CS5590 APL

-

Python Programming /Deep Learning

LAB2

Team Members:

Ahmed Siddiqui (30)

Vamsi Draksharam (14)

Maseerah Muradabadi (23)

Introduction: -

The assignment is targeted to cover D/L concepts and trying hands on neural network models with different dataset using KERAS.

Objectives: -

- Creation of regression and evaluating the model with MAE and Loss score.
- Creation of CNN models
- Training text classification & image classification model with different dataset and check the accuracy score for each model.
- Creating LSTM model for text classification.
- Comparing CNN and LSTM model for text classification

Methods: -

- Created notebooks in GoogleColab.
- Downloaded the dataset from Kaggle.
- Trained model with that dataset.
- Plotted using TensorBoard.

Workflow: -

Problem 1 Statement:

1. Build a Sequential model using keras to implement Linear Regression with any data set of your choice except the datasets being discussed in the class or used before

a. Show the graph on TensorBoard
b. Plot the loss and then change the below parameter and report your view how the result changes in each case
   a. learning rate
   b. batch size
   c. optimizer
   d. activation function

Objective:

The objective for the first problem is analysing the Linear Regression Model on any Sample given Dataset.
Showing the graph in TensorBoard Plot the loss and then change the below parameter to see how result changes in each case a. learning rate b. batch size c. optimizer d. activation function

Approach:

To implement Linear Regression model we have taken Abalone Dataset from UCI Data Repository. Using this dataset we can predict the age of abalone from physical measurement. We have used the various Deep Learning libraries and packages like Keras Tensorboard, optimizer and Activation functions to evaluate the model.
More details are given in workflow section.

Output/ Workflow:

A. Loading the dataset and creating the Panda dataframe.

+ Code + Text

```
# Linear Regression using Abalone dataset using Keras
import os
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense, Dropout,LeakyReLU
from keras import metrics
import matplotlib.pyplot as plt
from keras.optimizers import Adam, RMSprop
from keras.callbacks import TensorBoard

#Dataframe is created using abalone.csv
abalone_Data = pd.read_csv('abalone.csv')
```

B. Created the Age Column in abalone data panda dataframe.

```
# We need to calculate Age , lets first compute the  'Age' and assign it to dataset abalone_Data.
abalone_Data['Age'] = abalone_Data['Rings']+1.5
abalone_Data.drop('Rings', axis=1, inplace=True)
```

C. Feature Statistic on given dataset

```
# Feature wise statistics using builtin tools
print(abalone_Data.columns)
print(abalone_Data.head())
print(abalone_Data.info())
print(abalone_Data.describe())
```

D. Creating X,Y train and validation dataframe

```
#There are no Missing values and all Feature are numeric except sex
# Creating X and y
feature_column = ['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight', 'Viscera weight', 'Shell weight']
X = abalone_Data[feature_column]
y = abalone_Data['Age']
X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.30, random_state=42)
print(X_train.head())
print(y_valid.head())
np.random.seed(155)
```

E. Normalizing the train and validation dataframe.

```
# Normalization
def norm_s(df1, df2):
    dfs = df1.append(df2)
    min = np.min(dfs)
    max = np.max(dfs)
    me = np.mean(dfs)
    sigma = np.std(dfs)
    return (min, max, me, sigma)

def z_score(col, stats):
    m, M, me, s = stats
    df2 = pd.DataFrame()
    for c in col.columns:
        df2[c] = (col[c]-me[c])/s[c]
    return df2

stats = norm_s(X_train, X_valid)
arr_x_train = np.array(z_score(X_train, stats))
arr_y_train = np.array(y_train)
arr_x_valid = np.array(z_score(X_valid, stats))
arr_y_valid = np.array(y_valid)
print('Training shape:', arr_x_train.shape)
print('Validation',arr_y_train.shape)
print('Training samples: ', arr_x_train.shape[0])
print('Validation samples: ', arr_x_valid.shape[0])
```

F. Model Definition

```
# Defining the Model
def model(x_size, y_size):
    t_model = Sequential()
    t_model.add(Dense(100, activation="tanh", input_shape=(x_size,)))
    t_model.add(Dropout(0.1))
    t_model.add(Dense(50, activation="relu"))
    t_model.add(Dense(20, activation="relu"))
    t_model.add(Dense(y_size))
    t_model.compile(loss='mean_squared_error', optimizer=RMSprop(lr=0.004), metrics=[metrics.mae])
    return t_model
model = model(arr_x_train.shape[1], 1)
model.summary()
```

G. Defining the Number Epoch , batch size and defining tensorboard logic for graph.

```
# Batch Size and Epoch
epochs = 100
batch_size = 128

# Tensoboard Logic
LOG_DIR = os.getcwd()
tensorboard = TensorBoard(log_dir='LOG_DIR', histogram_freq=0,
                          write_graph=True, write_images=True)
```

H. Fit the Model and calculating the Train and validation score

```
# Fitting the Model
history = model.fit(arr_x_train, arr_y_train, batch_size=batch_size, epochs=epochs, shuffle=True, verbose=2,
                    callbacks=[tensorboard], validation_data=(arr_x_valid, arr_y_valid),)
train_score = model.evaluate(arr_x_train, arr_y_train, verbose=0)
valid_score = model.evaluate(arr_x_valid, arr_y_valid, verbose=0)
print('Train MAE: ', round(train_score[1], 4), ', Train Loss: ', round(train_score[0], 4))
print('Val MAE: ', round(valid_score[1], 4), ', Val Loss: ', round(valid_score[0], 4))
print(os.getcwd())
```

I.At the last we have plotted the loss graph

```
# Logic for Plotting the Loss
def plot_loss(h):
    plt.figure()
    plt.plot(h['loss'])
    plt.plot(h['val_loss'])
    plt.title('Training vs Validation Loss')
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Validation'])
    plt.draw()
    plt.show()
    return
plot_loss(history.history)
```

Evaluation:

In order to evaluate the we run our program and get the train loss statistics. As we can see below for RMSprop optimizer, epoch=800 and batch size=128 and learning rate of 0.004 we got the minimum train loss.

Changed the optimizer ,batch size activation function and learning rate and evaluated the train loss as shown below.

	Optimizer	Learning Rate	Batch Size	Activation Function	Epoch	Train Loss	Val Loss	Train Mean Absolute Error	Val Mean Absolute Error
2	Adam	0.002	130	sigmod,relu	200	3.9185	4.5463	1.4161	1.4906
3	Adam	0.001	140	tanh,sigmoid	400	3.8472	4.6826	1.3933	1.502
4	Adam	0.003	128	tanh,relu,	600	3.0858	5.1478	1.312	1.6404
5	Adam	0.004	120	tanh,relu,PReLU	800	2.5147	5.6156	1.157	1.6487
6									
7	Optimizer	Learning Rate	Batch Size	Activation Function	Epoch	Train Loss	Val Loss	Train Mean Absolute Error	Val Mean Absolute Error
8	RMSprop	0.001	120	sigmod,relu	200	4.3324	4.5158	1.5116	1.5024
9	RMSprop	0.002	140	tanh,sigmoid	400	3.8254	5.2734	1.3809	1.5752
10	RMSprop	0.003	130	tanh,relu	600	3.2284	5.5304	1.3286	1.6822
11	RMSprop	0.004	128	tanh,relu,PReLU	800	2.7559	6.3351	1.2245	1.7231
12									

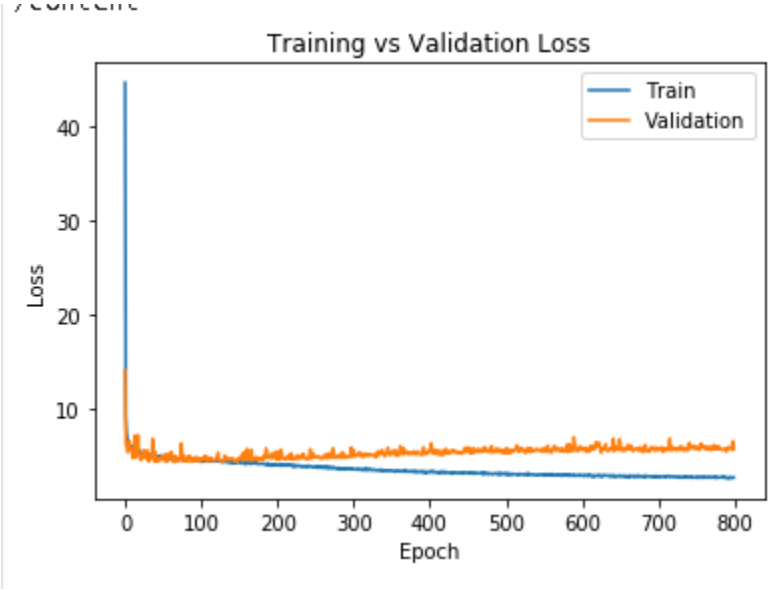
Other output :

```
Index(['Sex', 'Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
       'Viscera weight', 'Shell weight', 'Age'],
      dtype='object')
  Sex  Length  Diameter  ...  Viscera weight  Shell weight   Age
0   M   0.455     0.365  ...          0.1010         0.150  16.5
1   M   0.350     0.265  ...          0.0485         0.070   8.5
2   F   0.530     0.420  ...          0.1415         0.210  10.5
3   M   0.440     0.365  ...          0.1140         0.155  11.5
4   I   0.330     0.255  ...          0.0395         0.055   8.5

[5 rows x 9 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
Sex               4177 non-null object
Length            4177 non-null float64
Diameter          4177 non-null float64
Height            4177 non-null float64
Whole weight      4177 non-null float64
Shucked weight    4177 non-null float64
Viscera weight    4177 non-null float64
Shell weight      4177 non-null float64
Age               4177 non-null float64
dtypes: float64(8), object(1)
memory usage: 293.8+ KB
None
            Length     Diameter  ...  Shell weight          Age
count  4177.000000  4177.000000  ...   4177.000000  4177.000000
mean      0.523992     0.407881  ...      0.238831    11.433684
```

