Surface type detection for the robot's indoor navigation using Signal Processing and Machine Learning

A Project Submitted in Partial Fulfillment of the Requirement for the Degree of

BACHELOR OF TECHNOLOGY

Electrical Engineering

by

Akansh Maurya (1705220003)
Akash Singh (1705220004)
Lokendra Rathore (1705220025)

Under the Supervision of

Dr. Seethalekshmi K

(Head of Department, Electrical Engineering Department)

to the

Department of Electrical Engineering

INSTITUTE OF ENGINEERING & TECHNOLOGY

(Sitapur Road, Lucknow, Uttar Pradesh, India)

July, 2021

<u>CERTIFICATE</u>

Certified that **Akansh Maurya** (1705220003), **Akash Singh** (1705220004), **Lokendra Rathore** (1705220025) have carried out the project work presented in this project entitled **“Surface type detection for the robot’s indoor navigation using Signal Processing and Machine Learning”** for the award of **Bachelor of Technology** in **Electrical Engineering** at **Institute of Engineering & Technology, Lucknow** under my supervision. The project embodies results of original work, and studies are carried out by the students themselves and the contents of the project do not form the basis for the award of any other degree to the candidates or to anybody else from this or any other University/Institution.

Date: 29 July 2021

Place: Lucknow

Signature

Dr. Seethalekshmi K

(Head Of Department)

(Institute of Engineering And Technology, Lucknow)

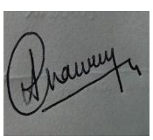
ABSTRACT

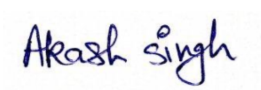
Humans have many sensory organs such as eyes, nose, skin, etc., that make us feel. In nature, these senses are so comprehensive that humans can sense our environment without any effort; for example, we can tell the difference between a plastic and a steel bottle by merely touching it or discriminating between the floor by walking, whether it is a concrete or an unpaved path. In contrast to this, the robots we have created to date do not have sensory systems comparable to humans. However, to better understand and correctly navigate a task, they need input about their surroundings. In this project, attempts are made to help robots to sense the environment. Specifically, the floor surface they are moving on using data collected from Inertial Measurement Units (acceleration, velocity, etc.; collectively ten sensor channels). The data is managed by the Department of Automation and Mechanical Engineering at Tampere University, Finland by driving a small mobile robot on different surfaces (9 classes). We will be processing the data in both the frequency and time domain to obtain features to train our machine learning model. Furthermore, we will use advanced deep learning techniques like one-dimensional convolutional neural networks and later compare them with traditional methods.

ACKNOWLEDGEMENT

I, Akansh Maurya(1705220003) along with my teammates Akash Singh(1705220004) and Lokendra Rathore(1705220025) would like to express a deep sense of thanks and gratitude to our project guide **Dr. Seethalekshmi K Ma'am** for guiding us immensely through the course of the project. Our sincere thanks goes to **Dr. Nitin Anand Shrivastava Sir** and **Dr. Shweta Ma'am** for their coordination in extending every possible support for the completion of this project. We must thank our classmates for their timely help & support in achieving the project's objectives. Last but not the least, we would like to thank all those who had helped directly or indirectly towards the fulfillment of this project.

Thanking You





Akansh Maurya	Akash Singh	Lokendra Rathore
1705220003	1705220004	1705220025

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

1. FFT: Fast Fourier Transform
2. IFFT: Inverse Fast Fourier Transform
3. IMU: Inertial Measurement Unit
4. MEMS: Micro-Electro-Mechanical Systems
5. CWT: Continuous Wavelet Transform
6. DWT: Discrete Wavelet Transform
7. CV: Cross Validation
8. SVC: Support Vector Classifier
9. SGD: Stochastic Gradient Descent
10. NB: Naive Bias
11. ACC: Accuracy
12. NN: Neural Network
13. CNN/ConvNet: Convolutional Neural Network
14. LSTM: Long Short Term Memory

CHAPTER – 1

INTRODUCTION

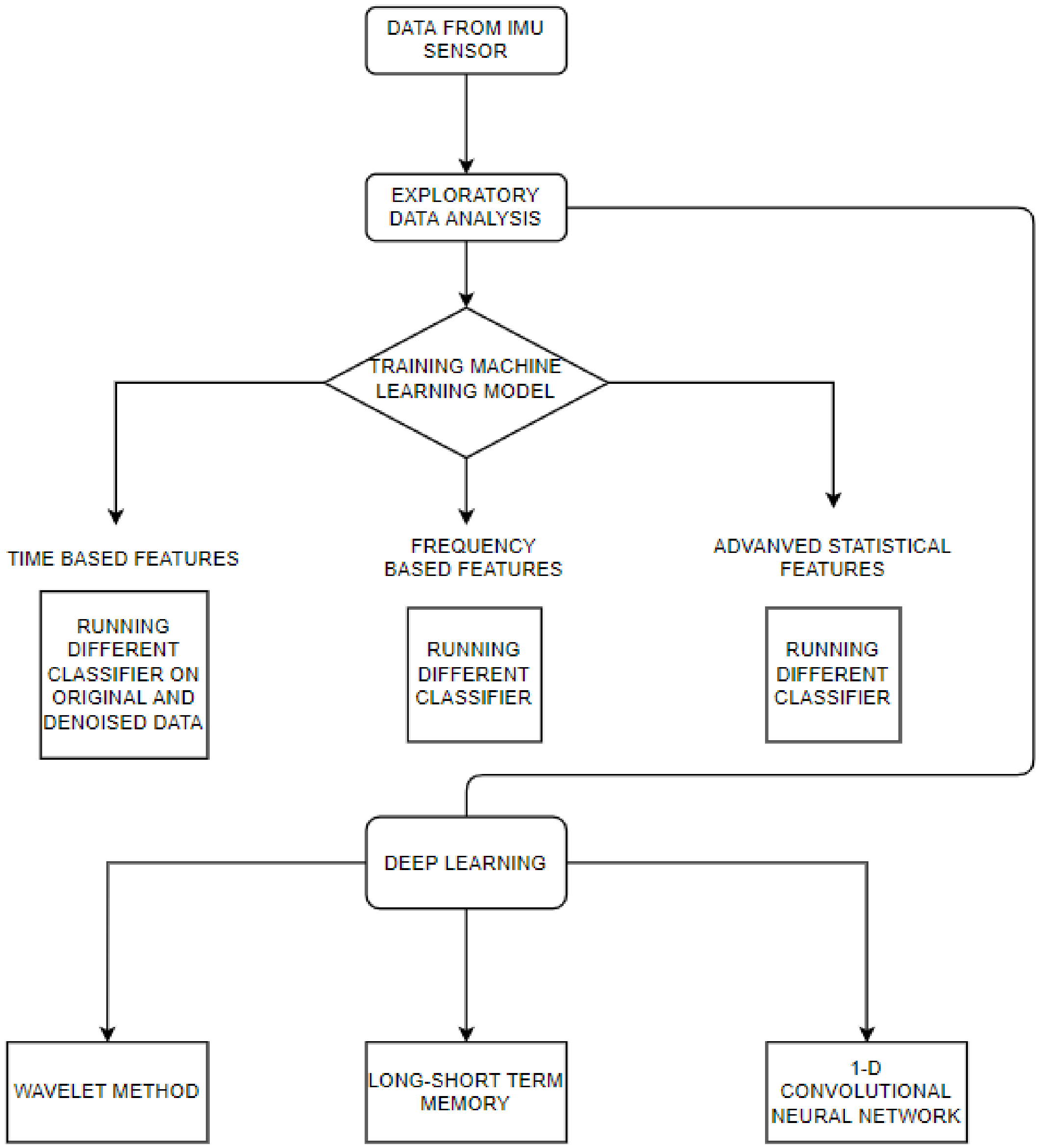
Humans have many sensory organs such as eyes, nose, skin, etc., that make us feel. In nature, these senses are so comprehensive that humans can sense our environment without any effort; for example, we can tell the difference between a plastic and a steel bottle by merely touching it or discriminating between the floor by walking, whether it is a concrete or an unpaved path. In contrast to this, the robots we have created to date do not have sensory systems comparable to humans. However, to better understand and correctly navigate a task, they need input about their surroundings. In this project, attempts are made to help robots to sense the environment. Specifically, the floor surface they are moving on using data collected from Inertial Measurement Units (acceleration, velocity, etc.; collectively ten sensor channels). The data is managed by the Department of Automation and Mechanical Engineering at Tampere University, Finland by driving a small mobile robot on different surfaces (9 classes). We will be processing the data in both the frequency and time domain to obtain features to train our machine learning model. Furthermore, we will use advanced deep learning techniques like one-dimensional convolutional neural networks and later compare them with traditional methods.

1.1 Project Objective:

1. To build a better sensory recognition system for robots.
2. Learning about IMU sensors and in general know the concepts of MEMS (Micro-electromechanical system)
3. Use signal De-noising, FFT techniques for better classification and feature calculation.
4. Using Traditional Machine Learning to classify signals.
5. Using advanced Deep Learning Techniques to classify signals:
    - Continuous Wavelet Transform(CWT) + Scalogram and Deep convolutional network.
    - LSTM(Long Short Term Memory).
    - 1-D ConvNet

1.2. Problem Approach Methodology:

Our methodology can be illustrated by the flow chart shown below. Project utilizes the concepts of Signal Processing, Machine Learning and Deep Learning. All the code is written in Python.

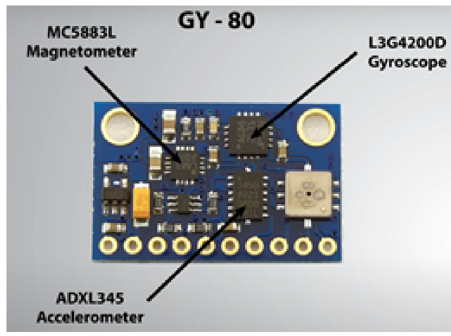


1.3: Study of Data Acquisition Source:

1.3.1: IMU: Inertial Measurement Unit

Consists of three sensors:

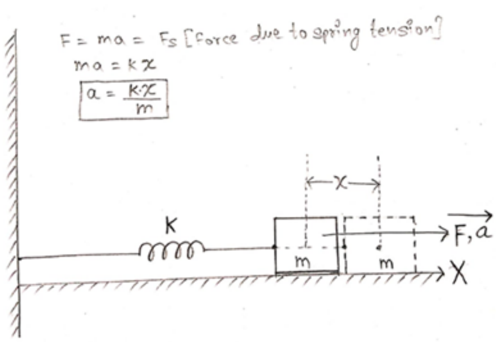
1. Accelerometer
2. Gyroscope
3. Magnetometer



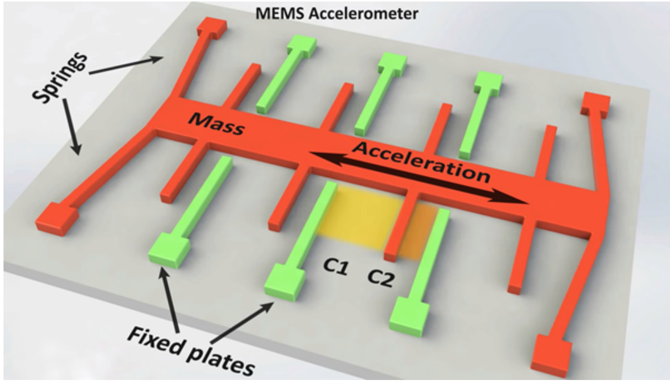
We have learned about several common sensors that are integrated into our phones, including accelerometers, gyroscopes, and magnetometers. Cell phone processors find the location and orientation of the phone in a three-dimensional space by using these sensors.

