Importing Libraries and Dataset

In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import cv2 as cv
import os
import csv
import math
import tensorflow as tf
from tensorflow import keras
import skimage
from skimage.feature import greycomatrix, greycoprops
from skimage import io
from datetime import datetime
from functools import reduce
import datetime as dt
import seaborn as sns
from itertools import chain
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
from imblearn.over_sampling import RandomOverSampler
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from keras.layers import CuDNNLSTM, Dense, Dropout, LSTM
from sklearn.ensemble import RandomForestClassifier
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

In [ ]:

```
## Vehicle 1 ##
v1_lin_acc = pd.read_csv("16/aceleracaoLinear_terra.csv")
v1_acc = pd.read_csv("16/acelerometro_terra.csv")
v1_magneto = pd.read_csv("16/campoMagnetico_terra.csv")
v1_gyro = pd.read_csv("16/giroscopio_terra.csv")
v1_label = pd.read_csv("16/groundTruth.csv")

## Vehicle 2 ##
v2_lin_acc = pd.read_csv("17/aceleracaoLinear_terra.csv")
v2_acc = pd.read_csv("17/acelerometro_terra.csv")
v2_magneto = pd.read_csv("17/campoMagnetico_terra.csv")
v2_gyro = pd.read_csv("17/giroscopio_terra.csv")
v2_label = pd.read_csv("17/groundTruth.csv")

## Vehicle 3 ##
v3_lin_acc = pd.read_csv("20/aceleracaoLinear_terra.csv")
v3_acc = pd.read_csv("20/acelerometro_terra.csv")
v3_magneto = pd.read_csv("20/campoMagnetico_terra.csv")
v3_gyro = pd.read_csv("20/giroscopio_terra.csv")
v3_label = pd.read_csv("20/groundTruth.csv")

## Vehicle 4 ##
v4_lin_acc = pd.read_csv("21/aceleracaoLinear_terra.csv")
v4_acc = pd.read_csv("21/acelerometro_terra.csv")
v4_magneto = pd.read_csv("21/campoMagnetico_terra.csv")
v4_gyro = pd.read_csv("21/giroscopio_terra.csv")
v4_label = pd.read_csv("21/groundTruth.csv")
```

Data Preprocessing

In [ ]:

```
new_label_column_names = ["event","start_time","end_time"]
new_lin_acc_names = ["timestamp","time_nano","x_lin_acc","y_lin_acc","z_lin_acc"]
new_acc_names = ["timestamp","time_nano","x_acc","y_acc","z_acc"]
new_magneto_names = ["timestamp","time_nano","x_magneto","y_magneto","z_magneto"]
new_gyro_names = ["timestamp","time_nano","x_gyro","y_gyro","z_gyro"]


v1_label.columns = v2_label.columns = v3_label.columns = v4_label.columns = new_labe
v1_lin_acc.columns = v2_lin_acc.columns = v3_lin_acc.columns = v4_lin_acc.columns =
v1_acc.columns = v2_acc.columns = v3_acc.columns = v4_acc.columns = new_acc_names
v1_magneto.columns = v2_magneto.columns = v3_magneto.columns = v4_magneto.columns =
v1_gyro.columns = v2_gyro.columns = v3_gyro.columns = v4_gyro.columns = new_gyro_nam

old_label_names = ["evento_nao_agressivo","curva_direita_agressiva","curva_esquerda_
new_label_names = ["Non aggressive","Agressive right turn","Agressive left turn","Ag

v1_label["event"] = v1_label["event"].replace(old_label_names,new_label_names)
v2_label["event"] = v2_label["event"].replace(old_label_names,new_label_names)
v3_label["event"] = v3_label["event"].replace(old_label_names,new_label_names)
v4_label["event"] = v4_label["event"].replace(old_label_names,new_label_names)

v1_label
```

Out[ ]:

	event	start_time	end_time
0	Non aggressive	2.0	6.5
1	Agressive right turn	19.5	23.5
2	Non aggressive	30.0	33.5
3	Agressive right turn	95.0	98.0
4	Agressive left turn	247.0	251.5
5	Agressive left turn	348.7	352.3
6	Non aggressive	485.0	489.0
7	Agressive left turn	496.0	499.5
8	Agressive right turn	587.0	590.0
9	Agressive left turn	750.0	753.8
10	Agressive right turn	840.7	844.0
11	Agressive right turn	980.0	983.2
12	Agressive left turn	1087.4	1090.9
13	Aggressive right lane change	1139.8	1142.0
14	Aggressive right lane change	1201.0	1202.9
15	Aggressive right lane change	1211.4	1213.5

In [ ]:

```
v1_label["event"] = v1_label["event"].replace(old_label_names,new_label_names)
v2_label["event"] = v2_label["event"].replace(old_label_names,new_label_names)
v3_label["event"] = v3_label["event"].replace(old_label_names,new_label_names)
v4_label["event"] = v4_label["event"].replace(old_label_names,new_label_names)
```

In [ ]:

```
len(v1_lin_acc), len(v1_acc), len(v1_gyro),len(v1_magneto["time_nano"].unique()),len
```

Out[ ]:
```
(64645,
 64645,
 64645,
 64645,
 20675,
 20675,
 20675,
 20675,
 30014,
 30014,
 30014,
 30014,
 41178,
 41178,
 41178,
 41178)
```

In [ ]:
```
v1_magneto['time_centiseconds'] = (np.array(v1_magneto['time_nano'])/100000000).asty
v1_magneto
```

Out[ ]:

	timestamp	time_nano	x_magneto	y_magneto	z_magneto	time_centiseconds
0	14/05/2016 10:54:33	11537628641797	-1.184853e-07	40.994006	-7.966440	115376
1	14/05/2016 10:54:33	11537648266397	-6.832270e-07	41.030238	-8.112972	115376
2	14/05/2016 10:54:33	11537667890997	-1.365388e-07	40.863243	-7.899277	115376
3	14/05/2016 10:54:33	11537667890997	-6.260234e-08	40.862746	-7.901860	115376
4	14/05/2016 10:54:33	11537687485076	-9.552314e-07	40.778226	-8.027510	115376
...	...	...	...	...	...	...
128951	14/05/2016 11:15:42	12806689529792	-1.117587e-08	15.251873	-10.477536	128066
128952	14/05/2016 11:15:42	12806709337514	-1.117587e-08	15.045683	-10.811708	128067
128953	14/05/2016 11:15:42	12806709337514	-4.228204e-07	15.057556	-10.795168	128067
128954	14/05/2016 11:15:42	12806728840032	-4.479662e-07	15.153189	-10.435536	128067
128955	14/05/2016 11:15:42	12806728840032	2.672896e-07	15.156427	-10.430830	128067

128956 rows × 6 columns

In [ ]:
```
v1_lin_acc['time_centiseconds'] = (np.array(v1_lin_acc['time_nano'])/100000000).asty
v1_lin_acc
```

Out[ ]:

	timestamp	time_nano	x_lin_acc	y_lin_acc	z_lin_acc	time_centiseconds
0	14/05/2016 10:54:33	11537640270059	-0.161602	0.120174	-0.209893	115376
1	14/05/2016 10:54:33	11537650128140	-0.122628	0.315638	-0.380996	115376

	timestamp	time_nano	x_lin_acc	y_lin_acc	z_lin_acc	time_centiseconds
2	14/05/2016 10:54:33	11537659894659	-0.178777	0.330181	-0.360696	115376
3	14/05/2016 10:54:33	11537679549779	0.016043	0.038759	-0.278204	115376
4	14/05/2016 10:54:33	11537699204899	0.141716	-0.162492	-0.049796	115376
...	...	...	...	...	...	...
64640	14/05/2016 11:15:42	12806642253734	-0.462898	-0.330468	-0.780452	128066
64641	14/05/2016 11:15:42	12806661878334	0.274546	0.864909	0.532109	128066
64642	14/05/2016 11:15:42	12806681472414	0.711429	0.930382	1.186323	128066
64643	14/05/2016 11:15:42	12806701066493	0.674247	-0.342384	0.097795	128067
64644	14/05/2016 11:15:42	12806720843694	0.203667	-1.021377	-0.435942	128067

64645 rows × 6 columns

In [ ]:

```
v1_acc['time_centiseconds'] = (np.array(v1_acc['time_nano'])/100000000).astype(int)
v1_acc
```

Out[ ]:

	timestamp	time_nano	x_acc	y_acc	z_acc	time_centiseconds
0	14/05/2016 10:54:33	11537640270059	-0.161602	0.120174	9.596758	115376
1	14/05/2016 10:54:33	11537650128140	-0.122628	0.315638	9.425655	115376
2	14/05/2016 10:54:33	11537659894659	-0.178777	0.330180	9.445955	115376
3	14/05/2016 10:54:33	11537679549779	0.016043	0.038759	9.528445	115376
4	14/05/2016 10:54:33	11537699204899	0.141716	-0.162492	9.756854	115376
...	...	...	...	...	...	...
64640	14/05/2016 11:15:42	12806642253734	-0.462898	-0.330468	9.026200	128066
64641	14/05/2016 11:15:42	12806661878334	0.274546	0.864909	10.338760	128066
64642	14/05/2016 11:15:42	12806681472414	0.711429	0.930382	10.992975	128066
64643	14/05/2016 11:15:42	12806701066493	0.674247	-0.342384	9.904447	128067
64644	14/05/2016 11:15:42	12806720843694	0.203667	-1.021377	9.370710	128067

64645 rows × 6 columns

In [ ]:

```
v1_gyro['time_centiseconds'] = (np.array(v1_gyro['time_nano'])/100000000).astype(int
v1_gyro
```

Out[ ]:

	timestamp	time_nano	x_gyro	y_gyro	z_gyro	time_centiseconds
0	14/05/2016 10:54:33	11537635386799	-0.070372	0.000844	0.029619	115376
1	14/05/2016 10:54:33	11537645580604	-0.058695	0.009130	0.024406	115376
2	14/05/2016 10:54:33	11537667280589	0.006625	-0.002283	-0.015018	115376
3	14/05/2016 10:54:33	11537684829803	0.064933	0.033172	-0.040503	115376
4	14/05/2016 10:54:33	11537704912208	0.039454	-0.013078	-0.007681	115377

	timestamp	time_nano	x_gyro	y_gyro	z_gyro	time_centiseconds
...	...	...	...	...	...	...
64640	14/05/2016 11:15:42	12806647564279	-0.194610	0.150351	-0.012446	128066
64641	14/05/2016 11:15:42	12806667158359	-0.167431	0.199986	0.020079	128066
64642	14/05/2016 11:15:42	12806686752439	0.064197	0.053792	0.079096	128066
64643	14/05/2016 11:15:42	12806706438078	0.179586	-0.106454	0.048294	128067
64644	14/05/2016 11:15:42	12806726062678	0.067985	-0.133239	-0.087171	128067

64645 rows × 6 columns

In [ ]:

```
len(v1_lin_acc['time_centiseconds'].unique())
```

Out[ ]: 12692

In [ ]:

```
v1_lin_acc['time_centiseconds'] = (np.array(v1_lin_acc['time_nano'])/100000000).asty
v1_acc['time_centiseconds'] = (np.array(v1_acc['time_nano'])/100000000).astype(int)
v1_magneto['time_centiseconds'] = (np.array(v1_magneto['time_nano'])/100000000).asty
v1_gyro['time_centiseconds'] = (np.array(v1_gyro['time_nano'])/100000000).astype(int

v2_lin_acc['time_centiseconds'] = (np.array(v2_lin_acc['time_nano'])/100000000).asty
v2_acc['time_centiseconds'] = (np.array(v2_acc['time_nano'])/100000000).astype(int)
v2_magneto['time_centiseconds'] = (np.array(v2_magneto['time_nano'])/100000000).asty
v2_gyro['time_centiseconds'] = (np.array(v2_gyro['time_nano'])/100000000).astype(int

v3_lin_acc['time_centiseconds'] = (np.array(v3_lin_acc['time_nano'])/100000000).asty
v3_acc['time_centiseconds'] = (np.array(v3_acc['time_nano'])/100000000).astype(int)
v3_magneto['time_centiseconds'] = (np.array(v3_magneto['time_nano'])/100000000).asty
v3_gyro['time_centiseconds'] = (np.array(v3_gyro['time_nano'])/100000000).astype(int

v4_lin_acc['time_centiseconds'] = (np.array(v4_lin_acc['time_nano'])/100000000).asty
v4_acc['time_centiseconds'] = (np.array(v4_acc['time_nano'])/100000000).astype(int)
v4_magneto['time_centiseconds'] = (np.array(v4_magneto['time_nano'])/100000000).asty
v4_gyro['time_centiseconds'] = (np.array(v4_gyro['time_nano'])/100000000).astype(int
```

In [ ]:

```
v1_magneto = v1_magneto.groupby('time_centiseconds', as_index=False, sort=False)['x_
v2_magneto = v2_magneto.groupby('time_centiseconds', as_index=False, sort=False)['x_
v3_magneto = v3_magneto.groupby('time_centiseconds', as_index=False, sort=False)['x_
v4_magneto = v4_magneto.groupby('time_centiseconds', as_index=False, sort=False)['x_

v1_gyro = v1_gyro.groupby('time_centiseconds', as_index=False, sort=False)['x_gyro',
v2_gyro = v2_gyro.groupby('time_centiseconds', as_index=False, sort=False)['x_gyro',
v3_gyro = v3_gyro.groupby('time_centiseconds', as_index=False, sort=False)['x_gyro',
v4_gyro = v4_gyro.groupby('time_centiseconds', as_index=False, sort=False)['x_gyro',


v1_acc = v1_acc.groupby('time_centiseconds', as_index=False, sort=False)['x_acc','y_
v2_acc = v2_acc.groupby('time_centiseconds', as_index=False, sort=False)['x_acc','y_
v3_acc = v3_acc.groupby('time_centiseconds', as_index=False, sort=False)['x_acc','y_
v4_acc = v4_acc.groupby('time_centiseconds', as_index=False, sort=False)['x_acc','y_


v1_lin_acc = v1_lin_acc.groupby('time_centiseconds', as_index=False, sort=False)['x_
v2_lin_acc = v2_lin_acc.groupby('time_centiseconds', as_index=False, sort=False)['x_
v3_lin_acc = v3_lin_acc.groupby('time_centiseconds', as_index=False, sort=False)['x_
v4_lin_acc = v4_lin_acc.groupby('time_centiseconds', as_index=False, sort=False)['x_
```

In [ ]:
```
v1_magneto
```

Out[ ]:

	time_centiseconds	x_magneto	y_magneto	z_magneto
0	115376	-2.544296e-07	40.885231	-7.985273
1	115377	-9.658916e-08	40.611182	-7.537545
2	115378	-1.441941e-07	40.690295	-7.378162
3	115379	-1.575346e-07	41.221873	-7.740907
4	115380	3.883603e-08	41.311051	-8.667088
...	...	...	...	...
12687	128063	1.620967e-07	14.970525	-10.320365
12688	128064	-1.456589e-07	15.004607	-10.723637
12689	128065	4.847534e-08	15.097503	-10.660156
12690	128066	-1.252629e-07	15.290653	-10.628135
12691	128067	-1.536682e-07	15.103214	-10.618311

12692 rows × 4 columns

In [ ]:
```
len(v1_lin_acc), len(v1_acc), len(v1_gyro),len(v1_magneto),len(v2_lin_acc), len(v2_a
```

Out[ ]:
```
(12692,
 12692,
 12692,
 12692,
 4059,
 4059,
 4059,
 4059,
 5892,
 5892,
 5892,
 5892,
 8084,
 8084,
 8084,
 8084)
```