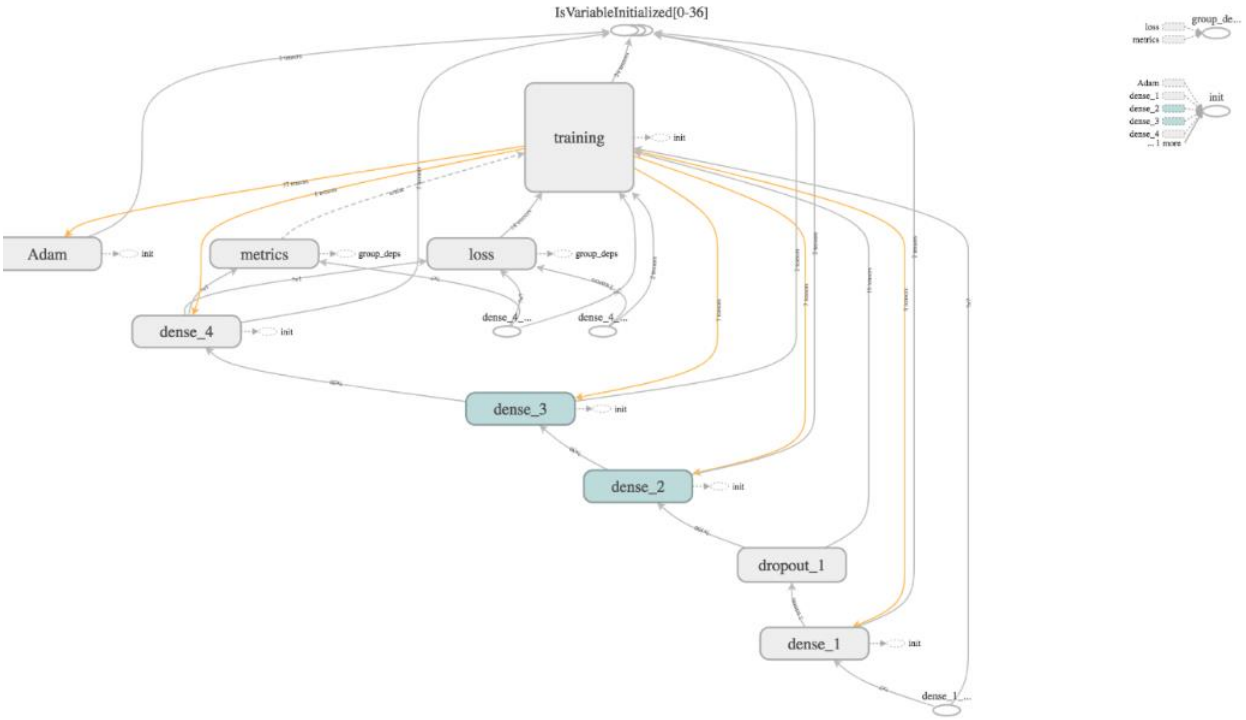
```
None
            Length     Diameter  ...  Shell weight          Age
count  4177.000000  4177.000000  ...   4177.000000  4177.000000
mean      0.523992     0.407881  ...      0.238831    11.433684
std       0.120093     0.099240  ...      0.139203     3.224169
min       0.075000     0.055000  ...      0.001500     2.500000
25%       0.450000     0.350000  ...      0.130000     9.500000
50%       0.545000     0.425000  ...      0.234000    10.500000
75%       0.615000     0.480000  ...      0.329000    12.500000
max       0.815000     0.650000  ...      1.005000    30.500000

[8 rows x 8 columns]
      Length  Diameter  Height  ...  Shucked weight  Viscera weight  Shell weight
2830   0.525     0.430   0.135  ...          0.4325          0.1800        0.1815
925    0.430     0.325   0.100  ...          0.1575          0.0825        0.1050
3845   0.455     0.350   0.105  ...          0.1625          0.0970        0.1450
547    0.205     0.155   0.045  ...          0.0170          0.0055        0.0155
2259   0.590     0.465   0.160  ...          0.5060          0.2525        0.2950

[5 rows x 7 columns]
866     10.5
1483     9.5
599     17.5
1702    10.5
670     15.5
Name: Age, dtype: float64
Training shape: (2923, 7)
Validation (2923,)
Training samples:  2923
Validation samples:  1254
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 100)               800
_________________________________________________________________
dropout_1 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_2 (Dense)              (None, 50)                5050
_________________________________________________________________
dense_3 (Dense)              (None, 20)                1020
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 21
=================================================================
Total params: 6,891
Trainable params: 6,891
Non-trainable params: 0
```

Loss Graph:



Tensorboard Graph:



Conclusion:

Using the evaluation above we can see that the RMSprop gives the better training loss as compared to Adam optimizer and also Epoch and learning rate plays very significant role in same.

Program-2

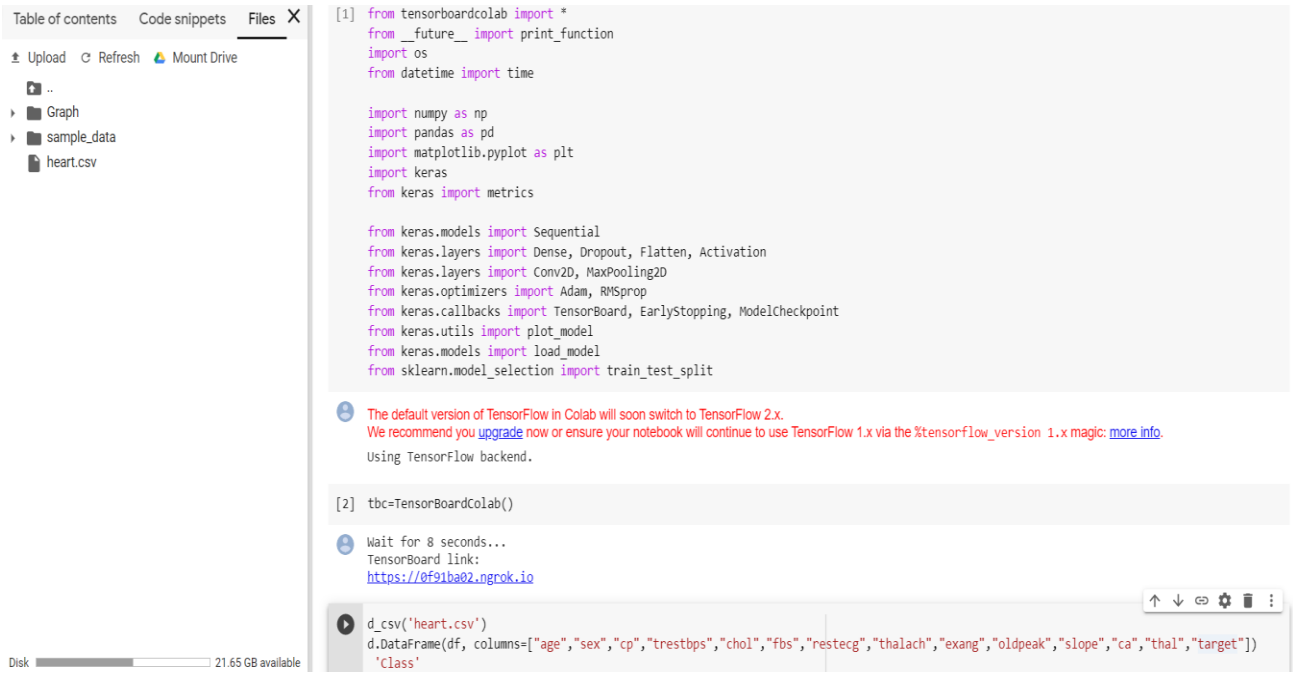
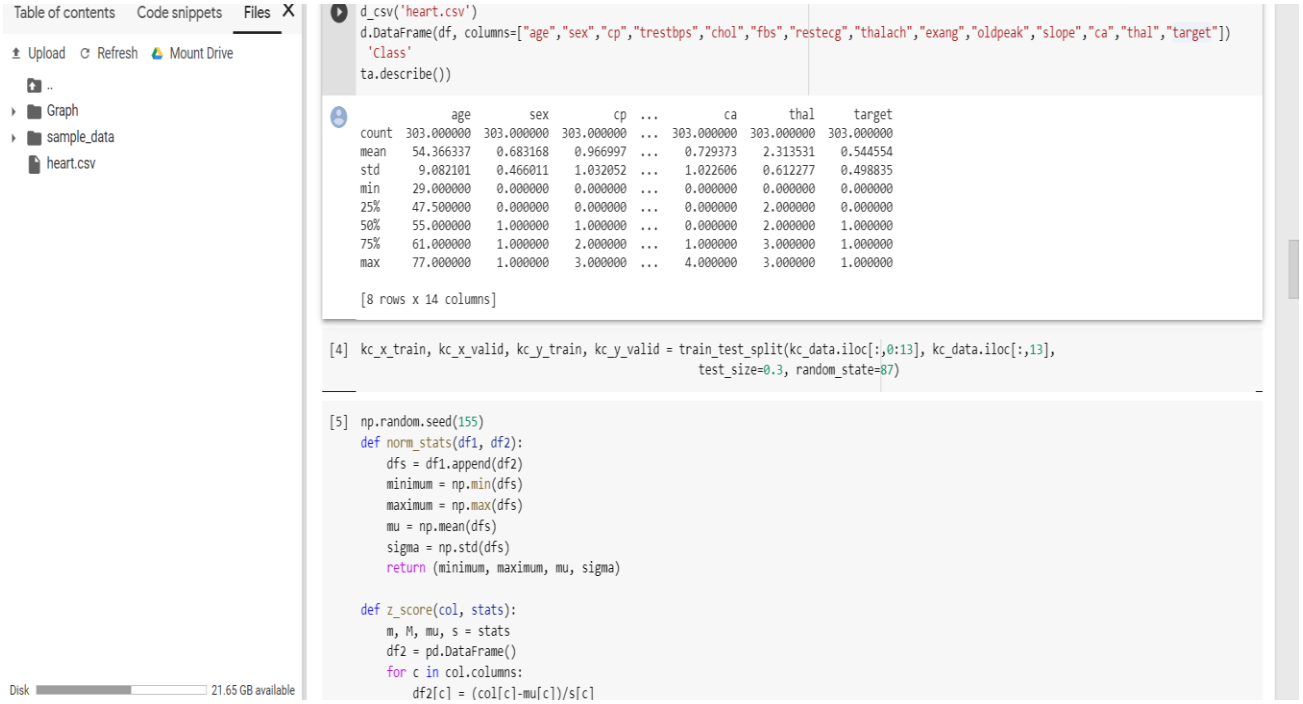
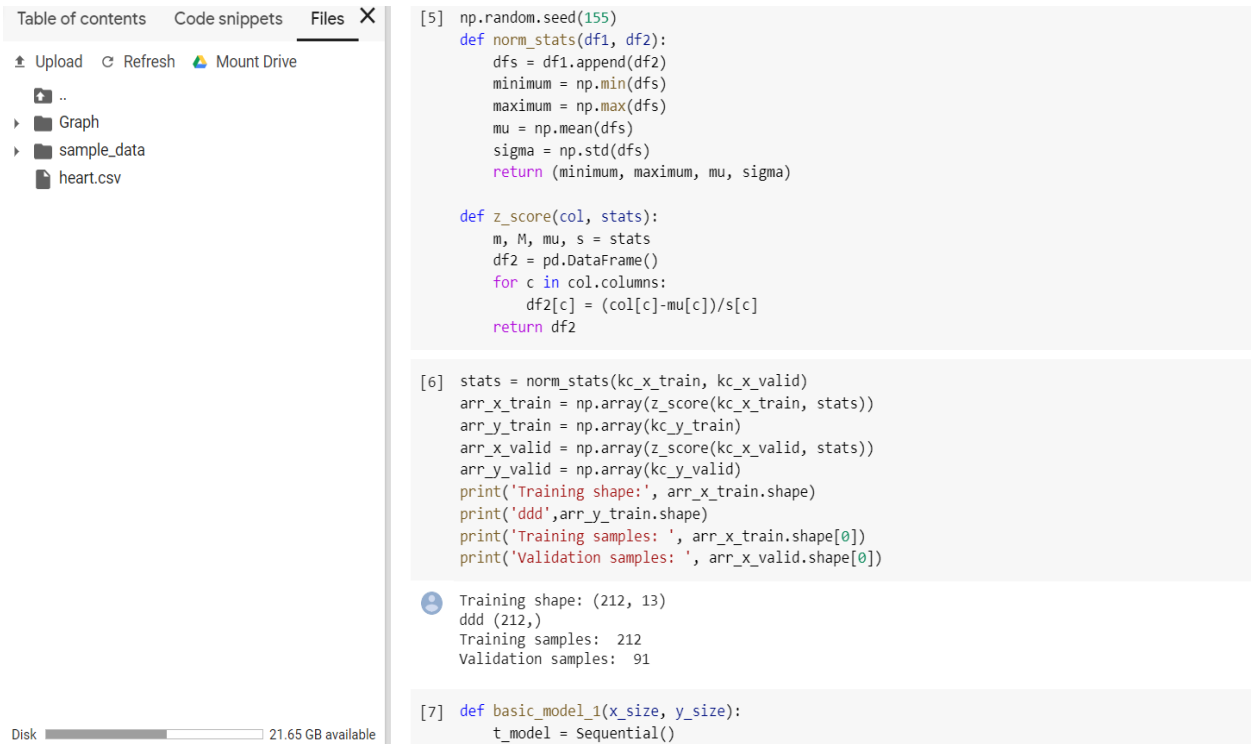
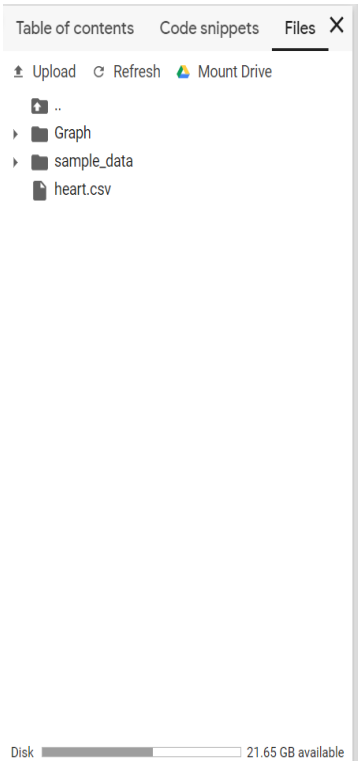