1.3.2: Accelerometer Working Process:

We try to calculate acceleration using a spring and mass device. A body of mass ‘m’ is connected to a wall by a spring with the spring coefficient k, as seen in the diagram.
a=f(x)



If it is somehow possible to quantify displacement, we will determine the body's acceleration.



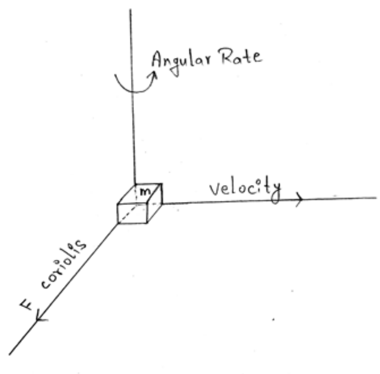
We know that,

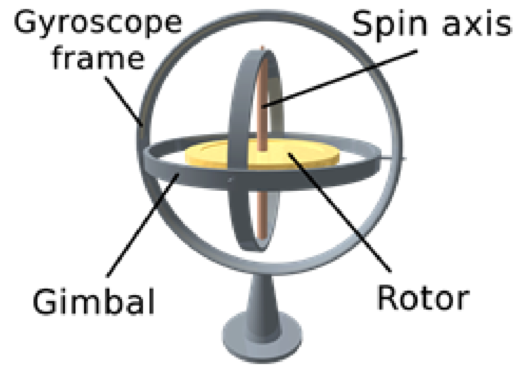
$C = \varepsilon o(A / d)$

The capacitance C varies through the fixed and movable plates due to 'x'. We can measure the value of 'x 'by calculating this C, which allows us to infer the value of ' a '. However, within accelerometer IC's, we could not employ such large spring-mass structures. This is where MEMS are introduced. MEMS stands for the micro-electro-mechanical system. These devices include mechanical as well as electronic parts but are built on a micrometre scale. Getting several MEMS structures of this type in separate planes, i.e. The accelerometer X, Y, Z offers the acceleration measurements as seen on the body in various directions.

1.3.3: Gyroscope working:

One of the sensors used inside an IMU is the gyroscope (Inertial measurement unit). We notice several gadgets that have motion control built into them.

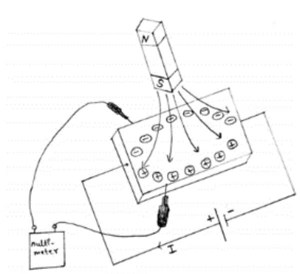




By measuring the change in displacement, we can calculate the magnitude of Coriolis force and thereby calculate the angular rotation. This whole assembly is built as MEMS of the scale of a micrometer. This makes it possible to manufacture small IC's for gyroscopes.
Having such systems placed in all three axes helps to find the angular velocity in the X, Y, and Z direction.

1.3.4: Magnetometer working:

The magnetometer is used to detect the magnetic fields near the body or object that are present. We can find the north direction by observing the magnetic fields and thereby recognize our orientation and position. In order to sense directions, several smartphones are fitted with a magnetometer.
There are two ways to calculate magnetic field, 'HALL' effect and the magneto-resistive effect. In short, 90 percent of the market sensors use the Hall effect, and the rest 10% use the magneto-resistive effect.



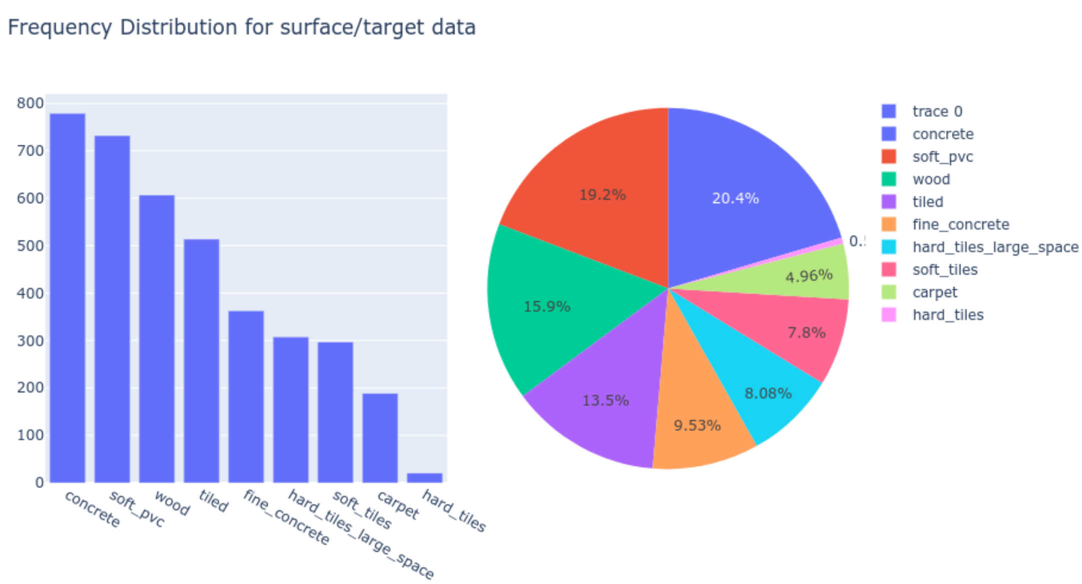
CHAPTER 2

DATA AND SIGNAL ANALYSIS

2.1: Exploratory Data Analysis

The data used in this project is collected by the Department of Automation and Mechanical Engineering at Tampere University, Finland by driving a small mobile robot on different surfaces (9 classes). These 9 classes are namely and there counts:

	target	surface
0	concrete	779
1	soft_pvc	732
2	wood	607
3	tiled	514
4	fine_concrete	363
5	hard_tiles_large_space	308
6	soft_tiles	297
7	carpet	189
8	hard_tiles	21

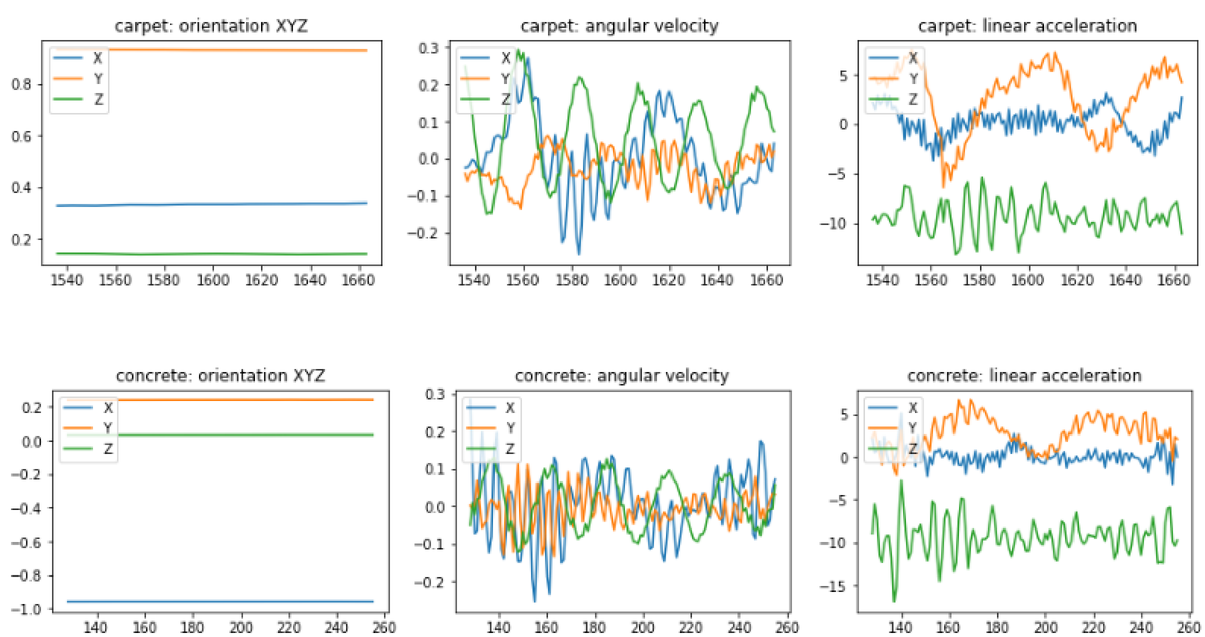


There are 9 different classes whose names and distribution are shown in the above figure. From the data, we can observe that Concrete is the most repeating class while hard-tiles occur the least.