In [ ]:
```
v1_lin_acc.set_index('time_centiseconds',inplace=True)
v1_acc.set_index('time_centiseconds',inplace=True)
v1_magneto.set_index('time_centiseconds',inplace=True)
v1_gyro.set_index('time_centiseconds',inplace=True)
vehicle_1 = pd.concat([v1_lin_acc, v1_acc, v1_magneto, v1_gyro], axis=1, sort=False)
vehicle_1.rename(columns = {'index':'time_centiseconds'})
#vehicle_1['time_centiseconds'] = pd.to_datetime(vehicle_1['time_centiseconds'])
initial_time1 = [vehicle_1['time_centiseconds'][0]]*len(vehicle_1)
#initial_time1 = pd.to_datetime(initial_time1)
vehicle_1['Time seconds'] = np.array(vehicle_1['time_centiseconds']-initial_time1)/1

v2_lin_acc.set_index('time_centiseconds',inplace=True)
v2_acc.set_index('time_centiseconds',inplace=True)
v2_magneto.set_index('time_centiseconds',inplace=True)
v2_gyro.set_index('time_centiseconds',inplace=True)
vehicle_2 = pd.concat([v2_lin_acc, v2_acc, v2_magneto, v2_gyro], axis=1, sort=False)
```

```
vehicle_2.rename(columns = {'index':'time_centiseconds'})
#vehicle_2['time_centiseconds'] = pd.to_datetime(vehicle_1['time_centiseconds'])
initial_time2 = [vehicle_2['time_centiseconds'][0]]*len(vehicle_2)
#initial_time2 = pd.to_datetime(initial_time2)
vehicle_2['Time seconds'] = np.array(vehicle_2['time_centiseconds']-initial_time2)/1

v3_lin_acc.set_index('time_centiseconds',inplace=True)
v3_acc.set_index('time_centiseconds',inplace=True)
v3_magneto.set_index('time_centiseconds',inplace=True)
v3_gyro.set_index('time_centiseconds',inplace=True)
vehicle_3 = pd.concat([v3_lin_acc, v3_acc, v3_magneto, v3_gyro], axis=1, sort=False)
vehicle_3.rename(columns = {'index':'time_centiseconds'})
#vehicle_3['time_centiseconds'] = pd.to_datetime(vehicle_3['time_centiseconds'])
initial_time3 = [vehicle_3['time_centiseconds'][0]]*len(vehicle_3)
#initial_time3 = pd.to_datetime(initial_time3)
vehicle_3['Time seconds'] = np.array(vehicle_3['time_centiseconds']-initial_time3)/1

v4_lin_acc.set_index('time_centiseconds',inplace=True)
v4_acc.set_index('time_centiseconds',inplace=True)
v4_magneto.set_index('time_centiseconds',inplace=True)
v4_gyro.set_index('time_centiseconds',inplace=True)
vehicle_4 = pd.concat([v4_lin_acc, v4_acc, v4_magneto, v4_gyro], axis=1, sort=False)
vehicle_4.rename(columns = {'index':'time_centiseconds'})
#vehicle_4['time_centiseconds'] = pd.to_datetime(vehicle_4['time_centiseconds'])
initial_time4 = [vehicle_4['time_centiseconds'][0]]*len(vehicle_4)
#initial_time4 = pd.to_datetime(initial_time4)
vehicle_4['Time seconds'] = np.array(vehicle_4['time_centiseconds']-initial_time4)/1
```

In [ ]:

```
vehicle_3
```

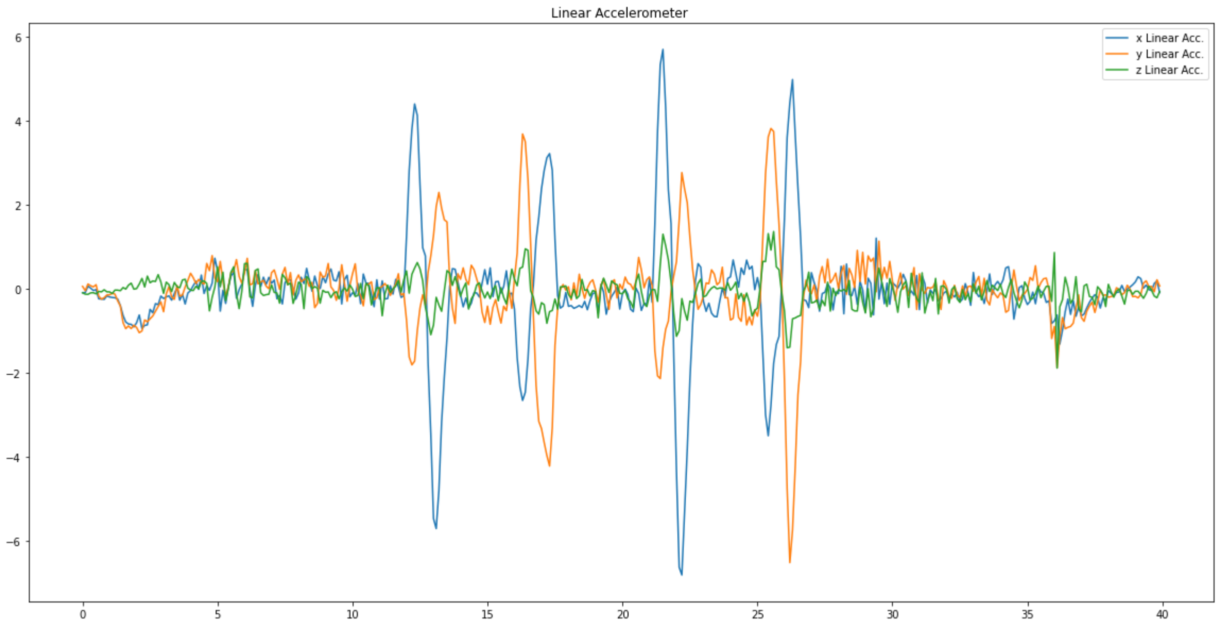
Out[ ]:

	time_centiseconds	x_lin_acc	y_lin_acc	z_lin_acc	x_acc	y_acc	z_acc	x_magneto
0	101969	-0.057714	-0.262420	-0.122779	-0.057714	-0.262420	9.683872	5.634502e-08
1	101970	-0.084763	-0.123271	-0.197237	-0.084763	-0.123271	9.609414	8.915085e-08
2	101971	-0.045146	-0.016721	-0.126474	-0.045146	-0.016721	9.680177	-8.381903e-10
3	101972	0.030745	-0.182721	-0.214915	0.030745	-0.182721	9.591736	7.235212e-08
4	101973	0.055466	-0.032278	-0.147409	0.055466	-0.032278	9.659242	-1.245838e-07
...	...	...	...	...	...	...	...	...
5887	107856	0.012417	-0.022259	-0.192566	0.012417	-0.022259	9.614084	-1.864741e-07
5888	107857	0.029609	0.016640	-0.142156	0.029609	0.016640	9.664494	1.533306e-07
5889	107858	-0.014249	0.119766	-0.152492	-0.014249	0.119766	9.654159	-1.196982e-07
5890	107859	0.078582	-0.010420	-0.178355	0.078582	-0.010420	9.628296	-1.957756e-07
5891	107860	-0.042076	-0.243651	-0.293133	-0.042076	-0.243651	9.513519	1.045846e-07

5892 rows × 14 columns

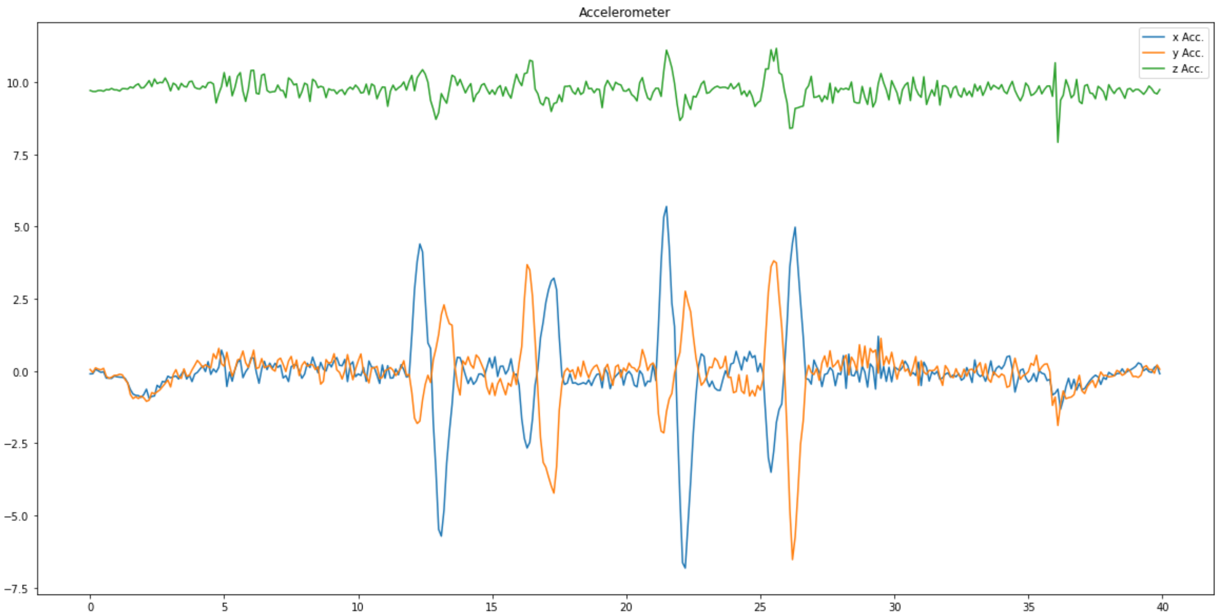
In [ ]:

```
#### visualisation #####
###used vehicle 2 data

y_points1 = np.array(vehicle_2['x_lin_acc'][0:400])
y_points2 = np.array(vehicle_2['y_lin_acc'][0:400])
y_points3 = np.array(vehicle_2['z_lin_acc'][0:400])
x_points = np.array(vehicle_2['Time seconds'][0:400])

plt.figure(figsize=(20,10))
plt.plot(x_points, y_points1, label="x Linear Acc.")
plt.plot(x_points, y_points2, label="y Linear Acc.")
plt.plot(x_points, y_points3, label="z Linear Acc.")
plt.legend(loc='upper right')
plt.title("Linear Accelerometer ")
plt.show()
```



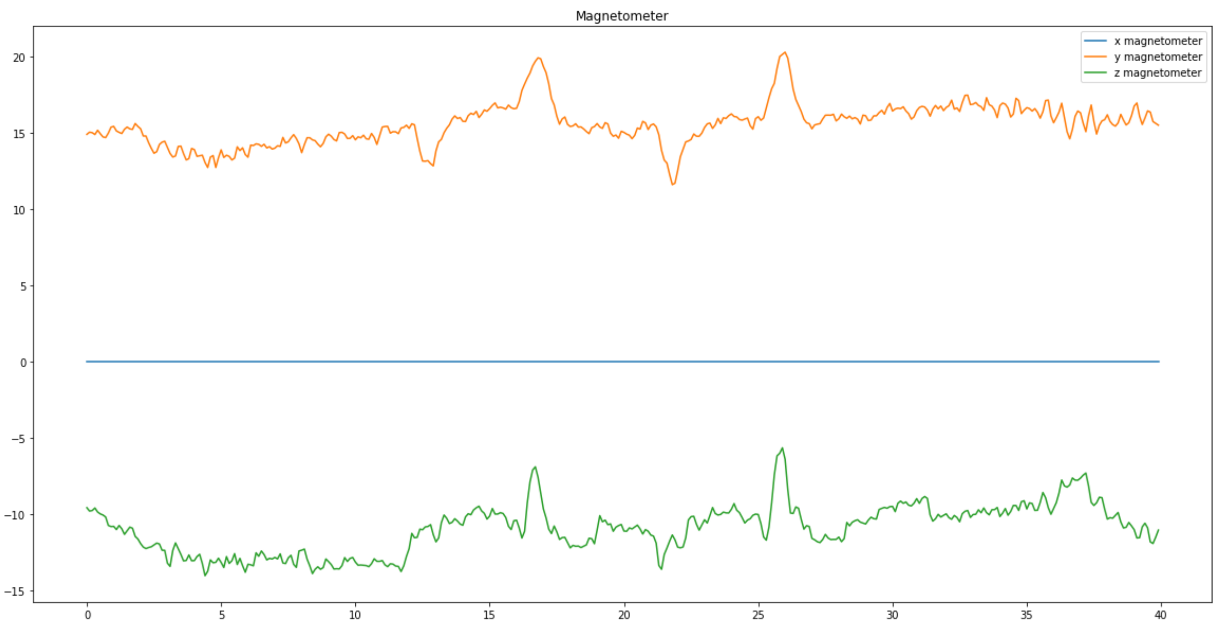
In [ ]:

```
y_points4 = np.array(vehicle_2['x_acc'][0:400])
y_points5 = np.array(vehicle_2['y_acc'][0:400])
y_points6 = np.array(vehicle_2['z_acc'][0:400])
x_points = np.array(vehicle_2['Time seconds'][0:400])

plt.figure(figsize=(20,10))
plt.plot(x_points, y_points4, label="x Acc.")
plt.plot(x_points, y_points5, label="y Acc.")
plt.plot(x_points, y_points6, label="z Acc.")
plt.legend(loc='upper right')
plt.title("Accelerometer")
plt.show()
```



In [ ]:

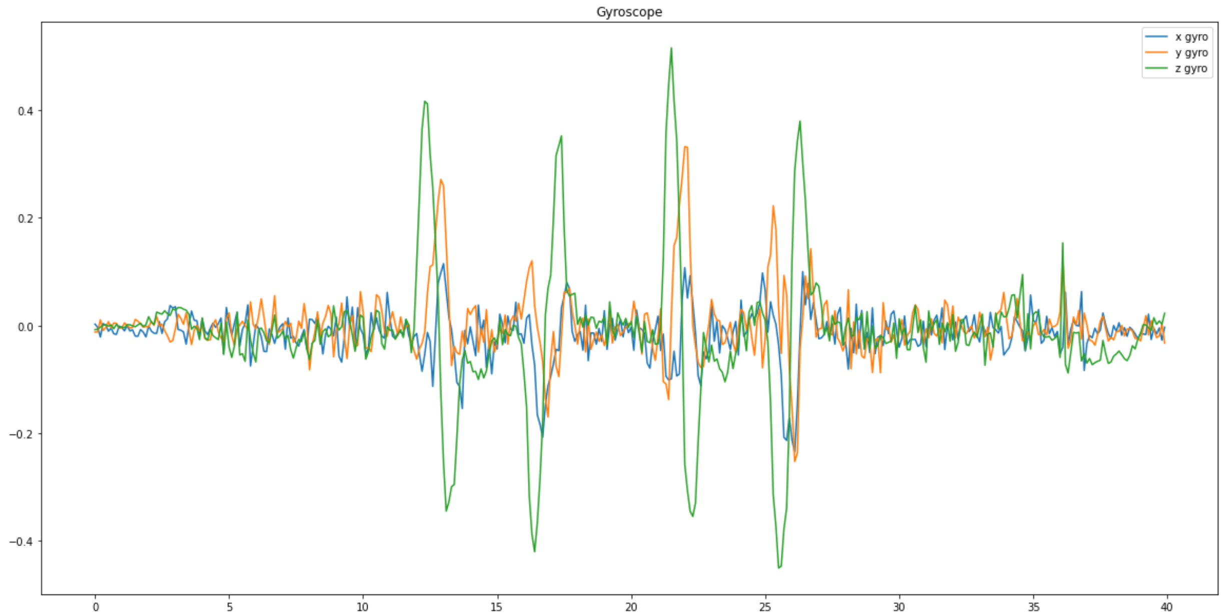
```
y_points7 = np.array(vehicle_2['x_magneto'][0:400])
y_points8 = np.array(vehicle_2['y_magneto'][0:400])
y_points9 = np.array(vehicle_2['z_magneto'][0:400])
x_points = np.array(vehicle_2['Time seconds'][0:400])

plt.figure(figsize=(20,10))
plt.plot(x_points, y_points7, label="x magnetometer")
plt.plot(x_points, y_points8, label="y magnetometer")
plt.plot(x_points, y_points9, label="z magnetometer")
plt.legend(loc='upper right')
plt.title("Magnetometer")
plt.show()
```



In [ ]:

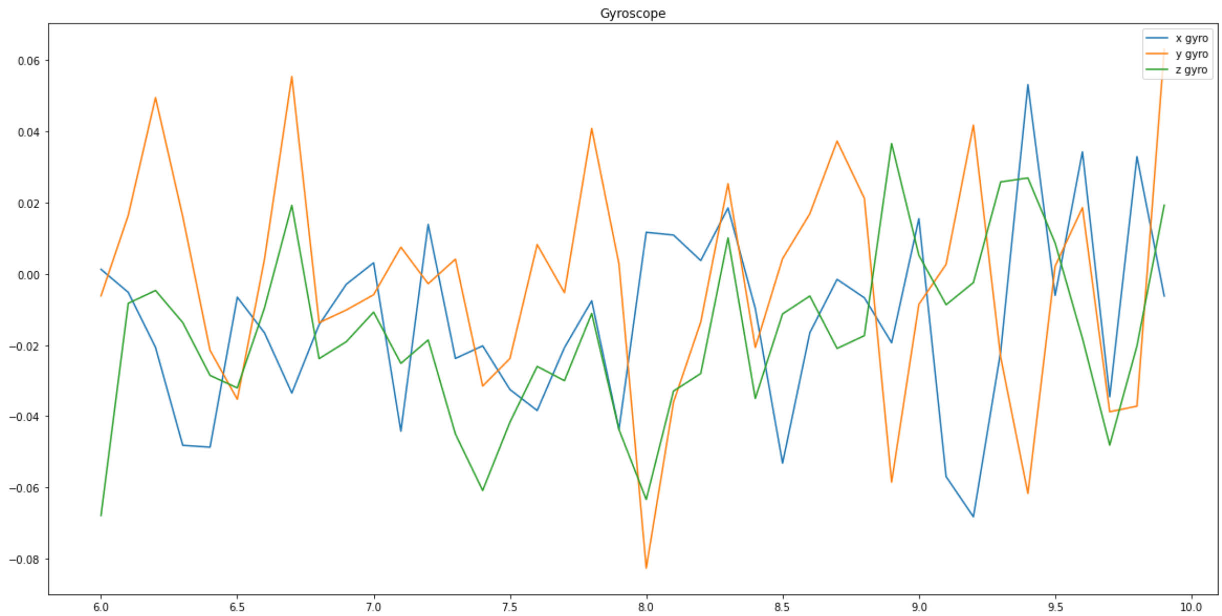
```
y_points10 = np.array(vehicle_2['x_gyro'][0:400])
y_points11 = np.array(vehicle_2['y_gyro'][0:400])
y_points12 = np.array(vehicle_2['z_gyro'][0:400])
x_points = np.array(vehicle_2['Time seconds'][0:400])

plt.figure(figsize=(20,10))
plt.plot(x_points, y_points10, label="x gyro")
plt.plot(x_points, y_points11, label="y gyro")
plt.plot(x_points, y_points12, label="z gyro")
```

```
plt.legend(loc='upper right')
plt.title("Gyroscope")
plt.show()
```