Table of contents
Code snippets
Files
Upload
Refresh
Mount Drive
..
Graph
sample_data
heart.csv
Disk
21.65 GB available
[1] from tensorboardcolab import *
from __future__ import print_function
import os
from datetime import time
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import keras
from keras import metrics
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Activation
from keras.layers import Conv2D, MaxPooling2D
from keras.optimizers import Adam, RMSprop
from keras.callbacks import TensorBoard, EarlyStopping, ModelCheckpoint
from keras.utils import plot_model
from keras.models import load_model
from sklearn.model_selection import train_test_split
The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade now or ensure your notebook will continue to use TensorFlow 1.x via the %tensorflow_version 1.x magic: more info.
Using TensorFlow backend.
[2] tbc=TensorBoardColab()
Wait for 8 seconds...
TensorBoard link:
https://0f91ba02.ngrok.io
d_csv('heart.csv')
d.DataFrame(df, columns=["age","sex","cp","trestbps","chol","fbs","restecg","thalach","exang","oldpeak","slope","ca","thal","target"])
'Class'





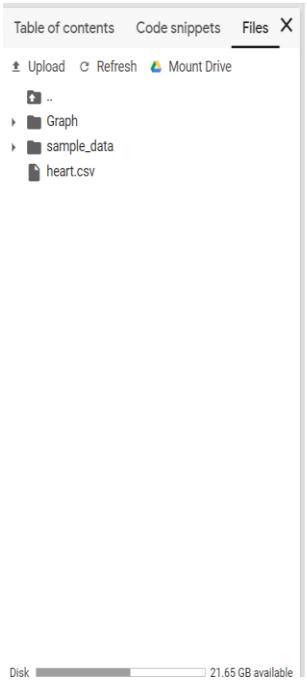

```
#The basic_model_2 is different from
the basic_model_1 but performs the same task with different structure
```




```
[7] def basic_model_1(x_size, y_size):
        t_model = Sequential()
        t_model.add(Dense(100, activation='sigmoid', input_shape=(x_size,)))

        t_model.add(Dense(50, activation="sigmoid"))
        t_model.add(Dense(y_size))
        t_model.compile(loss='binary_crossentropy',
            optimizer='rmsprop',
            metrics=[metrics.mae])
        return(t_model)
```

```
[8]

    def basic_model_2(x_size, y_size):



        t_model = Sequential()
        t_model.add(Dense(100, activation='sigmoid', input_shape=(x_size,)))
        t_model.add(Dropout(0.1))
        t_model.add(Dense(50, activation='sigmoid'))
        t_model.add(Dense(20, activation='sigmoid'))
        t_model.add(Dense(y_size))
        keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)
        optimizers = ['rmsprop', 'adam']
        t_model.compile(loss='binary_crossentropy',
            optimizer=Adam(),
            metrics=[metrics.mae])
        return(t_model)
```

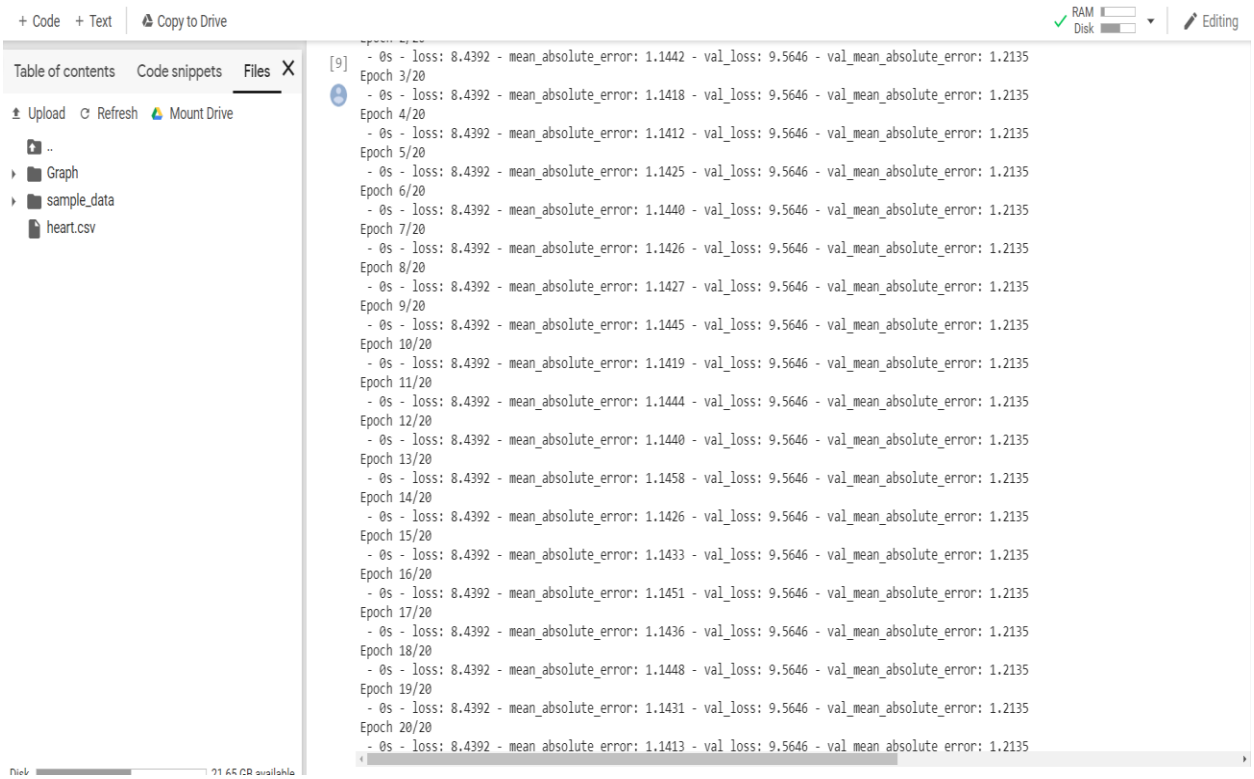
```
[9] model = basic_model_2(arr_x_train.shape[1], 1)

    model.summary()
```




```
[9] model = basic_model_2(arr_x_train.shape[1], 1)

    model.summary()
    epochs = 20
    batch_size =32

    history = model.fit(arr_x_train, arr_y_train,
        batch_size=batch_size,
        epochs=epochs,
        shuffle=True,
        verbose=2,
        validation_data=(arr_x_valid, arr_y_valid),callbacks=[TensorBoardColabCallback(tbc)])
```

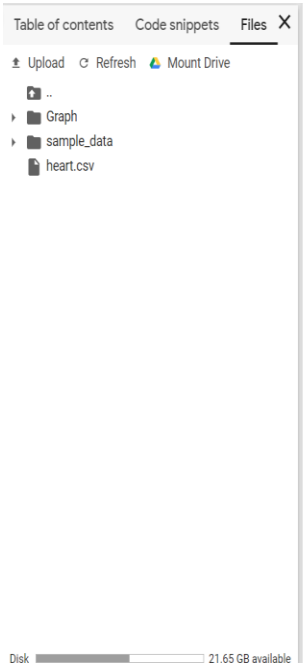
```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3005: The name tf.Session is deprecate

Train on 212 samples, validate on 91 samples
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorboardcolab/core.py:49: The name tf.summary.FileWriter is deprecated.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:190: The name tf.get_default_session i

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:197: The name tf.ConfigProto is deprec

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:207: The name tf.global_variables is d

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:216: The name tf.is_variable_initializ

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:223: The name tf.variables_initializer

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callbacks.py:1122: The name tf.summary.merge_all is deprecated. Plea

Epoch 1/20
 - 1s - loss: 8.4392 - mean_absolute_error: 1.1443 - val_loss: 9.5646 - val_mean_absolute_error: 1.2135
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorboardcolab/callbacks.py:51: The name tf.Summary is deprecated. Pleas

Epoch 2/20
 - 0s - loss: 8.4392 - mean_absolute_error: 1.1442 - val_loss: 9.5646 - val_mean_absolute_error: 1.2135
Epoch 3/20
```

```
# Changed verbose to 2 and observed execution.
```



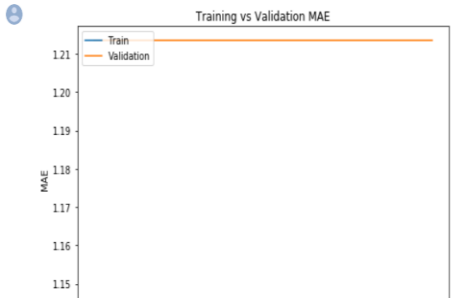

```
# Prepared the plotting

# Summarized history for MAE

# Summarized history for loss
```

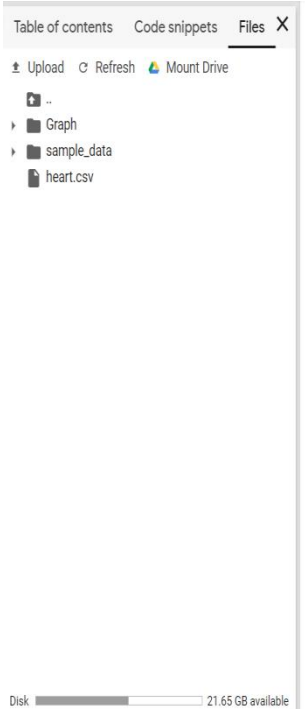
```
# Plotted it all in IPython (non-interactive)

# Created hyperparameter space

# Created hyperparameter options
```



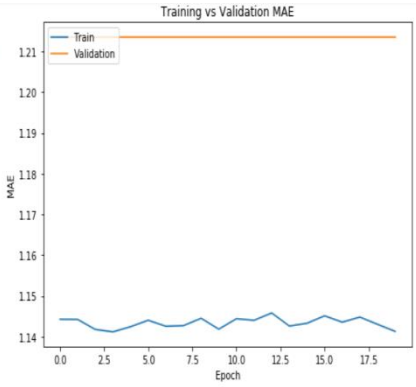

```
[12]    plt.subplot(212)
        plt.plot(h['loss'])
        plt.plot(h['val_loss'])
        plt.title('Training vs Validation Loss')
        plt.ylabel('Loss')
        plt.xlabel('Epoch')
        plt.legend(['Train', 'Validation'], loc='upper left')


        plt.draw()
        plt.show()

        return
```

