DEEP LEARNING LAB MANUAL

20AD56: DEEP LEARNING LAB

Lab Instructor: Dr. P. Bhagath M. Tech (IITG), Ph. D (IITG)

Department of Computer Science and Engineering
Lakireddy Bali Reddy College of Engineering
Mylavaram, A. P., India

Contents

1 Tensor Examples

[1]:

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] ='2'
import tensorflow as tf
x=tf.constant(4.0)
x=tf.constant(4,shape=(1,1),dtype=tf.float32)
y=tf.constant([[1,2,3],[4,5,6]])
print(y.shape)
x=tf.ones((3,3))
print(x)
x=tf.zeros((2,3)) # Creates matrix with all zeros
print(x)
x=tf.eye(3) # Create Identity Matrix
print(x)
x=tf.random.normal((3,3),mean=0,stddev=1)
print(x)
```

```
(2, 3)
tf.Tensor(
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]], shape=(3, 3), dtype=float32)
tf.Tensor(
[[0. 0. 0.]
 [0. 0. 0.]], shape=(2, 3), dtype=float32)
tf.Tensor(
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]], shape=(3, 3), dtype=float32)
tf.Tensor(
[[ 0.7543301   2.466944   -0.38838404]
 [ 1.543006    1.2886018  -0.8451891 ]
 [-2.0536883  -0.65441424 -0.5126149 ]], shape=(3, 3), dtype=float32)
```

[2]:

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] ='2'
import tensorflow as tf


x=tf.range(9)
print(x)

x=tf.range(start=1,limit=10,delta=2)
print(x)

x=4.5
y=tf.cast(x,dtype=tf.float64)
```

```
print(y)
```

```
tf.Tensor([0 1 2 3 4 5 6 7 8], shape=(9,), dtype=int32)
tf.Tensor([1 3 5 7 9], shape=(5,), dtype=int32)
tf.Tensor(4.5, shape=(), dtype=float64)
```

[3]:

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] ='2'
import tensorflow as tf


#Matrix Multiplication
x1=tf.random.normal((2,3))
x2=tf.random.normal((3,4))

print(x1)
print(x2)
z4=tf.matmul(x1,x2)
print(z4)
```

```
tf.Tensor(
[[ 0.9194401   0.49792027 -0.19131818]
 [ 0.49041578  0.8510263  -0.7529493 ]], shape=(2, 3), dtype=float32)
tf.Tensor(
[[ 1.3130509  -0.19392137  0.91854215  0.00565816]
 [ 1.2041862   0.49116    -1.8775115  -0.37181705]
 [-1.2677083   0.07924955  0.701386    0.76560897]], shape=(3, 4),
dtype=float32)
tf.Tensor(
[[ 2.049396    0.05109756 -0.22449447 -0.32640782]
 [ 2.623255    0.2632171  -1.6754522  -0.890116  ]], shape=(2, 4),
dtype=float32)
```

[4]:

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] ='2'
import tensorflow as tf

#Indexing
x=tf.constant([0,1,1,2,3,1,2,3])

print(x[:]) #Print all the elements
print(x[1:]) #Print all the elements except first

print(x[1:3])

print(x[::2]) #Skip every second element
print(x[::-1]) #Print in reverse order
```

```
tf.Tensor([0 1 1 2 3 1 2 3], shape=(8,), dtype=int32)
tf.Tensor([1 1 2 3 1 2 3], shape=(7,), dtype=int32)
tf.Tensor([1 1], shape=(2,), dtype=int32)
tf.Tensor([0 1 3 2], shape=(4,), dtype=int32)
tf.Tensor([3 2 1 3 2 1 1 0], shape=(8,), dtype=int32)
```

2 Principal Component Analysis

[1]:

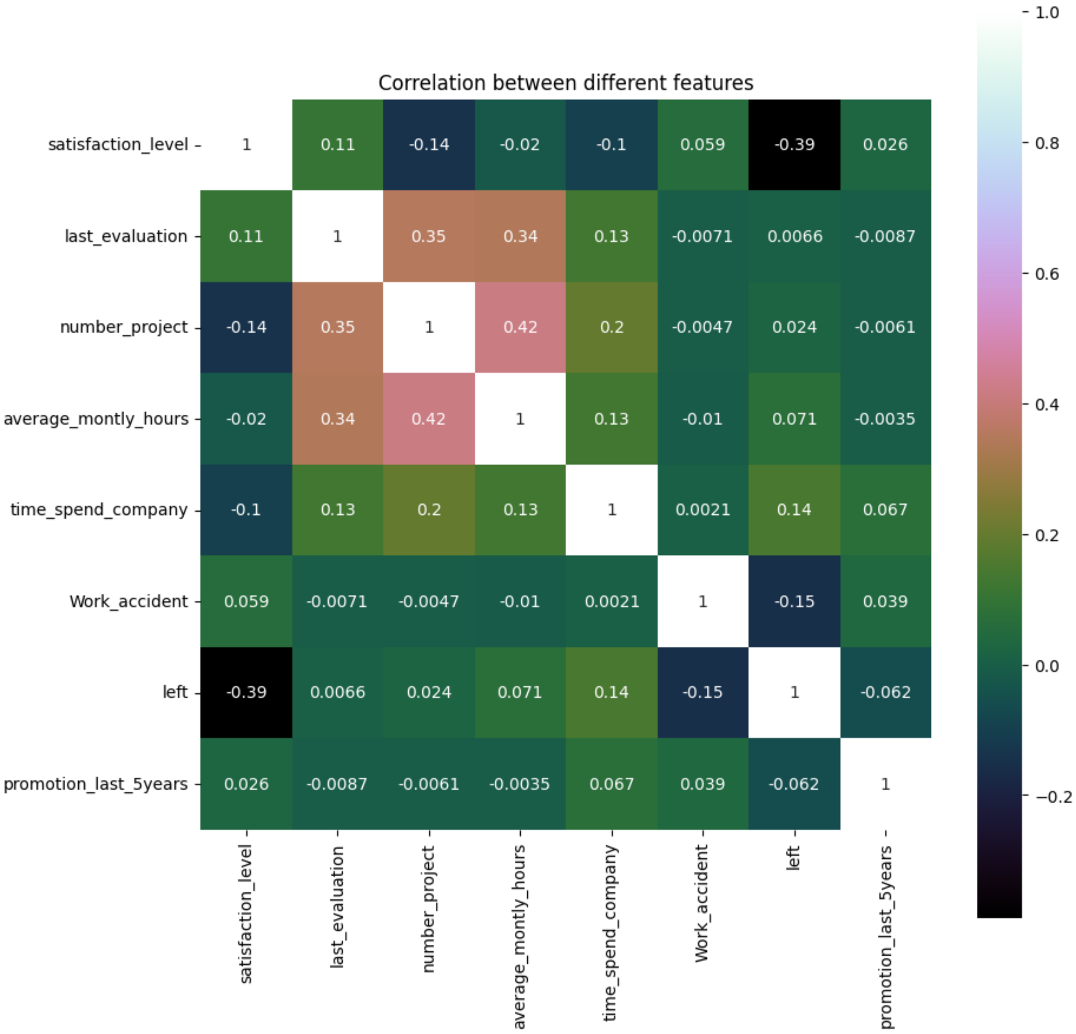
```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler

df=pd.read_csv('HR_comma_sep.csv')
print(df)

column_names=df.columns.tolist()
print(column_names)

correlation=df.corr()
plt.figure(figsize=(10,10))
sns.heatmap(correlation, vmax=1,square=True,annot=True,cmap='cubehelix')
plt.title('Correlation between different features')

X=df.iloc[:,0:8]
X_std=StandardScaler().fit_transform(X)
print(X_std)
np.shape(X)
```

```
       satisfaction_level  last_evaluation  number_project  \
0                    0.38             0.53               2
1                    0.80             0.86               5
2                    0.11             0.88               7
3                    0.72             0.87               5
4                    0.37             0.52               2
...                   ...              ...             ...
14994                0.40             0.57               2
14995                0.37             0.48               2
14996                0.37             0.53               2
14997                0.11             0.96               6
14998                0.37             0.52               2

       average_montly_hours  time_spend_company  Work_accident  left  \
0                       157                   3              0     1
1                       262                   6              0     1
2                       272                   4              0     1
3                       223                   5              0     1
4                       159                   3              0     1
...                     ...                 ...            ...   ...
14994                   151                   3              0     1
14995                   160                   3              0     1
14996                   143                   3              0     1
14997                   280                   4              0     1
14998                   158                   3              0     1

       promotion_last_5years    sales  salary
0                          0    sales     low
1                          0    sales  medium
2                          0    sales  medium
3                          0    sales     low
4                          0    sales     low
...                      ...      ...     ...
14994                      0  support     low
14995                      0  support     low
14996                      0  support     low
14997                      0  support     low
14998                      0  support     low

[14999 rows x 10 columns]
['satisfaction_level', 'last_evaluation', 'number_project',
'average_montly_hours', 'time_spend_company', 'Work_accident', 'left',
'promotion_last_5years', 'sales', 'salary']
```

```
/tmp/ipykernel_3937/155348698.py:14: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  correlation=df.corr()
```

```
[[-0.93649469 -1.08727529 -1.46286291 ... -0.41116529  1.788917
  -0.14741182]
 [ 0.75281433  0.84070693  0.97111292 ... -0.41116529  1.788917
  -0.14741182]
 [-2.02247906  0.95755433  2.59376348 ... -0.41116529  1.788917
  -0.14741182]
 ...
 [-0.97671633 -1.08727529 -1.46286291 ... -0.41116529  1.788917
  -0.14741182]
 [-2.02247906  1.42494396  1.7824382  ... -0.41116529  1.788917
  -0.14741182]
 [-0.97671633 -1.14569899 -1.46286291 ... -0.41116529  1.788917
  -0.14741182]]
```

```
[1]: (14999, 8)
```



```
[4]: #Find the covariance matrix
     mean_vec=np.mean(X_std,axis=0)

     cov_mat=(X_std-mean_vec).T.dot((X_std-mean_vec))/(X_std.shape[0]-1)

     print(cov_mat)

     eig_vals,eig_vecs=np.linalg.eig(cov_mat)


     print('EigenVectors\n%s'%eig_vecs)
     print('\n\n Eigen Values \n%s'%eig_vals)
```

```
[[ 1.00006668  0.10502822 -0.14297912 -0.02004945 -0.1008728   0.05870115
  -0.38840088  0.02560689]
 [ 0.10502822  1.00006668  0.34935588  0.33976445  0.1315995  -0.00710476
   0.00656756 -0.00868435]
 [-0.14297912  0.34935588  1.00006668  0.41723845  0.19679901 -0.00474086
   0.02378877 -0.00606436]
 [-0.02004945  0.33976445  0.41723845  1.00006668  0.12776343 -0.01014356
   0.07129193 -0.00354465]
 [-0.1008728   0.1315995   0.19679901  0.12776343  1.00006668  0.00212056
   0.14483183  0.06743742]
 [ 0.05870115 -0.00710476 -0.00474086 -0.01014356  0.00212056  1.00006668
  -0.15463194  0.03924805]
 [-0.38840088  0.00656756  0.02378877  0.07129193  0.14483183 -0.15463194
   1.00006668 -0.06179223]
 [ 0.02560689 -0.00868435 -0.00606436 -0.00354465  0.06743742  0.03924805
  -0.06179223  1.00006668]]
EigenVectors
[[-0.18956186 -0.60825815  0.51043559  0.14578963 -0.2534991  -0.32268329
  -0.2910217   0.2433296 ]
 [ 0.46363715 -0.31222881 -0.27367838  0.15715943 -0.10307248 -0.06471173
   0.54777287  0.52257837]
 [ 0.55704703 -0.12254292  0.58883958  0.0129521   0.09858338  0.1887942
   0.24157676 -0.47335058]
 [ 0.52559587 -0.17853674 -0.30588994  0.11339814  0.0120681   0.25349244
  -0.72147388  0.02274205]
 [ 0.33395132  0.11709262 -0.11038416 -0.44415687 -0.04569912 -0.79303045
  -0.09314767 -0.16013636]
 [-0.06443923 -0.28140442  0.07016424 -0.42577604  0.81315664  0.06549289
  -0.02938544  0.25312908]
 [ 0.2163394   0.61631274  0.45356155  0.01069646  0.00816191  0.01364792
  -0.16219105  0.58392171]
 [-0.00870881 -0.11358933  0.03780465 -0.74989628 -0.50186771  0.39801173
   0.02283486  0.11154387]]


 Eigen Values
[1.86103997 1.46419116 0.47748369 1.06065738 0.95604748 0.84555567
 0.62652988 0.70902817]
```