In our training set, we have 487680 rows and 13 columns.487680 rows represent 3810 different series of 128 data points, also these series are labeled with corresponding series ID, surface name, and group ID. Out of 13 columns, 10 columns represent different channel values obtained from IMU sensors. These sensor channels and example of a series for a surface type "Concrete" is shown:

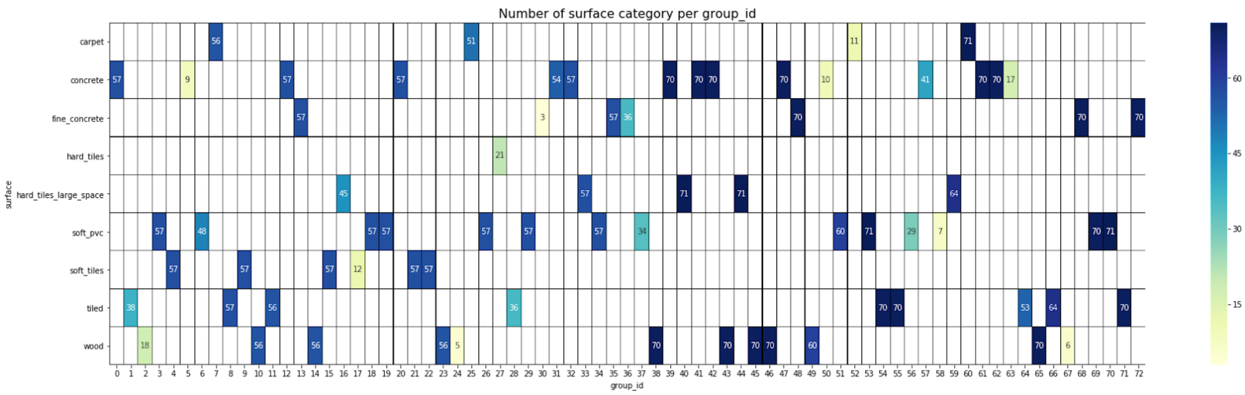
Sensor channels:

- Sensor Channel 0:orientation_X
- Sensor Channel 1:orientation_Y
- Sensor Channel 2:orientation_Z
- Sensor Channel 3:orientation_W
- Sensor Channel 4:angular_velocity_X
- Sensor Channel 5:angular_velocity_Y
- Sensor Channel 6:angular_velocity_Z
- Sensor Channel 7:linear_acceleration_X
- Sensor Channel 8:linear_acceleration_Y
- Sensor Channel 9:linear_acceleration_Z

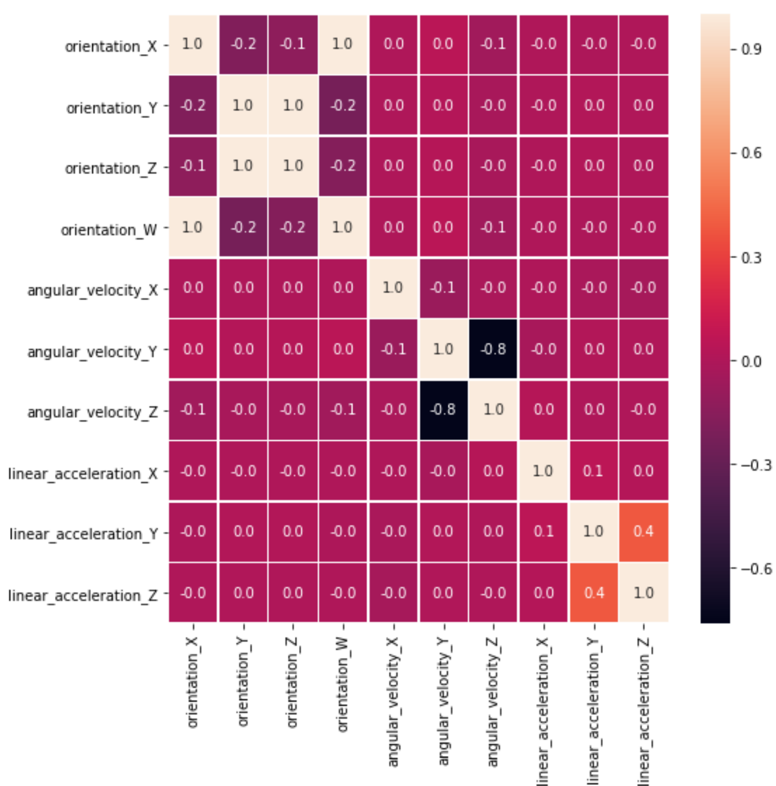


Group ID denotes the session for recording. For eg, recording session 1 only hovers over the iled surface of the robot. This is important to know, as IMU sensor calculation is highly influenced by factors such as battery level, climate, etc. We need to break the set in terms of category ID in order to do a valid cross-validation set.

In the project, different signals obtained from different channels were thoroughly analyzed, and also correlation coefficient between signals was calculated. There was a strong correlation between:



1. Angular_velocity_Z and Angular_velocity_Y
2. Orientation_X and orientation_Y
3. Orientation_Y and orientation_Z



CHAPTER- 3

TIME AND FREQUENCY BASED FEATURES

3.1: Feature Engineering (Time based)

We first computed basic statistical features on the dataset. These Statistical features are as follows:

- Mean
- Standard deviation
- Maximum
- Minimum
- Max to Min Ratio
- First-order derivative
- Second-order derivative

After computing these features on signal data we have used 20 different classifiers:

GradientBoostingClassifier	KNeighborsClassifier
XGBClassifier	QuadraticDiscriminantAnalysis
RandomForestClassifier	GaussianProcessClassifier
ExtraTreesClassifier	SGDClassifier
BaggingClassifier	GaussianNB
DecisionTreeClassifier	PassiveAggressiveClassifier
ExtraTreeClassifier	RidgeClassifierCV
LinearSVC	Perceptron
LogisticRegressionCV	SVC
LinearDiscriminantAnalysis	BernoulliNB

	MLA Name	MLA Parameters	MLA Train Accuracy	MLA Train Accuracy Mean	MLA Test Accuracy	MLA Test Accuracy Mean	MLA Test Accuracy Std
3	GradientBoostingClassifier	{'ccp_alpha': 0.0, 'criterion': 'friedman_mse'...	[0.9973761889143982, 0.9908136482939632, 0.996...	0.993635	[0.48751642575558474, 0.37139107611548555, 0.4...	0.476381	0.0558823
20	XGBClassifier	{'objective': 'binary:logistic', 'use_label_en...	[1.0, 1.0, 1.0, 1.0, 1.0]	1	[0.49934296977660975, 0.37270341207349084, 0.4...	0.471669	0.0576054
4	RandomForestClassifier	{'bootstrap': True, 'ccp_alpha': 0.0, 'class_w...	[1.0, 1.0, 1.0, 1.0, 1.0]	1	[0.4980289093298292, 0.34908136482939633, 0.42...	0.469048	0.0722208
2	ExtraTreesClassifier	{'bootstrap': False, 'ccp_alpha': 0.0, 'class_...	[1.0, 1.0, 1.0, 1.0, 1.0]	1	[0.5256241787122208, 0.3320209973753281, 0.424...	0.460394	0.0749429
1	BaggingClassifier	{'base_estimator': None, 'bootstrap': True, 'b...	[0.9931124959002952, 0.9957349081364829, 0.994...	0.993635	[0.507227332457293, 0.30708661417322836, 0.385...	0.431791	0.0754181
16	DecisionTreeClassifier	{'ccp_alpha': 0.0, 'class_weight': None, 'crit...	[1.0, 1.0, 1.0, 1.0, 1.0]	1	[0.43626806833114323, 0.3320209973753281, 0.39...	0.407885	0.0426353
17	ExtraTreeClassifier	{'ccp_alpha': 0.0, 'class_weight': None, 'crit...	[1.0, 1.0, 1.0, 1.0, 1.0]	1	[0.4152431011826544, 0.2874015748031496, 0.363...	0.394239	0.0627825
15	LinearSVC	{'C': 1.0, 'class_weight': None, 'dual': True,...	[0.6188914398163332, 0.6551837270341208, 0.668...	0.646394	[0.4336399474375821, 0.34120734908136485, 0.30...	0.393211	0.0606288
6	LogisticRegressionCV	{'Cs': 10, 'class_weight': None, 'cv': None, '...	[0.6448015742866514, 0.6850393700787402, 0.689...	0.670541	[0.43889618922470436, 0.3451443569553806, 0.30...	0.391637	0.0625872
18	LinearDiscriminantAnalysis	{'covariance_estimator': None, 'n_components':...	[0.5998688094457199, 0.6322178477690289, 0.641...	0.617457	[0.43889618922470436, 0.35170603674540685, 0.3...	0.382187	0.0533244
13	KNeighborsClassifier	{'algorithm': 'auto', 'leaf_size': 30, 'metric...	[0.8209248934076746, 0.8389107611548556, 0.823...	0.824672	[0.4244415243101183, 0.2979002624671916, 0.348...	0.37771	0.0555544
19	QuadraticDiscriminantAnalysis	{'priors': None, 'reg_param': 0.0, 'store_cova...	[0.7094129222695966, 0.6978346456692913, 0.746...	0.711485	[0.38633377135348224, 0.21916010498687663, 0.3...	0.37009	0.0864973
5	GaussianProcessClassifier	{'copy_X_train': True, 'kernel': None, 'max_it...	[0.9954083306001967, 0.9967191601049868, 0.997...	0.996851	[0.43101182654402104, 0.28083989501312334, 0.3...	0.366952	0.052384
14	SVC	{'C': 1.0, 'break_ties': False, 'cache_size': ...	[0.7566415218104297, 0.7680446194225722, 0.759...	0.763583	[0.4244415243101183, 0.23622047244094488, 0.32...	0.361444	0.0726443
9	SGDClassifier	{'alpha': 0.0001, 'average': False, 'class_wei...	[0.5241062643489669, 0.5406824146981627, [illegible]	0.531628	[0.38370565045992117, 0.28346456692913385, 0.3...	0.361416	0.0396081

With these features **Gradient boosting classifier** performs best in this case with **Mean test accuracy** on 5 fold set up to **47.638 %** and **standard deviation of 5.58 %.** The above figure shows the best 4 classifiers with the same data.