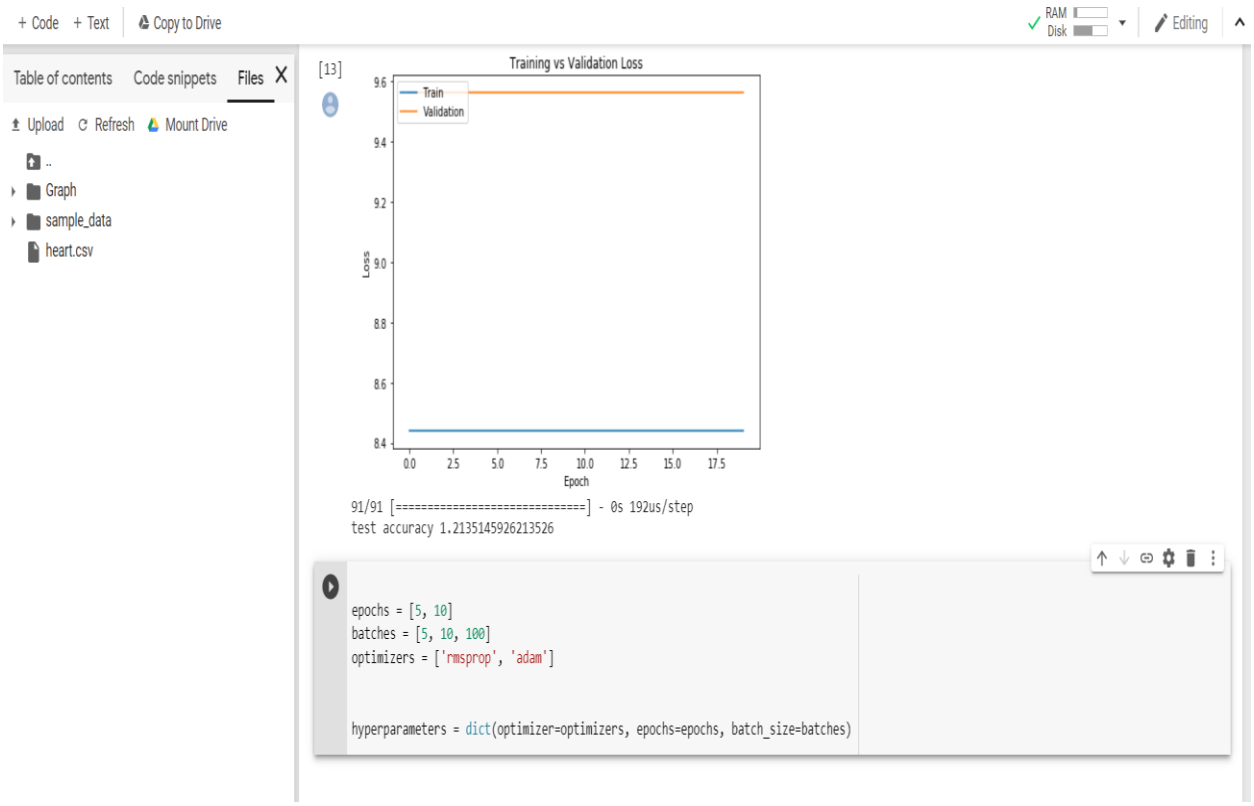
```
[13] plot_hist(history.history, xsize=8, ysize=12)
     score = model.evaluate(arr_x_valid,arr_y_valid)
     print('test accuracy',score[1])
```





[13]





Program-3

Implement the image classification with CNN model on anyone of the following datasets

Code:

```
# Import appropriate libraries
import numpy as np
import os
import cv2
import matplotlib.pyplot as plt
from keras.callbacks import TensorBoard
from keras.models import Sequential
from keras.layers import Dense, MaxPool2D
from keras.layers import Dropout
from keras.layers import Flatten
from keras.constraints import maxnorm
from keras.optimizers import SGD
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils, to_categorical
from keras import backend as K
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# Load data of natural images
labels = os.listdir('/content/drive/My Drive/natural_images')
x = [] # Feature predictor variables array
y = [] # Target variables array
```

```
for label in labels:
    pics = os.listdir('/content/drive/My Drive/natural_images/{}/'.format(label))
    for pic in pics:
        image = cv2.imread('/content/drive/My Drive/natural_images/{}/{}'.format(label, pic))
        image_resized = cv2.resize(image, (32, 32))
        x.append(np.array(image_resized))
        y.append(label)

x = np.array(x)
y = np.array(y)

x = x.astype('float32') / 255
enc = LabelEncoder().fit(y)
y_encoded = enc.transform(y)
y = to_categorical(y_encoded)

# Splitting data set into training and test data set
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33)

# Build the neural network model
# Define the model being built
model = Sequential()
# Convolutional layer
model.add(Conv2D(filters=32, kernel_size=(5, 5), activation='relu', input_shape=x_train.shape[1:]))
```

```
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
# Flatten layer
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(8, activation='softmax'))
# Compile the model defined
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())

# Fit the model defined on the training data set
history = model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=5, batch_size=2000)

# Final evaluation of the model using the test data set
scores = model.evaluate(x_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))

[test_loss, test_acc] = model.evaluate(x_test, y_test)
print("Evaluation result on Test Data : Loss = {}, accuracy = {}".format(test_loss, test_acc))
```

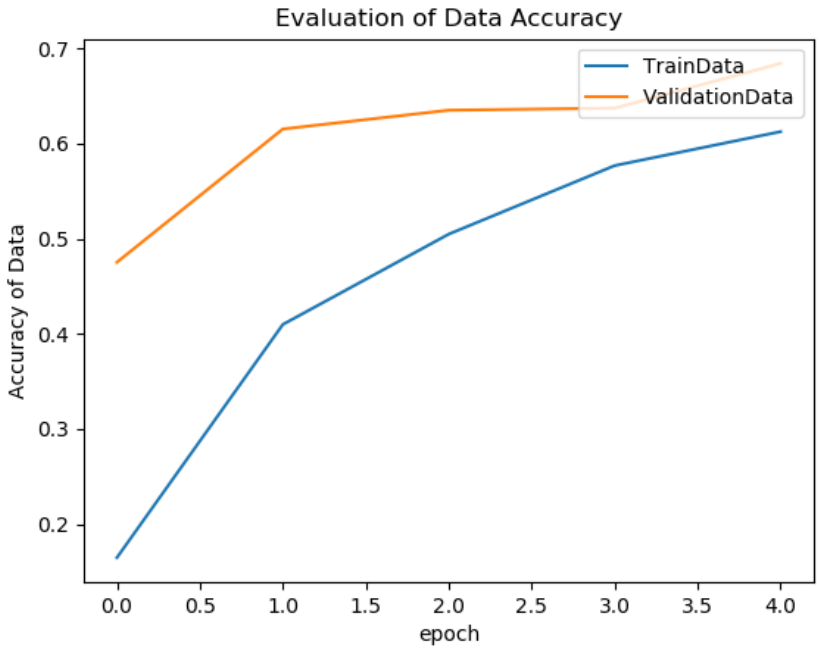
```
# Listing all the components of data present in history
print('The data components present in history are', history.history.keys())

# Graphical evaluation of accuracy associated with training and validation data
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Evaluation of Data Accuracy')
plt.xlabel('epoch')
plt.ylabel('Accuracy of Data')
plt.legend(['TrainData', 'ValidationData'], loc='upper right')
plt.show()

# Graphical evaluation of loss associated with training and validation data
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel('epoch')
plt.ylabel('Loss of Data')
plt.title('Evaluation of Data Loss')
plt.legend(['TrainData', 'ValidationData'], loc='upper right')
plt.show()
```

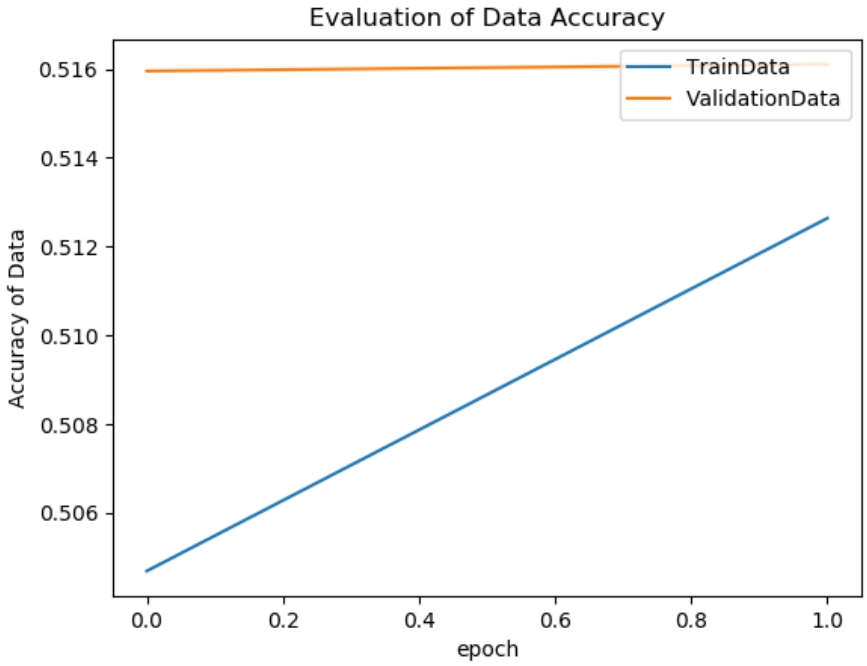
```
# Visualization of the model using tensor board
tbCallBack = TensorBoard(log_dir='./lab2_1', histogram_freq=0, write_graph=True, write_images=True)

# Fitting the model defined using the training data along with validation using test data
history = model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=10, verbose=0, initial_epc

# Evaluation of the loss and accuracy associated to the test data set
[test_loss, test_acc] = model.evaluate(x_test, y_test)
print("Evaluation result on Test Data using Tensorflow : Loss = {}, accuracy = {}".format(test_loss, test

# Listing all the components of data present in history
print('The data components present in history using Tensorflow are', history.history.keys())

# Graphical evaluation of accuracy associated with training and validation data
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Evaluation of Data Accuracy using Tensorflow')
plt.xlabel('epoch')
plt.ylabel('Accuracy of Data')
plt.legend(['TrainData', 'ValidationData'], loc='upper right')
plt.show()
```

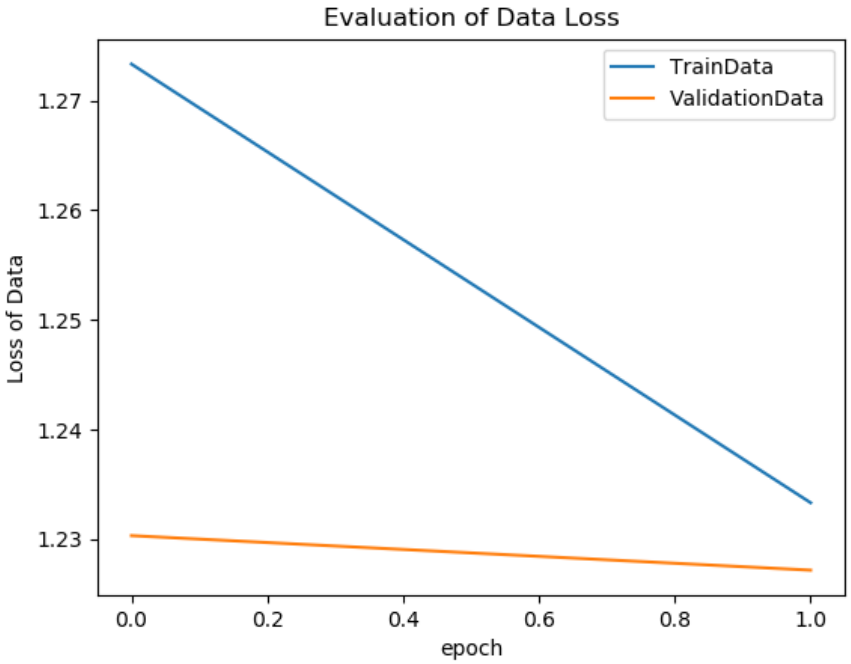
```
# Graphical evaluation of loss associated with training and validation data
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel('epoch')
plt.ylabel('Loss of Data')
plt.title('Evaluation of Data Loss using Tensorflow')
plt.legend(['TrainData', 'ValidationData'], loc='upper right')
plt.show()
```

