GEAR FAULT DETECTION
USING MACHINE LEARNING

A PROJECT REPORT ON NAAN MUDHALVAN

Submitted by

Kamesh A - 2020509023

Prakash S - 2020509035

Pranav Dharshan B - 2020509036

Vishal Fernando M - 2020509050

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

MECHANICAL ENGINEERING

MADRAS INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY: CHENNAI 600 044

NOVEMBER 2022

ANNA UNIVERSITY: CHENNAI 600 044

BONAFIDE CERTIFICATE

Certified that this project report "**GEAR FAULT DETECTION USING MACHINE LEARNING**" is the bonafide work of "**Kamesh A (2020509023), Prakash S (2020509035), and Pranav Dharshan B (2020509036), Vishal Fernando M (2020509050)"** who carried out the Naan Mudhalan project work under my supervision.

SIGNATURE

Dr. A. Siddharthan
Professor and Head
Department of Production Technology
MIT Campus, Anna University
Chennai 600 044

SIGNATURE

Dr. P. Ganesh
Machine Learning In-Charge
Assistant Professor (Sl. Gr.)
Department of Production Technology
MIT Campus, Anna University
Chennai 600 044

SIGNATURE

Mr. S. Mohamed Shazuli
Machine Learning In-Charge
Teaching Fellow
Department of Production Technology
MIT Campus, Anna University
Chennai 600 044

ACKNOWLEDGEMENT

I feel it as a great pleasure to thank our Professor and Head of Department Prof.Dr.A.Siddharthan, Department of Production Technology, M.I.T. Campus, Anna University, Chennai, for his constant encouragement and the involvement that he has shown us to undergo Naan Mudhalvan Skill Upliftment scheme.

I would like to thank Prof.Dr.P.Ganesh, Assistant Professor (Sl. Gr.), Department of Production Technology, M.I.T. Campus, Anna University, Chennai, for his immense guidance in preparing this Machine Learning project report.

I would also like to thank Mr. S. Mohamed Shazuli, Teaching Fellow, Department of Production Technology, M.I.T. Campus, Anna University, Chennai, for his immense guidance in preparing this Machine Learning project report.

ABSTRACT

Gearbox is one of the most important parts of the rotating machinery, so health monitoring of the gearbox is essential. The fault diagnostic system's correct location of gear tooth failure is a crucial component. In order to identify and pinpoint gear tooth failure, this research suggests a detection approach based on specially developed convolutional neural networks. The detection approach aims to compare the characteristic gap between the normal gear and the defective gear in the same period extracted by the convolutional neural network and assign weights to the vibration signal of the defective gear to obtain the weight sequence of the defective vibration signal, in order to obtain the faulty tooth weight. Finally, comparing the weight of each gear tooth will allow you to assess the gear's overall health. Through simulation vibration signal and experiment vibration signal, the suggested detection approach is evaluated. The outcome demonstrates that the suggested method can accurately identify gear failure and single gear tooth failure.

TABLE OF CONTENTS

CHAPTER NO NO	TITLE	PAGE
	ABSTRACT	4
1.	**INTRODUCTION**	**6**
2.	**MOTIVE**	**7**
	2.1 PROBLEM DEFINITION	7
	2.2 OBJECTIVES	7
3.	**DATA COLLECTION**	8
4.	**METHODOLOGY**	17
	3.1 CODE	18
5.	**CONCLUSION**	29

1.INTRODUCTION

Gearboxes are used to transport power between shafts in mechanical transmission systems and are anticipated to operate continuously throughout a manufacturing system. Any gearbox problems might result in unneeded downtime, costly repairs, or even fatalities. Therefore, it's crucial to find and fix problems as soon as possible. The fault diagnosis has drawn a lot of attention for the safe operation of the gearboxes as a useful component for condition-based maintenance.

The study on the application of convolutional neural networks in the detection and classification of gearbox faults is presented in this paper. An example of a feed-forward artificial neural network is the convolutional neural network (CNN). The way in which its individual neurons are tiled causes them to react to overlapping areas of the visual field. For image and video recognition, CNN and its variants are frequently used models. It serves as a classifier for the diagnosis of gearbox defects in this work.

