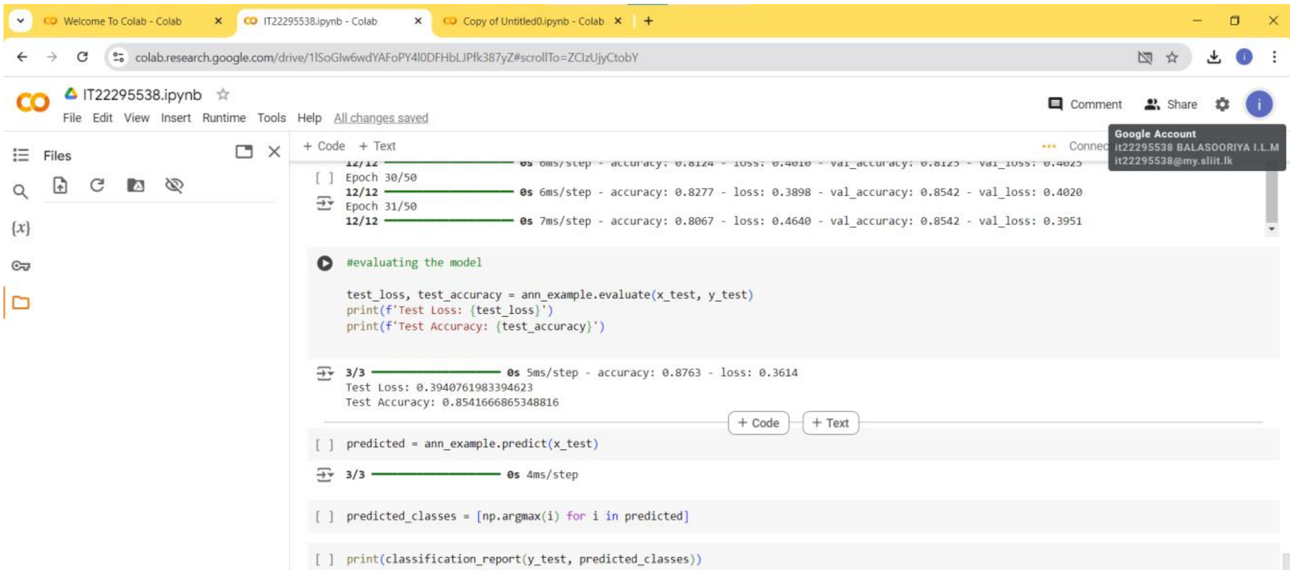
5. Evaluation

The trained model was evaluated on the test set, and the following metrics were calculated:

Test Loss: 0.3940761983394623

Test Accuracy: 0.8541666865348816

Precision, Recall, F1-score: These metrics were calculated for each class using classification_report().



```
Epoch 30/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8277 - loss: 0.3898 - val_accuracy: 0.8542 - val_loss: 0.4020
Epoch 31/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8067 - loss: 0.4640 - val_accuracy: 0.8542 - val_loss: 0.3951
```

```python
#evaluating the model

test_loss, test_accuracy = ann_example.evaluate(x_test, y_test)
print(f'Test Loss: {test_loss}')
print(f'Test Accuracy: {test_accuracy}')
```