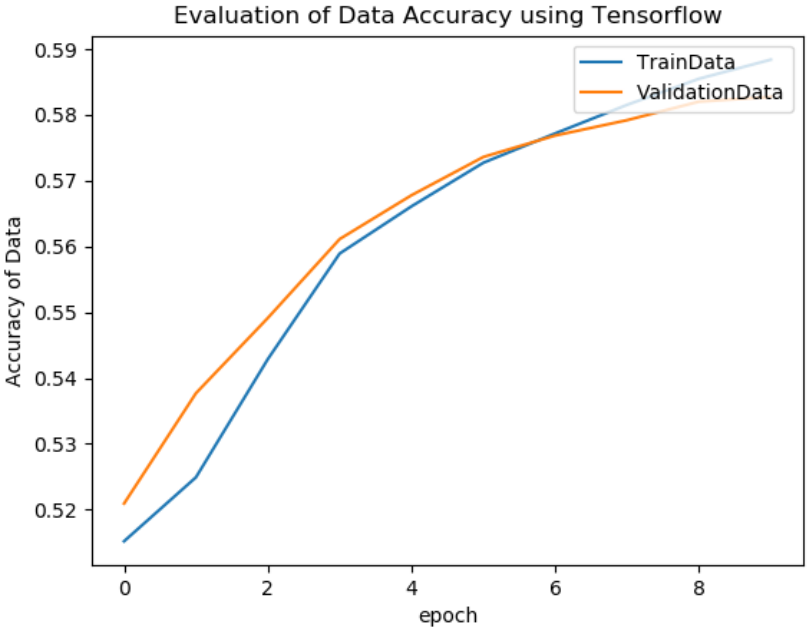
```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:66: T

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:541:

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4432:

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4267:

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:148:

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3733:
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:793: The name tf.trai

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3576:
```

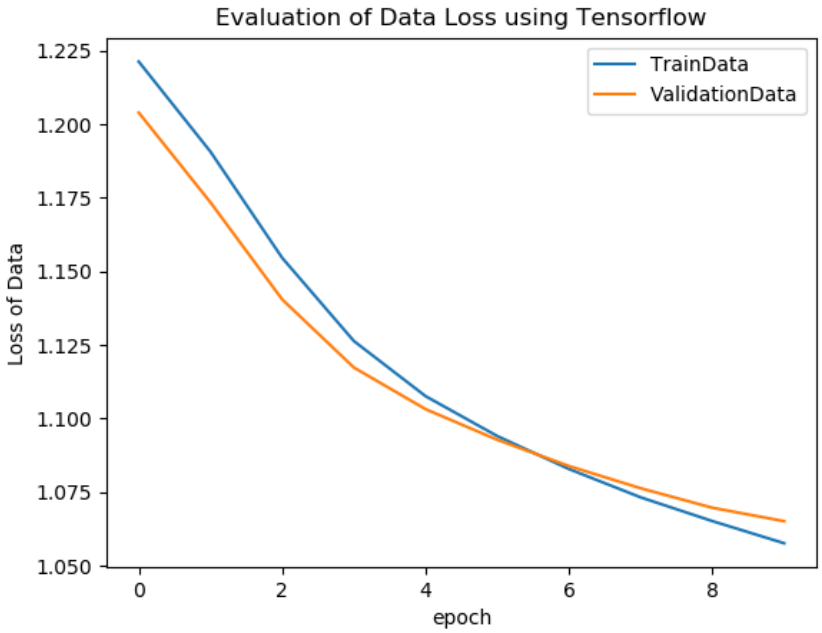
Output:

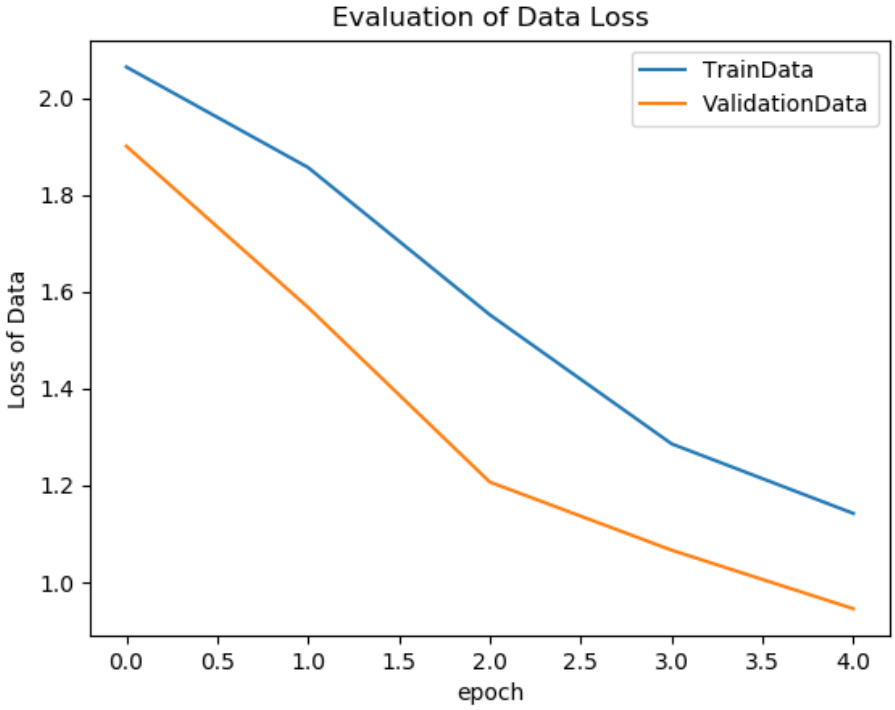

Evaluation of Data Accuracy
TrainData
ValidationData
Accuracy of Data
epoch
0.7
0.6
0.5
0.4
0.3
0.2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

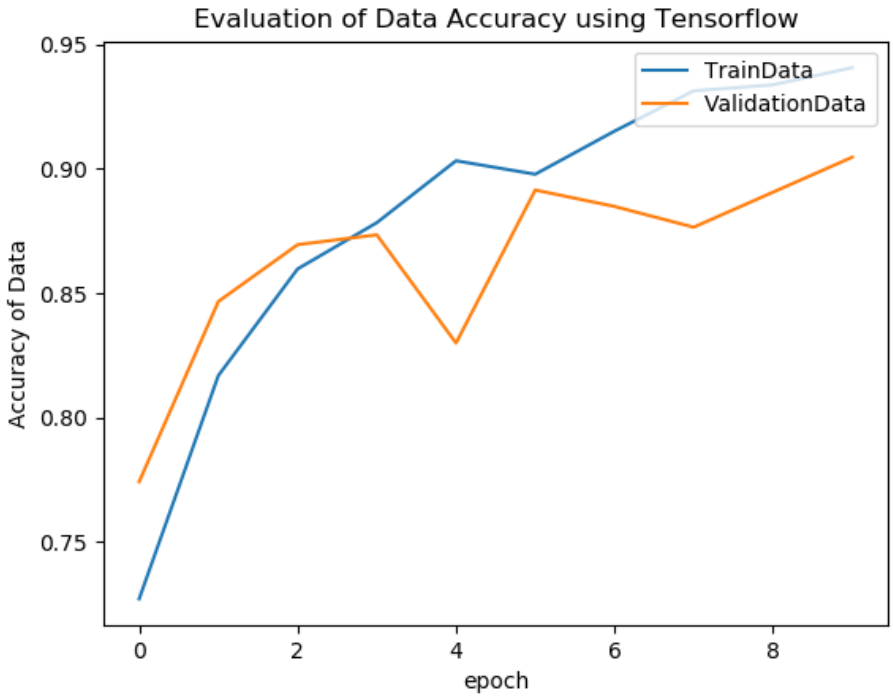

Evaluation of Data Accuracy
TrainData
ValidationData
Accuracy of Data
epoch
0.516
0.514
0.512
0.510
0.508
0.506
0.0
0.2
0.4
0.6
0.8
1.0

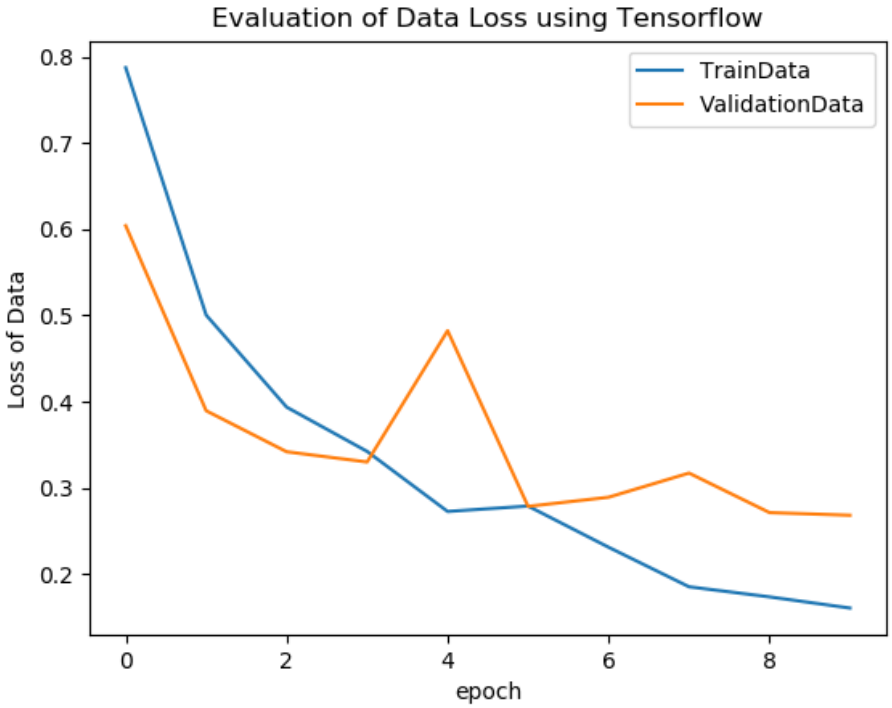

Evaluation of Data Loss
TrainData
ValidationData
Loss of Data
1.27
1.26
1.25
1.24
1.23
0.0
0.2
0.4
0.6
0.8
1.0
epoch


Evaluation of Data Accuracy using Tensorflow
TrainData
ValidationData
Accuracy of Data
0.59
0.58
0.57
0.56
0.55
0.54
0.53
0.52
0
2
4
6
8
epoch


Evaluation of Data Loss using Tensorflow
TrainData
ValidationData
Loss of Data
epoch
1.225
1.200
1.175
1.150
1.125
1.100
1.075
1.050
0
2
4
6
8


Evaluation of Data Loss
TrainData
ValidationData
Loss of Data
epoch
2.0
1.8
1.6
1.4
1.2
1.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Evaluation of Data Accuracy using Tensorflow
TrainData
ValidationData
Accuracy of Data
epoch
0.95
0.90
0.85
0.80
0.75
0
2
4
6
8


Evaluation of Data Loss using Tensorflow
TrainData
ValidationData
Loss of Data
epoch
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
2
4
6
8

Program-4

Code:

Implement the text classification with CNN model on the following movie reviews dataset

```
[1] import re
    from keras.layers.convolutional import MaxPooling1D
    from keras.optimizers import Adam
    from keras.layers import Dense, Dropout, Reshape, Flatten, concatenate, Input, Conv1D, GlobalMaxPooling1D, Embedding
    import matplotlib
    import numpy as np
    import pandas as pd
    from keras.preprocessing import sequence
    from keras.models import Sequential
    from keras.layers import Dense, Embedding
    from keras.layers import LSTM
    from keras.utils import to_categorical
    from keras.preprocessing.text import Tokenizer
    from keras.preprocessing.sequence import pad_sequences
```

```
Using TensorFlow backend.
The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade now or ensure your notebook will continue to use TensorFlow 1.x via the %tensorflow_version 1.x magic: more info.
```

```
[2] import nltk
    nltk.download('stopwords')
    from nltk.corpus import stopwords
    from nltk.tokenize import word_tokenize
    from sklearn.feature_extraction.text import CountVectorizer
    vect = CountVectorizer()
    from sklearn import metrics
    import seaborn as sns
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
[3] import matplotlib.pyplot as plt
    #git hub https://github.com/Stass88/lattelecom
    df_train = pd.read_csv('train.tsv', sep='\t')
```