3.2: Feature Engineering (Frequency based)

These Statistical features will be used on FFT of signals :

- Mean
- Standard deviation
- Maximum
- Minimum
- Max to Min Ratio
- First-order derivative
- Second-order derivative

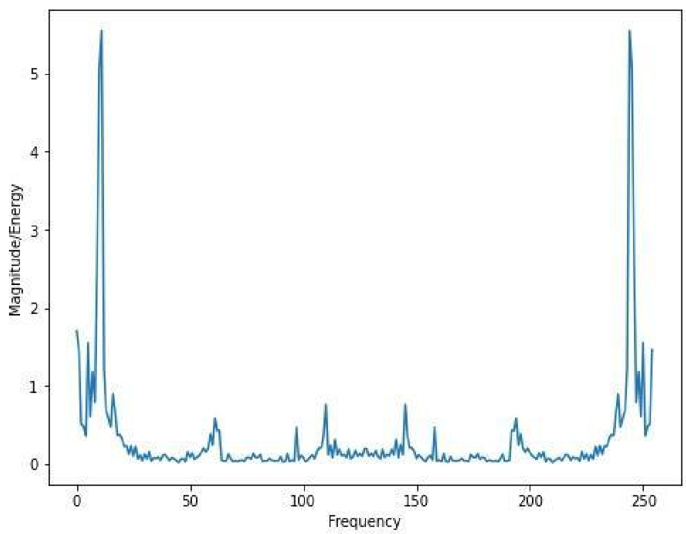
	MLA Name	MLA Parameters	MLA Train Accuracy	MLA Train Accuracy Mean	MLA Test Accuracy	MLA Test Accuracy Mean	MLA Test Accuracy Std
4	RandomForestClassifier	{'bootstrap': True, 'ccp_alpha': 0.0, 'class_w...	[1.0, 1.0, 1.0, 1.0, 1.0]	1	[0.545335085413929, 0.33727034120734906, 0.463...	0.477974	0.0773815
3	GradientBoostingClassifier	{'ccp_alpha': 0.0, 'criterion': 'friedman_mse'...	[0.979993440472286, 0.979002624671916, 0.97472...	0.976443	[0.5229960578186597, 0.3779527559055118, 0.484...	0.475601	0.0551668
2	ExtraTreesClassifier	{'bootstrap': False, 'ccp_alpha': 0.0, 'class_...	[1.0, 1.0, 1.0, 1.0, 1.0]	1	[0.5243101182654402, 0.32808398950131235, 0.47...	0.470615	0.0728671
5	GaussianProcessClassifier	{'copy_X_train': True, 'kernel': None, 'max_it...	[0.8950475565759265, 0.8996062992125984, 0.893...	0.897703	[0.5256241787122208, 0.3438320209973753, 0.440...	0.469051	0.0709422
1	BaggingClassifier	{'base_estimator': None, 'bootstrap': True, 'b...	[0.9924565431288948, 0.9908136482939632, 0.994...	0.993504	[0.533508541392904, 0.3123359580052493, 0.4692...	0.465371	0.0817689
20	XGBClassifier	{'objective': 'binary:logistic', 'use_label_en...	[1.0, 1.0, 1.0, 1.0, 1.0]	1	[0.4888304862023653, 0.35039370078740156, 0.47...	0.465358	0.061768
14	SVC	{'C': 1.0, 'break_ties': False, 'cache_size': ...	[0.7176123319121023, 0.7509842519685039, 0.726...	0.732481	[0.5045992115637319, 0.32677165354330706, 0.42...	0.449889	0.0672817
13	KNeighborsClassifier	{'algorithm': 'auto', 'leaf_size': 30, 'metric...	[0.875696949819613, 0.8848425196850394, 0.8831...	0.880053	[0.4520367936925099, 0.32545931758530183, 0.42...	0.43964	0.0658387
6	LogisticRegressionCV	{'Cs': 10, 'class_weight': None, 'cv': None, '...	[0.5677271236470974, 0.6069553805774278, 0.581...	0.574476	[0.41655716162943496, 0.3556430446194226, 0.42...	0.41312	0.0312195
16	DecisionTreeClassifier	{'ccp_alpha': 0.0, 'class_weight': None, 'crit...	[1.0, 1.0, 1.0, 1.0, 1.0]	1	[0.44415243101182655, 0.3136482939632546, 0.44...	0.410499	0.0490865
15	LinearSVC	{'C': 1.0, 'class_weight': None, 'dual': True,...	[0.5401771072482782, 0.5656167979002624, 0.561...	0.545408	[0.36136662286465177, 0.3188976377952756, 0.41...	0.394474	0.0587496
17	ExtraTreeClassifier	{'ccp_alpha': 0.0, 'class_weight': None, 'crit...	[1.0, 1.0, 1.0, 1.0, 1.0]	1	[0.44021024967148487, 0.2847769028871391, 0.38...	0.382954	0.0538393
18	LinearDiscriminantAnalysis	{'covariance_estimator': None, 'n_components':...	[0.5152509019350606, 0.5446194225721784, 0.522...	0.523098	[0.36662286465177396, 0.32677165354330706, 0.3...	0.37112	0.0309412
8	RidgeClassifierCV	{'alphas': array([0.1, 1. , 10.]), 'class_w...	[0.461462774680223, 0.511482939632546, 0.48244...	0.476773	[0.36268068331143233, 0.2388451443569554, 0.36...	0.355119	0.0761521
12	GaussianNB	{'priors': None, 'var_smoothing': 1e-09}	[0.43096097081010165, 0.45013123359580054, 0.4...	0.424409	[0.25098554533508544, 0.29133858267716534, 0.3...	0.351416	0.0809647

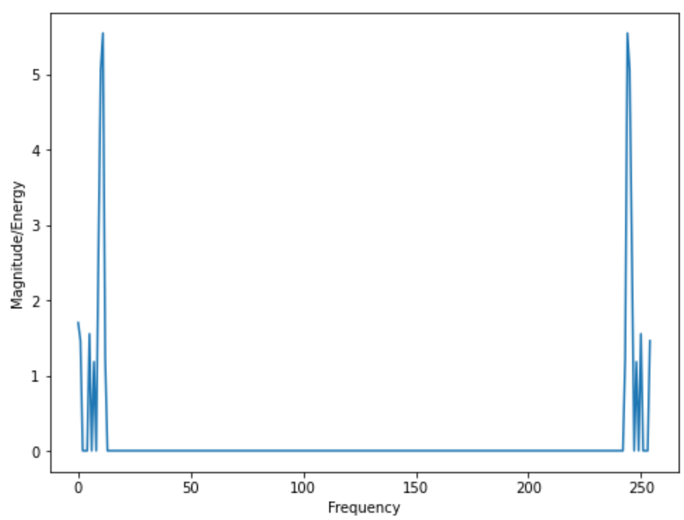
3.3: Both Frequency and time level concatenation feature

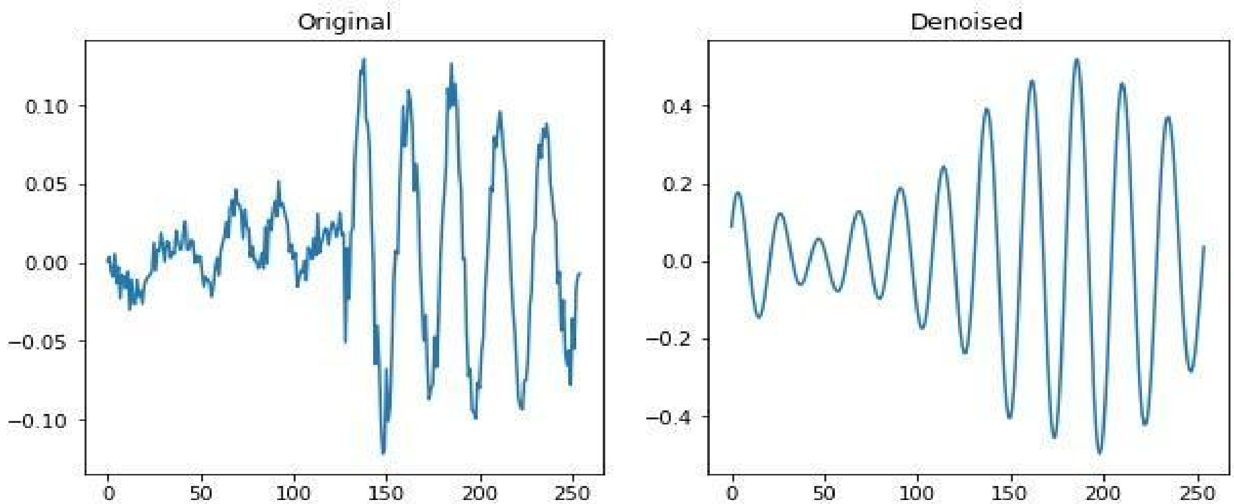
	MLA Name	MLA Parameters	MLA Train Accuracy	MLA Train Accuracy Mean	MLA Test Accuracy	MLA Test Accuracy Mean	MLA Test Accuracy Std
3	GradientBoostingClassifier	{'ccp_alpha': 0.0, 'criterion': 'friedman_mse'...	[0.9983601180714988, 0.994750656167979, 0.9957...	0.994882	[0.48226018396846254, 0.37139107611548555, 0.4...	0.472703	0.0525797
2	ExtraTreesClassifier	{'bootstrap': False, 'ccp_alpha': 0.0, 'class_...	[1.0, 1.0, 1.0, 1.0, 1.0]	1	[0.5400788436268068, 0.31758530183727035, 0.43...	0.46617	0.0835424
20	XGBClassifier	{'objective': 'binary:logistic', 'use_label_en...	[1.0, 1.0, 1.0, 1.0, 1.0]	1	[0.5019710906701709, 0.36089238845144356, 0.43...	0.465635	0.0610009
4	RandomForestClassifier	{'bootstrap': True, 'ccp_alpha': 0.0, 'class_w...	[1.0, 1.0, 1.0, 1.0, 1.0]	1	[0.533508541392904, 0.3241469816272966, 0.4154...	0.461711	0.0810807
1	BaggingClassifier	{'base_estimator': None, 'bootstrap': True, 'b...	[0.9921285667431945, 0.9954068241469817, 0.995...	0.993898	[0.5111695137976346, 0.3136482939632546, 0.398...	0.436775	0.0767904
6	LogisticRegressionCV	{'Cs': 10, 'class_weight': None, 'cv': None, '...	[0.7412266316825189, 0.7893700787401575, 0.793...	0.766408	[0.4783180026281209, 0.2874015748031496, 0.407...	0.424166	0.0745984
15	LinearSVC	{'C': 1.0, 'class_weight': None, 'dual': True,...	[0.7015414890127911, 0.7526246719160105, 0.754...	0.731434	[0.4507227332457293, 0.2887139107611549, 0.359...	0.418659	0.0822999
14	SVC	{'C': 1.0, 'break_ties': False, 'cache_size': ...	[0.830108232207281, 0.833989501312336, 0.82868...	0.832087	[0.47963206307490147, 0.27165354330708663, 0.3...	0.417616	0.0884374
13	KNeighborsClassifier	{'algorithm': 'auto', 'leaf_size': 30, 'metric...	[0.8743850442768121, 0.8884514435695539, 0.877...	0.876509	[0.4783180026281209, 0.3530183727034121, 0.346...	0.417358	0.0586299
16	DecisionTreeClassifier	{'ccp_alpha': 0.0, 'class_weight': None, 'crit...	[1.0, 1.0, 1.0, 1.0, 1.0]	1	[0.4533508541392904, 0.30446194225721784, 0.39...	0.414714	0.0670718
18	LinearDiscriminantAnalysis	{'covariance_estimator': None, 'n_components':...	[0.6634962282715644, 0.719488188976378, 0.7065...	0.686814	[0.492772667542707, 0.2677165354330709, 0.3787...	0.409741	0.0820533
5	GaussianProcessClassifier	{'copy_X_train': True, 'kernel': None, 'max_it...	[0.9990160708428993, 0.9990157480314961, 0.999...	0.999213	[0.4651773981603154, 0.3359580052493438, 0.357...	0.403965	0.0510329
9	SGDClassifier	{'alpha': 0.0001, 'average': False, 'class_wei...	[0.5952771400459167, 0.6374671916010499, 0.670...	0.624021	[0.328515111695138, 0.24015748031496062, 0.365...	0.387129	0.102188
17	ExtraTreeClassifier	{'ccp_alpha': 0.0, 'class_weight': None, 'crit...	[1.0, 1.0, 1.0, 1.0, 1.0]	1	[0.4375821287779238, 0.2230971128608924, 0.397...	0.385837	0.0851408

3.4: Denoising Data

In the next step, we used Fast Fourier transform(FFT) to denoise the signal. In the frequency domain, a filter is applied to an image by multiplying the FFT of that image by the FFT of the filter. Whenever the FFT of an image is multiplied by the FFT of a filter to perform convolution, this process is known as windowing. In this process, we remove those frequencies in the signal that have very little energy.