Extraction of the sensitive characteristics and classification of the condition patterns make up the two primary components of the most effective vibration-based fault diagnosis techniques. The most frequently utilised features in vibration-based defect diagnostics have been produced using temporal, spectral, wavelet, and other representations of the signals. It is possible to think of various representations as representing various vibration signal observations. In this study, Dataset was collected using four vibration sensors positioned in four different directions with a range of loads from 0% to 90%. There are two alternative scenarios:

1) Healthy condition and
2) Broken Tooth condition

There are a total of 20 files, 10 for a working gearbox and 10 for a damaged one. Each file is assigned a load that ranges from 0% to 90% in 10% increments. The pre-processed signal's characteristics are converted into vectors that serve as the CNN's input parameters.

2.MOTIVE

2.1 PROBLEM DEFINITIION:

Due to the difficulties of installing vibration sensors near to the fault and the existence of a sizable background noise caused by several mechanical excitations inside the system, gear defects are difficult to detect. A vibration signature known as the gear mesh frequency is produced by healthy gears. This type of signal typically has a low amplitude, is wideband, and is non-periodic and non-stationary. So, a CNN model is made to help with the classification of the healthy gear and broken tooth gear.

Figure 2.1: Broken tooth gear

2.2 OBJECTIVES:

- To learn the use of Machine Learning in the field of Mechanical Engineering.
- To examine the vibration that the running gears.
- To use CNN (Convolutional Neural Network) for vibration data processing.

3.DATA COLLECTION

The Dataset consist of two folders namely: Broken tooth and Healthy

3.1 BROKEN TOOTH:

It consists of dataset collected using four vibration sensors positioned in four different directions with a range of loads from 0% to 90% of a broken tooth gear. The .csv file name are:

- b30hz00.csv
- b30hz10.csv
- b30hz20.csv
- b30hz30.csv
- b30hz40.csv
- b30hz50.csv
- b30hz60.csv
- b30hz70.csv
- b30hz80.csv
- b30hz90.csv

The following are the first three rows of the .csv file

b30hz00.csv:

a1	**a2**	**a3**	**a4**
2.35039	1.45487	-1.66708	-2.05561
2.45297	1.4001	-2.8251	0.984487
-0.24128	-0.26739	0.79354	0.605862

b30hz10.csv:

a1	**a2**	**a3**	**a4**
1.26041	-1.35726	-1.91633	1.8457
-0.1262	-2.27283	0.536155	1.53092
-0.90316	-1.04204	-0.74134	1.65011

b30hz20.csv:

a1	**a2**	**a3**	**a4**
-2.72429	1.00036	0.258655	1.07957
-0.1333	0.467942	1.41364	0.569072
0.393993	-1.28957	0.34612	0.087828

b30hz30.csv:

a1	**a2**	**a3**	**a4**
-0.01614	3.27684	-2.9585	-10.4088
4.90723	-2.15351	-3.41721	-5.88936
-2.02462	-1.54883	2.74703	1.03744

b30hz40.csv:

a1	**a2**	**a3**	**a4**
-3.17043	1.24134	-3.55791	-0.34255
-7.43122	1.18426	-0.5525	-3.66744
-4.92843	2.64079	0.459975	-0.48999

b30hz50.csv:

a1	**a2**	**a3**	**a4**
-3.93468	6.55216	-1.23798	20.3103
2.40285	9.99438	-3.24265	8.3132
6.24273	-3.17577	-0.68697	-4.19382

b30hz60.csv:

a1	a2	a3	a4
6.84186	4.17358	-3.73527	1.02471
0.470408	-11.2805	0.664213	-2.72565
-1.23806	-15.445	8.016	-12.1506
-0.60466	1.34391	1.13617	-0.73092

b30hz70.csv:

a1	a2	a3	a4
-0.15749	1.52661	1.53455	-1.01746
1.16074	0.023445	1.70493	-3.50295
0.033562	0.045544	-0.43122	-5.58287

b30hz80.csv:

a1	a2	a3	a4
8.31242	-4.66927	-0.27957	-2.16663
3.77807	-7.09694	-6.34683	-0.80073
-4.83836	0.570051	-9.59574	8.78741

b30hz90.csv:

a1	a2	a3	a4
-5.51068	4.80685	-1.42651	0.508883
-9.42926	8.52212	1.43432	7.74571
-7.40029	5.08099	0.232017	3.16229