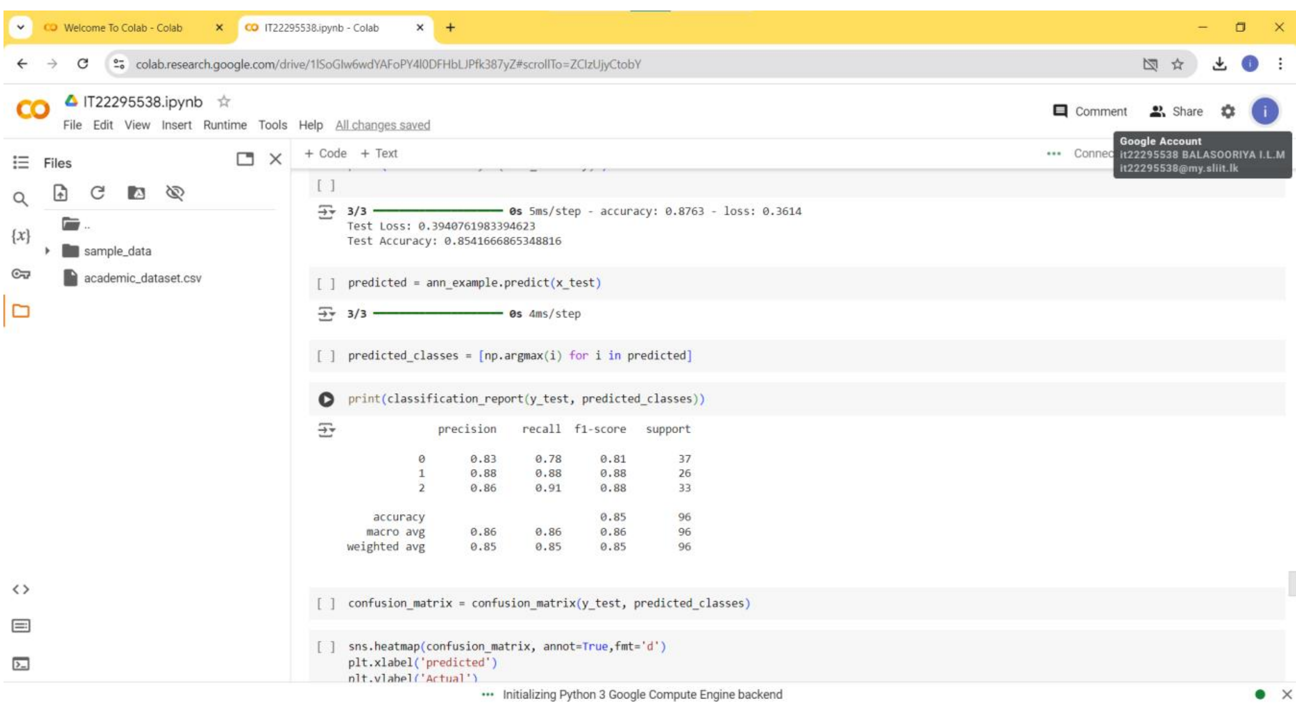
```
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8763 - loss: 0.3614
Test Loss: 0.3940761983394623
Test Accuracy: 0.8541666865348816
```

```python
predicted = ann_example.predict(x_test)
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
```

```python
predicted_classes = [np.argmax(i) for i in predicted]
```

```python
print(classification_report(y_test, predicted_classes))
```



```
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8763 - loss: 0.3614
Test Loss: 0.3940761983394623
Test Accuracy: 0.8541666865348816
```

```python
predicted = ann_example.predict(x_test)
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
```