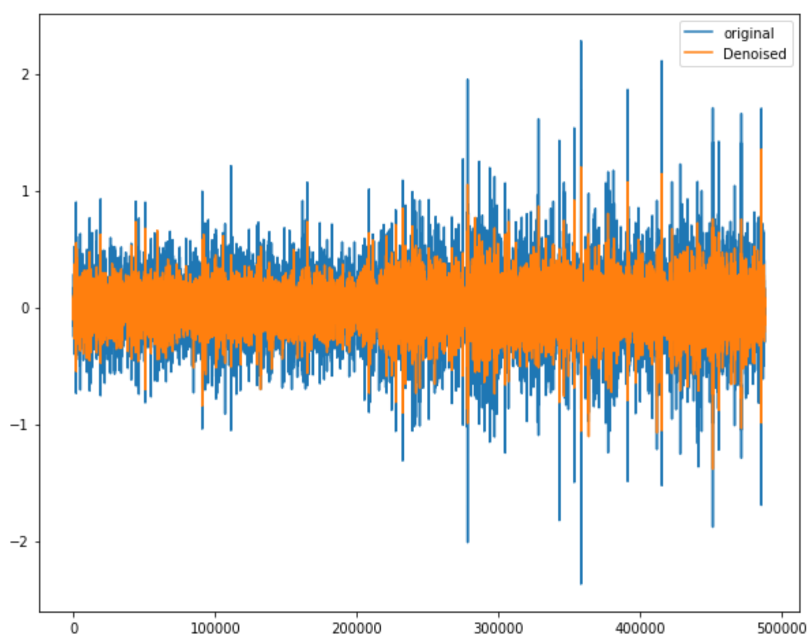




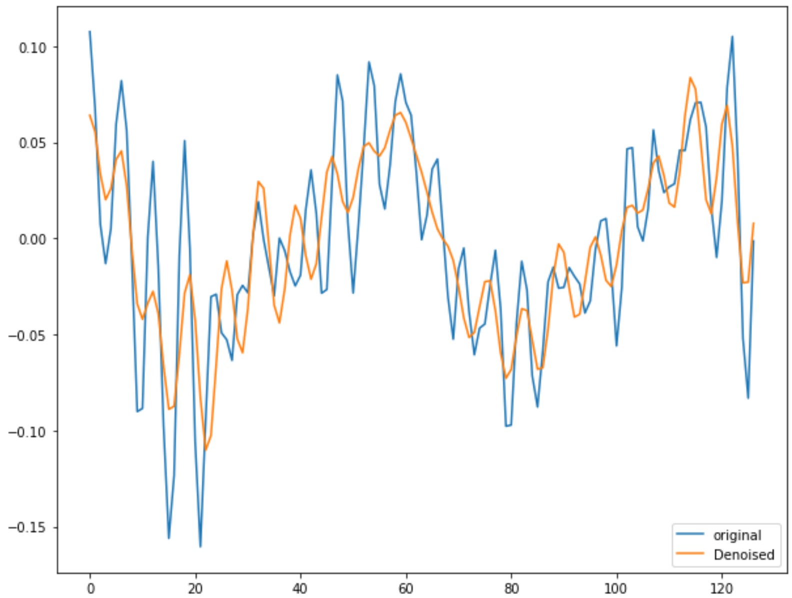


The result of denoising can be seen in the following figure: This figure represents a denoised signal for a series of **128 data points** while the next figure represents all **487680 data points.** Using the same basic features **(Mean, Standard deviation, Maximum, Minimum, Max to Min Ratio, First-order derivative, Second-order derivative)** we trained our Machine learning model on the new denoised signal.

3.4.1 FFT: Fast Fourier Transform Results



Since the dataset above has huge number of values(around 5 lac), the narrowed graph (128 values) for better visualization is shown below:



	MLA Name	MLA Parameters	MLA Train Accuracy	MLA Train Accuracy Mean	MLA Test Accuracy	MLA Test Accuracy Mean	MLA Test Accuracy Std
2	ExtraTreesClassifier	{'bootstrap': False, 'ccp_alpha': 0.0, 'class_...	[1.0, 1.0, 1.0, 1.0, 1.0]	1	[0.5282522996057819, 0.2979002624671916, 0.428...	0.457769	0.0885261
3	GradientBoostingClassifier	{'ccp_alpha': 0.0, 'criterion': 'friedman_mse'...	[0.9970482125286979, 0.9950787401574803, 0.995...	0.995079	[0.4980289093298292, 0.3293963254593176, 0.457...	0.454341	0.068532
20	XGBClassifier	{'objective': 'binary:logistic', 'use_label_en...	[1.0, 1.0, 1.0, 1.0, 1.0]	1	[0.4980289093298292, 0.33858267716535434, 0.42...	0.451987	0.0638282
4	RandomForestClassifier	{'bootstrap': True, 'ccp_alpha': 0.0, 'class_w...	[1.0, 1.0, 1.0, 1.0, 1.0]	1	[0.5006570302233903, 0.2874015748031496, 0.420...	0.448052	0.0883982
1	BaggingClassifier	{'base_estimator': None, 'bootstrap': True, 'b...	[0.9950803542144966, 0.994750656167979, 0.9940...	0.994751	[0.5111695137976346, 0.2992125984251969, 0.394...	0.435726	0.0800442
16	DecisionTreeClassifier	{'ccp_alpha': 0.0, 'class_weight': None, 'crit...	[1.0, 1.0, 1.0, 1.0, 1.0]	1	[0.4244415243101183, 0.2874015748031496, 0.382...	0.396336	0.0645472
19	QuadraticDiscriminantAnalysis	{'priors': None, 'reg_param': 0.0, 'store_cova...	[0.7395867497540177, 0.7477034120734908, 0.763...	0.735566	[0.392904073587385, 0.30708661417322836, 0.311...	0.380861	0.0636463
14	SVC	{'C': 1.0, 'break_ties': False, 'cache_size': ...	[0.7422105608396196, 0.770997375328084, 0.7650...	0.761484	[0.4323258869908016, 0.2335958005249344, 0.327...	0.370106	0.0792892
13	KNeighborsClassifier	{'algorithm': 'auto', 'leaf_size': 30, 'metric...	[0.8094457199081666, 0.8270997375328084, 0.805...	0.813648	[0.4126149802890933, 0.27165354330708663, 0.33...	0.363014	0.0626538
17	ExtraTreeClassifier	{'ccp_alpha': 0.0, 'class_weight': None, 'crit...	[1.0, 1.0, 1.0, 1.0, 1.0]	1	[0.41655716162943496, 0.18766404199475065, 0.3...	0.35933	0.0880142
6	LogisticRegressionCV	{'Cs': 10, 'class_weight': None, 'cv': None, '...	[0.6090521482453264, 0.6604330708661418, 0.675...	0.639309	[0.43626806833114323, 0.2887139107611549, 0.29...	0.357781	0.0596117
5	GaussianProcessClassifier	{'copy_X_train': True, 'kernel': None, 'max_it...	[0.9940964250573958, 0.9950787401574803, 0.996...	0.994948	[0.41392904073587383, 0.2979002624671916, 0.32...	0.357503	0.0488246
15	LinearSVC	{'C': 1.0, 'class_weight': None, 'dual': True,...	[0.5837979665464087, 0.635498687664042, 0.6465...	0.61175	[0.4231274638633377, 0.3005249343832021, 0.279...	0.350956	0.0553863
18	LinearDiscriminantAnalysis	{'covariance_estimator': None, 'n_components':...	[0.5808461790751066, 0.6010498687664042, 0.620...	0.589832	[0.41392904073587383, 0.32020997375328086, 0.2...	0.347277	0.0421703

The best performing classifier which in this case is the Extra Tree classifier gives an **accuracy of 45.7 %** with an **standard deviation of 8%**.

While Gradient boosting classifier only gives the accuracy of approximately 45 % which is a decrease of 2 % as compared to previous methods.

3.5 Advance Statistical Features (Time based)

In addition to the previously calculated features, we computed the following additional features on the dataset.

- Kurtosis
- Zero Crossing
- Skewness
- Absolute Energy
- Absolute Maximum
- Root Mean Square Energy
- Count Below Mean
- FFT_Coefficient
- Wavelength
- Number of Peaks
- Quantile (25,50,75)
- Mean_change_of_abs_change
- Max to Min

	MLA Name	MLA Parameters	MLA Train Accuracy	MLA Train Accuracy Mean	MLA Test Accuracy	MLA Test Accuracy Mean	MLA Test Accuracy Std
4	XGBClassifier	{'objective': 'binary:logistic', 'use_label_en...	[1.0, 1.0, 1.0, 1.0, 1.0]	1	[0.6110381077529566, 0.36220472440944884, 0.46...	0.506337	0.0856768
0	GradientBoostingClassifier	{'ccp_alpha': 0.0, 'criterion': 'friedman_mse'...	[1.0, 0.9983595800524935, 0.9993436166721366, ...	0.998819	[0.5965834428383706, 0.3648293963254593, 0.483...	0.503442	0.0782397
1	RandomForestClassifier	{'bootstrap': True, 'ccp_alpha': 0.0, 'class_w...	[1.0, 1.0, 1.0, 1.0, 1.0]	1	[0.5926412614980289, 0.3569553805774278, 0.422...	0.491121	0.0874737
2	SVC	{'C': 1.0, 'break_ties': False, 'cache_size': ...	[0.9307969826172515, 0.916994750656168, 0.9327...	0.927625	[0.5597897503285151, 0.33989501312335957, 0.41...	0.483764	0.0915117
3	DecisionTreeClassifier	{'ccp_alpha': 0.0, 'class_weight': None, 'crit...	[1.0, 1.0, 1.0, 1.0, 1.0]	1	[0.4664914586070959, 0.26246719160104987, 0.39...	0.412092	0.0812608

Advanced Statistical Features (Kurtosis, Skewness, Zero crossing, RMS Energy, etc) were then calculated in Time Domain and the **Mean test accuracy** thus obtained was **50.63%** with **standard deviation of 8.56%.**

NOTE: This test was performed with 20 features (7 Basic and 13 Advanced features).