[9]:

```
eig_pairs=[(np.abs(eig_vals[i]),eig_vecs[:,i]) for i in range(len(eig_vals))]
#print(eig_pairs)
eig_pairs.sort(key=lambda x:x[0], reverse=True)
for i in eig_pairs:
    print(i[0])
```

```
1.8610399673428657
1.4641911571613258
1.0606573786654157
```

```
0.9560474847067920
0.845555673284437
0.7090281741798209
0.6265298821915976
0.47748368692167753
```

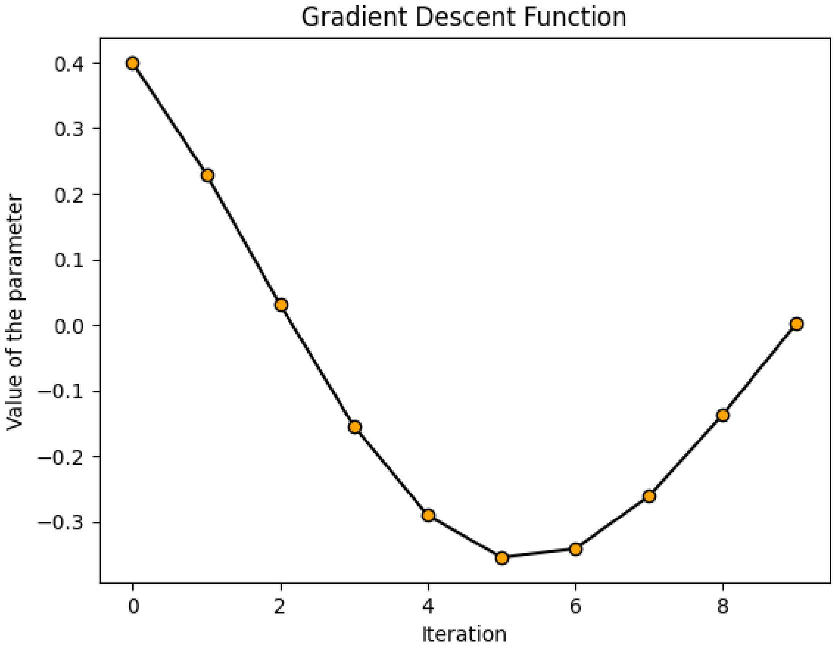
3 Gradient Descent Algorithm

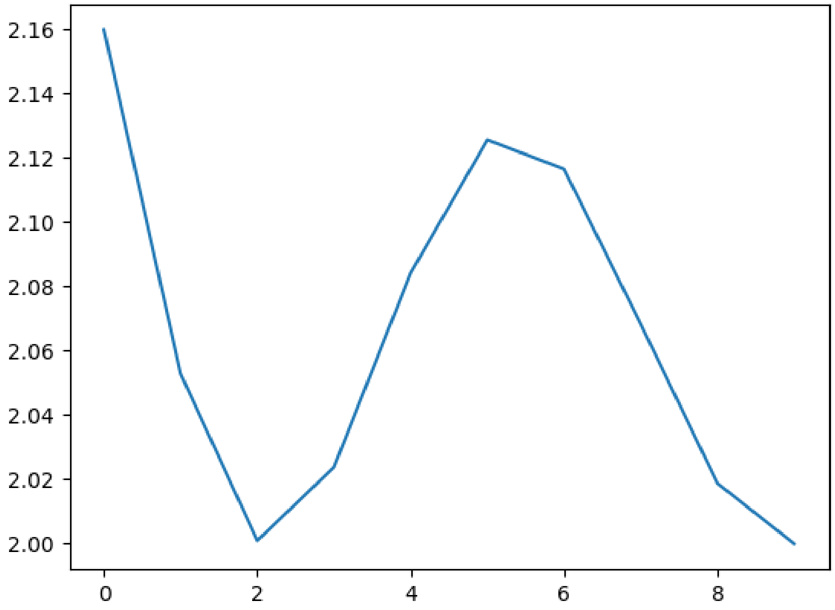
[1]:
```
import tensorflow as tf
import os
import matplotlib.pyplot as plt
os.environ['TF_CPP_MIN_LOG_LEVEL'] ='2'
# Create needed objects
sgd = tf.keras.optimizers.SGD(learning_rate=0.1, momentum=0.9)
var = tf.Variable(0.5)
cost = lambda: 2 + var ** 2

x=[]
y=[]
# Perform optimization
for _ in range(10):
    sgd.minimize(cost, var_list=[var])
    #print(var.numpy())
    #print(cost().numpy())
    x.append(var.numpy())
    y.append(cost().numpy())

# Extract results
print(x)
print(y)
plt.plot(x,marker='o',markerfacecolor='orange',color='black')
plt.title('Gradient Descent Function')
plt.xlabel('Iteration')
plt.ylabel('Value of the parameter')
plt.show()
plt.plot(y)
plt.show()
```

```
2023-06-01 10:10:35.495510: I tensorflow/core/platform/cpu_feature_guard.cc:193]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
2023-06-01 10:10:35.642913: W
tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could
not load dynamic library 'libcudart.so.11.0'; dlerror: libcudart.so.11.0: cannot
```

```
open shared object file: No such file or directory
2023-06-01 10:10:35.642942: I
tensorflow/compiler/xla/stream_executor/cuda/cudart_stub.cc:29] Ignore above
cudart dlerror if you do not have a GPU set up on your machine.
2023-06-01 10:10:37.518743: W
tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could
not load dynamic library 'libnvinfer.so.7'; dlerror: libnvinfer.so.7: cannot
open shared object file: No such file or directory
2023-06-01 10:10:37.518887: W
tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could
not load dynamic library 'libnvinfer_plugin.so.7'; dlerror:
libnvinfer_plugin.so.7: cannot open shared object file: No such file or
directory
2023-06-01 10:10:37.518897: W
tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Cannot
dlopen some TensorRT libraries. If you would like to use Nvidia GPU with
TensorRT, please make sure the missing libraries mentioned above are installed
properly.
2023-06-01 10:10:41.030442: W
tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could
not load dynamic library 'libcuda.so.1'; dlerror: libcuda.so.1: cannot open
shared object file: No such file or directory
2023-06-01 10:10:41.030469: W
tensorflow/compiler/xla/stream_executor/cuda/cuda_driver.cc:265] failed call to
cuInit: UNKNOWN ERROR (303)
2023-06-01 10:10:41.030485: I
tensorflow/compiler/xla/stream_executor/cuda/cuda_diagnostics.cc:156] kernel
driver does not appear to be running on this host (bhagath-VivoBook-
ASUSLaptop-X409DA-M409DA): /proc/driver/nvidia/version does not exist
2023-06-01 10:10:41.031353: I tensorflow/core/platform/cpu_feature_guard.cc:193]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
```

```
[0.4, 0.23, 0.031000003, -0.15429999, -0.29021, -0.354487, -0.3414389,
-0.2614078, -0.13709825, 0.0021999776]
[2.16, 2.0529, 2.000961, 2.0238085, 2.0842218, 2.1256611, 2.1165805, 2.068334,
2.018796, 2.0000048]
```