[2]
```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import CountVectorizer
vect = CountVectorizer()
from sklearn import metrics
import seaborn as sns
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

[3]
```
import matplotlib.pyplot as plt
#git hub https://github.com/Stass88/lattelecom
df_train = pd.read_csv('train.tsv', sep='\t')
df_test = pd.read_csv('test.tsv', sep='\t')
##Descriptive analyse
#this should help you to decide whether to use STOP WORDS or not.
#This part of code is just great analytical tool
```

[4]
```
##Data preprocessing
#we make text lower case and leave only letters from a-z and digits
df_train['Phrase'] = df_train['Phrase'].str.lower()
df_train['Phrase'] = df_train['Phrase'].apply((lambda x: re.sub('[^a-zA-z0-9\s]','',x)))
df_test['Phrase'] = df_test['Phrase'].str.lower()
df_test['Phrase'] = df_test['Phrase'].apply((lambda x: re.sub('[^a-zA-z0-9\s]','',x)))
X_train = df_train.Phrase
y_train = df_train.Sentiment
max_fatures = 2000
tokenize = Tokenizer(num_words=max_fatures, split=' ')
tokenize.fit_on_texts(X_train.values)
X_test = df_test.Phrase
X_train = tokenize.texts_to_sequences(X_train)
```

```
[4] X_train = tokenize.texts_to_sequences(X_train)
    X_test = tokenize.texts_to_sequences(X_test)
    max_lenght = max([len(s.split()) for s in df_train['Phrase']])
    X_train = pad_sequences(X_train, max_lenght)
    X_test = pad_sequences(X_test, max_lenght)
    print(X_train.shape)
    print(X_test.shape)
```

```
(156060, 48)
(66292, 48)
```

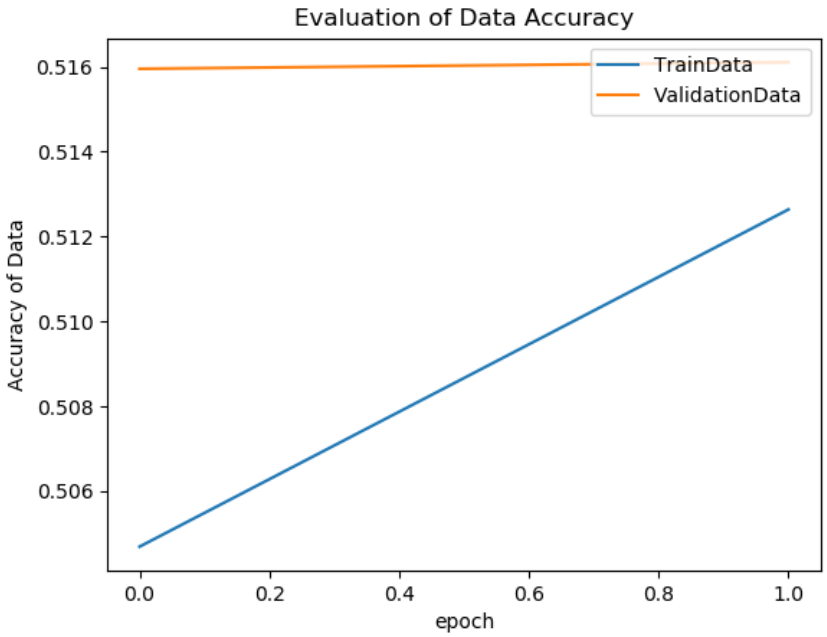
```
##Model building
model=Sequential()
model.add(Embedding(max_fatures, output_dim=100,input_length=48))
model.add(Conv1D(filters=64, kernel_size=5, activation='relu', padding='causal'))
model.add(MaxPooling1D(pool_size=2))
model.add(Conv1D(filters=64, kernel_size=5, activation='relu', padding='causal'))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.7))
model.add(Dense(100,activation='relu'))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(5,activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy',optimizer=Adam(lr=0.001),metrics=['accuracy'])
model.summary()
history = model.fit(X_train, y_train, epochs=40, verbose=True,  batch_size=1024)
```

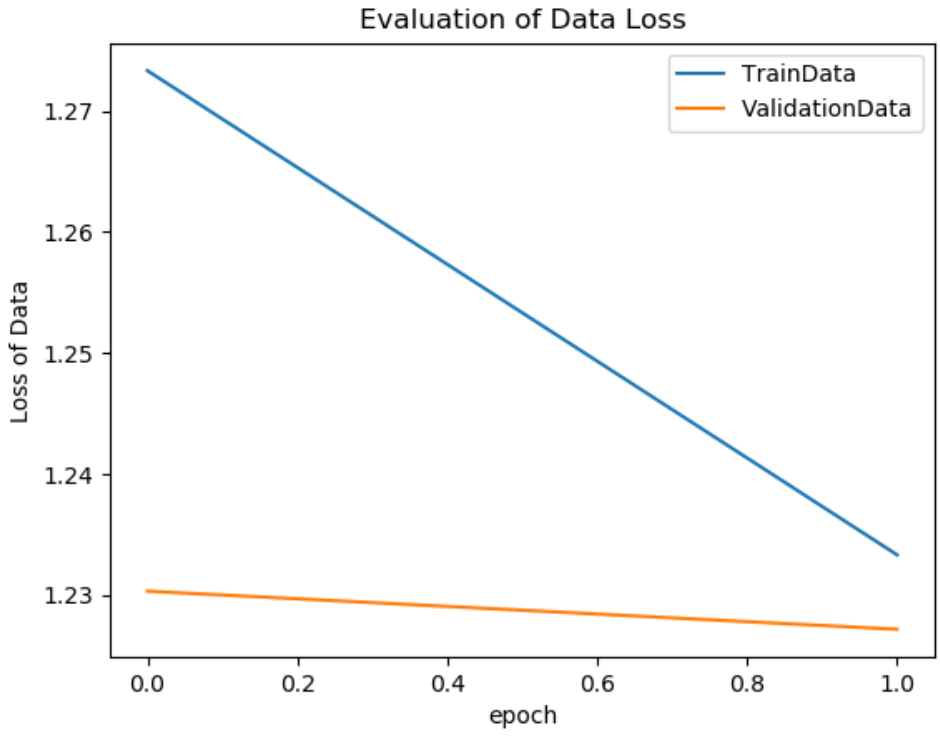
```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:148: The name tf.placeholder_with_defa

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3733: calling dropout (from tensorflow
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
WARNING:tensorflow:Large dropout rate: 0.7 (>0.5). In TensorFlow 2.x, dropout() uses dropout rate instead of keep_prob. Please ensure tha
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:793: The name tf.train.Optimizer is deprecated. Please
```

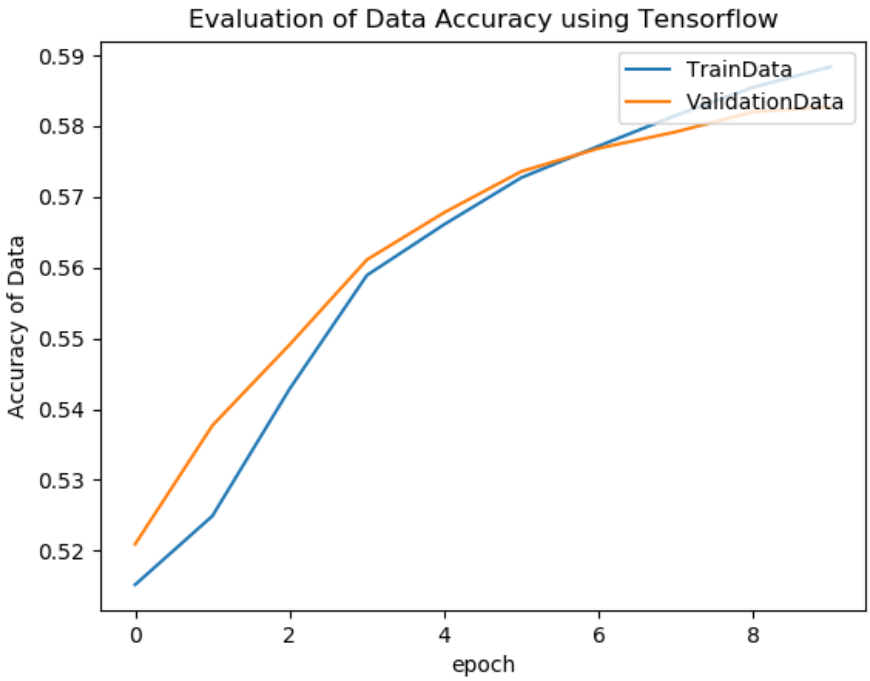
```
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 6005
=================================================================
Total params: 265,113
Trainable params: 265,113
Non-trainable params: 0
_________________________________________________________________
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python/ops/math_grad.py:1424: where (from tensorflow.pytho
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:1033: The name tf.assign_add is deprec

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:1020: The name tf.assign is deprecated

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3005: The name tf.Session is deprecate

Epoch 1/40
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:190: The name tf.get_default_session i

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:197: The name tf.ConfigProto is deprec

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:207: The name tf.global_variables is d

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:216: The name tf.is_variable_initializ

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:223: The name tf.variables_initializer

156060/156060 [==============================] - 70s 450us/step - loss: 1.2659 - acc: 0.5075
Epoch 2/40
156060/156060 [==============================] - 69s 439us/step - loss: 1.1396 - acc: 0.5536
Epoch 3/40
156060/156060 [==============================] - 69s 443us/step - loss: 1.0676 - acc: 0.5825
Epoch 4/40
156060/156060 [==============================] - 69s 441us/step - loss: 1.0101 - acc: 0.6055
Epoch 5/40
156060/156060 [==============================] - 69s 442us/step - loss: 0.9743 - acc: 0.6216
Epoch 6/40
 58368/156060 [==========>                   ] - ETA: 43s - loss: 0.9437 - acc: 0.6359
```

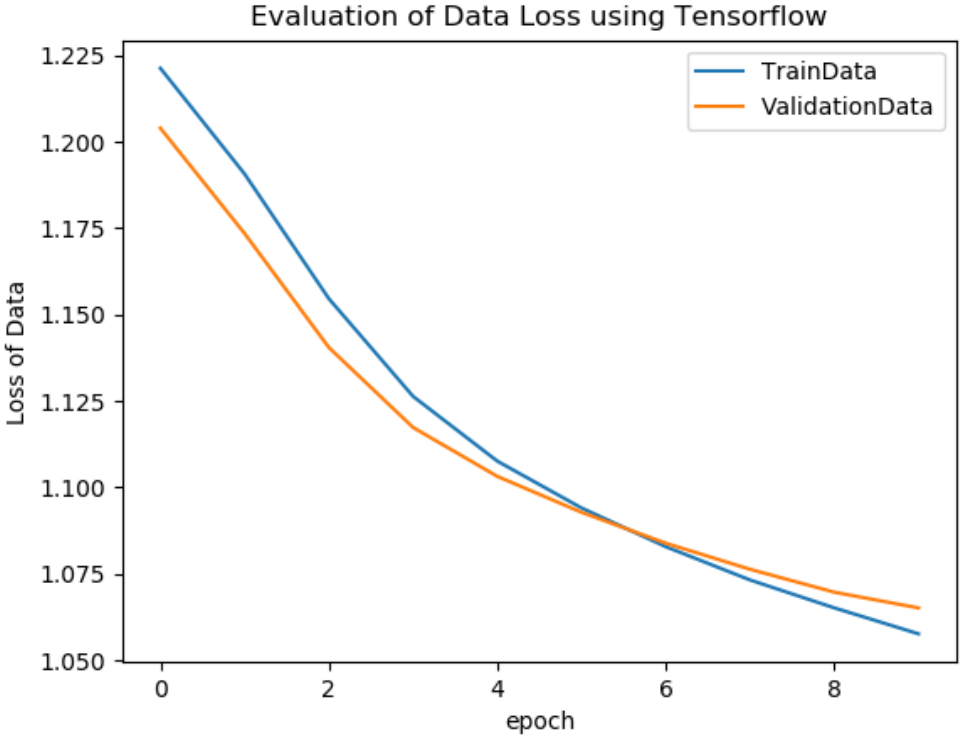
Output