In [ ]:

```
y_points10 = np.array(vehicle_2['x_gyro'][60:100])
y_points11 = np.array(vehicle_2['y_gyro'][60:100])
y_points12 = np.array(vehicle_2['z_gyro'][60:100])
x_points = np.array(vehicle_2['Time seconds'][60:100])

plt.figure(figsize=(20,10))
plt.plot(x_points, y_points10, label="x gyro")
plt.plot(x_points, y_points11, label="y gyro")
plt.plot(x_points, y_points12, label="z gyro")
plt.legend(loc='upper right')
plt.title("Gyroscope")
plt.show()
```



In [ ]:

```
labels = []
new_vehicle1 = pd.DataFrame()

for i in range(len(vehicle_1)):
    present = False
```

```python
    for j in range(len(v1_label)):
        if vehicle_1['Time seconds'][i]<=v1_label['end_time'][j] and vehicle_1['Time
            labels.append(v1_label['event'][j])
            row = vehicle_1.iloc[[i]]
            new_vehicle1 = new_vehicle1.append(row, ignore_index=True)
            present = True
            break
new_vehicle1['event'] = labels


###########
labels = []
new_vehicle2 = pd.DataFrame()

for i in range(len(vehicle_2)):
    present = False
    for j in range(len(v2_label)):
        if vehicle_2['Time seconds'][i]<=v2_label['end_time'][j] and vehicle_2['Time
            labels.append(v2_label['event'][j])
            row = vehicle_2.iloc[[i]]
            new_vehicle2 = new_vehicle2.append(row, ignore_index=True)
            present = True
            break
new_vehicle2['event'] = labels


###########
labels = []
new_vehicle3 = pd.DataFrame()

for i in range(len(vehicle_3)):
    present = False
    for j in range(len(v3_label)):
        if vehicle_3['Time seconds'][i]<=v3_label['end_time'][j] and vehicle_3['Time
            labels.append(v3_label['event'][j])
            row = vehicle_3.iloc[[i]]
            new_vehicle3 = new_vehicle3.append(row, ignore_index=True)
            present = True
            break
new_vehicle3['event'] = labels


############
labels = []
new_vehicle4 = pd.DataFrame()

for i in range(len(vehicle_4)):
    present = False
    for j in range(len(v4_label)):
        if vehicle_4['Time seconds'][i]<=v4_label['end_time'][j] and vehicle_4['Time
            labels.append(v4_label['event'][j])
            row = vehicle_4.iloc[[i]]
            new_vehicle4 = new_vehicle4.append(row, ignore_index=True)
            present = True
            break
new_vehicle4['event'] = labels
```

In [ ]:

```python
labels
```

Out[ ]:

```
['Aggressive left lane change',
 'Aggressive left lane change',
```

```
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
```

```
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Aggressive left lane change',
'Non aggressive',
```

```
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Aggressive left lane change',
 'Aggressive left lane change',
 'Aggressive left lane change',
 'Aggressive left lane change',
 'Aggressive left lane change',
 'Aggressive left lane change',
 'Aggressive left lane change',
 'Aggressive left lane change',
 'Aggressive left lane change',
 'Aggressive left lane change',
 'Aggressive left lane change',
 'Aggressive left lane change',
 'Aggressive left lane change',
 'Aggressive left lane change',
 'Aggressive left lane change',
 'Aggressive left lane change',
 'Aggressive left lane change',
 'Aggressive left lane change',
 'Aggressive left lane change',
 'Aggressive left lane change',
 'Aggressive left lane change',
 'Aggressive left lane change',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
```

```
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Non aggressive',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
```

```
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
```

```
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Agressive braking',
'Aggressive acceleration',
'Aggressive acceleration',
'Aggressive acceleration',
'Aggressive acceleration',
'Aggressive acceleration',
'Aggressive acceleration',
'Aggressive acceleration',
'Aggressive acceleration',
'Aggressive acceleration',
'Aggressive acceleration',
'Aggressive acceleration',
'Aggressive acceleration',
```

```
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
                    'Aggressive acceleration',
```

```
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
            'Aggressive acceleration',
```

```
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Aggressive acceleration',
        'Non aggressive',
        'Non aggressive',
        'Non aggressive',
        'Non aggressive',
        'Non aggressive',
        'Non aggressive',
        'Non aggressive',
        'Non aggressive',
        'Non aggressive',
        'Non aggressive',
        'Non aggressive',
        'Non aggressive',
        'Non aggressive',
        'Non aggressive',
        'Non aggressive',
        'Non aggressive',
        'Non aggressive',
```

```
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive',
 'Non aggressive']
```

In [ ]:

```
new_vehicle2
```

Out[ ]:

	time_centiseconds	x_lin_acc	y_lin_acc	z_lin_acc	x_acc	y_acc	z_acc	x_magneto
0	129093	-1.653038	0.840594	0.063234	-1.653038	0.840595	9.869885	-9.164214e-08
1	129094	-2.341093	2.488108	0.478537	-2.341093	2.488108	10.285188	2.058223e-08
2	129095	-2.667343	3.676871	0.506474	-2.667343	3.676871	10.313124	1.687557e-07

	time_centiseconds	x_lin_acc	y_lin_acc	z_lin_acc	x_acc	y_acc	z_acc	x_magneto
3	129096	-2.473614	3.493380	0.942907	-2.473614	3.493380	10.749558	-2.123415e-08
4	129097	-1.675308	2.593474	0.907588	-1.675308	2.593474	10.714238	1.490116e-08
...	...	...	...	...	...	...	...	...
426	132930	-0.024073	-2.214347	-0.237045	-0.024073	-2.214347	9.569605	1.549197e-07
427	132931	0.142740	-2.247065	-0.314202	0.142740	-2.247065	9.492449	1.036591e-07
428	132932	-0.331480	-1.940924	-0.689715	-0.331480	-1.940924	9.116937	-6.466435e-08
429	132933	-0.423305	-1.996444	-0.122956	-0.423305	-1.996444	9.683695	2.318606e-07
430	132934	-0.290728	-1.159422	-0.238115	-0.290728	-1.159422	9.568535	-1.138476e-07

431 rows × 15 columns

In [ ]:

```
vehicle_1.iloc[[10]]
```

Out[ ]:

	time_centiseconds	x_lin_acc	y_lin_acc	z_lin_acc	x_acc	y_acc	z_acc	x_magneto	y_m
10	115386	0.102623	0.390929	-0.174528	0.102623	0.390929	9.632123	-2.565648e-08	4

In [ ]:

```
new_vehicle3
```

Out[ ]:

	time_centiseconds	x_lin_acc	y_lin_acc	z_lin_acc	x_acc	y_acc	z_acc	x_magneto
0	102064	2.550530	-0.281751	0.116583	2.550530	-0.281752	9.923234	2.971850e-07
1	102065	2.979433	-0.806041	0.972374	2.979433	-0.806041	10.779025	1.987442e-07
2	102066	4.015142	0.171628	-0.546059	4.015142	0.171628	9.260591	-1.527369e-08
3	102067	3.419234	-1.490107	1.156678	3.419234	-1.490107	10.963328	-2.314337e-07
4	102068	4.743965	-2.326193	2.661753	4.743965	-2.326193	12.468404	-6.426126e-08
...	...	...	...	...	...	...	...	...

	time_centiseconds	x_lin_acc	y_lin_acc	z_lin_acc	x_acc	y_acc	z_acc	x_magneto
583	107309	-0.671471	1.734737	-0.922664	-0.671471	1.734737	8.883988	3.473316e-07
584	107310	-0.769552	1.695303	-0.709576	-0.769552	1.695303	9.097074	-2.575573e-07
585	107311	-1.046645	1.905795	-0.360317	-1.046645	1.905795	9.446333	-2.425164e-07
586	107312	-1.100073	2.158354	-0.152339	-1.100073	2.158355	9.654312	-3.560912e-07
587	107313	-0.576510	1.801576	0.592426	-0.576510	1.801576	10.399076	3.783498e-08

588 rows × 15 columns

In [ ]:

```
v3_label
```

Out[ ]:

	event	start_time	end_time
0	Agressive right turn	9.5	12.5
1	Non aggressive	19.0	23.0
2	Agressive right turn	91.6	94.9
3	Agressive right turn	120.9	124.1
4	Agressive right turn	135.4	139.0
5	Non aggressive	164.0	168.0
6	Non aggressive	187.0	190.5
7	Agressive right turn	219.4	223.9
8	Agressive right turn	232.6	236.7
9	Non aggressive	358.0	360.5
10	Agressive left turn	412.0	416.0
11	Agressive left turn	430.3	433.2
12	Agressive left turn	447.4	450.7
13	Non aggressive	463.5	465.6
14	Agressive left turn	496.1	499.2
15	Agressive left turn	508.8	512.0
16	Agressive left turn	531.6	534.4

In [ ]:

```
new_vehicle1['event_shift'] = new_vehicle1['event'].shift(1)
new_vehicle1['is_new_event'] = (new_vehicle1['event']!=new_vehicle1['event_shift'])
new_vehicle1['time_shift'] = new_vehicle1['Time seconds'].shift(1)
new_vehicle1['is_time_gap'] = ((new_vehicle1['Time seconds']-new_vehicle1['time_shif
new_vehicle1['is_new_event'] = new_vehicle1['is_new_event'] + new_vehicle1['is_time_
new_vehicle1.drop(['is_time_gap','event_shift','time_shift'],axis=1,inplace=True)
```

```
new_vehicle2['event_shift'] = new_vehicle2['event'].shift(1)
new_vehicle2['is_new_event'] = (new_vehicle2['event']!=new_vehicle2['event_shift'])
new_vehicle2['time_shift'] = new_vehicle2['Time seconds'].shift(1)
new_vehicle2['is_time_gap'] = ((new_vehicle2['Time seconds']-new_vehicle2['time_shif
new_vehicle2['is_new_event'] = new_vehicle2['is_new_event'] + new_vehicle2['is_time_
new_vehicle2.drop(['is_time_gap','event_shift','time_shift'],axis=1,inplace=True)

new_vehicle3['event_shift'] = new_vehicle3['event'].shift(1)
new_vehicle3['is_new_event'] = (new_vehicle3['event']!=new_vehicle3['event_shift'])
new_vehicle3['time_shift'] = new_vehicle3['Time seconds'].shift(1)
new_vehicle3['is_time_gap'] = ((new_vehicle3['Time seconds']-new_vehicle3['time_shif
new_vehicle3['is_new_event'] = new_vehicle3['is_new_event'] + new_vehicle3['is_time_
new_vehicle3.drop(['is_time_gap','event_shift','time_shift'],axis=1,inplace=True)


new_vehicle4['event_shift'] = new_vehicle4['event'].shift(1)
new_vehicle4['is_new_event'] = (new_vehicle4['event']!=new_vehicle4['event_shift'])
new_vehicle4['time_shift'] = new_vehicle4['Time seconds'].shift(1)
new_vehicle4['is_time_gap'] = ((new_vehicle4['Time seconds']-new_vehicle4['time_shif
new_vehicle4['is_new_event'] = new_vehicle4['is_new_event'] + new_vehicle4['is_time_
new_vehicle4.drop(['is_time_gap','event_shift','time_shift'],axis=1,inplace=True)
```

```
c:\Users\gg502\AppData\Local\Programs\Python\Python36\lib\site-packages\pandas\core
\computation\expressions.py:204: UserWarning: evaluating in Python space because the
'+' operator is not supported by numexpr for the bool dtype, use '|' instead
  f"evaluating in Python space because the {repr(op_str)} "
c:\Users\gg502\AppData\Local\Programs\Python\Python36\lib\site-packages\pandas\core
\computation\expressions.py:204: UserWarning: evaluating in Python space because the
'+' operator is not supported by numexpr for the bool dtype, use '|' instead
  f"evaluating in Python space because the {repr(op_str)} "
c:\Users\gg502\AppData\Local\Programs\Python\Python36\lib\site-packages\pandas\core
\computation\expressions.py:204: UserWarning: evaluating in Python space because the
'+' operator is not supported by numexpr for the bool dtype, use '|' instead
  f"evaluating in Python space because the {repr(op_str)} "
c:\Users\gg502\AppData\Local\Programs\Python\Python36\lib\site-packages\pandas\core
\computation\expressions.py:204: UserWarning: evaluating in Python space because the
'+' operator is not supported by numexpr for the bool dtype, use '|' instead
  f"evaluating in Python space because the {repr(op_str)} "
```

In [ ]:

```
data = pd.DataFrame
data = pd.concat([new_vehicle1,new_vehicle2,new_vehicle3,new_vehicle4], ignore_index
```

In [ ]:

```
data['is_new_event'].value_counts()
```

Out[ ]:

```
False    2174
True       69
Name: is_new_event, dtype: int64
```

In [ ]:

```
len(data)
```

Out[ ]:

```
2243
```

In [ ]:

```
label_summ = data[data['is_new_event']==True]
idx = [0,1,2,3,4,5,6]
label_summ = pd.DataFrame(data = label_summ['event'].value_counts())
label_summ.rename(columns = {'event':'no of sequences'},inplace=True)
label_summ.rename(columns={'index':'events'},inplace=True)
label_summ
```

Out[ ]:

	no of sequences
Non aggressive	14
Aggressive acceleration	12
Agressive braking	12
Agressive right turn	11
Agressive left turn	11
Aggressive right lane change	5
Aggressive left lane change	4

In [ ]:

```
data
```

Out[ ]:

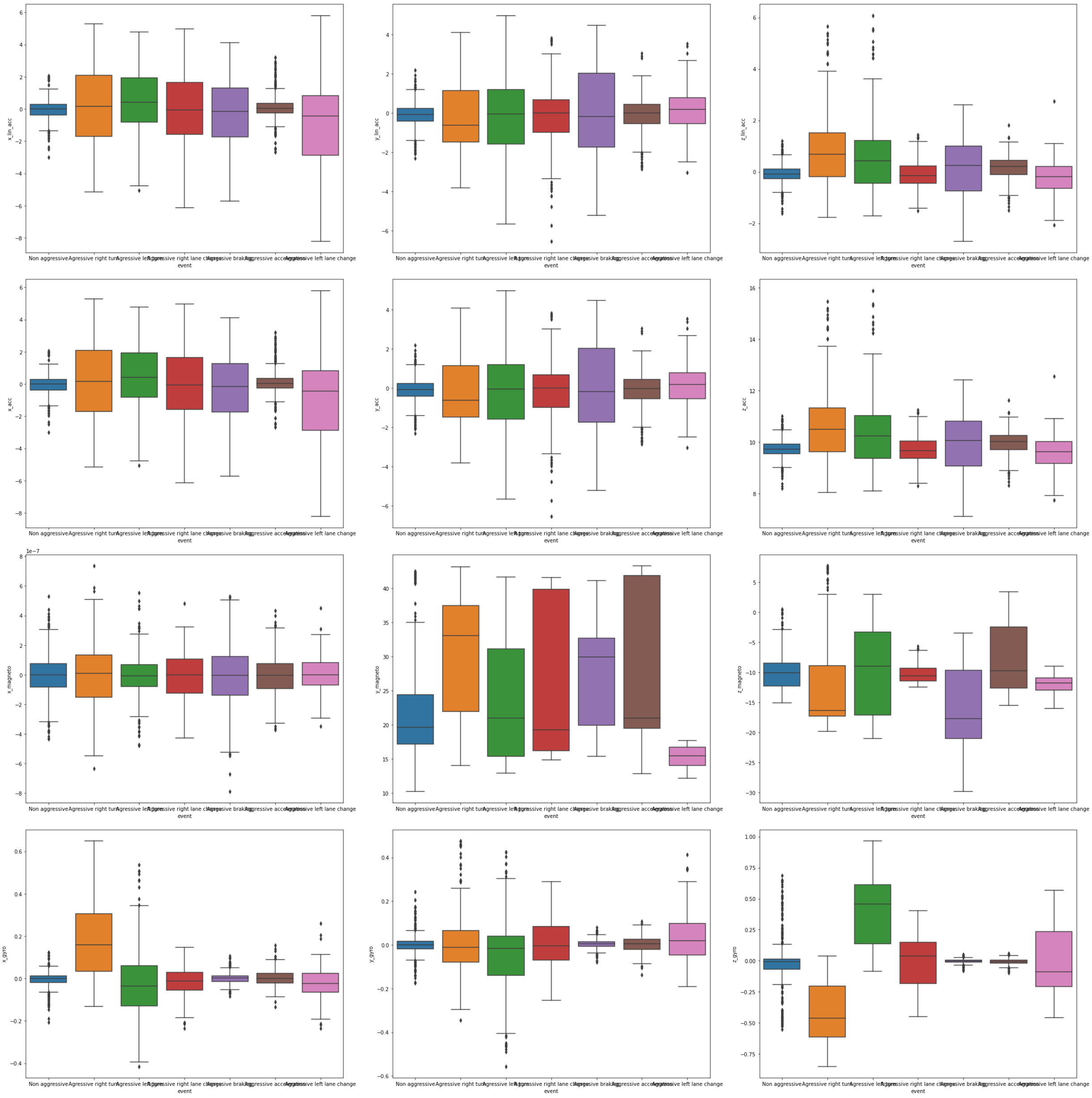
	time_centiseconds	x_lin_acc	y_lin_acc	z_lin_acc	x_acc	y_acc	z_acc	x_magneto
0	115396	0.132041	0.867996	-0.045091	0.132041	0.867996	9.761559	-1.259366e-07
1	115397	0.171421	1.122584	-0.051169	0.171421	1.122584	9.755482	1.040258e-07
2	115398	0.153374	1.082111	-0.044352	0.153374	1.082111	9.762299	1.226072e-08
3	115399	0.127719	1.099273	-0.034528	0.127720	1.099273	9.772121	8.466013e-08
4	115400	0.141839	1.072355	-0.059415	0.141839	1.072355	9.747235	-5.626498e-08
...	...	...	...	...	...	...	...	...
2238	119877	0.393265	0.754730	-0.203305	0.393265	0.754730	9.603346	1.142384e-07
2239	119878	0.852456	-0.133665	-0.558923	0.852456	-0.133665	9.247727	1.303270e-07
2240	119879	1.081358	-0.811304	-0.193637	1.081358	-0.811304	9.613014	-2.679146e-07
2241	119880	-0.966088	0.601259	-0.976800	-0.966088	0.601259	8.829851	-3.210880e-08
2242	119881	0.273143	-0.468067	0.111480	0.273143	-0.468067	9.918131	1.306571e-07

2243 rows × 16 columns

In [ ]:

```
fig, ax = plt.subplots(nrows=4, ncols=3, figsize=(30,30))

sns.boxplot(x='event', y='x_lin_acc', data=data, ax=ax[0][0])
sns.boxplot(x='event', y='y_lin_acc', data=data, ax=ax[0][1])
sns.boxplot(x='event', y='z_lin_acc', data=data, ax=ax[0][2])
sns.boxplot(x='event', y='x_acc', data=data, ax=ax[1][0])
sns.boxplot(x='event', y='y_acc', data=data, ax=ax[1][1])
sns.boxplot(x='event', y='z_acc', data=data, ax=ax[1][2])
```

```python
sns.boxplot(x='event', y='x_magneto', data=data, ax=ax[2][0])
sns.boxplot(x='event', y='y_magneto', data=data, ax=ax[2][1])
sns.boxplot(x='event', y='z_magneto', data=data, ax=ax[2][2])
sns.boxplot(x='event', y='x_gyro', data=data, ax=ax[3][0])
sns.boxplot(x='event', y='y_gyro', data=data, ax=ax[3][1])
sns.boxplot(x='event', y='z_gyro', data=data, ax=ax[3][2])

#plt.setp(ax.get_xticklabels(), rotation=45)
#ax.set_xticklabels(ax.get_xticks(), rotation = 50)
plt.tight_layout()
plt.show()
```



In [ ]:

```python
new_sequences = data.loc[data["is_new_event"]==True]
indices = new_sequences.index
seq_no, count = [], 0

for i in range(len(indices)-1):
    seq_len = indices[i+1]-indices[i]
    new_list = [i+1]*seq_len
    seq_no.append(new_list)
    count = i+1

rem_len = len(data)-indices[count]
```

```
seq_no.append([count+1]*rem_len)

flatten_list = list(chain.from_iterable(seq_no))
sequences = np.unique(flatten_list)

data["sequence no."] = flatten_list
```

In [ ]:

```
len(flatten_list)
```

Out[ ]: 2243

In [ ]:

```
data
```

Out[ ]:

	time_centiseconds	x_lin_acc	y_lin_acc	z_lin_acc	x_acc	y_acc	z_acc	x_magneto
0	115396	0.132041	0.867996	-0.045091	0.132041	0.867996	9.761559	-1.259366e-07
1	115397	0.171421	1.122584	-0.051169	0.171421	1.122584	9.755482	1.040258e-07
2	115398	0.153374	1.082111	-0.044352	0.153374	1.082111	9.762299	1.226072e-08
3	115399	0.127719	1.099273	-0.034528	0.127720	1.099273	9.772121	8.466013e-08
4	115400	0.141839	1.072355	-0.059415	0.141839	1.072355	9.747235	-5.626498e-08
...	...	...	...	...	...	...	...	...
2238	119877	0.393265	0.754730	-0.203305	0.393265	0.754730	9.603346	1.142384e-07
2239	119878	0.852456	-0.133665	-0.558923	0.852456	-0.133665	9.247727	1.303270e-07
2240	119879	1.081358	-0.811304	-0.193637	1.081358	-0.811304	9.613014	-2.679146e-07
2241	119880	-0.966088	0.601259	-0.976800	-0.966088	0.601259	8.829851	-3.210880e-08
2242	119881	0.273143	-0.468067	0.111480	0.273143	-0.468067	9.918131	1.306571e-07

2243 rows × 17 columns

In [ ]:

```
## partitioning data into 0.5 seconds windows

time_wd, seq_part_no = [], []

for seq in sequences:
    seq_df = data[data["sequence no."]==seq].reset_index(drop=True)
    min_time = seq_df["time_centiseconds"][0]
    min_time_lis = [min_time]*len(seq_df)

    time_diff = (seq_df["time_centiseconds"]-min_time_lis)
```

```
    wd_no = np.floor_divide(time_diff,5)+1
    wd_tostr = wd_no.apply(str)
    seq_str = [str(seq)]*len(seq_df)
    _str = ["_"]*len(seq_df)
    part_no = list(i+j for i,j in zip(seq_str , _str))
    part_no = list(i+j for i,j in zip(part_no , wd_tostr))

    time_wd.append(wd_no)
    seq_part_no.append(part_no)

time_wd = list(chain.from_iterable(time_wd))
seq_part_no = list(chain.from_iterable(seq_part_no))

data["partition no"] = time_wd
data["Partition label"] = seq_part_no
```

In [ ]:

```
data
```

Out[ ]:

	time_centiseconds	x_lin_acc	y_lin_acc	z_lin_acc	x_acc	y_acc	z_acc	x_magneto
0	115396	0.132041	0.867996	-0.045091	0.132041	0.867996	9.761559	-1.259366e-07
1	115397	0.171421	1.122584	-0.051169	0.171421	1.122584	9.755482	1.040258e-07
2	115398	0.153374	1.082111	-0.044352	0.153374	1.082111	9.762299	1.226072e-08
3	115399	0.127719	1.099273	-0.034528	0.127720	1.099273	9.772121	8.466013e-08
4	115400	0.141839	1.072355	-0.059415	0.141839	1.072355	9.747235	-5.626498e-08
...	...	...	...	...	...	...	...	...
2238	119877	0.393265	0.754730	-0.203305	0.393265	0.754730	9.603346	1.142384e-07
2239	119878	0.852456	-0.133665	-0.558923	0.852456	-0.133665	9.247727	1.303270e-07
2240	119879	1.081358	-0.811304	-0.193637	1.081358	-0.811304	9.613014	-2.679146e-07
2241	119880	-0.966088	0.601259	-0.976800	-0.966088	0.601259	8.829851	-3.210880e-08
2242	119881	0.273143	-0.468067	0.111480	0.273143	-0.468067	9.918131	1.306571e-07

2243 rows × 19 columns

In [ ]:

```
len(data['Partition label'].unique())
```

Out[ ]: 483

In [ ]:

```
label_dfs = [y for x,y in data.groupby(data['Partition label'])]
label = []
```

```
label_class = []
features = []
time_data = []

for label_df in label_dfs:
    label_df = label_df.reset_index(drop=True)

    label.append(label_df["Partition label"][0])
    label_class.append(label_df["event"][0])

    x_linacc_min = min(label_df["x_lin_acc"])
    y_linacc_min = min(label_df["y_lin_acc"])
    z_linacc_min = min(label_df["z_lin_acc"])

    x_linacc_max = max(label_df["x_lin_acc"])
    y_linacc_max = max(label_df["y_lin_acc"])
    z_linacc_max = max(label_df["z_lin_acc"])

    x_linacc_avg = np.mean(label_df["x_lin_acc"])
    y_linacc_avg = np.mean(label_df["y_lin_acc"])
    z_linacc_avg = np.mean(label_df["z_lin_acc"])

    x_linacc_std = np.std(label_df["x_lin_acc"])
    y_linacc_std = np.std(label_df["y_lin_acc"])
    z_linacc_std = np.std(label_df["z_lin_acc"])

    x_acc_min = min(label_df["x_acc"])
    y_acc_min = min(label_df["y_acc"])
    z_acc_min = min(label_df["z_acc"])

    x_acc_max = max(label_df["x_acc"])
    y_acc_max = max(label_df["y_acc"])
    z_acc_max = max(label_df["z_acc"])

    x_acc_avg = np.mean(label_df["x_acc"])
    y_acc_avg = np.mean(label_df["y_acc"])
    z_acc_avg = np.mean(label_df["z_acc"])

    x_acc_std = np.std(label_df["x_acc"])
    y_acc_std = np.std(label_df["y_acc"])
    z_acc_std = np.std(label_df["z_acc"])

    x_magneto_min = min(label_df["x_magneto"])
    y_magneto_min = min(label_df["y_magneto"])
    z_magneto_min = min(label_df["z_magneto"])

    x_magneto_max = max(label_df["x_magneto"])
    y_magneto_max = max(label_df["y_magneto"])
    z_magneto_max = max(label_df["z_magneto"])

    x_magneto_avg = np.mean(label_df["x_magneto"])
    y_magneto_avg = np.mean(label_df["y_magneto"])
    z_magneto_avg = np.mean(label_df["z_magneto"])

    x_magneto_std = np.std(label_df["x_magneto"])
    y_magneto_std = np.std(label_df["y_magneto"])
    z_magneto_std = np.std(label_df["z_magneto"])

    x_gyro_min = min(label_df["x_gyro"])
    y_gyro_min = min(label_df["y_gyro"])
    z_gyro_min = min(label_df["z_gyro"])

    x_gyro_max = max(label_df["x_gyro"])
    y_gyro_max = max(label_df["y_gyro"])
```

```python
    z_gyro_max = max(label_df["z_gyro"])

    x_gyro_avg = np.mean(label_df["x_gyro"])
    y_gyro_avg = np.mean(label_df["y_gyro"])
    z_gyro_avg = np.mean(label_df["z_gyro"])

    x_gyro_std = np.std(label_df["x_gyro"])
    y_gyro_std = np.std(label_df["y_gyro"])
    z_gyro_std = np.std(label_df["z_gyro"])


    features.append([x_linacc_min, y_linacc_min, z_linacc_min, x_linacc_max, y_linac
                     x_acc_min, y_acc_min, z_acc_min, x_acc_max, y_acc_max, z_acc_max
                     x_magneto_min, y_magneto_min, z_magneto_min, x_magneto_max, y_ma
                     x_gyro_min, y_gyro_min, z_gyro_min, x_gyro_max, y_gyro_max, z_gy

feature_columns = ["x_linacc_min", "y_linacc_min", "z_linacc_min", "x_linacc_max", "
                   "x_acc_min", "y_acc_min", "z_acc_min", "x_acc_max", "y_acc_max",
                   "x_magneto_min", "y_magneto_min", "z_magneto_min", "x_magneto_ma
                   "x_gyro_min", "y_gyro_min", "z_gyro_min", "x_gyro_max", "y_gyro_

features_df = pd.DataFrame(features, columns = feature_columns)
features_df["Label"] = label
features_df["event"] = label_class
```

In [ ]:

```python
features_df['x_magneto_min']
```

Out[ ]:

```
0     -2.119225e-07
1     -1.384877e-07
2     -1.143069e-07
3     -2.359351e-07
4     -1.840293e-07
           ...
478   -2.851710e-07
479   -5.327165e-08
480   -1.192093e-07
481   -3.241003e-07
482   -1.862645e-10
Name: x_magneto_min, Length: 483, dtype: float64
```

In [ ]:

```python
df = features_df.drop(['Label'],axis=1)
```

In [ ]:

```python
df
```

Out[ ]:

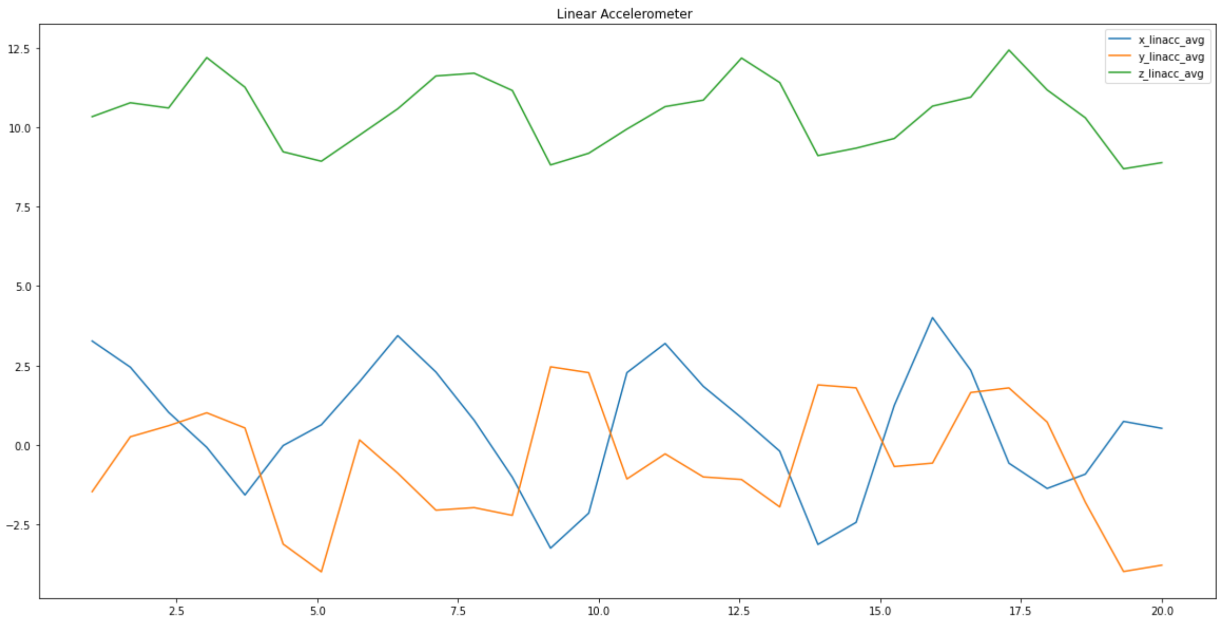
	x_linacc_min	y_linacc_min	z_linacc_min	x_linacc_max	y_linacc_max	z_linacc_max	x_linacc_avg
0	0.213532	-0.945080	-0.553991	1.798927	0.275360	0.077384	0.742918
1	2.627874	-1.932428	0.128613	3.617381	-1.055272	1.078540	3.271221
2	1.925024	-0.883151	0.721348	3.117764	0.790471	1.210040	2.446328
3	0.038199	0.369631	-0.681987	1.959833	0.928872	2.077756	1.027864
4	-0.705620	0.235897	1.158653	0.638396	1.682580	4.418347	-0.077547
...	...	...	...	...	...	...	...