Gradient Descent Function
Value of the parameter
Iteration
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
0
2
4
6
8


2.16
2.14
2.12
2.10
2.08
2.06
2.04
2.02
2.00
0
2
4
6
8

4 Convolutional Neural Networks

[56]:

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] ='2'

import tensorflow as tf

from tensorflow import keras
import matplotlib.pyplot as plt
import numpy as np




(x_train,y_train), (x_test,y_test) =keras.datasets.mnist.load_data()
x_train=x_train/255
x_test=x_test/255
print(len(x_train))
print(len(x_test))
x_train[0]
```

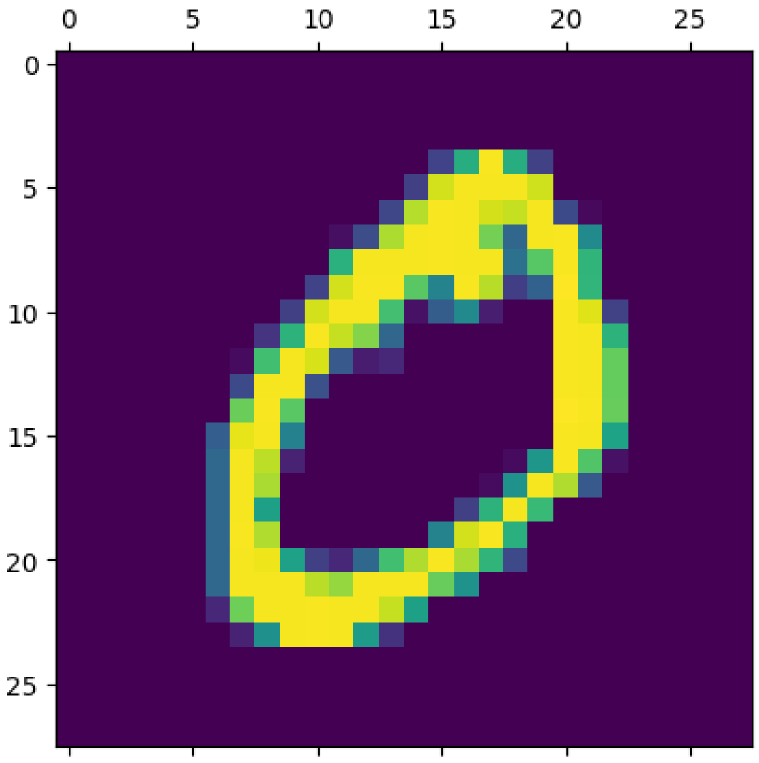
```
60000
10000
```

[56]:

```
array([[0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.01176471, 0.07058824, 0.07058824,
        0.07058824, 0.49411765, 0.53333333, 0.68627451, 0.10196078,
        0.65098039, 1.        , 0.96862745, 0.49803922, 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.11764706, 0.14117647,
        0.36862745, 0.60392157, 0.66666667, 0.99215686, 0.99215686,
        0.99215686, 0.99215686, 0.99215686, 0.88235294, 0.6745098 ,
        0.99215686, 0.94901961, 0.76470588, 0.25098039, 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.19215686, 0.93333333, 0.99215686,
        0.99215686, 0.99215686, 0.99215686, 0.99215686, 0.99215686,
        0.99215686, 0.99215686, 0.98431373, 0.36470588, 0.32156863,
        0.32156863, 0.21960784, 0.15294118, 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.07058824, 0.85882353, 0.99215686,
        0.99215686, 0.99215686, 0.99215686, 0.99215686, 0.77647059,
        0.71372549, 0.96862745, 0.94509804, 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.31372549, 0.61176471,
        0.41960784, 0.99215686, 0.99215686, 0.80392157, 0.04313725,
        0.        , 0.16862745, 0.60392157, 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.05490196,
        0.00392157, 0.60392157, 0.99215686, 0.35294118, 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.54509804, 0.99215686, 0.74509804, 0.00784314,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.04313725, 0.74509804, 0.99215686, 0.2745098 ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.1372549 , 0.94509804, 0.88235294,
        0.62745098, 0.42352941, 0.00392157, 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.31764706, 0.94117647,
        0.99215686, 0.99215686, 0.46666667, 0.09803922, 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.17647059,
        0.72941176, 0.99215686, 0.99215686, 0.58823529, 0.10588235,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.0627451 , 0.36470588, 0.98823529, 0.99215686, 0.73333333,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.97647059, 0.99215686, 0.97647059,
        0.25098039, 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.18039216,
        0.50980392, 0.71764706, 0.99215686, 0.99215686, 0.81176471,
        0.00784314, 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.15294118, 0.58039216, 0.89803922,
        0.99215686, 0.99215686, 0.99215686, 0.98039216, 0.71372549,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.09411765, 0.44705882, 0.86666667, 0.99215686, 0.99215686,
        0.99215686, 0.99215686, 0.78823529, 0.30588235, 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.09019608, 0.25882353,
        0.83529412, 0.99215686, 0.99215686, 0.99215686, 0.99215686,
        0.77647059, 0.31764706, 0.00784314, 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.07058824, 0.67058824, 0.85882353, 0.99215686,
        0.99215686, 0.99215686, 0.99215686, 0.76470588, 0.31372549,
        0.03529412, 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.21568627,
        0.6745098 , 0.88627451, 0.99215686, 0.99215686, 0.99215686,
        0.99215686, 0.95686275, 0.52156863, 0.04313725, 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.53333333,
        0.99215686, 0.99215686, 0.99215686, 0.83137255, 0.52941176,
        0.51764706, 0.0627451 , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ]])
```

[59]:
```
plt.matshow(x_train[1])
print(y_train[1])
```

```
0
```



[61]:
```
print(y_train[:10])
```

```
[5 0 4 1 9 2 1 3 1 4]
```

print(x$_t rain.shape$)

x$_t rain_f lattened = x_t rain.reshape(len(x_t rain), 28 * 28) x_t est_f lattened = x_t est.reshape(len(x_t est), 28 * 28)$

print(x$_t rain_f lattened.shape) print(x_t est_f lattened.shape$)

```
Epoch 1/10
1875/1875 [==============================] - 7s 3ms/step - loss: 9.7042 -
accuracy: 0.8409
```