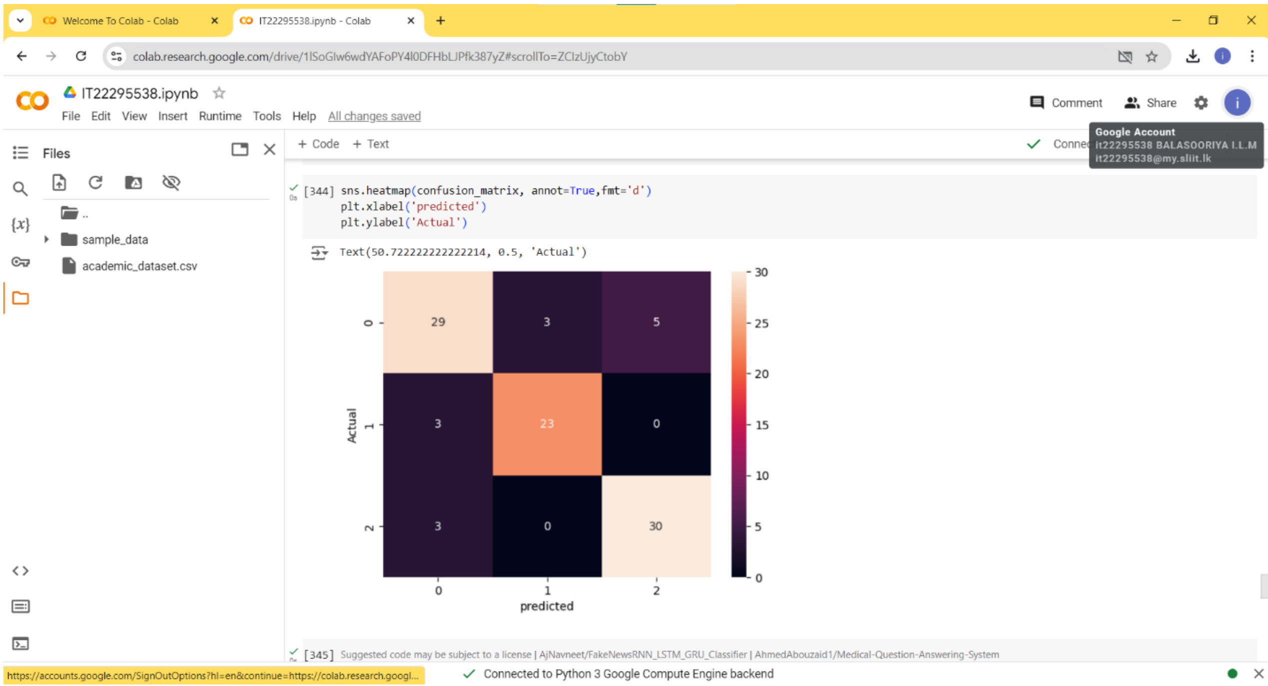
```python
predicted_classes = [np.argmax(i) for i in predicted]
```

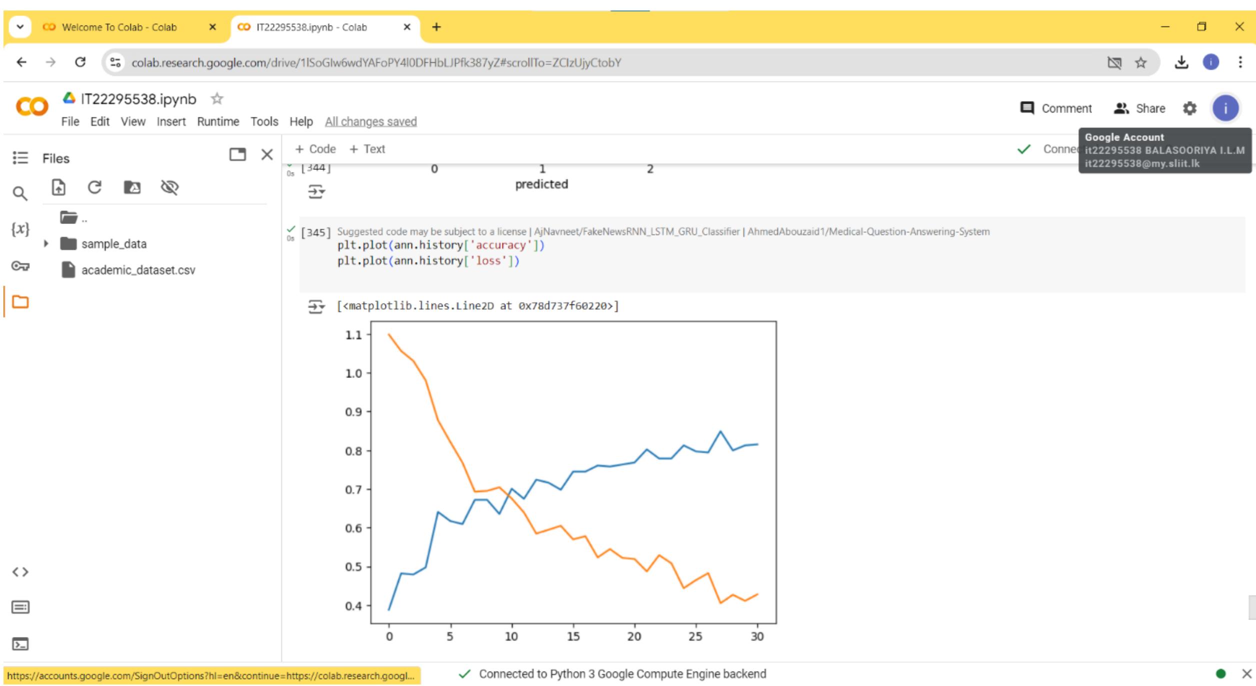
```python
print(classification_report(y_test, predicted_classes))
```