	x_linacc_min	y_linacc_min	z_linacc_min	x_linacc_max	y_linacc_max	z_linacc_max	x_linacc_avg
478	0.194214	-2.915153	-0.072153	1.176256	-0.660332	4.659564	0.594686
479	-1.315942	-2.204906	0.789823	0.950823	-1.220241	1.861554	-0.544849
480	-3.341690	-0.820263	-1.051386	-1.489183	2.555306	1.007381	-2.614278
481	-3.181725	1.390165	-1.027327	-1.672604	2.677557	-0.183479	-2.205725
482	-1.432755	1.653576	-0.335926	-1.432755	1.653576	-0.335926	-1.432755

483 rows × 49 columns

In [ ]:

```
y_points1 = np.array(df['x_acc_avg'][1:30])
y_points2 = np.array(df['y_acc_avg'][1:30])
y_points3 = np.array(df['z_acc_avg'][1:30])
x_points = np.linspace(1,20,29)

plt.figure(figsize=(20,10))
plt.plot(x_points, y_points1, label="x_linacc_avg")
plt.plot(x_points, y_points2, label="y_linacc_avg")
plt.plot(x_points, y_points3, label="z_linacc_avg")
plt.legend(loc='upper right')
plt.title("Linear Accelerometer ")
plt.show()
```



In [ ]:

```
df['event'][1:30]
```

Out[ ]:

```
1       Agressive left turn
2       Agressive left turn
3       Agressive left turn
4       Agressive left turn
5       Agressive left turn
6       Agressive left turn
7       Agressive left turn
```

```
8     Agressive right turn
9     Agressive right turn
10    Agressive right turn
11    Agressive right turn
12    Agressive right turn
13    Agressive right turn
14    Agressive right turn
15    Agressive right turn
16    Agressive right turn
17    Agressive right turn
18    Agressive right turn
19    Agressive right turn
20    Agressive right turn
21    Agressive right turn
22     Agressive left turn
23     Agressive left turn
24     Agressive left turn
25     Agressive left turn
26     Agressive left turn
27     Agressive left turn
28     Agressive left turn
29     Agressive left turn
Name: event, dtype: object
```

In [ ]:

```
cols_scale = ["x_linacc_min", "y_linacc_min", "z_linacc_min", "x_linacc_max", "y_lin
                "x_acc_min", "y_acc_min", "z_acc_min", "x_acc_max", "y_acc_max", "z_
                "x_magneto_min", "y_magneto_min", "z_magneto_min", "x_magneto_max",
                "x_gyro_min", "y_gyro_min", "z_gyro_min", "x_gyro_max", "y_gyro_max"
             ]

scaler = MinMaxScaler()
df[cols_scale] = scaler.fit_transform(df[cols_scale])
```

In [ ]:

```
label_encoder = LabelEncoder()
df["class"] = label_encoder.fit_transform(df["event"])
```

In [ ]:

```
df
```

Out[ ]:

	x_linacc_min	y_linacc_min	z_linacc_min	x_linacc_max	y_linacc_max	z_linacc_max	x_linacc_avg
0	0.697794	0.546702	0.448173	0.580582	0.487860	0.160141	0.648999
1	0.897954	0.450035	0.591370	0.771061	0.342637	0.300238	0.920619
2	0.839684	0.552765	0.715714	0.718727	0.544079	0.318640	0.832000
3	0.683258	0.675419	0.421322	0.597437	0.559184	0.440064	0.679611
4	0.621592	0.662326	0.807452	0.459019	0.641442	0.767596	0.560855
...	...	...	...	...	...	...	...
478	0.696192	0.353821	0.549253	0.515359	0.385740	0.801351	0.633074
479	0.570994	0.423358	0.730079	0.491745	0.324633	0.409810	0.510652

	x_linacc_min	y_linacc_min	z_linacc_min	x_linacc_max	y_linacc_max	z_linacc_max	x_linacc_avg
480	0.403050	0.558922	0.343829	0.236161	0.736690	0.290280	0.288329
481	0.416312	0.775335	0.348877	0.216948	0.750033	0.123637	0.332221
482	0.561310	0.801124	0.493919	0.242071	0.638277	0.102304	0.415262

483 rows × 50 columns

In [ ]:

```
X = df.drop(['event','class'],axis=1)
Y = df['class']
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, stratify=df
```

In [ ]:

```
y_train.value_counts()
```

Out[ ]:

```
6    90
0    75
5    68
4    67
3    51
2    20
1    15
Name: class, dtype: int64
```

In [ ]:

```
y_test.value_counts()
```

Out[ ]:

```
6    22
0    19
5    17
4    17
3    13
2     5
1     4
Name: class, dtype: int64
```

In [ ]:

```
over_sampler = RandomOverSampler(random_state=42)
X_train, Y_train = over_sampler.fit_resample(x_train, y_train)
```

In [ ]:

```
Y_train.value_counts()
```

Out[ ]:

```
6    90
5    90
4    90
3    90
2    90
1    90
0    90
Name: class, dtype: int64
```

In [ ]:

```
X_train
```

Out[ ]:

x_linacc_min	y_linacc_min	z_linacc_min	x_linacc_max	y_linacc_max	z_linacc_max	x_linacc_avg

	x_linacc_min	y_linacc_min	z_linacc_min	x_linacc_max	y_linacc_max	z_linacc_max	x_linacc_avg
0	0.674155	0.430854	0.571117	0.662502	0.407645	0.793664	0.687773
1	0.747232	0.524273	0.698709	0.641927	0.554332	0.607313	0.736250
2	0.616106	0.579775	0.532495	0.406725	0.469788	0.233945	0.550526
3	0.340214	0.715768	0.400069	0.070293	0.628103	0.157923	0.202534
4	0.637887	0.580290	0.226857	0.461024	0.557530	0.287488	0.562707
...	...	...	...	...	...	...	...
625	0.674155	0.430854	0.571117	0.662502	0.407645	0.793664	0.687773
626	0.542585	0.502784	0.811512	0.265664	0.410735	0.521112	0.416184
627	0.643004	0.526702	0.764935	0.485639	0.488245	0.492473	0.596764
628	0.347746	0.540109	0.236229	0.188574	0.863443	0.292491	0.232118
629	0.416606	0.711201	0.271586	0.162163	0.806648	0.187518	0.286230