```
Epoch 2/10
1875/1875 [==============================] - 6s 3ms/step - loss: 6.0889 -
accuracy: 0.8784
Epoch 3/10
1875/1875 [==============================] - 6s 3ms/step - loss: 5.7107 -
accuracy: 0.8825
Epoch 4/10
1875/1875 [==============================] - 6s 3ms/step - loss: 5.5727 -
accuracy: 0.8844
Epoch 5/10
1875/1875 [==============================] - 6s 3ms/step - loss: 5.2911 -
accuracy: 0.8866
Epoch 6/10
1875/1875 [==============================] - 6s 3ms/step - loss: 5.4390 -
accuracy: 0.8870
Epoch 7/10
1875/1875 [==============================] - 6s 3ms/step - loss: 5.2113 -
accuracy: 0.8896
Epoch 8/10
1875/1875 [==============================] - 6s 3ms/step - loss: 5.2926 -
accuracy: 0.8891
Epoch 9/10
1875/1875 [==============================] - 6s 3ms/step - loss: 5.2121 -
accuracy: 0.8902
Epoch 10/10
1875/1875 [==============================] - 6s 3ms/step - loss: 5.2102 -
accuracy: 0.8913
```

[23]: `<keras.callbacks.History at 0x7f0788489850>`

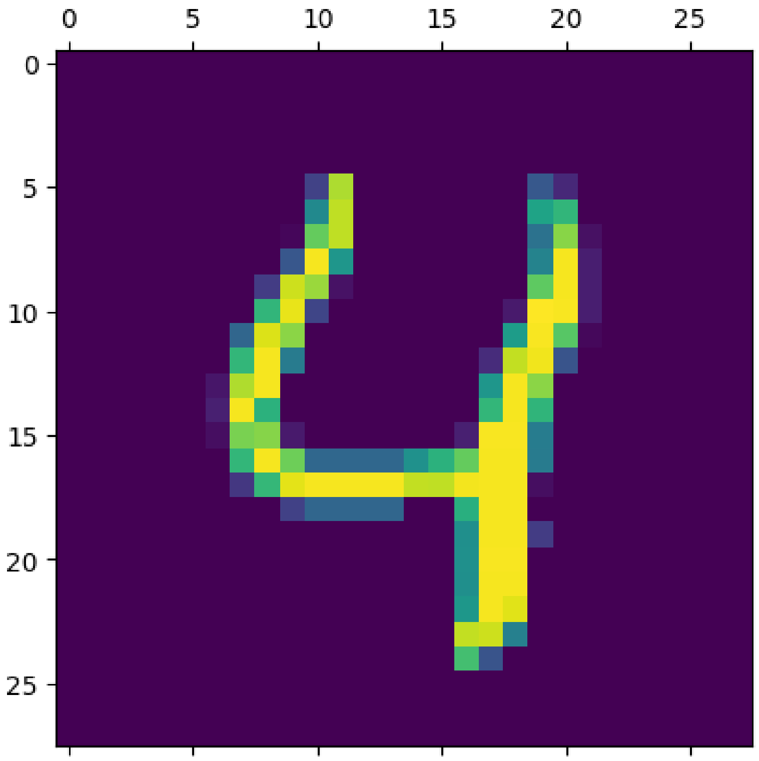
[26]:
```
model.evaluate(x_test_flattened,y_test)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 6.0120 -
accuracy: 0.8869
```

[26]: `[6.011996269226074, 0.886900007724762]`

[48]:
```
plt.matshow(x_test[4])
y_predicted=model.predict(x_test_flattened)
print(y_predicted[4])
print(np.argmax(y_predicted[4]))
```

```
313/313 [==============================] - 0s 1ms/step
[1.5413564e-08 0.0000000e+00 2.5811736e-11 1.2735150e-12 1.0000000e+00
 4.3093944e-28 1.0000000e+00 1.0000000e+00 1.0000000e+00 1.0000000e+00]
4
```



```
[52]: y_predicted_labels=[np.argmax(i) for i in y_predicted]
      print(y_predicted_labels[:10])
      print(y_test[:10])

      cm=tf.math.confusion_matrix(labels=y_test,predictions=y_predicted_labels)
      cm
```

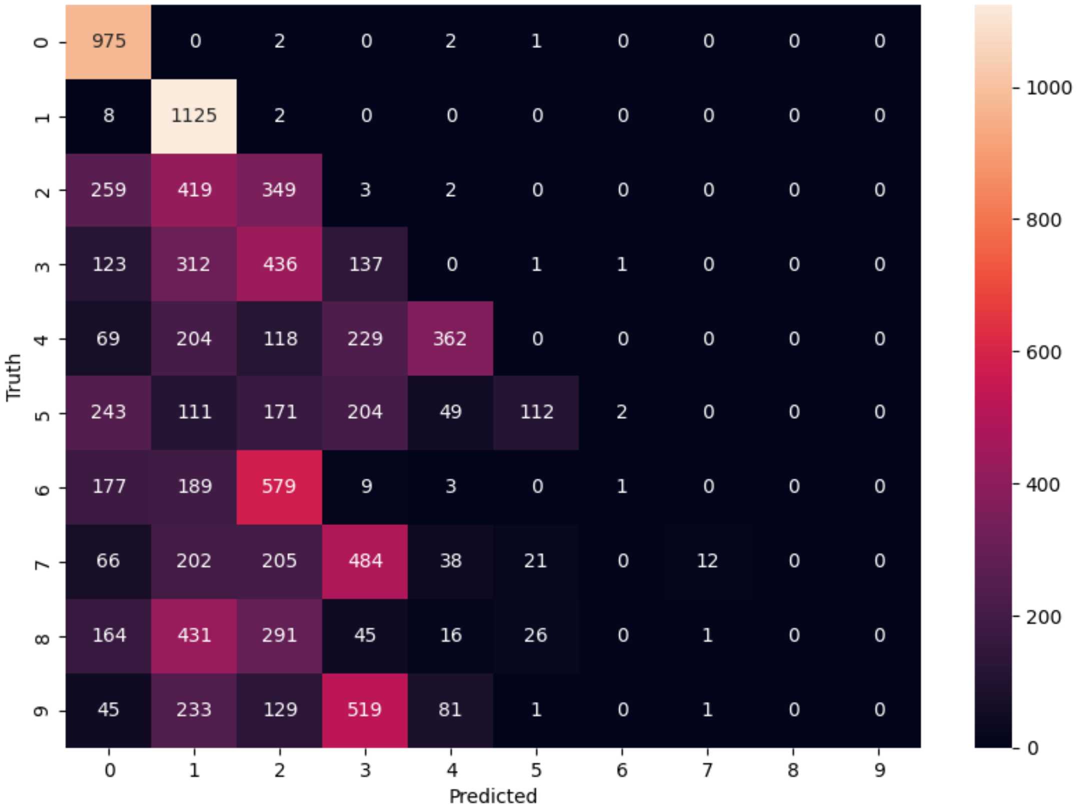
```
     [3, 0, 1, 0, 4, 1, 4, 1, 0, 4]
     [7 2 1 0 4 1 4 9 5 9]
[52]: <tf.Tensor: shape=(10, 10), dtype=int32, numpy=
      array([[ 975,    0,    2,    0,    2,    1,    0,    0,    0,    0],
             [   8, 1125,    2,    0,    0,    0,    0,    0,    0,    0],
             [ 259,  419,  349,    3,    2,    0,    0,    0,    0,    0],
             [ 123,  312,  436,  137,    0,    1,    1,    0,    0,    0],
             [  69,  204,  118,  229,  362,    0,    0,    0,    0,    0],
             [ 243,  111,  171,  204,   49,  112,    2,    0,    0,    0],
             [ 177,  189,  579,    9,    3,    0,    1,    0,    0,    0],
             [  66,  202,  205,  484,   38,   21,    0,   12,    0,    0],
             [ 164,  431,  291,   45,   16,   26,    0,    1,    0,    0],
             [  45,  233,  129,  519,   81,    1,    0,    1,    0,    0]],
            dtype=int32)>
```

[55]:

```
import seaborn as sn
plt.figure(figsize=(10,7))
sn.heatmap(cm,annot=True,fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

[55]:

```
Text(95.72222222222221, 0.5, 'Truth')
```



5 Image Classification using CNN

[1]:

```
# Library for plotting the images and the loss function
import matplotlib.pyplot as plt

# We import the data set from tensorflow and build the model there
import tensorflow as tf
from tensorflow.keras import datasets, layers, models

# Download the data set
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.
 ↪load_data()

# Normalize pixel values between 0 and 1
```

```
train_images, test_images = train_images / 255.0, test_images / 255.0

# Define the 10 image classes
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']

# Show the first 10 images
plt.figure(figsize=(10,10))
for i in range(10):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i])
    # Die CIFAR Labels sind Arrays, deshalb benötigen wir den extra Index
    plt.xlabel(class_names[train_labels[i][0]])
plt.show()
```

```
2023-03-14 12:30:19.138920: I tensorflow/core/platform/cpu_feature_guard.cc:193]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
2023-03-14 12:30:19.520290: W
tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could
not load dynamic library 'libcudart.so.11.0'; dlerror: libcudart.so.11.0: cannot
open shared object file: No such file or directory
2023-03-14 12:30:19.520322: I
tensorflow/compiler/xla/stream_executor/cuda/cudart_stub.cc:29] Ignore above
cudart dlerror if you do not have a GPU set up on your machine.
2023-03-14 12:30:21.418244: W
tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could
not load dynamic library 'libnvinfer.so.7'; dlerror: libnvinfer.so.7: cannot
open shared object file: No such file or directory
2023-03-14 12:30:21.418569: W
tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could
not load dynamic library 'libnvinfer_plugin.so.7'; dlerror:
libnvinfer_plugin.so.7: cannot open shared object file: No such file or
directory
2023-03-14 12:30:21.418592: W
tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Cannot
dlopen some TensorRT libraries. If you would like to use Nvidia GPU with
TensorRT, please make sure the missing libraries mentioned above are installed
properly.
```

[2]:

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        896

 max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 4, 4, 64)          36928

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 64)                65600
 dense_1 (Dense)             (None, 10)                650