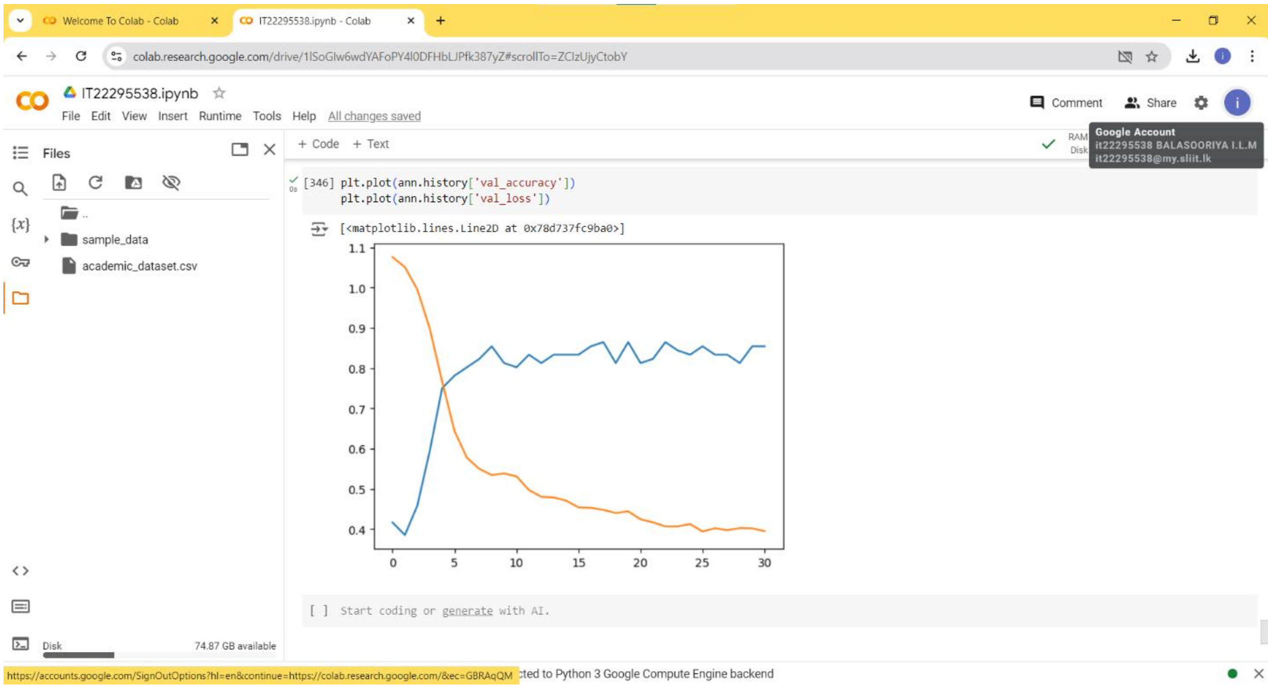
```
              precision    recall  f1-score   support

           0       0.83      0.78      0.81        37
           1       0.88      0.88      0.88        26
           2       0.86      0.91      0.88        33

    accuracy                           0.85        96
   macro avg       0.86      0.86      0.86        96
weighted avg       0.85      0.85      0.85        96
```

```python
confusion_matrix = confusion_matrix(y_test, predicted_classes)
```

```python
sns.heatmap(confusion_matrix, annot=True,fmt='d')
plt.xlabel('predicted')
plt.ylabel('Actual')
```