630 rows × 48 columns

ANN

In [ ]:

```
ann = keras.Sequential([
    keras.layers.Dense(28, input_shape = (48,), activation = 'relu'),
    keras.layers.BatchNormalization(),
    keras.layers.Dense(40, activation='relu'),
    keras.layers.BatchNormalization(),
    keras.layers.Dense(30, activation='relu'),
    keras.layers.Dense(20, activation='relu'),
    keras.layers.Dense(10, activation='relu'),
    keras.layers.BatchNormalization(),
    keras.layers.Dense(7, activation='softmax')
])

ann.compile(
    optimizer = 'adam',
    loss = 'sparse_categorical_crossentropy',
    metrics = ['accuracy']
)
#tf.keras.optimizers.RMSprop()

ann.fit(X_train, Y_train, epochs = 300, validation_split=0.1, batch_size=16)
```

```
Epoch 1/300
36/36 [==============================] - 2s 14ms/step - loss: 1.9451 - accuracy: 0.2
099 - val_loss: 1.9344 - val_accuracy: 0.0476
Epoch 2/300
36/36 [==============================] - 0s 4ms/step - loss: 1.6368 - accuracy: 0.37
04 - val_loss: 1.9598 - val_accuracy: 0.0000e+00
Epoch 3/300
36/36 [==============================] - 0s 3ms/step - loss: 1.4889 - accuracy: 0.46
74 - val_loss: 1.9888 - val_accuracy: 0.0159
Epoch 4/300
36/36 [==============================] - 0s 3ms/step - loss: 1.4514 - accuracy: 0.45
15 - val_loss: 1.9838 - val_accuracy: 0.0159
Epoch 5/300
36/36 [==============================] - 0s 3ms/step - loss: 1.2966 - accuracy: 0.52
```

```
91 - val_loss: 1.9274 - val_accuracy: 0.0317
Epoch 6/300
36/36 [==============================] - 0s 3ms/step - loss: 1.2107 - accuracy: 0.58
73 - val_loss: 1.9113 - val_accuracy: 0.0794
Epoch 7/300
36/36 [==============================] - 0s 3ms/step - loss: 1.1657 - accuracy: 0.62
43 - val_loss: 1.8744 - val_accuracy: 0.2222
Epoch 8/300
36/36 [==============================] - 0s 4ms/step - loss: 1.0861 - accuracy: 0.63
14 - val_loss: 1.7144 - val_accuracy: 0.3810
Epoch 9/300
36/36 [==============================] - 0s 4ms/step - loss: 1.0064 - accuracy: 0.68
96 - val_loss: 1.5800 - val_accuracy: 0.4286
Epoch 10/300
36/36 [==============================] - 0s 4ms/step - loss: 0.9582 - accuracy: 0.69
14 - val_loss: 1.3212 - val_accuracy: 0.5238
Epoch 11/300
36/36 [==============================] - 0s 4ms/step - loss: 0.9355 - accuracy: 0.69
49 - val_loss: 1.2143 - val_accuracy: 0.5714
Epoch 12/300
36/36 [==============================] - 0s 5ms/step - loss: 0.8889 - accuracy: 0.71
78 - val_loss: 1.1123 - val_accuracy: 0.6190
Epoch 13/300
36/36 [==============================] - 0s 4ms/step - loss: 0.8456 - accuracy: 0.69
49 - val_loss: 0.9130 - val_accuracy: 0.6984
Epoch 14/300
36/36 [==============================] - 0s 3ms/step - loss: 0.7877 - accuracy: 0.74
60 - val_loss: 0.9114 - val_accuracy: 0.7302
Epoch 15/300
36/36 [==============================] - 0s 3ms/step - loss: 0.7789 - accuracy: 0.76
19 - val_loss: 0.7073 - val_accuracy: 0.7778
Epoch 16/300
36/36 [==============================] - 0s 3ms/step - loss: 0.7571 - accuracy: 0.76
54 - val_loss: 0.6611 - val_accuracy: 0.7778
Epoch 17/300
36/36 [==============================] - 0s 3ms/step - loss: 0.7358 - accuracy: 0.76
01 - val_loss: 0.4986 - val_accuracy: 0.8254
Epoch 18/300
36/36 [==============================] - 0s 4ms/step - loss: 0.7554 - accuracy: 0.76
19 - val_loss: 0.4655 - val_accuracy: 0.8889
Epoch 19/300
36/36 [==============================] - 0s 3ms/step - loss: 0.7243 - accuracy: 0.75
66 - val_loss: 0.4481 - val_accuracy: 0.8571
Epoch 20/300
36/36 [==============================] - 0s 4ms/step - loss: 0.6693 - accuracy: 0.77
25 - val_loss: 0.3750 - val_accuracy: 0.8889
Epoch 21/300
36/36 [==============================] - 0s 3ms/step - loss: 0.6616 - accuracy: 0.78
66 - val_loss: 0.3960 - val_accuracy: 0.9206
Epoch 22/300
36/36 [==============================] - 0s 4ms/step - loss: 0.6048 - accuracy: 0.79
72 - val_loss: 0.2595 - val_accuracy: 0.9206
Epoch 23/300
36/36 [==============================] - 0s 3ms/step - loss: 0.5623 - accuracy: 0.81
31 - val_loss: 0.3439 - val_accuracy: 0.9048
Epoch 24/300
36/36 [==============================] - 0s 3ms/step - loss: 0.6297 - accuracy: 0.77
60 - val_loss: 0.3470 - val_accuracy: 0.8889
Epoch 25/300
36/36 [==============================] - 0s 3ms/step - loss: 0.5935 - accuracy: 0.79
89 - val_loss: 0.2596 - val_accuracy: 0.9365
Epoch 26/300
36/36 [==============================] - 0s 3ms/step - loss: 0.5447 - accuracy: 0.83
60 - val_loss: 0.2710 - val_accuracy: 0.9365
Epoch 27/300
36/36 [==============================] - 0s 3ms/step - loss: 0.5349 - accuracy: 0.83
07 - val_loss: 0.2107 - val_accuracy: 0.9524
Epoch 28/300
36/36 [==============================] - 0s 3ms/step - loss: 0.5658 - accuracy: 0.79
```

```
89 - val_loss: 0.2229 - val_accuracy: 0.9365
Epoch 29/300
36/36 [==============================] - 0s 4ms/step - loss: 0.5694 - accuracy: 0.81
31 - val_loss: 0.2592 - val_accuracy: 0.9365
Epoch 30/300
36/36 [==============================] - 0s 3ms/step - loss: 0.4840 - accuracy: 0.84
13 - val_loss: 0.2247 - val_accuracy: 0.9048
Epoch 31/300
36/36 [==============================] - 0s 3ms/step - loss: 0.5505 - accuracy: 0.81
13 - val_loss: 0.2597 - val_accuracy: 0.9206
Epoch 32/300
36/36 [==============================] - 0s 3ms/step - loss: 0.4588 - accuracy: 0.85
19 - val_loss: 0.1843 - val_accuracy: 0.9206
Epoch 33/300
36/36 [==============================] - 0s 3ms/step - loss: 0.4806 - accuracy: 0.83
07 - val_loss: 0.1524 - val_accuracy: 0.9524
Epoch 34/300
36/36 [==============================] - 0s 3ms/step - loss: 0.4792 - accuracy: 0.85
71 - val_loss: 0.1473 - val_accuracy: 0.9683
Epoch 35/300
36/36 [==============================] - 0s 4ms/step - loss: 0.5155 - accuracy: 0.83
95 - val_loss: 0.1289 - val_accuracy: 0.9524
Epoch 36/300
36/36 [==============================] - 0s 4ms/step - loss: 0.5413 - accuracy: 0.82
01 - val_loss: 0.0865 - val_accuracy: 0.9841
Epoch 37/300
36/36 [==============================] - 0s 3ms/step - loss: 0.4984 - accuracy: 0.82
19 - val_loss: 0.1586 - val_accuracy: 0.9683
Epoch 38/300
36/36 [==============================] - 0s 3ms/step - loss: 0.4208 - accuracy: 0.85
36 - val_loss: 0.1200 - val_accuracy: 0.9841
Epoch 39/300
36/36 [==============================] - 0s 3ms/step - loss: 0.4719 - accuracy: 0.84
83 - val_loss: 0.1081 - val_accuracy: 0.9683
Epoch 40/300
36/36 [==============================] - 0s 3ms/step - loss: 0.4179 - accuracy: 0.86
95 - val_loss: 0.1062 - val_accuracy: 1.0000
Epoch 41/300
36/36 [==============================] - 0s 4ms/step - loss: 0.4978 - accuracy: 0.82
19 - val_loss: 0.1588 - val_accuracy: 0.9524
Epoch 42/300
36/36 [==============================] - 0s 3ms/step - loss: 0.4534 - accuracy: 0.84
66 - val_loss: 0.1316 - val_accuracy: 0.9524
Epoch 43/300
36/36 [==============================] - 0s 3ms/step - loss: 0.3998 - accuracy: 0.87
48 - val_loss: 0.1188 - val_accuracy: 0.9683
Epoch 44/300
36/36 [==============================] - 0s 3ms/step - loss: 0.3870 - accuracy: 0.87
30 - val_loss: 0.1400 - val_accuracy: 0.9524
Epoch 45/300
36/36 [==============================] - 0s 3ms/step - loss: 0.4025 - accuracy: 0.85
36 - val_loss: 0.2226 - val_accuracy: 0.9365
Epoch 46/300
36/36 [==============================] - 0s 3ms/step - loss: 0.4012 - accuracy: 0.85
89 - val_loss: 0.1069 - val_accuracy: 0.9683
Epoch 47/300
36/36 [==============================] - 0s 5ms/step - loss: 0.3924 - accuracy: 0.86
60 - val_loss: 0.0959 - val_accuracy: 0.9683
Epoch 48/300
36/36 [==============================] - 0s 3ms/step - loss: 0.3553 - accuracy: 0.88
18 - val_loss: 0.0728 - val_accuracy: 1.0000
Epoch 49/300
36/36 [==============================] - 0s 3ms/step - loss: 0.4120 - accuracy: 0.86
24 - val_loss: 0.1196 - val_accuracy: 0.9524
Epoch 50/300
36/36 [==============================] - 0s 3ms/step - loss: 0.3630 - accuracy: 0.87
48 - val_loss: 0.0448 - val_accuracy: 1.0000
Epoch 51/300
36/36 [==============================] - 0s 3ms/step - loss: 0.3615 - accuracy: 0.88
```

```
71 - val_loss: 0.0916 - val_accuracy: 0.9841
Epoch 52/300
36/36 [==============================] - 0s 3ms/step - loss: 0.4328 - accuracy: 0.84
48 - val_loss: 0.3589 - val_accuracy: 0.8730
Epoch 53/300
36/36 [==============================] - 0s 4ms/step - loss: 0.3671 - accuracy: 0.87
48 - val_loss: 0.0578 - val_accuracy: 0.9841
Epoch 54/300
36/36 [==============================] - 0s 4ms/step - loss: 0.3520 - accuracy: 0.87
48 - val_loss: 0.0512 - val_accuracy: 1.0000
Epoch 55/300
36/36 [==============================] - 0s 4ms/step - loss: 0.3239 - accuracy: 0.89
42 - val_loss: 0.1281 - val_accuracy: 0.9683
Epoch 56/300
36/36 [==============================] - 0s 5ms/step - loss: 0.3490 - accuracy: 0.88
89 - val_loss: 0.0660 - val_accuracy: 0.9841
Epoch 57/300
36/36 [==============================] - 0s 4ms/step - loss: 0.3222 - accuracy: 0.88
54 - val_loss: 0.1130 - val_accuracy: 0.9841
Epoch 58/300
36/36 [==============================] - 0s 4ms/step - loss: 0.4283 - accuracy: 0.87
48 - val_loss: 0.0411 - val_accuracy: 1.0000
Epoch 59/300
36/36 [==============================] - 0s 3ms/step - loss: 0.3935 - accuracy: 0.86
24 - val_loss: 0.0521 - val_accuracy: 1.0000
Epoch 60/300
36/36 [==============================] - 0s 3ms/step - loss: 0.3242 - accuracy: 0.88
54 - val_loss: 0.0547 - val_accuracy: 1.0000
Epoch 61/300
36/36 [==============================] - 0s 4ms/step - loss: 0.3332 - accuracy: 0.89
59 - val_loss: 0.0798 - val_accuracy: 1.0000
Epoch 62/300
36/36 [==============================] - 0s 4ms/step - loss: 0.3798 - accuracy: 0.86
42 - val_loss: 0.0786 - val_accuracy: 0.9683
Epoch 63/300
36/36 [==============================] - 0s 5ms/step - loss: 0.3335 - accuracy: 0.88
54 - val_loss: 0.0604 - val_accuracy: 0.9841
Epoch 64/300
36/36 [==============================] - 0s 4ms/step - loss: 0.3037 - accuracy: 0.89
59 - val_loss: 0.0292 - val_accuracy: 1.0000
Epoch 65/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2940 - accuracy: 0.90
12 - val_loss: 0.0523 - val_accuracy: 0.9841
Epoch 66/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2860 - accuracy: 0.90
48 - val_loss: 0.0267 - val_accuracy: 1.0000
Epoch 67/300
36/36 [==============================] - 0s 3ms/step - loss: 0.3377 - accuracy: 0.86
77 - val_loss: 0.0372 - val_accuracy: 1.0000
Epoch 68/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2856 - accuracy: 0.89
07 - val_loss: 0.0793 - val_accuracy: 1.0000
Epoch 69/300
36/36 [==============================] - 0s 3ms/step - loss: 0.3202 - accuracy: 0.89
42 - val_loss: 0.0505 - val_accuracy: 0.9841
Epoch 70/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2933 - accuracy: 0.89
07 - val_loss: 0.0294 - val_accuracy: 1.0000
Epoch 71/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2356 - accuracy: 0.92
06 - val_loss: 0.0232 - val_accuracy: 1.0000
Epoch 72/300
36/36 [==============================] - 0s 3ms/step - loss: 0.3085 - accuracy: 0.89
42 - val_loss: 0.0256 - val_accuracy: 1.0000
Epoch 73/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2990 - accuracy: 0.90
48 - val_loss: 0.0497 - val_accuracy: 1.0000
Epoch 74/300
36/36 [==============================] - 0s 3ms/step - loss: 0.3242 - accuracy: 0.88
```

```
01 - val_loss: 0.0311 - val_accuracy: 1.0000
Epoch 75/300
36/36 [==============================] - 0s 3ms/step - loss: 0.3220 - accuracy: 0.88
36 - val_loss: 0.0465 - val_accuracy: 1.0000
Epoch 76/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2834 - accuracy: 0.91
01 - val_loss: 0.1403 - val_accuracy: 0.9365
Epoch 77/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2659 - accuracy: 0.91
01 - val_loss: 0.0269 - val_accuracy: 1.0000
Epoch 78/300
36/36 [==============================] - 0s 4ms/step - loss: 0.2719 - accuracy: 0.91
71 - val_loss: 0.0495 - val_accuracy: 1.0000
Epoch 79/300
36/36 [==============================] - 0s 3ms/step - loss: 0.3364 - accuracy: 0.89
95 - val_loss: 0.1156 - val_accuracy: 0.9841
Epoch 80/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2979 - accuracy: 0.90
65 - val_loss: 0.0897 - val_accuracy: 0.9683
Epoch 81/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2782 - accuracy: 0.90
12 - val_loss: 0.0448 - val_accuracy: 1.0000
Epoch 82/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2448 - accuracy: 0.91
18 - val_loss: 0.0224 - val_accuracy: 1.0000
Epoch 83/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2202 - accuracy: 0.92
95 - val_loss: 0.0216 - val_accuracy: 1.0000
Epoch 84/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2833 - accuracy: 0.91
18 - val_loss: 0.0339 - val_accuracy: 1.0000
Epoch 85/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2970 - accuracy: 0.88
18 - val_loss: 0.0183 - val_accuracy: 1.0000
Epoch 86/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2559 - accuracy: 0.91
01 - val_loss: 0.0151 - val_accuracy: 1.0000
Epoch 87/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2812 - accuracy: 0.90
30 - val_loss: 0.0274 - val_accuracy: 1.0000
Epoch 88/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2620 - accuracy: 0.91
53 - val_loss: 0.0277 - val_accuracy: 1.0000
Epoch 89/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2302 - accuracy: 0.91
36 - val_loss: 0.0152 - val_accuracy: 1.0000
Epoch 90/300
36/36 [==============================] - 0s 4ms/step - loss: 0.2121 - accuracy: 0.92
95 - val_loss: 0.0677 - val_accuracy: 0.9841
Epoch 91/300
36/36 [==============================] - 0s 4ms/step - loss: 0.2919 - accuracy: 0.89
77 - val_loss: 0.0185 - val_accuracy: 1.0000
Epoch 92/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2940 - accuracy: 0.90
48 - val_loss: 0.0395 - val_accuracy: 0.9841
Epoch 93/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2593 - accuracy: 0.92
06 - val_loss: 0.0254 - val_accuracy: 1.0000
Epoch 94/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2471 - accuracy: 0.89
77 - val_loss: 0.0111 - val_accuracy: 1.0000
Epoch 95/300
36/36 [==============================] - 0s 5ms/step - loss: 0.2206 - accuracy: 0.93
30 - val_loss: 0.0290 - val_accuracy: 1.0000
Epoch 96/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2244 - accuracy: 0.92
95 - val_loss: 0.0132 - val_accuracy: 1.0000
Epoch 97/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2499 - accuracy: 0.91
```

```
36 - val_loss: 0.0454 - val_accuracy: 1.0000
Epoch 98/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2055 - accuracy: 0.93
47 - val_loss: 0.0471 - val_accuracy: 1.0000
Epoch 99/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2559 - accuracy: 0.91
53 - val_loss: 0.0102 - val_accuracy: 1.0000
Epoch 100/300
36/36 [==============================] - 0s 4ms/step - loss: 0.2759 - accuracy: 0.90
30 - val_loss: 0.0358 - val_accuracy: 1.0000
Epoch 101/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2325 - accuracy: 0.91
53 - val_loss: 0.0360 - val_accuracy: 1.0000
Epoch 102/300
36/36 [==============================] - 0s 4ms/step - loss: 0.2704 - accuracy: 0.89
77 - val_loss: 0.0317 - val_accuracy: 0.9841
Epoch 103/300
36/36 [==============================] - 0s 6ms/step - loss: 0.2098 - accuracy: 0.91
89 - val_loss: 0.0197 - val_accuracy: 1.0000
Epoch 104/300
36/36 [==============================] - 0s 4ms/step - loss: 0.2685 - accuracy: 0.90
65 - val_loss: 0.0979 - val_accuracy: 0.9683
Epoch 105/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2127 - accuracy: 0.92
42 - val_loss: 0.0102 - val_accuracy: 1.0000
Epoch 106/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2771 - accuracy: 0.89
07 - val_loss: 0.0448 - val_accuracy: 0.9841
Epoch 107/300
36/36 [==============================] - 0s 4ms/step - loss: 0.2151 - accuracy: 0.92
24 - val_loss: 0.0818 - val_accuracy: 0.9683
Epoch 108/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2295 - accuracy: 0.92
06 - val_loss: 0.0633 - val_accuracy: 0.9683
Epoch 109/300
36/36 [==============================] - 0s 3ms/step - loss: 0.3053 - accuracy: 0.89
42 - val_loss: 0.0180 - val_accuracy: 1.0000
Epoch 110/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2513 - accuracy: 0.91
71 - val_loss: 0.0195 - val_accuracy: 1.0000
Epoch 111/300
36/36 [==============================] - 0s 4ms/step - loss: 0.2386 - accuracy: 0.91
71 - val_loss: 0.0446 - val_accuracy: 0.9683
Epoch 112/300
36/36 [==============================] - 0s 4ms/step - loss: 0.2400 - accuracy: 0.90
48 - val_loss: 0.0244 - val_accuracy: 1.0000
Epoch 113/300
36/36 [==============================] - 0s 4ms/step - loss: 0.2379 - accuracy: 0.92
77 - val_loss: 0.0243 - val_accuracy: 1.0000
Epoch 114/300
36/36 [==============================] - 0s 4ms/step - loss: 0.2275 - accuracy: 0.92
06 - val_loss: 0.0174 - val_accuracy: 1.0000
Epoch 115/300
36/36 [==============================] - 0s 4ms/step - loss: 0.2064 - accuracy: 0.92
59 - val_loss: 0.0373 - val_accuracy: 1.0000
Epoch 116/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2823 - accuracy: 0.91
36 - val_loss: 0.0469 - val_accuracy: 1.0000
Epoch 117/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2121 - accuracy: 0.91
18 - val_loss: 0.0218 - val_accuracy: 1.0000
Epoch 118/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1797 - accuracy: 0.94
00 - val_loss: 0.0216 - val_accuracy: 1.0000
Epoch 119/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2282 - accuracy: 0.91
71 - val_loss: 0.0538 - val_accuracy: 0.9683
Epoch 120/300
36/36 [==============================] - 0s 4ms/step - loss: 0.2079 - accuracy: 0.92
```

```
06 - val_loss: 0.0211 - val_accuracy: 1.0000
Epoch 121/300
36/36 [==============================] - ETA: 0s - loss: 0.2618 - accuracy: 0.92 - 0
s 3ms/step - loss: 0.2733 - accuracy: 0.9153 - val_loss: 0.0144 - val_accuracy: 1.00
00
Epoch 122/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1857 - accuracy: 0.94
89 - val_loss: 0.0110 - val_accuracy: 1.0000
Epoch 123/300
36/36 [==============================] - 0s 4ms/step - loss: 0.2231 - accuracy: 0.92
59 - val_loss: 0.0202 - val_accuracy: 1.0000
Epoch 124/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1855 - accuracy: 0.94
18 - val_loss: 0.0605 - val_accuracy: 0.9683
Epoch 125/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1694 - accuracy: 0.94
89 - val_loss: 0.0091 - val_accuracy: 1.0000
Epoch 126/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1930 - accuracy: 0.91
89 - val_loss: 0.0093 - val_accuracy: 1.0000
Epoch 127/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1801 - accuracy: 0.94
53 - val_loss: 0.0197 - val_accuracy: 1.0000
Epoch 128/300
36/36 [==============================] - 0s 4ms/step - loss: 0.2115 - accuracy: 0.92
77 - val_loss: 0.0540 - val_accuracy: 1.0000
Epoch 129/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1898 - accuracy: 0.93
30 - val_loss: 0.0336 - val_accuracy: 1.0000
Epoch 130/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2089 - accuracy: 0.91
89 - val_loss: 0.0334 - val_accuracy: 1.0000
Epoch 131/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1647 - accuracy: 0.94
18 - val_loss: 0.0123 - val_accuracy: 1.0000
Epoch 132/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1998 - accuracy: 0.92
95 - val_loss: 0.0277 - val_accuracy: 1.0000
Epoch 133/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1613 - accuracy: 0.94
53 - val_loss: 0.0165 - val_accuracy: 1.0000
Epoch 134/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1776 - accuracy: 0.93
12 - val_loss: 0.0299 - val_accuracy: 1.0000
Epoch 135/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1929 - accuracy: 0.93
65 - val_loss: 0.0506 - val_accuracy: 0.9841
Epoch 136/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1824 - accuracy: 0.93
30 - val_loss: 0.0100 - val_accuracy: 1.0000
Epoch 137/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1811 - accuracy: 0.93
83 - val_loss: 0.0228 - val_accuracy: 1.0000
Epoch 138/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1637 - accuracy: 0.95
06 - val_loss: 0.0125 - val_accuracy: 1.0000
Epoch 139/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1701 - accuracy: 0.94
53 - val_loss: 0.0117 - val_accuracy: 1.0000
Epoch 140/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1292 - accuracy: 0.94
71 - val_loss: 0.0252 - val_accuracy: 1.0000
Epoch 141/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1804 - accuracy: 0.94
36 - val_loss: 0.0160 - val_accuracy: 1.0000
Epoch 142/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1723 - accuracy: 0.94
71 - val_loss: 0.0392 - val_accuracy: 1.0000
Epoch 143/300
```

```
36/36 [==============================] - 0s 3ms/step - loss: 0.2404 - accuracy: 0.91
53 - val_loss: 0.0048 - val_accuracy: 1.0000
Epoch 144/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1894 - accuracy: 0.93
30 - val_loss: 0.0016 - val_accuracy: 1.0000
Epoch 145/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1332 - accuracy: 0.95
59 - val_loss: 0.0032 - val_accuracy: 1.0000
Epoch 146/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1543 - accuracy: 0.95
77 - val_loss: 0.0141 - val_accuracy: 1.0000
Epoch 147/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1829 - accuracy: 0.94
00 - val_loss: 0.0571 - val_accuracy: 0.9683
Epoch 148/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1762 - accuracy: 0.94
71 - val_loss: 0.0504 - val_accuracy: 0.9841
Epoch 149/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1570 - accuracy: 0.94
89 - val_loss: 0.0125 - val_accuracy: 1.0000
Epoch 150/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2068 - accuracy: 0.92
77 - val_loss: 0.0036 - val_accuracy: 1.0000
Epoch 151/300
36/36 [==============================] - 0s 4ms/step - loss: 0.2077 - accuracy: 0.92
06 - val_loss: 0.0151 - val_accuracy: 1.0000
Epoch 152/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1642 - accuracy: 0.94
36 - val_loss: 0.0226 - val_accuracy: 1.0000
Epoch 153/300
36/36 [==============================] - 0s 5ms/step - loss: 0.1761 - accuracy: 0.92
24 - val_loss: 0.0094 - val_accuracy: 1.0000
Epoch 154/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1463 - accuracy: 0.94
71 - val_loss: 0.0209 - val_accuracy: 0.9841
Epoch 155/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2063 - accuracy: 0.94
00 - val_loss: 0.0029 - val_accuracy: 1.0000
Epoch 156/300
36/36 [==============================] - 0s 4ms/step - loss: 0.2108 - accuracy: 0.91
89 - val_loss: 0.0108 - val_accuracy: 1.0000
Epoch 157/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1601 - accuracy: 0.94
36 - val_loss: 0.0029 - val_accuracy: 1.0000
Epoch 158/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1521 - accuracy: 0.95
06 - val_loss: 0.0039 - val_accuracy: 1.0000
Epoch 159/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1881 - accuracy: 0.93
12 - val_loss: 0.0152 - val_accuracy: 1.0000
Epoch 160/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1785 - accuracy: 0.94
18 - val_loss: 0.0097 - val_accuracy: 1.0000
Epoch 161/300
36/36 [==============================] - 0s 4ms/step - loss: 0.2322 - accuracy: 0.92
77 - val_loss: 0.0737 - val_accuracy: 0.9683
Epoch 162/300
36/36 [==============================] - 0s 5ms/step - loss: 0.1779 - accuracy: 0.93
83 - val_loss: 0.0195 - val_accuracy: 0.9841
Epoch 163/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1660 - accuracy: 0.94
18 - val_loss: 0.0082 - val_accuracy: 1.0000
Epoch 164/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1688 - accuracy: 0.94
18 - val_loss: 0.0847 - val_accuracy: 0.9683
Epoch 165/300
36/36 [==============================] - 0s 5ms/step - loss: 0.1705 - accuracy: 0.93
65 - val_loss: 0.0457 - val_accuracy: 0.9683
Epoch 166/300
```

```
36/36 [==============================] - 0s 5ms/step - loss: 0.1580 - accuracy: 0.95
77 - val_loss: 0.0084 - val_accuracy: 1.0000
Epoch 167/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1711 - accuracy: 0.94
53 - val_loss: 0.0192 - val_accuracy: 1.0000
Epoch 168/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1581 - accuracy: 0.94
00 - val_loss: 0.0032 - val_accuracy: 1.0000
Epoch 169/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1506 - accuracy: 0.95
94 - val_loss: 0.0056 - val_accuracy: 1.0000
Epoch 170/300
36/36 [==============================] - 0s 3ms/step - loss: 0.2315 - accuracy: 0.92
24 - val_loss: 0.0246 - val_accuracy: 1.0000
Epoch 171/300
36/36 [==============================] - ETA: 0s - loss: 0.2503 - accuracy: 0.91 - 0
s 3ms/step - loss: 0.2550 - accuracy: 0.9171 - val_loss: 0.0032 - val_accuracy: 1.00
00
Epoch 172/300
36/36 [==============================] - 0s 4ms/step - loss: 0.2134 - accuracy: 0.92
77 - val_loss: 0.0062 - val_accuracy: 1.0000
Epoch 173/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1438 - accuracy: 0.94
89 - val_loss: 0.0247 - val_accuracy: 1.0000
Epoch 174/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1325 - accuracy: 0.95
06 - val_loss: 0.0184 - val_accuracy: 0.9841
Epoch 175/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1724 - accuracy: 0.93
30 - val_loss: 0.0063 - val_accuracy: 1.0000
Epoch 176/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1642 - accuracy: 0.94
36 - val_loss: 0.0285 - val_accuracy: 1.0000
Epoch 177/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1673 - accuracy: 0.93
12 - val_loss: 0.0254 - val_accuracy: 1.0000
Epoch 178/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1929 - accuracy: 0.93
12 - val_loss: 0.0063 - val_accuracy: 1.0000
Epoch 179/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1590 - accuracy: 0.95
24 - val_loss: 0.0050 - val_accuracy: 1.0000
Epoch 180/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1819 - accuracy: 0.94
00 - val_loss: 0.0206 - val_accuracy: 1.0000
Epoch 181/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1732 - accuracy: 0.94
00 - val_loss: 0.0484 - val_accuracy: 0.9841
Epoch 182/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1603 - accuracy: 0.94
71 - val_loss: 0.0087 - val_accuracy: 1.0000
Epoch 183/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1485 - accuracy: 0.95
77 - val_loss: 0.0088 - val_accuracy: 1.0000
Epoch 184/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1898 - accuracy: 0.92
59 - val_loss: 0.0091 - val_accuracy: 1.0000
Epoch 185/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1265 - accuracy: 0.95
06 - val_loss: 0.0080 - val_accuracy: 1.0000
Epoch 186/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1570 - accuracy: 0.93
83 - val_loss: 0.0044 - val_accuracy: 1.0000
Epoch 187/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1307 - accuracy: 0.95
59 - val_loss: 0.0429 - val_accuracy: 0.9841
Epoch 188/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1509 - accuracy: 0.94
71 - val_loss: 0.0546 - val_accuracy: 0.9683
```

```
Epoch 189/300
36/36 [==============================] - 0s 5ms/step - loss: 0.1345 - accuracy: 0.94
89 - val_loss: 0.0052 - val_accuracy: 1.0000
Epoch 190/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1741 - accuracy: 0.95
24 - val_loss: 0.0152 - val_accuracy: 1.0000
Epoch 191/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1951 - accuracy: 0.94
18 - val_loss: 0.0048 - val_accuracy: 1.0000
Epoch 192/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1905 - accuracy: 0.93
12 - val_loss: 0.0381 - val_accuracy: 0.9683
Epoch 193/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1765 - accuracy: 0.94
53 - val_loss: 0.0030 - val_accuracy: 1.0000
Epoch 194/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1701 - accuracy: 0.94
53 - val_loss: 0.0063 - val_accuracy: 1.0000
Epoch 195/300
36/36 [==============================] - 0s 5ms/step - loss: 0.1566 - accuracy: 0.94
00 - val_loss: 0.0134 - val_accuracy: 1.0000
Epoch 196/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1435 - accuracy: 0.95
77 - val_loss: 0.0083 - val_accuracy: 1.0000
Epoch 197/300
36/36 [==============================] - 0s 5ms/step - loss: 0.1773 - accuracy: 0.92
77 - val_loss: 0.0097 - val_accuracy: 1.0000
Epoch 198/300
36/36 [==============================] - 0s 6ms/step - loss: 0.1475 - accuracy: 0.95
59 - val_loss: 0.0104 - val_accuracy: 1.0000
Epoch 199/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1490 - accuracy: 0.95
41 - val_loss: 0.0121 - val_accuracy: 1.0000
Epoch 200/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1564 - accuracy: 0.94
53 - val_loss: 0.0188 - val_accuracy: 1.0000
Epoch 201/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1280 - accuracy: 0.95
41 - val_loss: 0.0035 - val_accuracy: 1.0000
Epoch 202/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1231 - accuracy: 0.95
59 - val_loss: 0.0018 - val_accuracy: 1.0000
Epoch 203/300
36/36 [==============================] - 0s 5ms/step - loss: 0.1339 - accuracy: 0.95
24 - val_loss: 0.0279 - val_accuracy: 1.0000
Epoch 204/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1245 - accuracy: 0.95
24 - val_loss: 0.0020 - val_accuracy: 1.0000
Epoch 205/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1217 - accuracy: 0.96
12 - val_loss: 0.0036 - val_accuracy: 1.0000
Epoch 206/300
36/36 [==============================] - 0s 5ms/step - loss: 0.1339 - accuracy: 0.95
77 - val_loss: 0.0358 - val_accuracy: 1.0000
Epoch 207/300
36/36 [==============================] - 0s 4ms/step - loss: 0.2131 - accuracy: 0.92
24 - val_loss: 0.0096 - val_accuracy: 1.0000
Epoch 208/300
36/36 [==============================] - 0s 5ms/step - loss: 0.1544 - accuracy: 0.95
94 - val_loss: 0.0071 - val_accuracy: 1.0000
Epoch 209/300
36/36 [==============================] - 0s 6ms/step - loss: 0.2349 - accuracy: 0.91
71 - val_loss: 0.0060 - val_accuracy: 1.0000
Epoch 210/300
36/36 [==============================] - 0s 5ms/step - loss: 0.1508 - accuracy: 0.95
24 - val_loss: 0.0316 - val_accuracy: 0.9841
Epoch 211/300
36/36 [==============================] - 0s 4ms/step - loss: 0.2485 - accuracy: 0.91
01 - val_loss: 0.0148 - val_accuracy: 1.0000
```

```
Epoch 212/300
36/36 [==============================] - 0s 4ms/step - loss: 0.2420 - accuracy: 0.90
65 - val_loss: 0.0137 - val_accuracy: 1.0000
Epoch 213/300
36/36 [==============================] - 0s 4ms/step - loss: 0.2366 - accuracy: 0.92
24 - val_loss: 0.0246 - val_accuracy: 1.0000
Epoch 214/300
36/36 [==============================] - 0s 5ms/step - loss: 0.1547 - accuracy: 0.95
41 - val_loss: 0.0066 - val_accuracy: 1.0000
Epoch 215/300
36/36 [==============================] - 0s 5ms/step - loss: 0.1221 - accuracy: 0.95
41 - val_loss: 0.0054 - val_accuracy: 1.0000
Epoch 216/300
36/36 [==============================] - 0s 5ms/step - loss: 0.1882 - accuracy: 0.92
95 - val_loss: 0.0951 - val_accuracy: 0.9683
Epoch 217/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1802 - accuracy: 0.95
06 - val_loss: 0.0101 - val_accuracy: 1.0000
Epoch 218/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1216 - accuracy: 0.95
41 - val_loss: 0.0135 - val_accuracy: 1.0000
Epoch 219/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1513 - accuracy: 0.95
41 - val_loss: 0.0096 - val_accuracy: 1.0000
Epoch 220/300
36/36 [==============================] - 0s 5ms/step - loss: 0.1643 - accuracy: 0.94
53 - val_loss: 0.0404 - val_accuracy: 0.9683
Epoch 221/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1584 - accuracy: 0.94
18 - val_loss: 0.0053 - val_accuracy: 1.0000
Epoch 222/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1434 - accuracy: 0.94
71 - val_loss: 0.0066 - val_accuracy: 1.0000
Epoch 223/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1312 - accuracy: 0.95
77 - val_loss: 0.0056 - val_accuracy: 1.0000
Epoch 224/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1800 - accuracy: 0.94
53 - val_loss: 0.0052 - val_accuracy: 1.0000
Epoch 225/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1104 - accuracy: 0.96
65 - val_loss: 0.0016 - val_accuracy: 1.0000
Epoch 226/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1426 - accuracy: 0.96
12 - val_loss: 0.0020 - val_accuracy: 1.0000
Epoch 227/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1253 - accuracy: 0.95
94 - val_loss: 0.0054 - val_accuracy: 1.0000
Epoch 228/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1485 - accuracy: 0.95
77 - val_loss: 0.0393 - val_accuracy: 0.9683
Epoch 229/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1493 - accuracy: 0.94
36 - val_loss: 0.0133 - val_accuracy: 1.0000
Epoch 230/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1395 - accuracy: 0.95
06 - val_loss: 0.0135 - val_accuracy: 1.0000
Epoch 231/300
36/36 [==============================] - 0s 4ms/step - loss: 0.2110 - accuracy: 0.94
18 - val_loss: 0.0203 - val_accuracy: 1.0000
Epoch 232/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1764 - accuracy: 0.93
12 - val_loss: 0.0077 - val_accuracy: 1.0000
Epoch 233/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1379 - accuracy: 0.96
30 - val_loss: 0.0082 - val_accuracy: 1.0000
Epoch 234/300
36/36 [==============================] - 0s 5ms/step - loss: 0.1673 - accuracy: 0.93
30 - val_loss: 0.0015 - val_accuracy: 1.0000
```

```
Epoch 235/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1010 - accuracy: 0.95
94 - val_loss: 9.8755e-04 - val_accuracy: 1.0000
Epoch 236/300
36/36 [==============================] - 0s 5ms/step - loss: 0.0900 - accuracy: 0.96
83 - val_loss: 0.0025 - val_accuracy: 1.0000
Epoch 237/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1362 - accuracy: 0.95
24 - val_loss: 0.0056 - val_accuracy: 1.0000
Epoch 238/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1260 - accuracy: 0.95
24 - val_loss: 0.0186 - val_accuracy: 1.0000
Epoch 239/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1117 - accuracy: 0.95
94 - val_loss: 0.0049 - val_accuracy: 1.0000
Epoch 240/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1377 - accuracy: 0.95
24 - val_loss: 0.0014 - val_accuracy: 1.0000
Epoch 241/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1562 - accuracy: 0.94
36 - val_loss: 0.0067 - val_accuracy: 1.0000
Epoch 242/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1201 - accuracy: 0.95
94 - val_loss: 0.0049 - val_accuracy: 1.0000
Epoch 243/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1792 - accuracy: 0.93
30 - val_loss: 0.0019 - val_accuracy: 1.0000
Epoch 244/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1212 - accuracy: 0.95
24 - val_loss: 0.0028 - val_accuracy: 1.0000
Epoch 245/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1272 - accuracy: 0.95
59 - val_loss: 0.0051 - val_accuracy: 1.0000
Epoch 246/300
36/36 [==============================] - 0s 4ms/step - loss: 0.0967 - accuracy: 0.97
00 - val_loss: 8.7818e-04 - val_accuracy: 1.0000
Epoch 247/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1198 - accuracy: 0.95
77 - val_loss: 0.0014 - val_accuracy: 1.0000
Epoch 248/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1179 - accuracy: 0.95
94 - val_loss: 0.0184 - val_accuracy: 0.9841
Epoch 249/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1072 - accuracy: 0.95
59 - val_loss: 0.0032 - val_accuracy: 1.0000
Epoch 250/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1164 - accuracy: 0.95
59 - val_loss: 0.0013 - val_accuracy: 1.0000
Epoch 251/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1716 - accuracy: 0.94
53 - val_loss: 0.0018 - val_accuracy: 1.0000
Epoch 252/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1220 - accuracy: 0.96
12 - val_loss: 0.0034 - val_accuracy: 1.0000
Epoch 253/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1410 - accuracy: 0.95
24 - val_loss: 0.0026 - val_accuracy: 1.0000
Epoch 254/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1151 - accuracy: 0.95
41 - val_loss: 0.0021 - val_accuracy: 1.0000
Epoch 255/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1520 - accuracy: 0.94
18 - val_loss: 0.0035 - val_accuracy: 1.0000
Epoch 256/300
36/36 [==============================] - 0s 5ms/step - loss: 0.1199 - accuracy: 0.95
06 - val_loss: 3.6406e-04 - val_accuracy: 1.0000
Epoch 257/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1323 - accuracy: 0.95
77 - val_loss: 0.0018 - val_accuracy: 1.0000
```

```
Epoch 258/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1253 - accuracy: 0.94
89 - val_loss: 0.0013 - val_accuracy: 1.0000
Epoch 259/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1090 - accuracy: 0.97
18 - val_loss: 0.0031 - val_accuracy: 1.0000
Epoch 260/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1883 - accuracy: 0.93
65 - val_loss: 0.0036 - val_accuracy: 1.0000
Epoch 261/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1557 - accuracy: 0.95
41 - val_loss: 0.0043 - val_accuracy: 1.0000
Epoch 262/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1088 - accuracy: 0.95
41 - val_loss: 0.0027 - val_accuracy: 1.0000
Epoch 263/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1522 - accuracy: 0.94
89 - val_loss: 0.0049 - val_accuracy: 1.0000
Epoch 264/300
36/36 [==============================] - 0s 4ms/step - loss: 0.0932 - accuracy: 0.97
53 - val_loss: 0.0081 - val_accuracy: 1.0000
Epoch 265/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1003 - accuracy: 0.97
18 - val_loss: 0.0111 - val_accuracy: 1.0000
Epoch 266/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1460 - accuracy: 0.95
06 - val_loss: 0.0071 - val_accuracy: 1.0000
Epoch 267/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1226 - accuracy: 0.95
77 - val_loss: 0.0038 - val_accuracy: 1.0000
Epoch 268/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1572 - accuracy: 0.93
83 - val_loss: 0.0290 - val_accuracy: 1.0000
Epoch 269/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1318 - accuracy: 0.95
77 - val_loss: 0.0069 - val_accuracy: 1.0000
Epoch 270/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1417 - accuracy: 0.95
06 - val_loss: 0.0200 - val_accuracy: 0.9841
Epoch 271/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1000 - accuracy: 0.96
65 - val_loss: 0.0046 - val_accuracy: 1.0000
Epoch 272/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1314 - accuracy: 0.95
77 - val_loss: 0.0094 - val_accuracy: 1.0000
Epoch 273/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1089 - accuracy: 0.96
12 - val_loss: 0.0040 - val_accuracy: 1.0000
Epoch 274/300
36/36 [==============================] - 0s 3ms/step - loss: 0.0856 - accuracy: 0.97
00 - val_loss: 0.0049 - val_accuracy: 1.0000
Epoch 275/300
36/36 [==============================] - ETA: 0s - loss: 0.0829 - accuracy: 0.97 - 0
s 3ms/step - loss: 0.1067 - accuracy: 0.9630 - val_loss: 0.0026 - val_accuracy: 1.00
00
Epoch 276/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1126 - accuracy: 0.96
30 - val_loss: 0.0154 - val_accuracy: 0.9841
Epoch 277/300
36/36 [==============================] - 0s 4ms/step - loss: 0.0852 - accuracy: 0.96
65 - val_loss: 0.0017 - val_accuracy: 1.0000
Epoch 278/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1386 - accuracy: 0.95
06 - val_loss: 0.0071 - val_accuracy: 1.0000
Epoch 279/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1275 - accuracy: 0.95
41 - val_loss: 0.0019 - val_accuracy: 1.0000
Epoch 280/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1504 - accuracy: 0.95
```

```
06 - val_loss: 0.0019 - val_accuracy: 1.0000
Epoch 281/300
36/36 [==============================] - 0s 7ms/step - loss: 0.1188 - accuracy: 0.96
12 - val_loss: 0.0012 - val_accuracy: 1.0000
Epoch 282/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1476 - accuracy: 0.95
77 - val_loss: 0.0018 - val_accuracy: 1.0000
Epoch 283/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1084 - accuracy: 0.95
59 - val_loss: 0.0032 - val_accuracy: 1.0000
Epoch 284/300
36/36 [==============================] - 0s 3ms/step - loss: 0.0959 - accuracy: 0.96
83 - val_loss: 0.0021 - val_accuracy: 1.0000
Epoch 285/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1250 - accuracy: 0.95
41 - val_loss: 9.3564e-04 - val_accuracy: 1.0000
Epoch 286/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1278 - accuracy: 0.95
59 - val_loss: 6.6924e-04 - val_accuracy: 1.0000
Epoch 287/300
36/36 [==============================] - 0s 3ms/step - loss: 0.0911 - accuracy: 0.96
65 - val_loss: 0.0028 - val_accuracy: 1.0000
Epoch 288/300
36/36 [==============================] - 0s 4ms/step - loss: 0.0746 - accuracy: 0.98
06 - val_loss: 0.0092 - val_accuracy: 1.0000
Epoch 289/300
36/36 [==============================] - 0s 5ms/step - loss: 0.1399 - accuracy: 0.94
89 - val_loss: 0.0035 - val_accuracy: 1.0000
Epoch 290/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1051 - accuracy: 0.96
12 - val_loss: 9.3830e-04 - val_accuracy: 1.0000
Epoch 291/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1004 - accuracy: 0.97
35 - val_loss: 0.0040 - val_accuracy: 1.0000
Epoch 292/300
36/36 [==============================] - 0s 3ms/step - loss: 0.1462 - accuracy: 0.95
41 - val_loss: 0.0013 - val_accuracy: 1.0000
Epoch 293/300
36/36 [==============================] - 0s 5ms/step - loss: 0.1382 - accuracy: 0.95
24 - val_loss: 9.4749e-04 - val_accuracy: 1.0000
Epoch 294/300
36/36 [==============================] - 0s 4ms/step - loss: 0.0777 - accuracy: 0.97
18 - val_loss: 8.2144e-04 - val_accuracy: 1.0000
Epoch 295/300
36/36 [==============================] - 0s 4ms/step - loss: 0.0971 - accuracy: 0.96
65 - val_loss: 0.0017 - val_accuracy: 1.0000
Epoch 296/300
36/36 [==============================] - 0s 5ms/step - loss: 0.1374 - accuracy: 0.95
06 - val_loss: 0.0471 - val_accuracy: 0.9841
Epoch 297/300
36/36 [==============================] - 0s 5ms/step - loss: 0.1260 - accuracy: 0.96
30 - val_loss: 0.0032 - val_accuracy: 1.0000
Epoch 298/300
36/36 [==============================] - 0s 4ms/step - loss: 0.1366 - accuracy: 0.95
06 - val_loss: 0.0072 - val_accuracy: 1.0000
Epoch 299/300
36/36 [==============================] - 0s 6ms/step - loss: 0.0835 - accuracy: 0.97
71 - val_loss: 7.3829e-04 - val_accuracy: 1.0000
Epoch 300/300
36/36 [==============================] - 0s 6ms/step - loss: 0.1083 - accuracy: 0.95
94 - val_loss: 0.0011 - val_accuracy: 1.0000
```