CHAPTER- 4

CONTINUOUS WAVELET TRANSFORM

4.1 Continuous Wavelet Transform:

In previous steps, the team has performed classification based on both frequency and time-domain features. The limitations of the Fourier transform are that it can only tell the frequency components that are present in the signal, but not its evolution with time. Short-time Fourier transform attempts to perform this shortcoming but because of fixed window size, there is a dilemma of resolutions. To overcome this Continuous wavelet transform(CWT) is performed which outputs 3D matrices, where one dimension represents the time, the other represents the coefficients and scale. It is computed by the following formula:

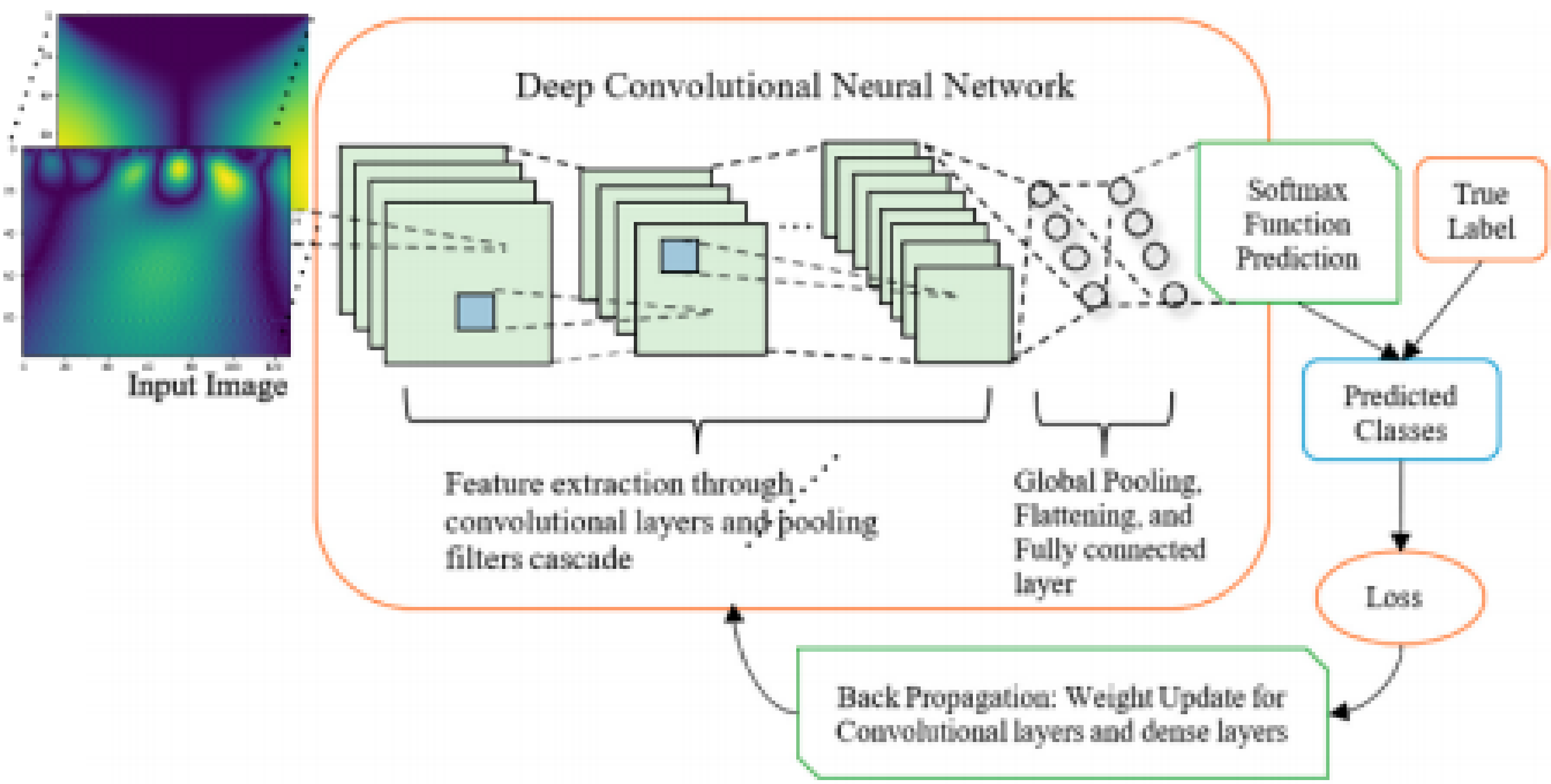
Continuous Wavelet Transform (CWT)

$$T(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t)\, \psi^{*} \frac{(t-b)}{a} dt$$

Discrete Wavelet Transform (DWT)

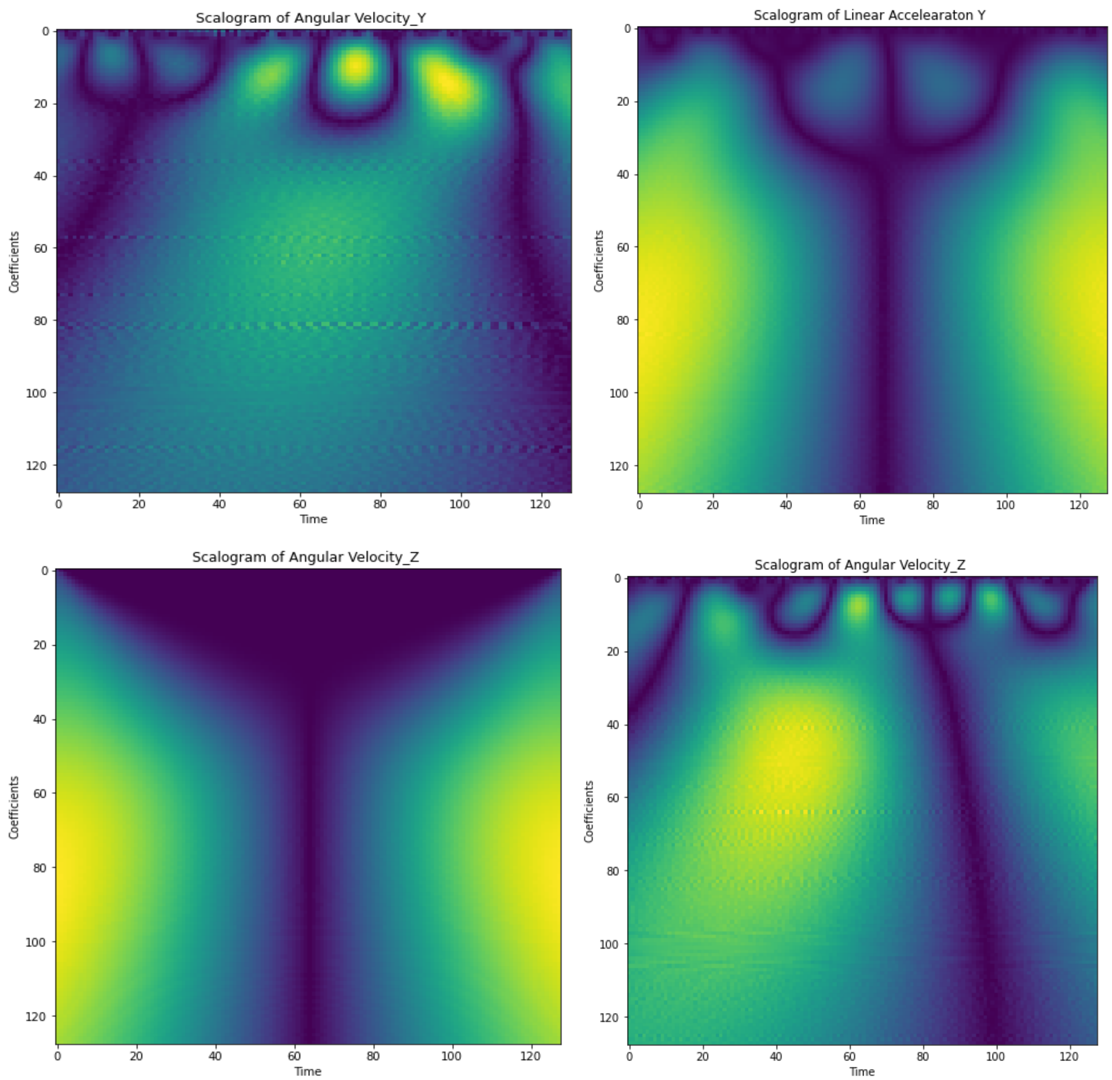
$$T_{m,n} = \int_{-\infty}^{\infty} x(t)\, \psi_{m,n}(t)\, dt$$

1. We take CWT of each 1D signal and all the coefficients are arranged to form a CWT Scalogram, which is represented in colormap image of format .jpg.
2. But unlike the ECG signal, in this problem statement, there are 10 different IMU sensor readings that in combination with each other represent the surface type. So all CWT of all the 1D signals representing one surface is performed.
3. Further, these 10 different images are stacked making it a tensor of dimension (w,h,10), where w, h represents width and height of an image. This serves as an input to the Deep convolutional neural network.