```
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 6005
=================================================================
Total params: 265,113
Trainable params: 265,113
Non-trainable params: 0
_________________________________________________________________
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python/ops/math_grad.py:1424: where (from tensorflow.pytho
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:1033: The name tf.assign_add is deprec

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:1020: The name tf.assign is deprecated

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3005: The name tf.Session is deprecate

Epoch 1/40
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:190: The name tf.get_default_session i

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:197: The name tf.ConfigProto is deprec

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:207: The name tf.global_variables is d

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:216: The name tf.is_variable_initializ

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:223: The name tf.variables_initializer

156060/156060 [==============================] - 70s 450us/step - loss: 1.2659 - acc: 0.5075
Epoch 2/40
156060/156060 [==============================] - 69s 439us/step - loss: 1.1396 - acc: 0.5536
Epoch 3/40
156060/156060 [==============================] - 69s 443us/step - loss: 1.0676 - acc: 0.5825
Epoch 4/40
156060/156060 [==============================] - 69s 441us/step - loss: 1.0101 - acc: 0.6055
Epoch 5/40
156060/156060 [==============================] - 69s 442us/step - loss: 0.9743 - acc: 0.6216
Epoch 6/40
 58368/156060 [==========>                   ] - ETA: 43s - loss: 0.9437 - acc: 0.6359
```


Evaluation of Data Accuracy
TrainData
ValidationData
Accuracy of Data
0.516
0.514
0.512
0.510
0.508
0.506
0.0
0.2
0.4
0.6
0.8
1.0
epoch

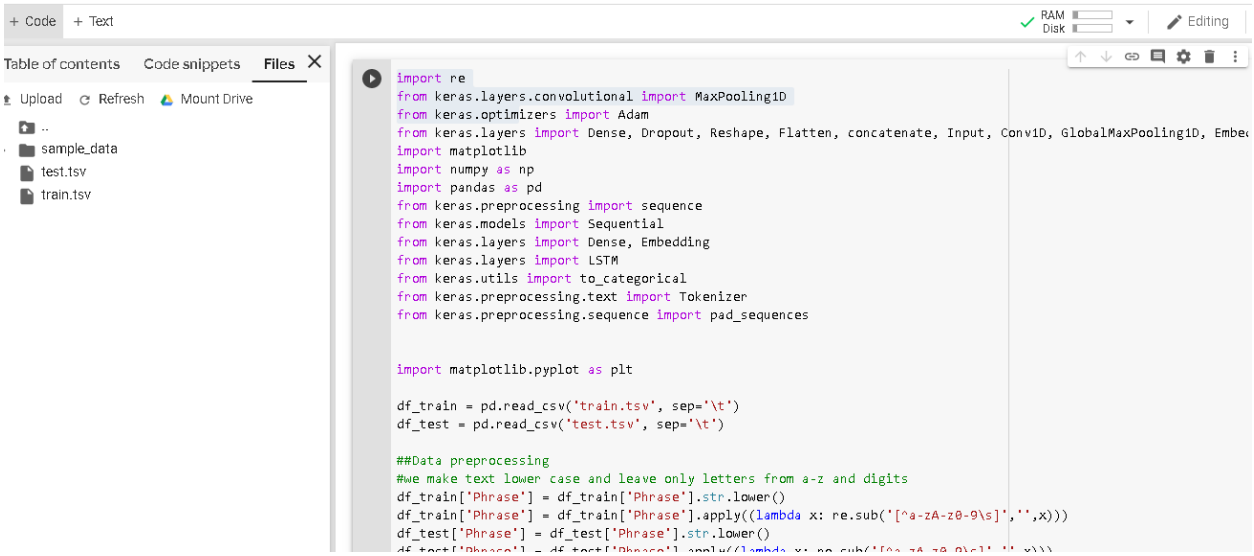

Evaluation of Data Loss
TrainData
ValidationData
Loss of Data
1.27
1.26
1.25
1.24
1.23
0.0
0.2
0.4
0.6
0.8
1.0
epoch


Evaluation of Data Accuracy using Tensorflow
TrainData
ValidationData
Accuracy of Data
0.59
0.58
0.57
0.56
0.55
0.54
0.53
0.52
0
2
4
6
8
epoch


Evaluation of Data Loss using Tensorflow
TrainData
ValidationData
Loss of Data
1.225
1.200
1.175
1.150
1.125
1.100
1.075
1.050
0
2
4
6
8
epoch

Program-5

Code




```
import matplotlib.pyplot as plt

df_train = pd.read_csv('train.tsv', sep='\t')
df_test = pd.read_csv('test.tsv', sep='\t')

##Data preprocessing
#we make text lower case and leave only letters from a-z and digits
df_train['Phrase'] = df_train['Phrase'].str.lower()
df_train['Phrase'] = df_train['Phrase'].apply((lambda x: re.sub('[^a-zA-z0-9\s]','',x)))
df_test['Phrase'] = df_test['Phrase'].str.lower()
df_test['Phrase'] = df_test['Phrase'].apply((lambda x: re.sub('[^a-zA-z0-9\s]','',x)))
X_train = df_train.Phrase
y_train = df_train.Sentiment
max_fatures = 2000
tokenize = Tokenizer(num_words=max_fatures, split=' ')
tokenize.fit_on_texts(X_train.values)
X_test = df_test.Phrase
X_train = tokenize.texts_to_sequences(X_train)
X_test = tokenize.texts_to_sequences(X_test)
max_lenght = max([len(s.split()) for s in df_train['Phrase']])
X_train = pad_sequences(X_train, max_lenght)
X_test = pad_sequences(X_test, max_lenght)
print(X_train.shape)
print(X_test.shape)

##Model building
model=Sequential()
model.add(Embedding(max_fatures, output_dim=100,input_length=48))
```

```
##Model building
model=Sequential()
model.add(Embedding(max_fatures, output_dim=100,input_length=48))
model.add(LSTM(128,dropout=0.5, recurrent_dropout=0.5,return_sequences=True))
model.add(LSTM(64,dropout=0.5, recurrent_dropout=0.5,return_sequences=False))
model.add(Dense(100,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(5,activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy',optimizer=Adam(lr=0.004),metrics=['accuracy'])
model.summary()

history = model.fit(X_train, y_train, epochs=10, verbose=True,  batch_size=1024)
```

Output:

```
(156060, 48)
(66292, 48)
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 48, 100)           200000
_________________________________________________________________
lstm_3 (LSTM)                (None, 48, 128)           117248
_________________________________________________________________
lstm_4 (LSTM)                (None, 64)                49408
_________________________________________________________________
dense_3 (Dense)              (None, 100)               6500
_________________________________________________________________
dropout_2 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_4 (Dense)              (None, 5)                 505
=================================================================
Total params: 373,661
Trainable params: 373,661
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
156060/156060 [==============================] - 29s 184us/step - loss: 1.1672 - acc: 0.5451
Epoch 2/10
```