=================================================================
Total params: 122,570
Trainable params: 122,570
Non-trainable params: 0
_________________________________________________________________
2023-03-14 12:30:25.837111: W
tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could
not load dynamic library 'libcuda.so.1'; dlerror: libcuda.so.1: cannot open
shared object file: No such file or directory
2023-03-14 12:30:25.837617: W
tensorflow/compiler/xla/stream_executor/cuda/cuda_driver.cc:265] failed call to
cuInit: UNKNOWN ERROR (303)
2023-03-14 12:30:25.837684: I
tensorflow/compiler/xla/stream_executor/cuda/cuda_diagnostics.cc:156] kernel
driver does not appear to be running on this host (bhagath-VivoBook-
ASUSLaptop-X409DA-M409DA): /proc/driver/nvidia/version does not exist
2023-03-14 12:30:25.838800: I tensorflow/core/platform/cpu_feature_guard.cc:193]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
```

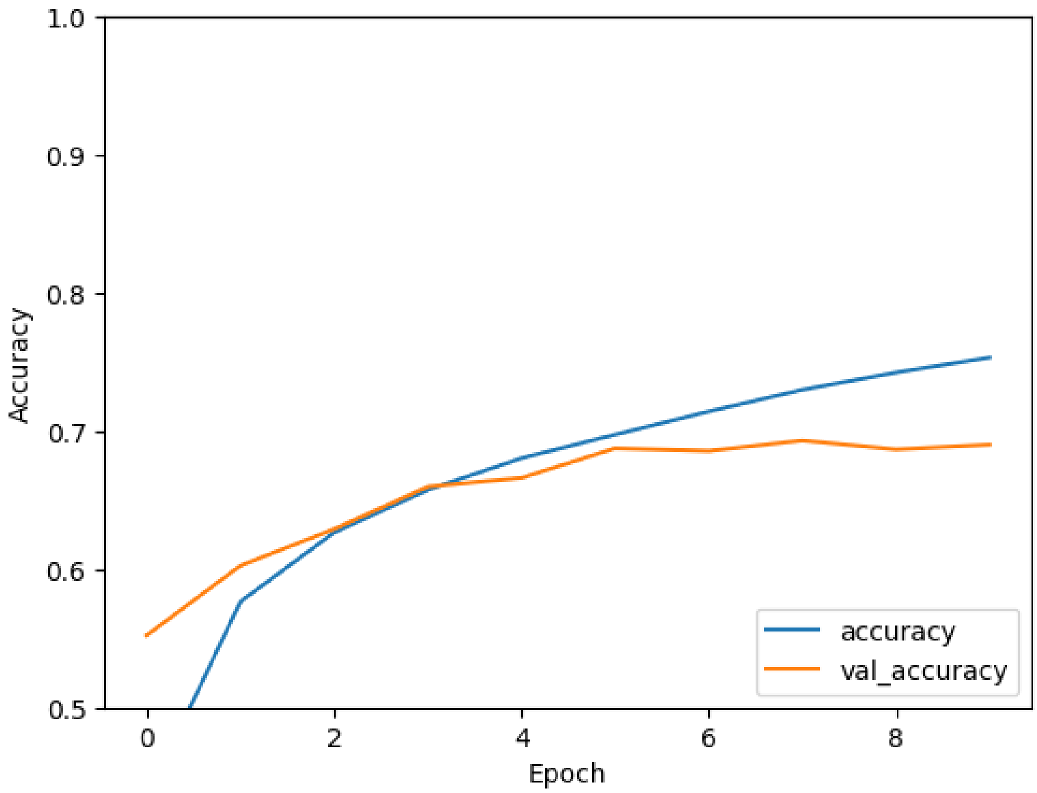
[3]:

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.
 ↪SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

history = model.fit(train_images, train_labels, epochs=10,
                    validation_data=(test_images, test_labels))
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')
```

```
Epoch 1/10

2023-03-14 12:30:26.599568: W tensorflow/tsl/framework/cpu_allocator_impl.cc:82]
Allocation of 614400000 exceeds 10% of free system memory.

1562/1563 [============================>.] - ETA: 0s - loss: 1.5428 - accuracy:
0.4340

2023-03-14 12:30:58.398192: W tensorflow/tsl/framework/cpu_allocator_impl.cc:82]
Allocation of 122880000 exceeds 10% of free system memory.
```

```
1563/1563 [==============================] - 36s 23ms/step - loss: 1.5427 -
accuracy: 0.4341 - val_loss: 1.2397 - val_accuracy: 0.5522
Epoch 2/10
1563/1563 [==============================] - 40s 25ms/step - loss: 1.1885 -
accuracy: 0.5765 - val_loss: 1.1224 - val_accuracy: 0.6026
Epoch 3/10
1563/1563 [==============================] - 39s 25ms/step - loss: 1.0569 -
accuracy: 0.6266 - val_loss: 1.0383 - val_accuracy: 0.6291
Epoch 4/10
1563/1563 [==============================] - 39s 25ms/step - loss: 0.9715 -
accuracy: 0.6574 - val_loss: 0.9661 - val_accuracy: 0.6598
Epoch 5/10
1563/1563 [==============================] - 39s 25ms/step - loss: 0.9072 -
accuracy: 0.6804 - val_loss: 0.9621 - val_accuracy: 0.6661
Epoch 6/10
1563/1563 [==============================] - 39s 25ms/step - loss: 0.8573 -
accuracy: 0.6975 - val_loss: 0.9137 - val_accuracy: 0.6876
Epoch 7/10
1563/1563 [==============================] - 39s 25ms/step - loss: 0.8101 -
accuracy: 0.7142 - val_loss: 0.9097 - val_accuracy: 0.6857
Epoch 8/10
1563/1563 [==============================] - 39s 25ms/step - loss: 0.7694 -
accuracy: 0.7300 - val_loss: 0.9027 - val_accuracy: 0.6932
Epoch 9/10
1563/1563 [==============================] - 39s 25ms/step - loss: 0.7345 -
accuracy: 0.7424 - val_loss: 0.9283 - val_accuracy: 0.6868
Epoch 10/10
1563/1563 [==============================] - 39s 25ms/step - loss: 0.7030 -
accuracy: 0.7533 - val_loss: 0.9350 - val_accuracy: 0.6902
```

```
[3]: <matplotlib.legend.Legend at 0x7f27e5664790>
```



[4]:
```
import numpy as np
model.predict(test_images[0:10])
y_predicted=model.predict(test_images)
print(y_predicted[1])
print(np.argmax(y_predicted[1]))
```

```
1/1 [==============================] - 0s 96ms/step
 15/313 [>...] - ETA: 2s
```

```
2023-03-14 12:36:52.987975: W tensorflow/tsl/framework/cpu_allocator_impl.cc:82]
Allocation of 122880000 exceeds 10% of free system memory.
```

```
313/313 [==============================] - 2s 8ms/step
[ 4.4609394  7.0889387 -3.2336006 -3.6716437 -6.599072  -5.5241814
 -5.16695   -8.889729  11.211372   3.9697392]
8
```

[6]:
```
print(test_labels[1])
```

```
[8]
```

6 Cat vs Dog Experiment

[4]:

```
import tensorflow as tf
from tensorflow import keras
from keras import Sequential
from keras.layers import Dense, Conv2D, MaxPooling2D, Flatten
```

[5]:

```
#Generators
train_data=keras.utils.image_dataset_from_directory(
        directory='CATDOGDATA/train',
        labels='inferred',
        label_mode='int',
        batch_size=32,
        image_size=(256,256)
)
validation_data=keras.utils.image_dataset_from_directory(
        directory='CATDOGDATA/test',
        labels='inferred',
        label_mode='int',
        batch_size=32,
        image_size=(256,256)
)
```

```
Found 557 files belonging to 2 classes.
Found 140 files belonging to 2 classes.
```

[6]:

```
#Normalize
def process(image,label):
    image=tf.cast(image/255,tf.float32)
    return image,label


train_data=train_data.map(process)
validataion_data=validation_data.map(process)
```

[7]:

```
#Create CNN Model
model=Sequential()
model.
 ↪add(Conv2D(32,kernel_size=(3,3),padding='valid',activation='relu',input_shape=(256,256,3)))
model.add(MaxPooling2D(pool_size=(2,2),strides=2,padding='valid'))

model.add(Conv2D(64,kernel_size=(3,3),padding='valid',activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2),strides=2,padding='valid'))

model.add(Conv2D(128,kernel_size=(3,3),padding='valid',activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2),strides=2,padding='valid'))

model.add(Flatten())
model.add(Dense(128,activation='relu'))
model.add(Dense(64,activation='relu'))
model.add(Dense(1,activation='sigmoid'))

model.summary()
```

```
Model: "sequential"
-----------------------------------------------------------------
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 32)      896

 max_pooling2d (MaxPooling2D  (None, 127, 127, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 125, 125, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 62, 62, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 60, 60, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 30, 30, 128)      0
 2D)

 flatten (Flatten)           (None, 115200)            0

 dense (Dense)               (None, 128)               14745728

 dense_1 (Dense)             (None, 64)                8256

 dense_2 (Dense)             (None, 1)                 65

=================================================================
Total params: 14,847,297
Trainable params: 14,847,297
Non-trainable params: 0
-----------------------------------------------------------------
```