IT22295538.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Files
sample_data
academic_dataset.csv
[344] sns.heatmap(confusion_matrix, annot=True,fmt='d')
plt.xlabel('predicted')
plt.ylabel('Actual')
Text(50.72222222222214, 0.5, 'Actual')
29 3 5
3 23 0
3 0 30
Actual
predicted
[345] Suggested code may be subject to a license | AjNavneet/FakeNewsRNN_LSTM_GRU_Classifier | AhmedAbouzaid1/Medical-Question-Answering-System
Connected to Python 3 Google Compute Engine backend


IT22295538.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Files
sample_data
academic_dataset.csv
[345] Suggested code may be subject to a license | AjNavneet/FakeNewsRNN_LSTM_GRU_Classifier | AhmedAbouzaid1/Medical-Question-Answering-System
plt.plot(ann.history['accuracy'])
plt.plot(ann.history['loss'])
[<matplotlib.lines.Line2D at 0x78d737f60220>]
Connected to Python 3 Google Compute Engine backend



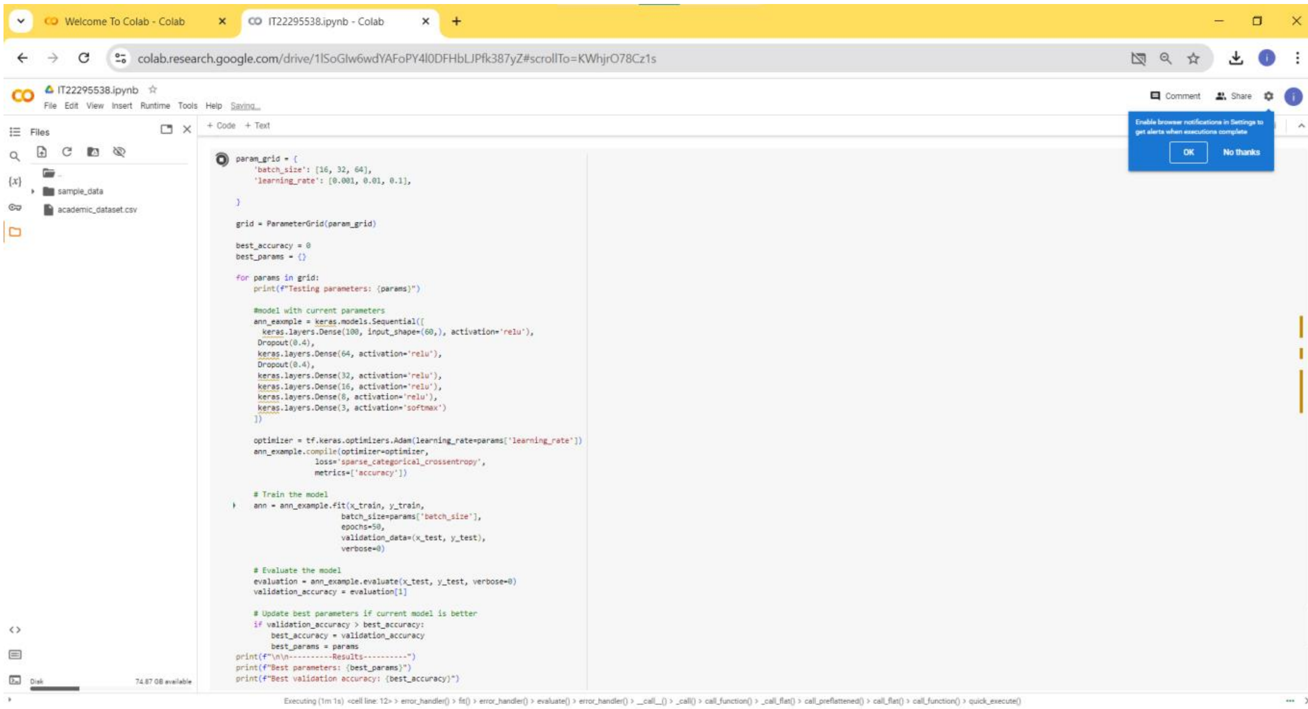
6. Hyperparameter Tuning

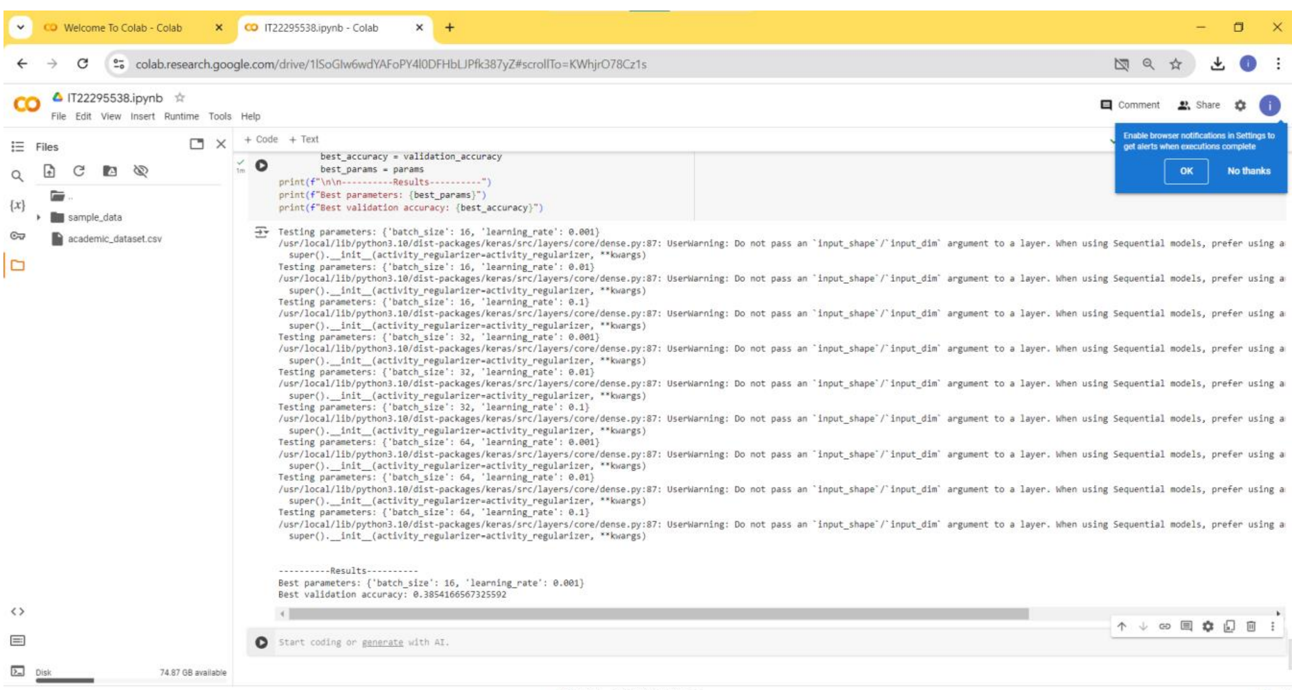
To further optimize performance, I experimented with different hyperparameters such as:

Batch size: 16, 32, 64

Learning rate: 0.001, 0.01,0.1

The best results were achieved with a batch size of 16 and a learning rate of 0.001.