3.1 HEALTHY:

- h30hz00.csv
- h30hz10.csv
- h30hz20.csv
- h30hz30.csv
- h30hz40.csv
- h30hz50.csv
- h30hz60.csv
- h30hz70.csv
- h30hz80.csv
- h30hz90.csv

The following pages consists of the first 3 rows of each .csv file of healthy tooth gear folder.

h30hz00.csv:

a1	a2	a3	a4
4.63671	0.516978	-3.20594	1.82241
1.9928	4.18466	-2.74061	2.80436
-3.76411	0.997335	-1.30309	1.83668

h30hz10.csv:

a1	a2	a3	a4
-0.16938	-1.28208	3.30282	-1.55699
3.94582	-0.22091	-0.00349	-0.17465
0.888728	0.694251	-0.03549	-0.47026

h30hz20.csv:

a1	a2	a3	a4
-7.82861	2.8795	1.21863	-1.53793
4.01181	2.77293	-2.37448	-1.15037
-0.86447	3.89715	-3.30984	-0.43506

h30hz30.csv:

a1	**a2**	**a3**	**a4**
9.97455	3.27094	0.433691	-1.78217
-1.34088	-3.14412	-3.6191	4.44778
-9.56382	4.70157	-3.23392	7.39967

h30hz40.csv:

a1	**a2**	**a3**	**a4**
2.14539	0.077986	2.94622	-0.28511
6.35557	4.54152	4.68974	-1.81114
0.842493	0.747476	6.71455	-2.65847

h30hz50.csv:

a1	**a2**	**a3**	**a4**
2.14416	-1.95821	-0.19053	-4.58475
-9.92015	-7.47519	1.79468	-7.47251
-1.33059	0.751472	-3.5574	0.328149

h30hz60.csv:

a1	a2	a3	a4
-18.9713	-0.24965	-0.18772	1.0615
-9.93899	8.53113	7.55448	4.69872
21.4341	-4.73962	-0.27836	5.95105

h30hz70.csv:

a1	a2	a3	a4
-3.62418	2.136	3.58575	-6.32608
-0.02433	-6.47083	0.813486	-4.35671
4.72101	5.37865	-1.55939	1.11916

h30hz80.csv:

a1	a2	a3	a4
5.18442	3.81187	6.13919	-3.45186
24.1875	-4.5581	0.057864	3.03233
-5.52775	1.13757	1.70977	9.44042

h30hz90.csv:

a1	**a2**	**a3**	**a4**
-0.78825	-3.72269	1.06864	-0.57133
5.43042	-0.0607	-4.77016	0.599285
-15.4611	1.90858	2.54934	-0.47952

4.METHODOLOGY

The structure of the visual system, and more specifically the models of it put out by, served as inspiration for the convolutional neural network. The initial computer models are based on hierarchically ordered picture modifications in Fukushima's noncognition and on local connection between neurons. Convolutional networks' processing approach is congruent with current knowledge of the physiology of the visual system. Convolutional neural network-based pattern recognition algorithms continue to rank among the highest performing systems today. This has been demonstrably shown for handwritten character recognition, which has long been used as a benchmark for machine learning.

The core of CNN is the use of many filters to extract spatial information from data that are concealed. The convolution layers of the CNN are enhanced during training to extract highly discriminative features, and the final layers mimic a multilayer perceptron to carry out the classification tasks.