[8]:

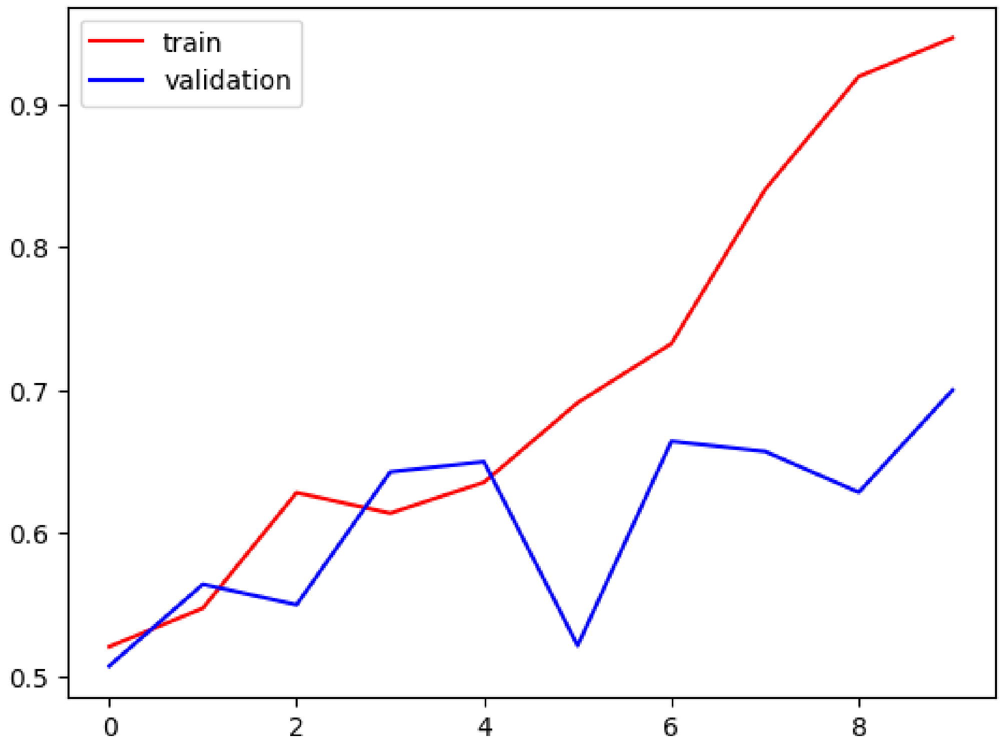
```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
history=model.fit(train_data,epochs=10,validation_data=validation_data)
```

```
Epoch 1/10
18/18 [==============================] - 42s 2s/step - loss: 1.1351 - accuracy:
0.5206 - val_loss: 13.6453 - val_accuracy: 0.5071
Epoch 2/10
18/18 [==============================] - 47s 3s/step - loss: 0.6818 - accuracy:
0.5476 - val_loss: 23.9026 - val_accuracy: 0.5643
Epoch 3/10
```

```
18/18 [==============================] - 45s 2s/step - loss: 0.6667 - accuracy:
0.6284 - val_loss: 59.0386 - val_accuracy: 0.5500
Epoch 4/10
18/18 [==============================] - 40s 2s/step - loss: 0.6624 - accuracy:
0.6140 - val_loss: 49.4017 - val_accuracy: 0.6429
Epoch 5/10
18/18 [==============================] - 45s 2s/step - loss: 0.6606 - accuracy:
0.6355 - val_loss: 31.4972 - val_accuracy: 0.6500
Epoch 6/10
18/18 [==============================] - 45s 2s/step - loss: 0.6146 - accuracy:
0.6912 - val_loss: 76.8635 - val_accuracy: 0.5214
Epoch 7/10
18/18 [==============================] - 42s 2s/step - loss: 0.5208 - accuracy:
0.7325 - val_loss: 74.2570 - val_accuracy: 0.6643
Epoch 8/10
18/18 [==============================] - 47s 2s/step - loss: 0.3430 - accuracy:
0.8402 - val_loss: 127.0416 - val_accuracy: 0.6571
Epoch 9/10
18/18 [==============================] - 43s 2s/step - loss: 0.2273 - accuracy:
0.9192 - val_loss: 180.2071 - val_accuracy: 0.6286
Epoch 10/10
18/18 [==============================] - 43s 2s/step - loss: 0.1428 - accuracy:
0.9461 - val_loss: 226.6793 - val_accuracy: 0.7000
```

[9]:

```
import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'],color='red',label='train')
plt.plot(history.history['val_accuracy'],color='blue',label='validation')
plt.legend()
plt.show()
```



```
[17]: import cv2
      test_img=cv2.imread('test/dog_181.jpg')
      plt.imshow(test_img)
      test_img.shape
      test_img=cv2.resize(test_img,(256,256))
      test_img.shape
      test_input=test_img.reshape((1,256,256,3))
      model.predict(test_input)
```

```
      1/1 [==============================] - 0s 41ms/step
[17]: array([[1.]], dtype=float32)
```

7 Google Stock Prediction using LSTM

[29]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset_train = pd.read_csv('Google_Stock_Price_Train.csv')
training_set=dataset_train.iloc[:,1:2].values
print(dataset_train)
print(training_set)
```

```
            Date    Open    High     Low   Close      Volume
0       1/3/2012  325.25  332.83  324.97  663.59   7,380,500
1       1/4/2012  331.27  333.87  329.08  666.45   5,749,400
2       1/5/2012  329.83  330.75  326.89  657.21   6,590,300
3       1/6/2012  328.34  328.77  323.68  648.24   5,405,900
4       1/9/2012  322.04  322.29  309.46  620.76  11,688,800
...          ...     ...     ...     ...     ...         ...
1253  12/23/2016  790.90  792.74  787.28  789.91     623,400
1254  12/27/2016  790.68  797.86  787.66  791.55     789,100
1255  12/28/2016  793.70  794.23  783.20  785.05   1,153,800
1256  12/29/2016  783.33  785.93  778.92  782.79     744,300
1257  12/30/2016  782.75  782.78  770.41  771.82   1,770,000

[1258 rows x 6 columns]
```

```
[[325.25]
 [331.27]
 [329.83]
 ...
 [793.7 ]
 [783.33]
 [782.75]]
```

[30]:

```
#Perform the feature scaling
from sklearn.preprocessing import MinMaxScaler
sc=MinMaxScaler(feature_range=(0,1))
training_set_scaled=sc.fit_transform(training_set)
print(training_set_scaled)
```

```
[[0.08581368]
 [0.09701243]
 [0.09433366]
 ...
 [0.95725128]
 [0.93796041]
 [0.93688146]]
```

[31]:

```
x_train=[]
y_train=[]

for i in range(60,len(training_set_scaled)):
    x_train.append(training_set_scaled[i-60:i,0])
    y_train.append(training_set_scaled[i, 0])
x_train, y_train=np.array(x_train), np.array(y_train)
print(x_train)
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
```