Welcome To Colab - Colab
IT22295538.ipynb - Colab
colab.research.google.com/drive/1lSoGlw6wdYAFoPY4l0DFHbLJPfk387yZ#scrollTo=KWhjrO78Cz1s
IT22295538.ipynb
File Edit View Insert Runtime Tools Help Saving...
Comment Share
Enable browser notifications in Settings to get alerts when executions complete
OK No thanks
Files
sample_data
academic_dataset.csv
param_grid = {
'batch_size': [16, 32, 64],
'learning_rate': [0.001, 0.01, 0.1],
}
grid = ParameterGrid(param_grid)
best_accuracy = 0
best_params = {}
for params in grid:
print(f"Testing parameters: {params}")
#model with current parameters
ann_eaxmple = keras.models.Sequential([
keras.layers.Dense(100, input_shape=(60,), activation='relu'),
Dropout(0.4),
keras.layers.Dense(64, activation='relu'),
Dropout(0.4),
keras.layers.Dense(32, activation='relu'),
keras.layers.Dense(16, activation='relu'),
keras.layers.Dense(8, activation='relu'),
keras.layers.Dense(3, activation='softmax')
])
optimizer = tf.keras.optimizers.Adam(learning_rate=params['learning_rate'])
ann_example.compile(optimizer=optimizer,
loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
# Train the model
ann = ann_example.fit(x_train, y_train,
batch_size=params['batch_size'],
epochs=50,
validation_data=(x_test, y_test),
verbose=0)
# Evaluate the model
evaluation = ann_example.evaluate(x_test, y_test, verbose=0)
validation_accuracy = evaluation[1]
# Update best parameters if current model is better
if validation_accuracy > best_accuracy:
best_accuracy = validation_accuracy
best_params = params
print(f"\n\n----------Results----------")
print(f"Best parameters: {best_params}")
print(f"Best validation accuracy: {best_accuracy}")
Executing (1m 1s) <cell line: 12> > error_handler() > fit() > error_handler() > evaluate() > error_handler() > __call__() > _call() > call_function() > _call_flat() > call_preflattened() > call_flat() > call_function() > quick_execute()


Welcome To Colab - Colab
IT22295538.ipynb - Colab
colab.research.google.com/drive/1lSoGlw6wdYAFoPY4l0DFHbLJPfk387yZ#scrollTo=KWhjrO78Cz1s
IT22295538.ipynb
File Edit View Insert Runtime Tools Help
Comment Share
Enable browser notifications in Settings to get alerts when executions complete
OK No thanks
Files
sample_data
academic_dataset.csv
best_accuracy = validation_accuracy
best_params = params
print(f"\n\n----------Results----------")
print(f"Best parameters: {best_params}")
print(f"Best validation accuracy: {best_accuracy}")
Testing parameters: {'batch_size': 16, 'learning_rate': 0.001}
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using a
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Testing parameters: {'batch_size': 16, 'learning_rate': 0.01}
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using a
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Testing parameters: {'batch_size': 16, 'learning_rate': 0.1}
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using a
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Testing parameters: {'batch_size': 32, 'learning_rate': 0.001}
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using a
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Testing parameters: {'batch_size': 32, 'learning_rate': 0.01}
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using a
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Testing parameters: {'batch_size': 32, 'learning_rate': 0.1}
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using a
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Testing parameters: {'batch_size': 64, 'learning_rate': 0.001}
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using a
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Testing parameters: {'batch_size': 64, 'learning_rate': 0.01}
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using a
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Testing parameters: {'batch_size': 64, 'learning_rate': 0.1}
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using a
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
----------Results----------
Best parameters: {'batch_size': 16, 'learning_rate': 0.001}
Best validation accuracy: 0.3854166567325592
Start coding or generate with AI.
1m 41s completed at 7:02 PM

7. Findings

Overfitting was mitigated using dropout layers and early stopping, which resulted in smoother convergence of the model during training.

8. Challenges Faced

Choosing the right architecture: Experimenting with different architectures and hyperparameters required time to find an optimal solution.

Early stopping: Fine-tuning early stopping criteria took several trials to prevent underfitting while ensuring good generalization.

9. Lessons Learned

Dropout and early stopping are effective techniques to combat overfitting.

Hyperparameter tuning can significantly impact model performance, and automating this process using grid search or random search can save time.

Monitoring both loss and accuracy metrics is crucial to assess model performance over time.