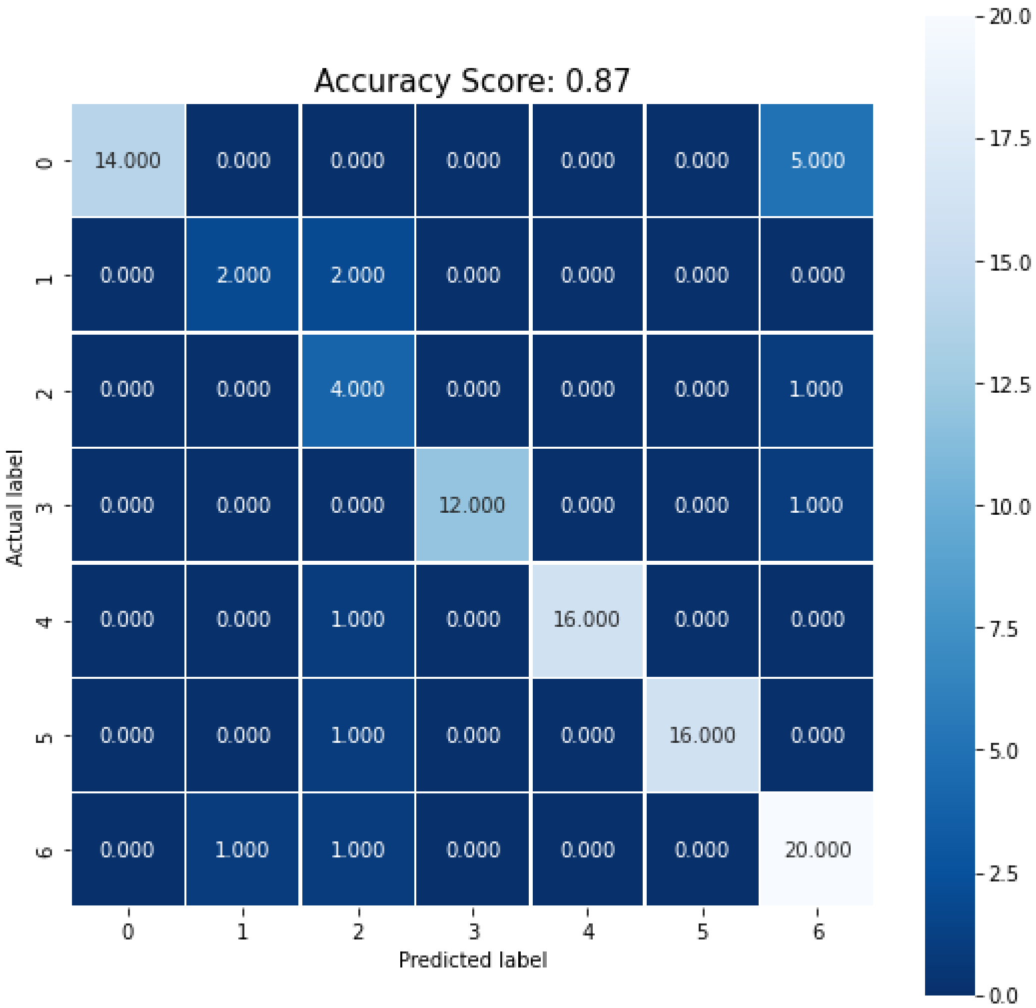
Out[ ]:
```
<keras.callbacks.History at 0x20d3e9b1588>
```