```
[[0.08581368 0.09701243 0.09433366 ... 0.07846566 0.08034452 0.08497656]
 [0.09701243 0.09433366 0.09156187 ... 0.08034452 0.08497656 0.08627874]
 [0.09433366 0.09156187 0.07984225 ... 0.08497656 0.08627874 0.08471612]
 ...
 [0.92106928 0.92438053 0.93048218 ... 0.95475854 0.95204256 0.95163331]
 [0.92438053 0.93048218 0.9299055  ... 0.95204256 0.95163331 0.95725128]
 [0.93048218 0.9299055  0.93113327 ... 0.95163331 0.95725128 0.93796041]]
```

[37]:

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
model = Sequential()

# Add LSTM layer
model.add(LSTM(units = 50, return_sequences=True, input_shape = (x_train.
 ↪shape[1], 1)))
```

```
# Add Regularization
model.add(Dropout(0.2))

model.add(LSTM(units = 50, return_sequences=True))
model.add(Dropout(0.2))

model.add(LSTM(units = 50, return_sequences=True))
model.add(Dropout(0.2))

model.add(LSTM(units = 50))
model.add(Dropout(0.2))

# Add output layer
model.add(Dense(units = 1))
model.summary()
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(x_train,y_train,epochs=40,batch_size=32)
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_11 (LSTM)              (None, 60, 50)            10400

 dropout_11 (Dropout)        (None, 60, 50)            0

 lstm_12 (LSTM)              (None, 60, 50)            20200

 dropout_12 (Dropout)        (None, 60, 50)            0

 lstm_13 (LSTM)              (None, 60, 50)            20200

 dropout_13 (Dropout)        (None, 60, 50)            0

 lstm_14 (LSTM)              (None, 50)                20200

 dropout_14 (Dropout)        (None, 50)                0

 dense_5 (Dense)             (None, 1)                 51

=================================================================
Total params: 71,051
Trainable params: 71,051
Non-trainable params: 0
_________________________________________________________________
Epoch 1/40
38/38 [==============================] - 15s 112ms/step - loss: 0.0501
```

```
Epoch 2/40
38/38 [==============================] - 4s 113ms/step - loss: 0.0063
Epoch 3/40
38/38 [==============================] - 4s 115ms/step - loss: 0.0050
Epoch 4/40
38/38 [==============================] - 4s 112ms/step - loss: 0.0053
Epoch 5/40
38/38 [==============================] - 4s 114ms/step - loss: 0.0046
Epoch 6/40
38/38 [==============================] - 4s 113ms/step - loss: 0.0043
Epoch 7/40
38/38 [==============================] - 4s 110ms/step - loss: 0.0045
Epoch 8/40
38/38 [==============================] - 5s 125ms/step - loss: 0.0048
Epoch 9/40
38/38 [==============================] - 4s 110ms/step - loss: 0.0047
Epoch 10/40
38/38 [==============================] - 4s 115ms/step - loss: 0.0042
Epoch 11/40
38/38 [==============================] - 4s 111ms/step - loss: 0.0039
Epoch 12/40
38/38 [==============================] - 4s 111ms/step - loss: 0.0045
Epoch 13/40
38/38 [==============================] - 4s 113ms/step - loss: 0.0039
Epoch 14/40
38/38 [==============================] - 4s 112ms/step - loss: 0.0039
Epoch 15/40
38/38 [==============================] - 4s 110ms/step - loss: 0.0037
Epoch 16/40
38/38 [==============================] - 4s 111ms/step - loss: 0.0035
Epoch 17/40
38/38 [==============================] - 5s 145ms/step - loss: 0.0039
Epoch 18/40
38/38 [==============================] - 5s 130ms/step - loss: 0.0033
Epoch 19/40
38/38 [==============================] - 5s 143ms/step - loss: 0.0039
Epoch 20/40
38/38 [==============================] - 5s 132ms/step - loss: 0.0034
Epoch 21/40
38/38 [==============================] - 4s 114ms/step - loss: 0.0030
Epoch 22/40
38/38 [==============================] - 5s 126ms/step - loss: 0.0038
Epoch 23/40
38/38 [==============================] - 5s 123ms/step - loss: 0.0031
Epoch 24/40
38/38 [==============================] - 5s 129ms/step - loss: 0.0032
Epoch 25/40
38/38 [==============================] - 5s 124ms/step - loss: 0.0032
```

```
Epoch 26/40
38/38 [==============================] - 5s 127ms/step - loss: 0.0033
Epoch 27/40
38/38 [==============================] - 4s 116ms/step - loss: 0.0029
Epoch 28/40
38/38 [==============================] - 4s 114ms/step - loss: 0.0033
Epoch 29/40
38/38 [==============================] - 5s 120ms/step - loss: 0.0034
Epoch 30/40
38/38 [==============================] - 5s 119ms/step - loss: 0.0030
Epoch 31/40
38/38 [==============================] - 4s 116ms/step - loss: 0.0030
Epoch 32/40
38/38 [==============================] - 5s 133ms/step - loss: 0.0028
Epoch 33/40
38/38 [==============================] - 5s 120ms/step - loss: 0.0033
Epoch 34/40
38/38 [==============================] - 4s 117ms/step - loss: 0.0027
Epoch 35/40
38/38 [==============================] - 5s 122ms/step - loss: 0.0025
Epoch 36/40
38/38 [==============================] - 4s 117ms/step - loss: 0.0027
Epoch 37/40
38/38 [==============================] - 5s 121ms/step - loss: 0.0029
Epoch 38/40
38/38 [==============================] - 4s 117ms/step - loss: 0.0026
Epoch 39/40
38/38 [==============================] - 5s 119ms/step - loss: 0.0025
Epoch 40/40
38/38 [==============================] - 5s 128ms/step - loss: 0.0034
```

[37]:
```
<keras.callbacks.History at 0x7f26ea6ae080>
```

[33]:
```
test_df=pd.read_csv('Google_Stock_Price_Test.csv')
test_df.head()
stock_price = test_df.iloc[:, 1:2].values
stock_price
```

[33]:
```
array([[778.81],
       [788.36],
       [786.08],
       [795.26],
       [806.4 ],
       [807.86],
       [805.  ],
       [807.14],
       [807.48],
       [807.08],
       [805.81],
       [805.12],
       [806.91],
       [807.25],
       [822.3 ],
       [829.62],
       [837.81],
       [834.71],
       [814.66],
       [796.86]])
```

[36]:

```
# Fetch 60 timesteps by combining train and test got prediction
total_df = pd.concat((dataset_train['Open'], test_df['Open']), axis = 0)
inputs = total_df[0:].values
inputs = inputs.reshape(-1, 1)
inputs = sc.transform(inputs)
# Reshape the dataset

x_test = []

for i in range(60, len(inputs)):
    x_test.append(inputs[i-60:i, 0])

x_test = np.array(x_test)

x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))
predicted_stock_price = model.predict(x_test)
print(predicted_stock_price)

predicted_stock_price = sc.inverse_transform(predicted_stock_price)
predicted_stock_price
```

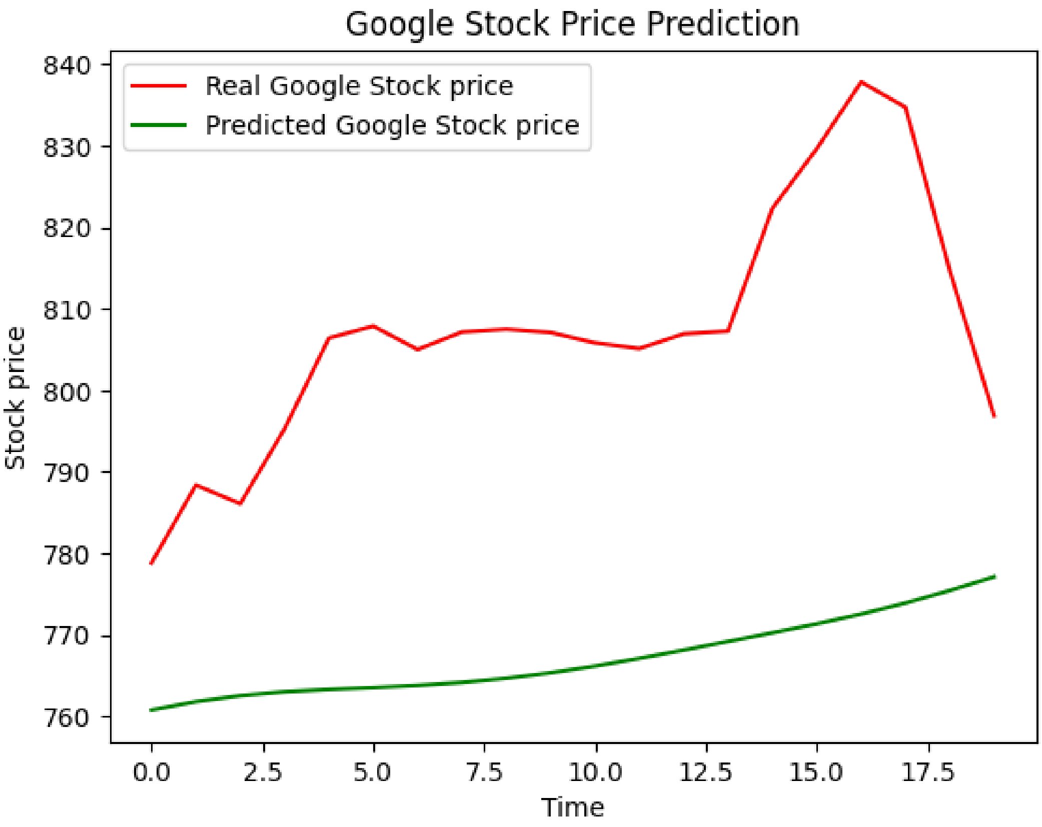
```
39/39 [==============================] - 2s 43ms/step
[[0.05899942]
 [0.06085769]
 [0.06290559]
 ...
 [0.9203946 ]
 [0.92327535]
 [0.9263818 ]]
```

[36]:

```
array([[310.83572],
       [311.83466],
       [312.93552],
       ...,
       [773.8873 ],
       [775.43585],
       [777.1058 ]], dtype=float32)
```

[35]:

```
plt.plot(stock_price, color = 'red', label = 'Real Google Stock price')
plt.plot(predicted_stock_price, color = 'green', label = 'Predicted Google Stock␣
 ↪price')
plt.title("Google Stock Price Prediction")
plt.xlabel('Time')
plt.ylabel('Stock price')
plt.legend()
plt.show()
```