4.1 CODE:

In [1]:

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

In [2]:

```
def MakeDataset(Directory,lab):
    df = pd.DataFrame(columns=['a1','a2','a3','a4'])

    for root, dirs, files in os.walk(Directory):
        for i in range (len(files)):

            path = os.path.join(root,files[i])

            df_temp = pd.read_csv(path)
            load_col = [int(files[i][5:-4])/100 for j in range(len(df_temp))]
            label_col = [lab for j in range(len(df_temp))]
            df_temp['load']= load_col
            df_temp['fault']= label_col

            df = pd.concat([df,df_temp],axis = 0)
            print(path)


    return df
```

In [3]:

```
Directory = 'C:\Gear data\BrokenTooth'
df_F = MakeDataset(Directory, lab = 'F')
```

```
C:\Gear data\BrokenTooth\b30hz00.csv
C:\Gear data\BrokenTooth\b30hz10.csv
C:\Gear data\BrokenTooth\b30hz20.csv
C:\Gear data\BrokenTooth\b30hz30.csv
C:\Gear data\BrokenTooth\b30hz40.csv
C:\Gear data\BrokenTooth\b30hz50.csv
C:\Gear data\BrokenTooth\b30hz60.csv
C:\Gear data\BrokenTooth\b30hz70.csv
C:\Gear data\BrokenTooth\b30hz80.csv
C:\Gear data\BrokenTooth\b30hz90.csv
```

In [4]:

```
Directory = 'C:\Gear data\Healthy'
df_H = MakeDataset(Directory, lab = 'H')
```

```
C:\Gear data\Healthy\h30hz00.csv
C:\Gear data\Healthy\h30hz10.csv
C:\Gear data\Healthy\h30hz20.csv
C:\Gear data\Healthy\h30hz30.csv
C:\Gear data\Healthy\h30hz40.csv
C:\Gear data\Healthy\h30hz50.csv
C:\Gear data\Healthy\h30hz60.csv
C:\Gear data\Healthy\h30hz70.csv
C:\Gear data\Healthy\h30hz80.csv
C:\Gear data\Healthy\h30hz90.csv
```

In [5]:

```
df = pd.read_csv('C:\Gear data\Gear_Fault_data.csv')

from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
df.iloc[:,:-2]=scaler.fit_transform(df.iloc[:,:-2])
```

In [6]:

```
df
```

Out[6]:

	a1	a2	a3	a4	load	fault
0	0.381468	0.329958	-0.421759	-0.460352	0.0	F
1	0.398126	0.317534	-0.713949	0.220273	0.0	F
2	-0.039401	-0.060712	0.199101	0.135505	0.0	F
3	0.183330	-0.202151	0.174735	0.137119	0.0	F
4	-0.210701	0.222349	-0.286385	-0.077817	0.0	F
...	...	...	...	...	...	...
2021114	0.109795	-0.733740	-0.436623	-0.703804	0.9	H
2021115	-1.717584	1.752341	-0.552190	0.575163	0.9	H
2021116	-0.655194	0.584480	0.369389	0.610819	0.9	H
2021117	0.303240	-1.154518	1.346836	-0.305877	0.9	H
2021118	1.230958	1.407677	-1.545650	2.585326	0.9	H