In [ ]:
```python
y_pred_ann = [np.argmax(i) for i in ann.predict(x_test)]
score_ann = accuracy_score(y_test, y_pred_ann)
score_ann
```

Out[ ]: 0.865979381443299

In [ ]:

```
cf_ann = confusion_matrix(y_test, y_pred_ann)
plt.figure(figsize=(9,9))
sns.heatmap(cf_ann, annot=True, fmt=".3f", linewidths=.5, square = True, cmap = 'Blu
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
all_sample_title = 'Accuracy Score: {0}'.format(round(score_ann,2))
plt.title(all_sample_title, size = 15);
```



In [ ]:

```
print(classification_report(y_test,y_pred_ann))
```

```
              precision    recall  f1-score   support

           0       1.00      0.74      0.85        19
           1       0.67      0.50      0.57         4
           2       0.44      0.80      0.57         5
           3       1.00      0.92      0.96        13
           4       1.00      0.94      0.97        17
           5       1.00      0.94      0.97        17
           6       0.74      0.91      0.82        22

    accuracy                           0.87        97
   macro avg       0.84      0.82      0.82        97
weighted avg       0.90      0.87      0.87        97
```

SVM

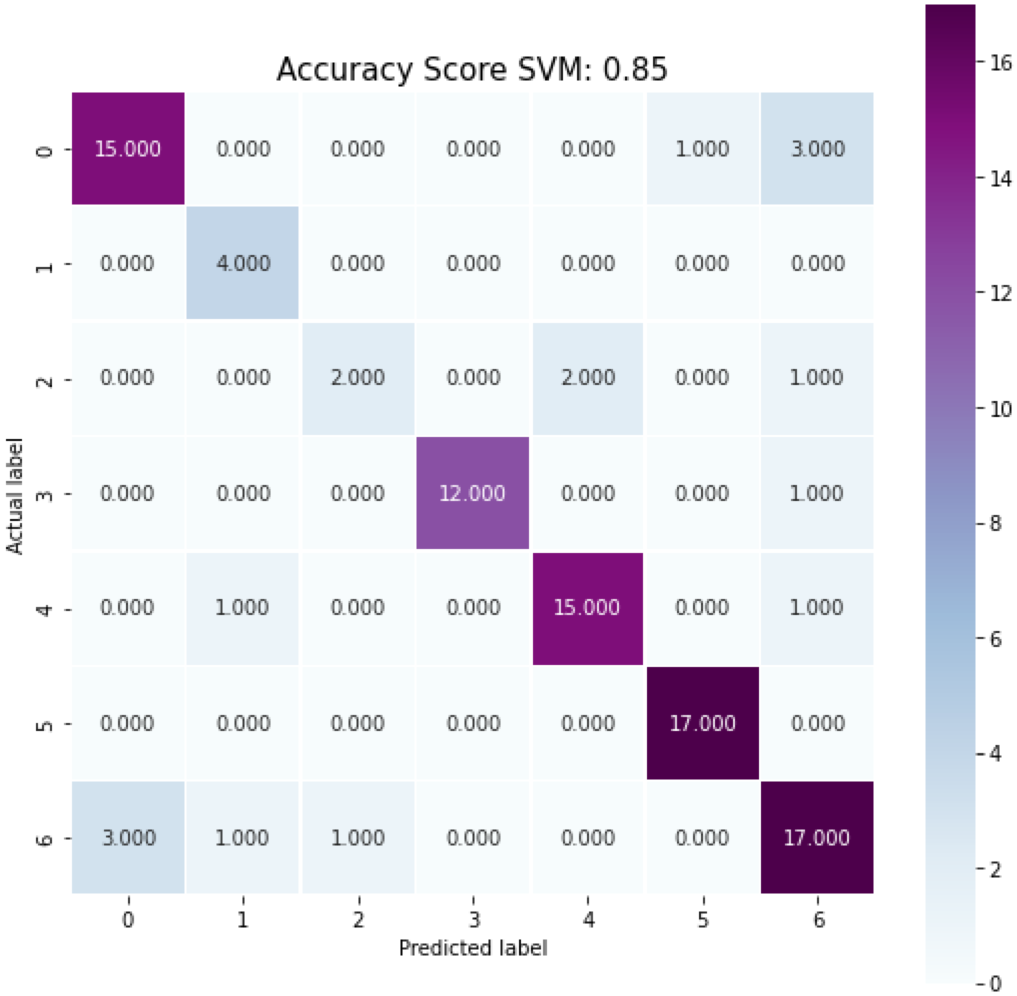
In [ ]:
```
classifier_svm = SVC(kernel = 'rbf', C=2.5, gamma=1.4, random_state = 0)
classifier_svm.fit(X_train, Y_train)
```

Out[ ]:
```
SVC(C=2.5, gamma=1.4, random_state=0)
```

In [ ]:
```
y_pred_svm = classifier_svm.predict(x_test)
cf_svm = confusion_matrix(y_test, y_pred_svm)
score_svm = accuracy_score(y_test, y_pred_svm)
score_svm
```

Out[ ]:
```
0.845360824742268
```

In [ ]:
```
plt.figure(figsize=(9,9))
sns.heatmap(cf_svm, annot=True, fmt=".3f", linewidths=.5, square = True, cmap = 'BuP
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
all_sample_title = 'Accuracy Score SVM: {0}'.format(round(score_svm,2))
plt.title(all_sample_title, size = 15);
```



In [ ]:
```
print(classification_report(y_test,y_pred_svm))
```

```
              precision    recall  f1-score   support

           0       0.83      0.79      0.81        19
           1       0.67      1.00      0.80         4
           2       0.67      0.40      0.50         5
           3       1.00      0.92      0.96        13
           4       0.88      0.88      0.88        17
```

```
           5       0.94      1.00      0.97        17
           6       0.74      0.77      0.76        22

    accuracy                           0.85        97
   macro avg       0.82      0.82      0.81        97
weighted avg       0.85      0.85      0.84        97
```

RF

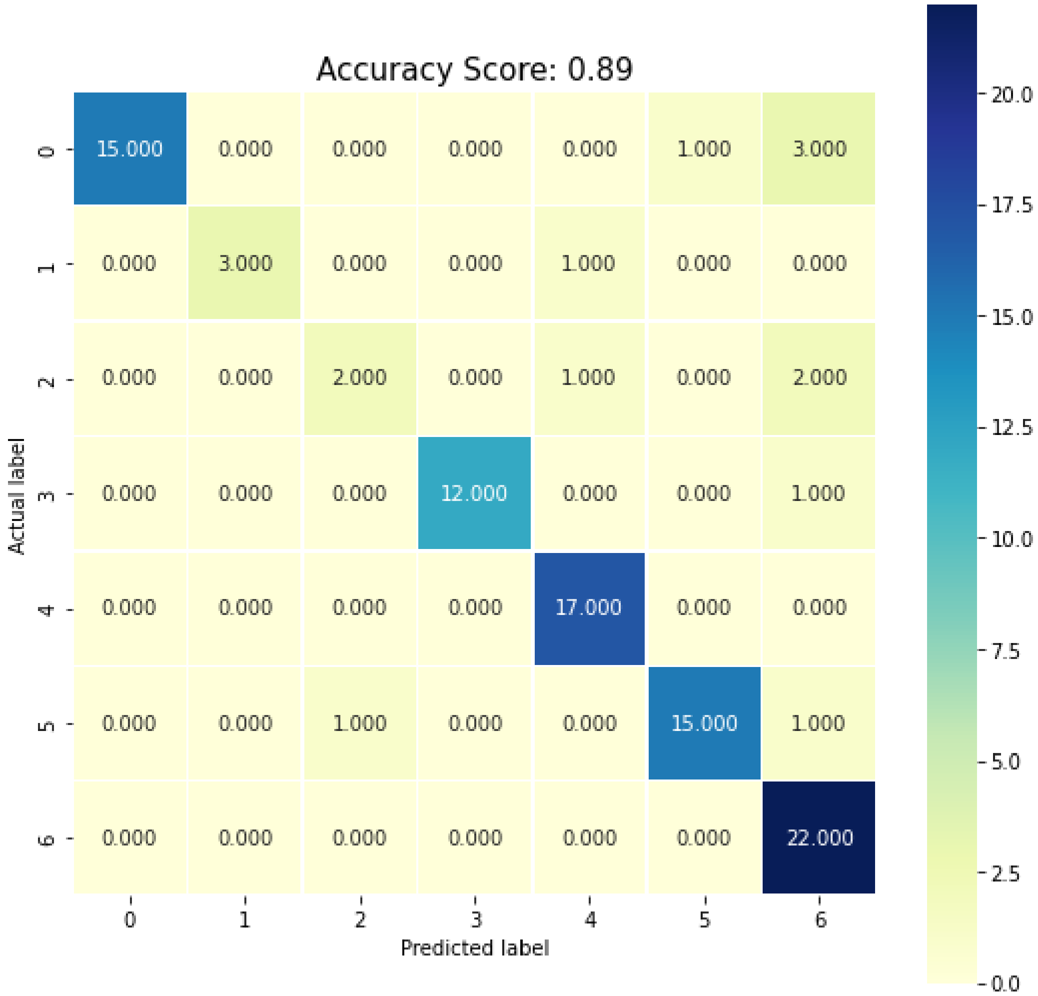
In [ ]:
```
rf = RandomForestClassifier(n_estimators = 300, criterion = 'gini', max_depth=10,  r
rf.fit(X_train, Y_train)
```

Out[ ]:
```
RandomForestClassifier(max_depth=10, n_estimators=300, random_state=42)
```

In [ ]:
```
y_pred_rf = rf.predict(x_test)
#y_pred_rf = [int(i) for i in y_pred_rf]
cf_rf = confusion_matrix(y_test, y_pred_rf)
score_rf = accuracy_score(y_test, y_pred_rf)
score_rf = round(score_rf,2)
score_rf
```

Out[ ]:
```
0.89
```

In [ ]:
```
cf_rf = confusion_matrix(y_test, y_pred_rf)
plt.figure(figsize=(9,9))
sns.heatmap(cf_rf, annot=True, fmt=".3f", linewidths=.5, square = True, cmap = 'YlGn
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
all_sample_title = 'Accuracy Score: {0}'.format(score_rf)
plt.title(all_sample_title, size = 15);
```



In [ ]:

```
print(classification_report(y_test,y_pred_rf))
```

```
              precision    recall  f1-score   support

           0       1.00      0.79      0.88        19
           1       1.00      0.75      0.86         4
           2       0.67      0.40      0.50         5
           3       1.00      0.92      0.96        13
           4       0.89      1.00      0.94        17
           5       0.94      0.88      0.91        17
           6       0.76      1.00      0.86        22

    accuracy                           0.89        97
   macro avg       0.89      0.82      0.85        97
weighted avg       0.90      0.89      0.88        97
```