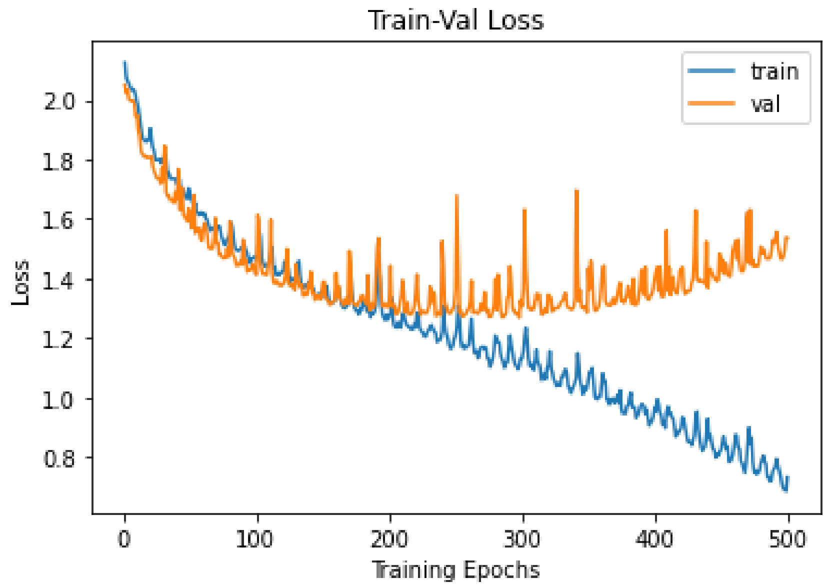


Deep Convolutional Neural Network
Input Image
Feature extraction through convolutional layers and pooling filters cascade
Global Pooling, Flattening, and Fully connected layer
Softmax Function Prediction
True Label
Predicted Classes
Loss
Back Propagation: Weight Update for Convolutional layers and dense layers

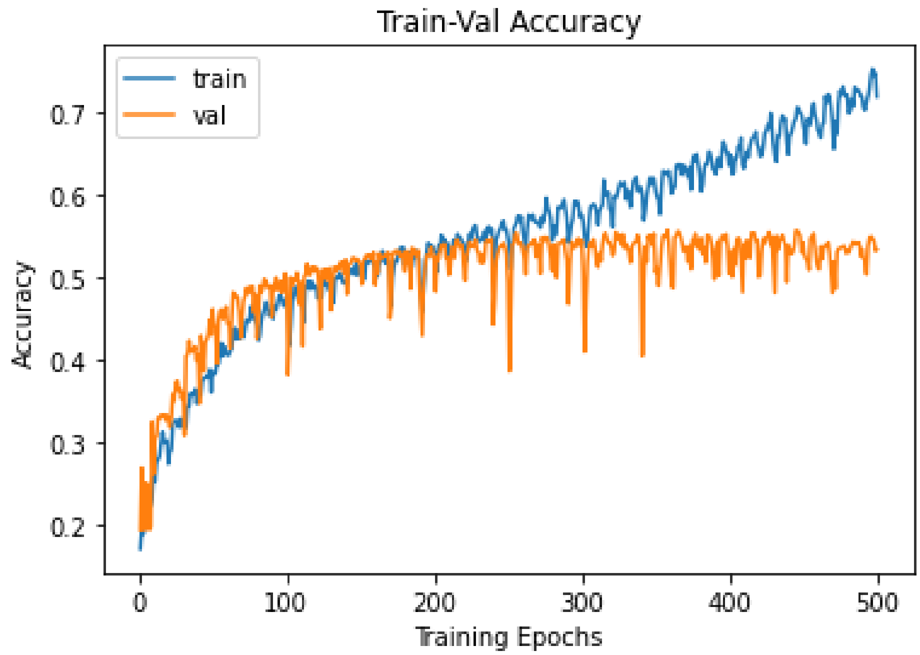
4.2: Plots of Scalogram produced:



4.3: Training and Testing Results:



# of Epochs: 500	
Train Loss(Min)	0.730759
Test Loss(Min)	1.366049



#of Epochs: 500	
Train Accuracy(Max)	73.14%
Test Accuracy(Max)	57.32%

CHAPTER- 5

LSTM AND 1-D CONVNET

5.1 Long Short term Memory(LSTM)

For the purpose we have used **many to one architecture**, we have 10 different sensor inputs and a series of 128 continuous data points that are fed into the LSTM network and it gives one output that is the label of the surface. The architecture of LSTM is as follows and its corresponding code:

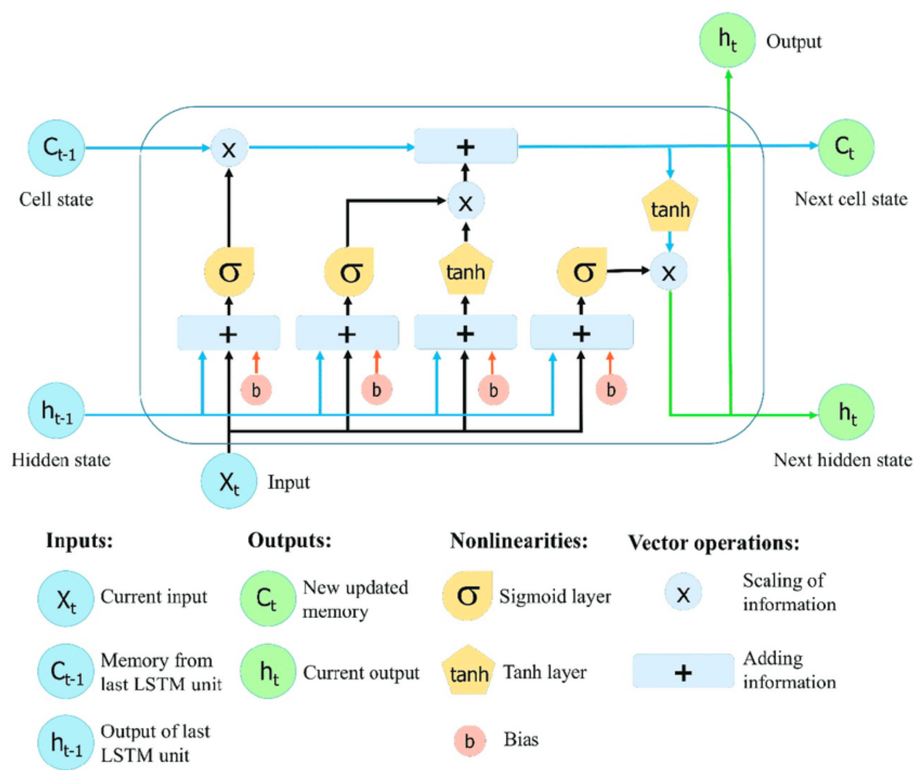


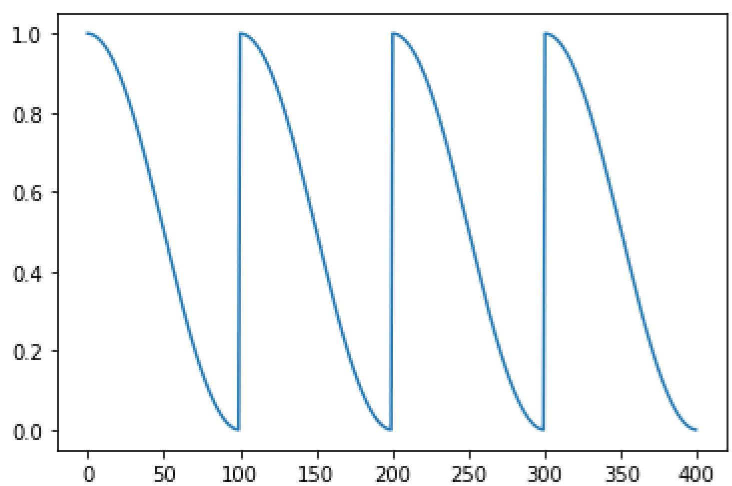
Image Source: Link

Code: There are 128 timesteps that represent one surface, hence use of 128 blocks of LSTM is required. Input will be sequential in nature. At first block data in all the 10 sensor channels is input for the first time step.

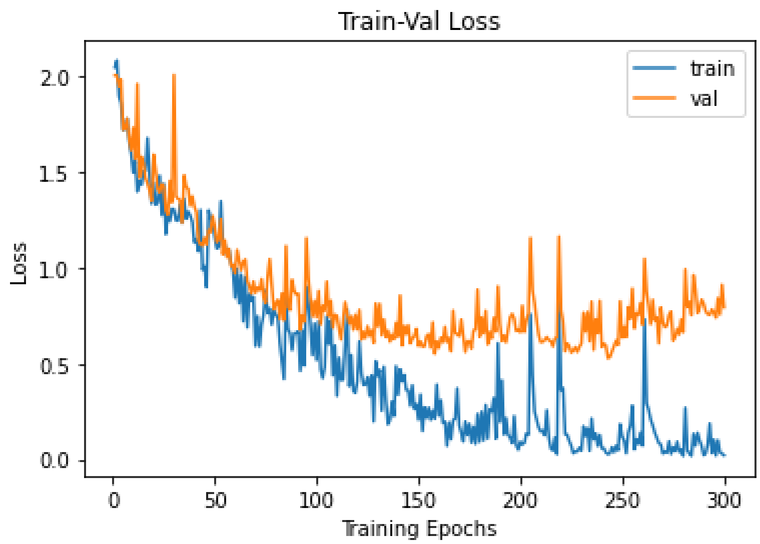
In [52]:
```python
class LSTMClassifier(nn.Module):
    """Very simple implementation of LSTM-based time-series classifier."""

    def __init__(self, input_dim, hidden_dim, layer_dim, output_dim):
        super().__init__()
        self.hidden_dim = hidden_dim
        self.layer_dim = layer_dim
        self.rnn = nn.LSTM(input_dim, hidden_dim, layer_dim, batch_first=True)
        self.fc = nn.Linear(hidden_dim, output_dim)
        self.batch_size = None
        self.hidden = None

    def forward(self, x):
        h0, c0 = self.init_hidden(x)
        out, (hn, cn) = self.rnn(x, (h0, c0))
        out = self.fc(out[:, -1, :])
        return out

    def init_hidden(self, x):
        h0 = torch.zeros(self.layer_dim, x.size(0), self.hidden_dim)
        c0 = torch.zeros(self.layer_dim, x.size(0), self.hidden_dim)
        return [t.cuda() for t in (h0, c0)]
```

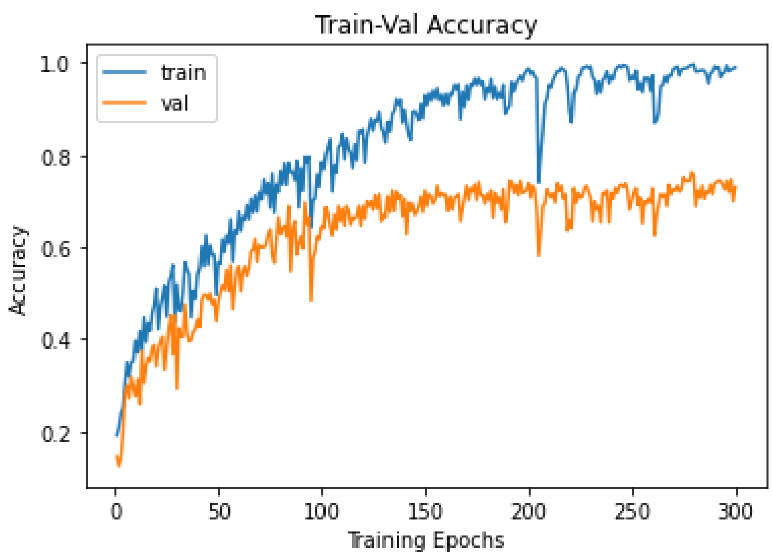
Other than this a cyclic learning rate scheduler is used. This helps in better and faster convergence of algorithms to the local minima.

```python
def cosine(t_max, eta_min=0):

    def scheduler(epoch, base_lr):
        t = epoch % t_max
        return eta_min + (base_lr - eta_min)*(1 + np.cos(np.pi*t/t_max))/2

    return scheduler
```



5.2: Training and Testing Results (LSTM):





Number of Epochs: 300	
Training Accuracy	98.251%
Testing Accuracy	76.251

5.3: One-Dimensional Convolution Neural Network(1-D ConvNet)

In this problem scenario we used the following architecture as referred to in the following image. We used both the time (Raw features) and Fourier transform of the signal. The input dimension is (128+65). Both of them are separately fed into different convnet which later is concatenated to produce one output that is surface label.