2021119 rows × 6 columns

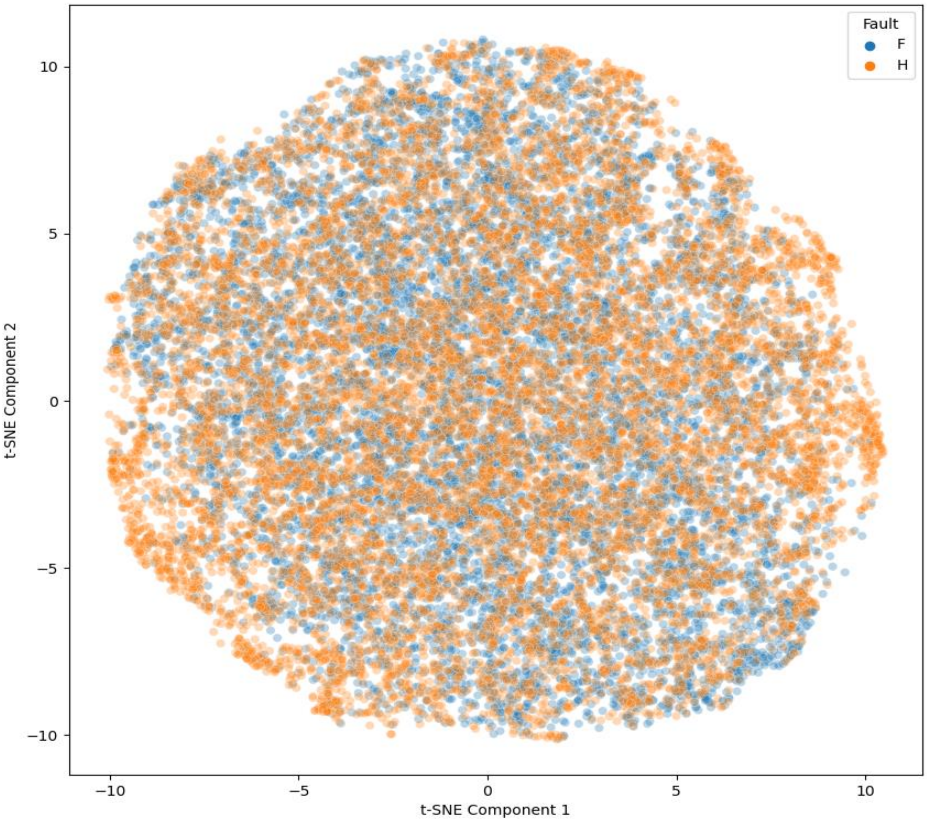
In [7]:

```
X = df.iloc[::100,:-1]
Y = df.iloc[::100,-1]
```

In [8]:

```
from sklearn.manifold import TSNE

X_t_sne = TSNE(n_components=2, learning_rate='auto', verbose=1, perplexity=40, n_iter=300).
tSNEdf = pd.DataFrame(data = X_t_sne, columns = ['t-SNE Component 1','t-SNE Component 2'])
tSNEdf['Fault'] = np.array(Y)

fig,ax = plt.subplots(figsize = (10,10))

sns.scatterplot(x=tSNEdf['t-SNE Component 1'], y=tSNEdf['t-SNE Component 2'], hue='Fault',
                data = tSNEdf,
                legend="full",
                alpha=0.3)
plt.show()
```

```
C:\Users\prakash\anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:780:
FutureWarning: The default initialization in TSNE will change from 'random'
to 'pca' in 1.2.
  warnings.warn(


[t-SNE] Computing 121 nearest neighbors...
[t-SNE] Indexed 20212 samples in 0.016s...
[t-SNE] Computed neighbors for 20212 samples in 1.288s...
[t-SNE] Computed conditional probabilities for sample 1000 / 20212
[t-SNE] Computed conditional probabilities for sample 2000 / 20212
[t-SNE] Computed conditional probabilities for sample 3000 / 20212
[t-SNE] Computed conditional probabilities for sample 4000 / 20212
[t-SNE] Computed conditional probabilities for sample 5000 / 20212
[t-SNE] Computed conditional probabilities for sample 6000 / 20212
[t-SNE] Computed conditional probabilities for sample 7000 / 20212
[t-SNE] Computed conditional probabilities for sample 8000 / 20212
[t-SNE] Computed conditional probabilities for sample 9000 / 20212
[t-SNE] Computed conditional probabilities for sample 10000 / 20212
[t-SNE] Computed conditional probabilities for sample 11000 / 20212
[t-SNE] Computed conditional probabilities for sample 12000 / 20212
[t-SNE] Computed conditional probabilities for sample 13000 / 20212
[t-SNE] Computed conditional probabilities for sample 14000 / 20212
[t-SNE] Computed conditional probabilities for sample 15000 / 20212
[t-SNE] Computed conditional probabilities for sample 16000 / 20212
[t-SNE] Computed conditional probabilities for sample 17000 / 20212
[t-SNE] Computed conditional probabilities for sample 18000 / 20212
[t-SNE] Computed conditional probabilities for sample 19000 / 20212
[t-SNE] Computed conditional probabilities for sample 20000 / 20212
[t-SNE] Computed conditional probabilities for sample 20212 / 20212
[t-SNE] Mean sigma: 0.255420
[t-SNE] KL divergence after 250 iterations with early exaggeration: 95.73569
5
[t-SNE] KL divergence after 300 iterations: 3.713595
```