8 Text Predicction using LSTM

[27]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from keras.models import Model
from keras.layers import LSTM, Activation, Dense, Dropout, Input, Embedding
from keras.preprocessing.text import Tokenizer
from keras.preprocessing import sequence
from keras.utils import to_categorical
from keras.callbacks import EarlyStopping
```

[19]:
```
df=pd.read_csv('spam.csv',delimiter=',',encoding='latin-1')
df.head()
df.drop(['Unnamed: 2','Unnamed: 3','Unnamed: 4'],axis=1, inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   v1      5572 non-null   object
 1   v2      5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
0        ham
1        ham
2       spam
3        ham
4        ham
        ...
5567    spam
5568     ham
5569     ham
5570     ham
5571     ham
Name: v1, Length: 5572, dtype: object
```

[20]:
```
X=df.v2
Y=df.v1
le=LabelEncoder()
Y=le.fit_transform(Y)
Y=Y.reshape(-1,1)
print(X)
```

```
0       Go until jurong point, crazy.. Available only ...
1                           Ok lar... Joking wif u oni...
2       Free entry in 2 a wkly comp to win FA Cup fina...
3       U dun say so early hor... U c already then say...
4       Nah I don't think he goes to usf, he lives aro...
                              ...
5567    This is the 2nd time we have tried 2 contact u...
5568                Will Ì_ b going to esplanade fr home?
5569    Pity, * was in mood for that. So...any other s...
5570    The guy did some bitching but I acted like i'd...
5571                               Rofl. Its true to its name
```

```
Name: v2, Length: 5572, dtype: object
```

[32]:
```python
import tensorflow as tf
#Split test and training data
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.15)
#Convert the train data into tokens
max_words=1000
max_len=150
tok=Tokenizer(num_words=max_words)
tok.fit_on_texts(X_train)
sequences=tok.texts_to_sequences(X_train)
sequences_matrix=tf.keras.utils.pad_sequences(sequences,maxlen=max_len)
print(sequences_matrix[0])
print(len(sequences_matrix),len(X_train))
```

```
[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0  13  83  10 116 188  21 145  41 832 194 265 288
  25 436 401 600 107  87]
4736 4736
```

[36]:
```python
def RNN():
    inputs = Input(name='inputs',shape=[max_len])
    layer=Embedding(max_words,50,input_length=max_len)(inputs)
    layer=LSTM(64)(layer)
    layer=Dense(256,name='FC1')(layer)
    layer=Activation('relu')(layer)
    layer=Dropout(0.5)(layer)
    layer=Dense(1,name='out_layer')(layer)
    layer=Activation('sigmoid')(layer)
    model=Model(inputs=inputs,outputs=layer)
    return model
```

[46]:
```python
model=RNN()
model.summary()
model.compile(loss='binary_crossentropy',optimizer=tf.keras.optimizers.
 ↪RMSprop(),metrics=['accuracy'])
model.fit(sequences_matrix,Y_train,batch_size=128,epochs=10,validation_split=0.
 ↪2) #callbacks=[EarlyStopping(monitor='val_loss',min_delta=0.0001)]
```

```
Model: "model_8"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inputs (InputLayer)         [(None, 150)]             0

 embedding_8 (Embedding)     (None, 150, 50)           50000

 lstm_8 (LSTM)               (None, 64)                29440

 FC1 (Dense)                 (None, 256)               16640

 activation_16 (Activation)  (None, 256)               0

 dropout_8 (Dropout)         (None, 256)               0

 out_layer (Dense)           (None, 1)                 257

 activation_17 (Activation)  (None, 1)                 0

=================================================================
Total params: 96,337
Trainable params: 96,337
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
30/30 [==============================] - 6s 137ms/step - loss: 0.3912 -
accuracy: 0.8638 - val_loss: 0.2848 - val_accuracy: 0.8513
Epoch 2/10
30/30 [==============================] - 4s 123ms/step - loss: 0.1849 -
accuracy: 0.9388 - val_loss: 0.1304 - val_accuracy: 0.9736
Epoch 3/10
30/30 [==============================] - 4s 126ms/step - loss: 0.0776 -
accuracy: 0.9802 - val_loss: 0.0717 - val_accuracy: 0.9778
Epoch 4/10
30/30 [==============================] - 4s 127ms/step - loss: 0.0498 -
accuracy: 0.9863 - val_loss: 0.1108 - val_accuracy: 0.9694
Epoch 5/10
30/30 [==============================] - 4s 128ms/step - loss: 0.0431 -
accuracy: 0.9892 - val_loss: 0.0657 - val_accuracy: 0.9800
Epoch 6/10
30/30 [==============================] - 4s 128ms/step - loss: 0.0305 -
accuracy: 0.9926 - val_loss: 0.0657 - val_accuracy: 0.9800
Epoch 7/10
30/30 [==============================] - 4s 128ms/step - loss: 0.0277 -
accuracy: 0.9934 - val_loss: 0.0663 - val_accuracy: 0.9831
Epoch 8/10
30/30 [==============================] - 4s 129ms/step - loss: 0.0204 -
accuracy: 0.9955 - val_loss: 0.0698 - val_accuracy: 0.9831
Epoch 9/10
30/30 [==============================] - 4s 127ms/step - loss: 0.0159 -
accuracy: 0.9963 - val_loss: 0.0826 - val_accuracy: 0.9810
```

```
Epoch 10/10
30/30 [==============================] - 4s 127ms/step - loss: 0.0160 -
accuracy: 0.9968 - val_loss: 0.0769 - val_accuracy: 0.9821
```

[46]:
```
<keras.callbacks.History at 0x7f8b262d86d0>
```

[47]:
```
test_sequences=tok.texts_to_sequences(X_test)
test_sequences_matrix=tf.keras.utils.pad_sequences(test_sequences,maxlen=max_len)
accr=model.evaluate(test_sequences_matrix,Y_test)
print('Test Set \n Loss: {:0.3f}\n Accuracy: {:0.3f}'.format(accr[0],accr[1]))
```

```
27/27 [==============================] - 1s 33ms/step - loss: 0.0545 - accuracy:
0.9868
Test Set
 Loss: 0.054
 Accuracy: 0.987
```

9 Word2Vec Model for Amazon Cell Phone data

[1]:
```
from gensim.models import word2vec, FastText
import gensim
import pandas as pd
import re

from sklearn.decomposition import PCA
from matplotlib import pyplot as plt
import plotly.graph_objects as go
import numpy as np
import warnings
```

[4]:
```
#Read the json file into a pandas dataframe
df=pd.read_json("Cell_Phones_and_Accessories_5.json",lines=True)
df.head()
#df.shape
```

[4]:
```
       reviewerID        asin      reviewerName helpful  \
0  A30TL5EWN6DFXT  120401325X         christina  [0, 0]
1   ASY55RVNIL0UD  120401325X          emily l.  [0, 0]
2  A2TMXE2AF07ONB  120401325X             Erica  [0, 0]
3   AWJ0WZQYMYFQ4  120401325X                JM  [4, 4]
4   ATX7CZYFXI1KW  120401325X  patrice m rogoza  [2, 3]

                                          reviewText  overall  \
0  They look good and stick good! I just don't li...        4
1  These stickers work like the review says they ...        5
2  These are awesome and make my phone look so st...        5
3  Item arrived in great time and was in perfect ...        4
4  awesome! stays on, and looks great. can be use...        5
```

```
                                  summary  unixReviewTime   reviewTime
0                              Looks Good      1400630400  05 21, 2014
1                    Really great product.      1389657600  01 14, 2014
2                          LOVE LOVE LOVE      1403740800  06 26, 2014
3                                   Cute!      1382313600  10 21, 2013
4  leopard home button sticker for iphone 4s      1359849600   02 3, 2013
```

[10]:

```
df.reviewText[1]
#Preprocess the text and store that in a variable called review_text
review_text=df.reviewText.apply(gensim.utils.simple_preprocess)
review_text
```

[10]:

```
0         [they, look, good, and, stick, good, just, don...
1         [these, stickers, work, like, the, review, say...
2         [these, are, awesome, and, make, my, phone, lo...
3         [item, arrived, in, great, time, and, was, in,...
4         [awesome, stays, on, and, looks, great, can, b...
                                ...
194434    [works, great, just, like, my, original, one, ...
194435    [great, product, great, packaging, high, quali...
194436    [this, is, great, cable, just, as, good, as, t...
194437    [really, like, it, becasue, it, works, well, w...
194438    [product, as, described, have, wasted, lot, of...
Name: reviewText, Length: 194439, dtype: object
```

[11]:

```
model=gensim.models.Word2Vec(
    window=10,
    min_count=2,
    workers=4
)
model.build_vocab(review_text,progress_per=1000)
model.train(review_text,total_examples=model.corpus_count,epochs=model.epochs)
```

[11]:

```
(61507408, 83868975)
```

[18]:

```
model.save("amazon_cell.model")
model.wv.most_similar("mobile")
```

[18]:

```
[('prepaid', 0.7679797410964966),
 ('gsm', 0.7454944849014282),
 ('cellular', 0.7317382097244263),
 ('att', 0.6899208426475525),
 ('virgin', 0.6877989768981934),
 ('broadband', 0.6869664788246155),
 ('uma', 0.6837249994277954),
 ('metropcs', 0.6831269860267639),
 ('sprint', 0.6756433248519897),
 ('tmobile', 0.6749399304389954)]
```