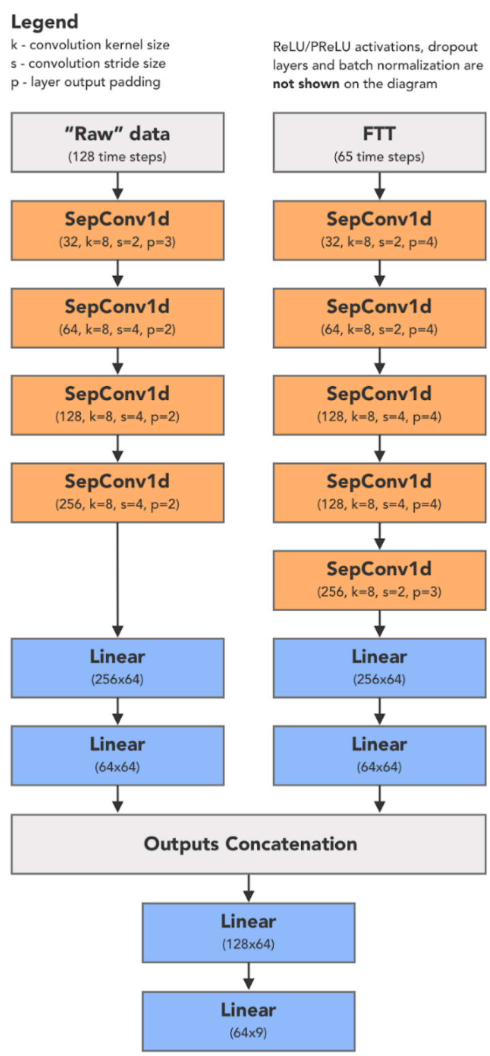
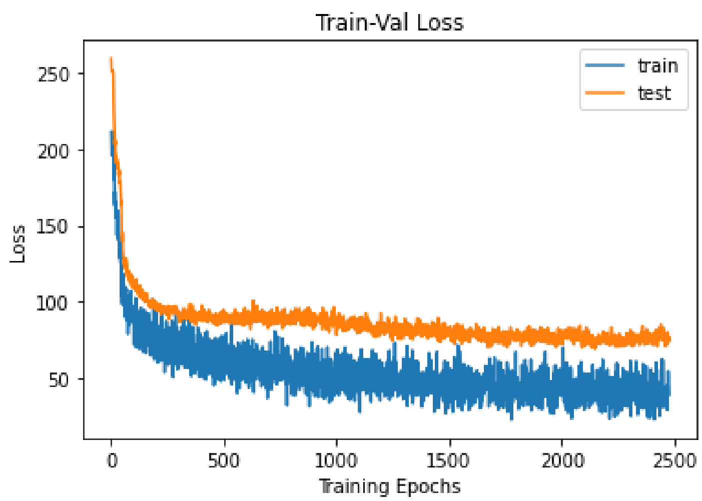
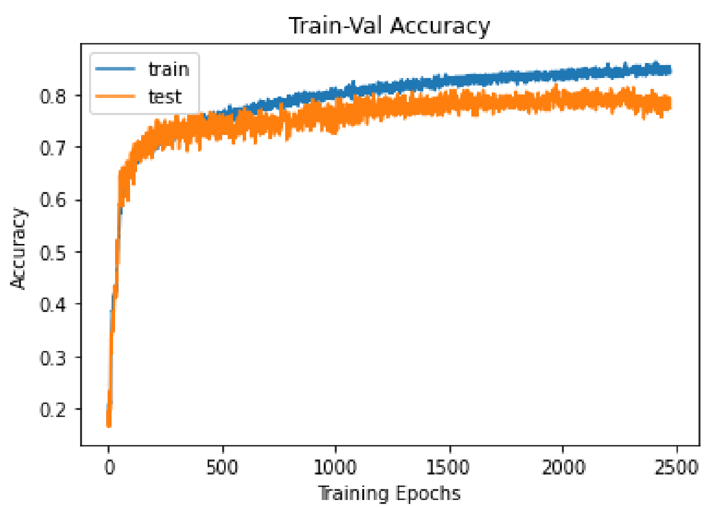


Image Source: Link

5.4: Training and Testing Results (1-D ConvNet):



# of Epochs: 2500	
Train Loss(Min)	97.32
Test Loss(Min)	43.12



#of Epochs: 2500	
Train Accuracy(Max)	88.32%
Test Accuracy(Max)	81.89%

CONCLUSION

- The data was in the form of time series, so in order to classify the series to corresponding surface type, **7 basic features (Mean, Standard deviation, Maximum, Minimum, Max to Min Ratio, First-order derivative, Second-order derivative)** were calculated. These features were calculated in the time domain.
- Model Training and Testing was performed and with these features **Gradient Boosting Classifier** performs best with **Mean test accuracy** on 5 fold set up to **47.638 %** and **standard deviation of 5.58%**.
- **De-noising** of the original data was then performed through Fast Fourier Transform(**FFT**) in order to get better accuracy. But the **Mean test accuracy** dropped down by around 2% to **45.77%** and **standard deviation of 8.85%**.
- **Advanced Statistical Features (Kurtosis, Skewness, Zero crossing, RMS Energy, etc)** were then calculated in Time Domain and the **Mean test accuracy** thus obtained was **50.63%** with **standard deviation of 8.56%.**
- **Continuous Wavelet transform,** a popular method that is used in classification of ECG signals was used in the case to classify the IMU sensor reading. Through this method the **accuracy** on the test data set is **57.32%.**
- **Long Short Term Memory (LSTM),** is a method used to classify signals and continuous sequences. In this dataset it performs way better than other methods discussed before with an **accuracy** of **76.251 %**.
- **One-Dimensional Convolution Neural Network(1-D ConvNet),** this is also a deep learning method that outperforms previous techniques. As an input both frequencies present in the signal and time based features were given in the input. Total classification **accuracy** on the test data set is 81.89 %.

FUTURE WORK

- The sensor used in the experiment recorded results in the **lab environment**. To fully evaluate the performance of the algorithm, one needs to train and validate the model on the outside lab environment. The whole motive of the work was to find a way through which the number of sensors can be reduced and maximum information acquisition can be done.
- The signal characteristics are affected by different voltage-current levels of the battery. One of the future aspects is to incorporate a changing system. An **online learning system** that can work with it.
- In the frequency domain other features can be also investigated and calculated.
- Wavelet transform based denoising can also be tried.
- As the Machine Learning and Deep Learning field is evolving new methodologies and techniques can also be tested.

REFERENCES

1. Francesco Lomio and Erjon Skenderi and Damoon Mohamadi and Jussi Collin and Reza Ghabcheloo and Heikki Huttunen, “Surface Type Classification for Autonomous Robot Indoor Navigation”, 2019, 1905.00252

2. D. Anthony, E. Basha, J. Ostdiek, J. Ore and C. Detweiler, "Surface classification for sensor deployment from UAV landings," 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, 2015, pp. 3464-3470, doi: 10.1109/ICRA.2015.7139678.

3. O'Reilly MA, Whelan DF, Ward TE, Delahunt E, Caulfield B. Classification of lunge biomechanics with multiple and individual inertial measurement units. Sports Biomech. 2017 Sep;16(3):342-360. doi: 10.1080/14763141.2017.1314544. Epub 2017 May 19. PMID: 28523981.

4. Kim, Minwoo; Cho, Jaechan; Lee, Seongjoo; Jung, Yunho. 2019. "IMU Sensor-Based Hand Gesture Recognition for Human-Machine Interfaces" Sensors 19, no. 18: 3827.

5. Byeon YH, Pan SB, Kwak KC. Intelligent Deep Models Based on Scalograms of Electrocardiogram Signals for Biometrics. Sensors (Basel). 2019;19(4):935. Published 2019 Feb 22. doi:10.3390/s19040935

6. MATLAB Implementation: Classify Time Series Using Wavelet Analysis and Deep Learning - MATLAB & Simulink Example

7. Türk Ö, Özerdem MS. Epilepsy Detection by Using Scalogram Based Convolutional Neural Network from EEG Signals. Brain Sci. 2019;9(5):115. Published 2019 May 17. doi:10.3390/brainsci9050115

8. Feature-Free Activity Classification of Inertial Sensor Data With Machine Vision Techniques: Method, Development, and Evaluation by Gunther Eysenbach, Lingfei Mo and Gregory Norman. Published online 2017 Aug 4. doi: 10.2196/mhealth.7521

9. Sijie Zhuo, Lucas Sherlock, Gillian Dobbie, Yun Sing Koh, Giovanni Russello, and Danielle Lottridge- “Real-time Smartphone Activity Classification Using Inertial Sensors”. Published online 2020 Jan 24. doi: 10.3390/s20030655

10. Lauro Ojeda, Johann Borenstein, Gary Witus, and Robert Karlsen.Terrain characterization and classification with a mobile robot. Journal of Field Robotics, 23(2):103–122, 2006.

11. Felipe G Oliveira, Elerson RS Santos, Armando Alves Neto, Mario FM Campos, and Douglas G Macharet. Speed-invariant terrain roughness classification and control based on inertial sensors. In 2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR), pages 1–6. IEEE, 2017.

12. Csaba Kertesz. Rigidity-based surface recognition for a domestic legged robot. IEEE Robotics and Automation Letters, 1(1):309–315, 2016