In [9]:

```
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.utils import to_categorical
```

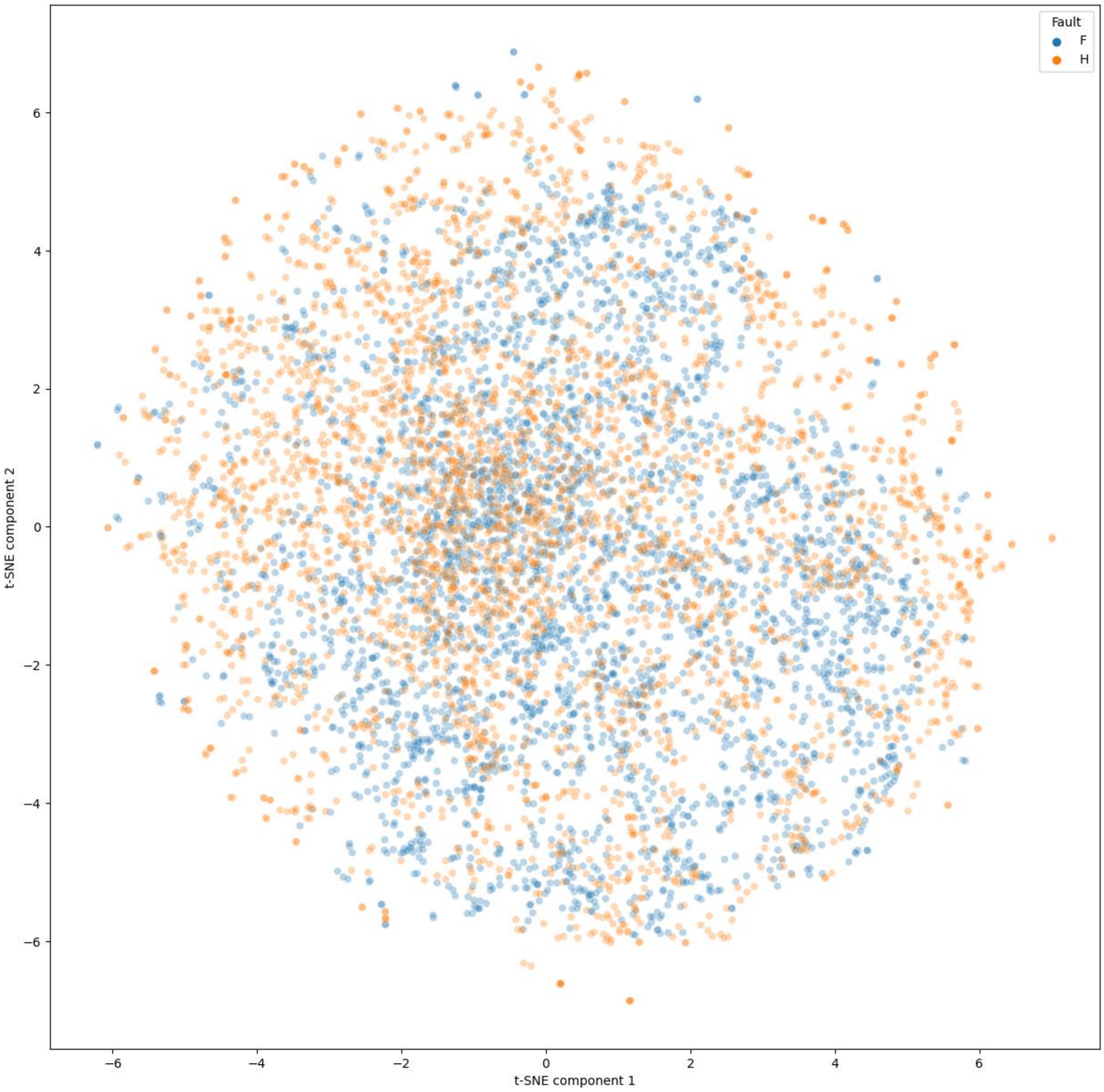
In [10]:

```
win_len= 100
stride= 200

X=[]
Y=[]


for k in ['F','H']:

    df_temp_1 = df[df['fault']==k]

    for j in (np.arange(0,1,0.1)):
        df_temp_2=df_temp_1[df_temp_1['load']==j]

        for i in np.arange(0,len(df_temp_2)-(win_len),stride):
            X.append(df_temp_2.iloc[i:i+win_len,:-1])
            Y.append(df_temp_2.iloc[i+win_len,-1])

X=np.array(X)
X=X.reshape((X.shape[0],X.shape[1],X.shape[2],1))



Y=np.array(Y)
encoder= LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)
OHE_Y = to_categorical(encoded_Y)
```