```
Epoch 7/10
156060/156060 [==============================] - 28s 178us/step - loss: 0.9310 - acc: 0.6323
Epoch 8/10
156060/156060 [==============================] - 28s 177us/step - loss: 0.9225 - acc: 0.6354
Epoch 9/10
156060/156060 [==============================] - 27s 175us/step - loss: 0.9148 - acc: 0.6382
Epoch 10/10
156060/156060 [==============================] - 28s 179us/step - loss: 0.9072 - acc: 0.6419
```

Program-6

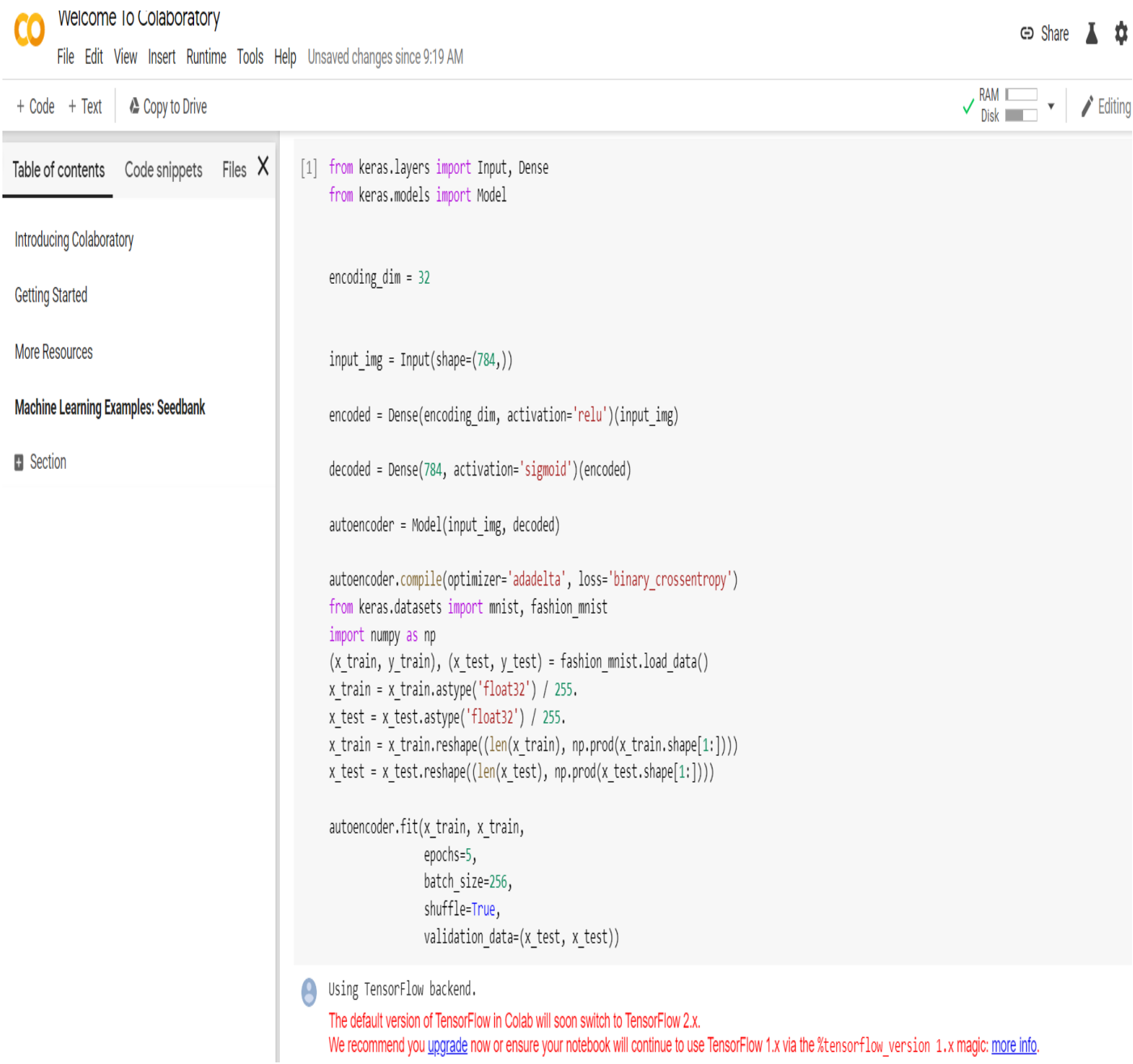
Model	Accuracy
CNN	63

LSTM	65

The accuracy of LSTM model of text classification is higher than of CNN, this is because of cell state and recurrent dropout. Dropping low priority feature over random dropping is always essential. So **LSTM** model is **best** for text classification.

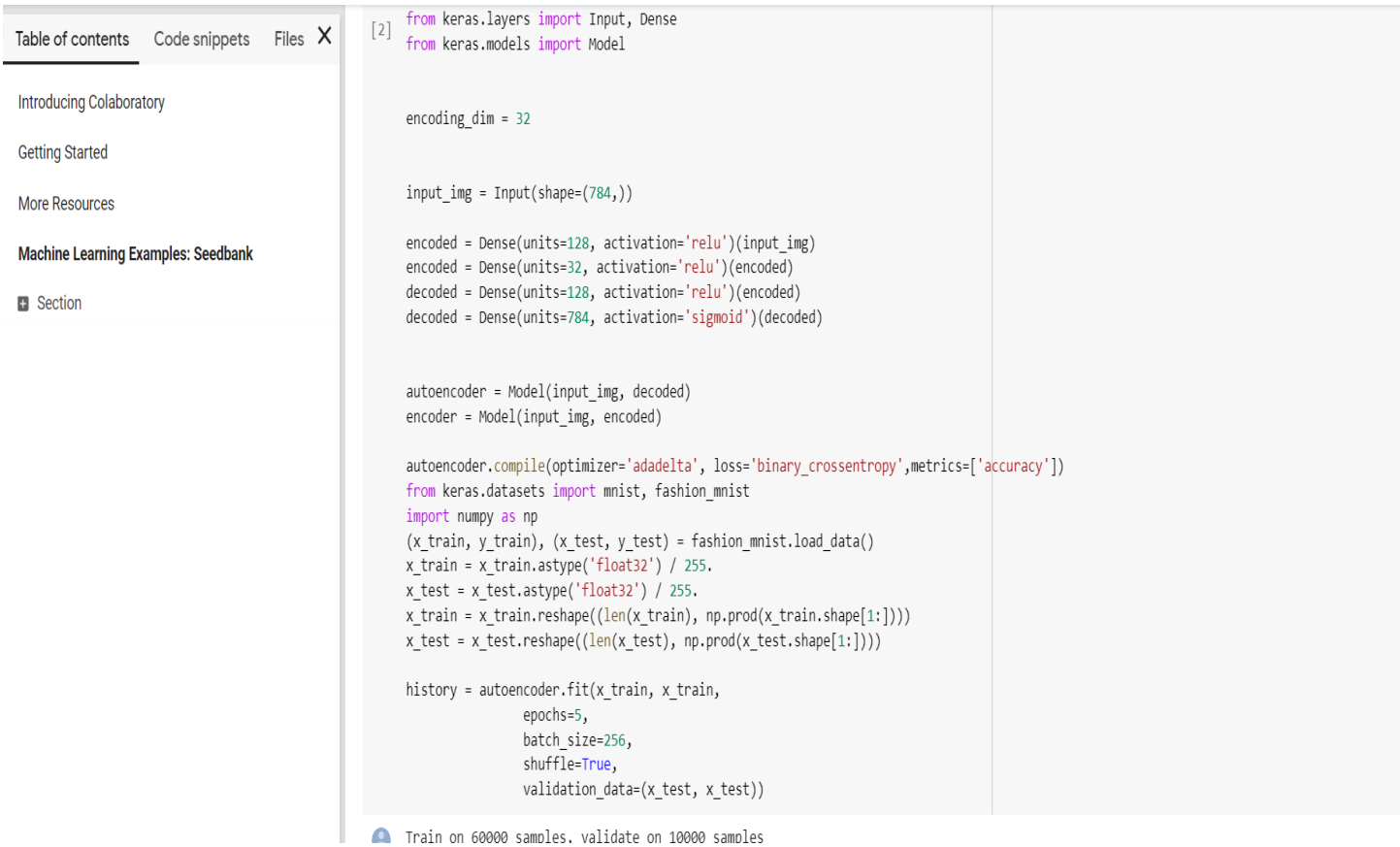
Increasing the Epoch value increased accuracy in both cases

Program-7




```
# here is the size of our encoded representations
encoding_dim = 32

# here is our input placeholder
input_img = Input(shape=(784,))
# "encoded" is the encoded representation of the input
encoded = Dense(encoding_dim, activation='relu')(input_img)
# "decoded" is the lossy reconstruction of the input
decoded = Dense(784, activation='sigmoid')(encoded)
# the model below maps an input to its reconstruction
autoencoder = Model(input_img, decoded)
# this model maps an input to its encoded representation
```

```
Instructions for updating:
[1] Use tf.where in 2.0, which has the same broadcast rule as np.where
Downloading data from http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-labels-idx1-ubyte.gz
32768/29515 [==============================] - 0s 0us/step
Downloading data from http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-images-idx3-ubyte.gz
26427392/26421880 [==============================] - 0s 0us/step
Downloading data from http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-labels-idx1-ubyte.gz
8192/5148 [===============================================] - 0s 0us/step
Downloading data from http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-images-idx3-ubyte.gz
4423680/4422102 [==============================] - 0s 0us/step
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:1033: The name tf.assign_add is deprecat

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:1020: The name tf.assign is deprecated.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3005: The name tf.Session is deprecated.

Train on 60000 samples, validate on 10000 samples
Epoch 1/5
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:190: The name tf.get_default_session is

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:197: The name tf.ConfigProto is deprecat

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:207: The name tf.global_variables is dep

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:216: The name tf.is_variable_initialized

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:223: The name tf.variables_initializer i

60000/60000 [==============================] - 4s 71us/step - loss: 0.5378 - val_loss: 0.4674
Epoch 2/5
60000/60000 [==============================] - 3s 57us/step - loss: 0.4348 - val_loss: 0.4106
Epoch 3/5
60000/60000 [==============================] - 3s 57us/step - loss: 0.3973 - val_loss: 0.3883
Epoch 4/5
60000/60000 [==============================] - 3s 58us/step - loss: 0.3784 - val_loss: 0.3721
Epoch 5/5
60000/60000 [==============================] - 3s 58us/step - loss: 0.3638 - val_loss: 0.3593
<keras.callbacks.History at 0x7f414a8ede80>
```




```
#Addition of one hidden layer
```

```
autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy',metrics=['accuracy'])
```

[2]
```
from keras.datasets import mnist, fashion_mnist
import numpy as np
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.
x_train = x_train.reshape((len(x_train), np.prod(x_train.shape[1:])))
x_test = x_test.reshape((len(x_test), np.prod(x_test.shape[1:])))

history = autoencoder.fit(x_train, x_train,
                epochs=5,
                batch_size=256,
                shuffle=True,
                validation_data=(x_test, x_test))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/5
60000/60000 [==============================] - 6s 95us/step - loss: 0.4939 - acc: 0.4580 - val_loss: 0.4029 - val_acc: 0.4889
Epoch 2/5
60000/60000 [==============================] - 5s 87us/step - loss: 0.3836 - acc: 0.4912 - val_loss: 0.3765 - val_acc: 0.4952
Epoch 3/5
60000/60000 [==============================] - 5s 86us/step - loss: 0.3662 - acc: 0.4952 - val_loss: 0.3575 - val_acc: 0.4928
Epoch 4/5
60000/60000 [==============================] - 5s 85us/step - loss: 0.3495 - acc: 0.4988 - val_loss: 0.3476 - val_acc: 0.5016
Epoch 5/5
60000/60000 [==============================] - 5s 85us/step - loss: 0.3385 - acc: 0.5005 - val_loss: 0.3368 - val_acc: 0.4961
```

[3]
```
encoded_imgs = encoder.predict(x_test)
predicted = autoencoder.predict(x_test)
```

```
from matplotlib import pyplot as plt
plt.figure(figsize=(40, 4))
for i in range(10):

    ax = plt.subplot(3, 20, i + 1)
```

Table of contents Code snippets Files X

Introducing Colaboratory

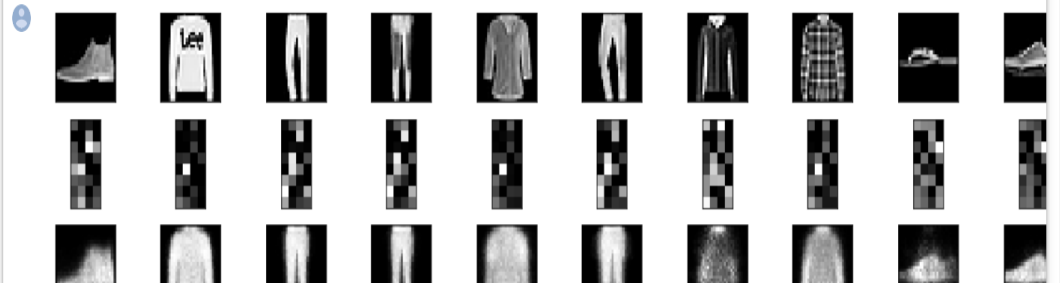
Getting Started

More Resources

Machine Learning Examples: Seedbank

Section

```
from matplotlib import pyplot as plt
plt.figure(figsize=(40, 4))
for i in range(10):

    ax = plt.subplot(3, 20, i + 1)
    plt.imshow(x_test[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)


    ax = plt.subplot(3, 20, i + 1 + 20)
    plt.imshow(encoded_imgs[i].reshape(8,4))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

    ax = plt.subplot(3, 20, 2*20 +i+ 1)
    plt.imshow(predicted[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
```

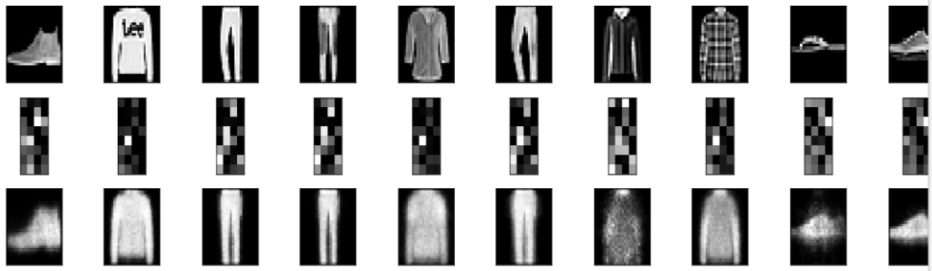


```
# Display of original images
    ax = plt.subplot(3, 20, i + 1)
    plt.imshow(x_test[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

    # Display of the encoded images
    ax = plt.subplot(3, 20, i + 1 + 20)
    plt.imshow(encoded_imgs[i].reshape(8,4))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
# Display of the reconstructed images
```

Table of contents Code snippets Files X

- Introducing Colaboratory
- Getting Started
- More Resources
- **Machine Learning Examples: Seedbank**
- Section

```
ax = plt.subplot(3, 20, 2*20 +i+ 1)
plt.imshow(predicted[i].reshape(28, 28))
plt.gray()
ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False)
```



```
[5] from matplotlib import pyplot as plt
    plt.plot(history.history['acc'])
    plt.plot(history.history['val_acc'])
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('model accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train_acc','val_acc', 'train_loss','val_loss'], loc='upper left')
    plt.show()
```

Visualization using Matplotlib

Table of contents | Code snippets | Files X

- Introducing Colaboratory
- Getting Started
- More Resources
- **Machine Learning Examples: Seedbank**
- Section

```
[5] from matplotlib import pyplot as plt
    plt.plot(history.history['acc'])
    plt.plot(history.history['val_acc'])
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('model accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train_acc','val_acc', 'train_loss','val_loss'], loc='upper left')
    plt.show()
```