In [11]:

```
X_pre_cnn = X.reshape(X.shape[0],X.shape[1]*X.shape[2])

from sklearn.manifold import TSNE

X_t_sne = TSNE(n_components=2, learning_rate='auto',verbose=1, perplexity=40, n_iter=300).f

tSNEdf = pd.DataFrame(data = X_t_sne, columns = ['t-SNE component 1', 't-SNE component 2'])

tSNEdf['Fault']=Y



fig, ax = plt.subplots(figsize=(15,15))
sns.scatterplot(x=tSNEdf['t-SNE component 1'],y=tSNEdf['t-SNE component 2'],hue='Fault',
    data=tSNEdf,
    legend="full",
    alpha=0.3)
plt.show()
```

```
C:\Users\prakash\anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:780:
FutureWarning: The default initialization in TSNE will change from 'random'
to 'pca' in 1.2.
  warnings.warn(

[t-SNE] Computing 121 nearest neighbors...
[t-SNE] Indexed 7138 samples in 0.007s...
[t-SNE] Computed neighbors for 7138 samples in 1.899s...
[t-SNE] Computed conditional probabilities for sample 1000 / 7138
[t-SNE] Computed conditional probabilities for sample 2000 / 7138
[t-SNE] Computed conditional probabilities for sample 3000 / 7138
[t-SNE] Computed conditional probabilities for sample 4000 / 7138
[t-SNE] Computed conditional probabilities for sample 5000 / 7138
[t-SNE] Computed conditional probabilities for sample 6000 / 7138
[t-SNE] Computed conditional probabilities for sample 7000 / 7138
[t-SNE] Computed conditional probabilities for sample 7138 / 7138
[t-SNE] Mean sigma: 4.793280
[t-SNE] KL divergence after 250 iterations with early exaggeration: 90.35430
9
[t-SNE] KL divergence after 300 iterations: 3.770233
```



[12]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,OHE_Y,test_size=0.3,shuffle=True)
```

In [13]:

```
from tensorflow.keras.models import Sequential,Model
from tensorflow.keras.layers import Input,Dense, Dropout, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D




no_classes = 2

cnn_model = Sequential()
cnn_model.add(Conv2D(32, kernel_size=(20, 3),activation='relu',input_shape=(X.shape[1],X.sh

cnn_model.add(MaxPooling2D((20, 2),strides=(5, 5),padding='same'))

cnn_model.add(Conv2D(64, (10, 3), activation='relu',padding='same'))

cnn_model.add(MaxPooling2D(pool_size=(10, 2),strides=(3, 3),padding='same'))


cnn_model.add(Flatten())

cnn_model.add(Dense(128, activation='relu'))


cnn_model.add(Dense(no_classes, activation='softmax'))

cnn_model.summary()

cnn_model.compile(loss='categorical_crossentropy', optimizer='adam',metrics=['accuracy'])
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 100, 5, 32)        1952

 max_pooling2d (MaxPooling2D  (None, 20, 1, 32)        0
 )

 conv2d_1 (Conv2D)           (None, 20, 1, 64)         61504

 max_pooling2d_1 (MaxPooling  (None, 7, 1, 64)         0
 2D)

 flatten (Flatten)           (None, 448)               0

 dense (Dense)               (None, 128)               57472

 dense_1 (Dense)             (None, 2)                 258

=================================================================
Total params: 121,186
Trainable params: 121,186
Non-trainable params: 0
_________________________________________________________________
```

In [14]:

```
batch_size = 128
epochs = 5
history = cnn_model.fit(X_train, y_train, batch_size=batch_size,epochs=epochs,verbose=1,val
```

```
Epoch 1/5
40/40 [==============================] - 10s 69ms/step - loss: 0.2664 - accu
racy: 0.8871 - val_loss: 0.0019 - val_accuracy: 1.0000
Epoch 2/5
40/40 [==============================] - 3s 65ms/step - loss: 7.7550e-04 - a
ccuracy: 1.0000 - val_loss: 3.7795e-04 - val_accuracy: 1.0000
Epoch 3/5
40/40 [==============================] - 1s 37ms/step - loss: 3.0475e-04 - a
ccuracy: 1.0000 - val_loss: 3.1443e-04 - val_accuracy: 1.0000
Epoch 4/5
40/40 [==============================] - 1s 37ms/step - loss: 2.2334e-04 - a
ccuracy: 1.0000 - val_loss: 2.2731e-04 - val_accuracy: 1.0000
Epoch 5/5
40/40 [==============================] - 2s 39ms/step - loss: 1.6598e-04 - a
ccuracy: 1.0000 - val_loss: 1.8685e-04 - val_accuracy: 1.0000
```

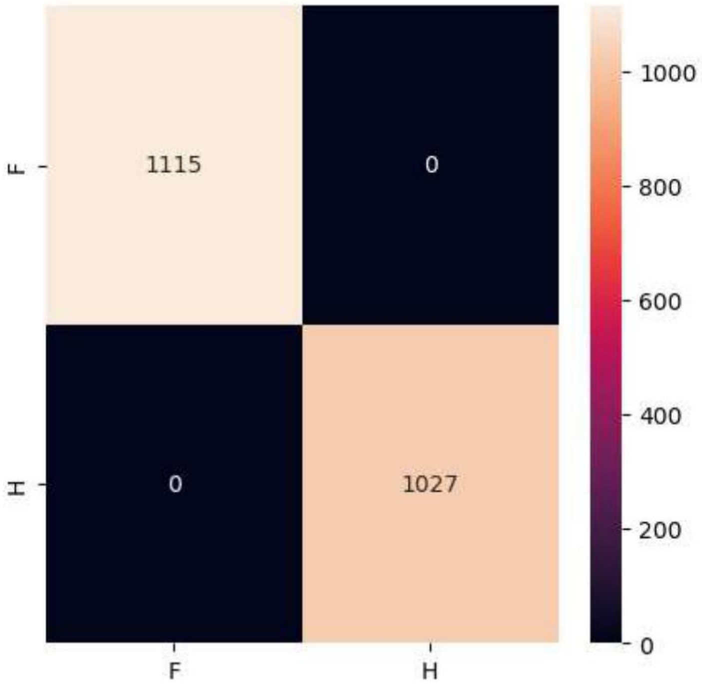
In [15]:

```
cnn_model.save(r'C:\Gear data\Trained Model\CNN_model_gear.h5')
```

In [16]:

```
def inv_Transform_result(y_pred):
    y_pred = y_pred.argmax(axis=1)
    y_pred = encoder.inverse_transform(y_pred)
    return y_pred



y_pred=cnn_model.predict(X_test)


Y_pred=inv_Transform_result(y_pred)
Y_test = inv_Transform_result(y_test)





from sklearn.metrics import confusion_matrix


plt.figure(figsize=(5,5))
cm = confusion_matrix(Y_test, Y_pred)
f = sns.heatmap(cm, annot=True, fmt='d',xticklabels=encoder.classes_,yticklabels=encoder.cl
plt.show()
```

```
67/67 [==============================] - 2s 2ms/step
```



In [17]:

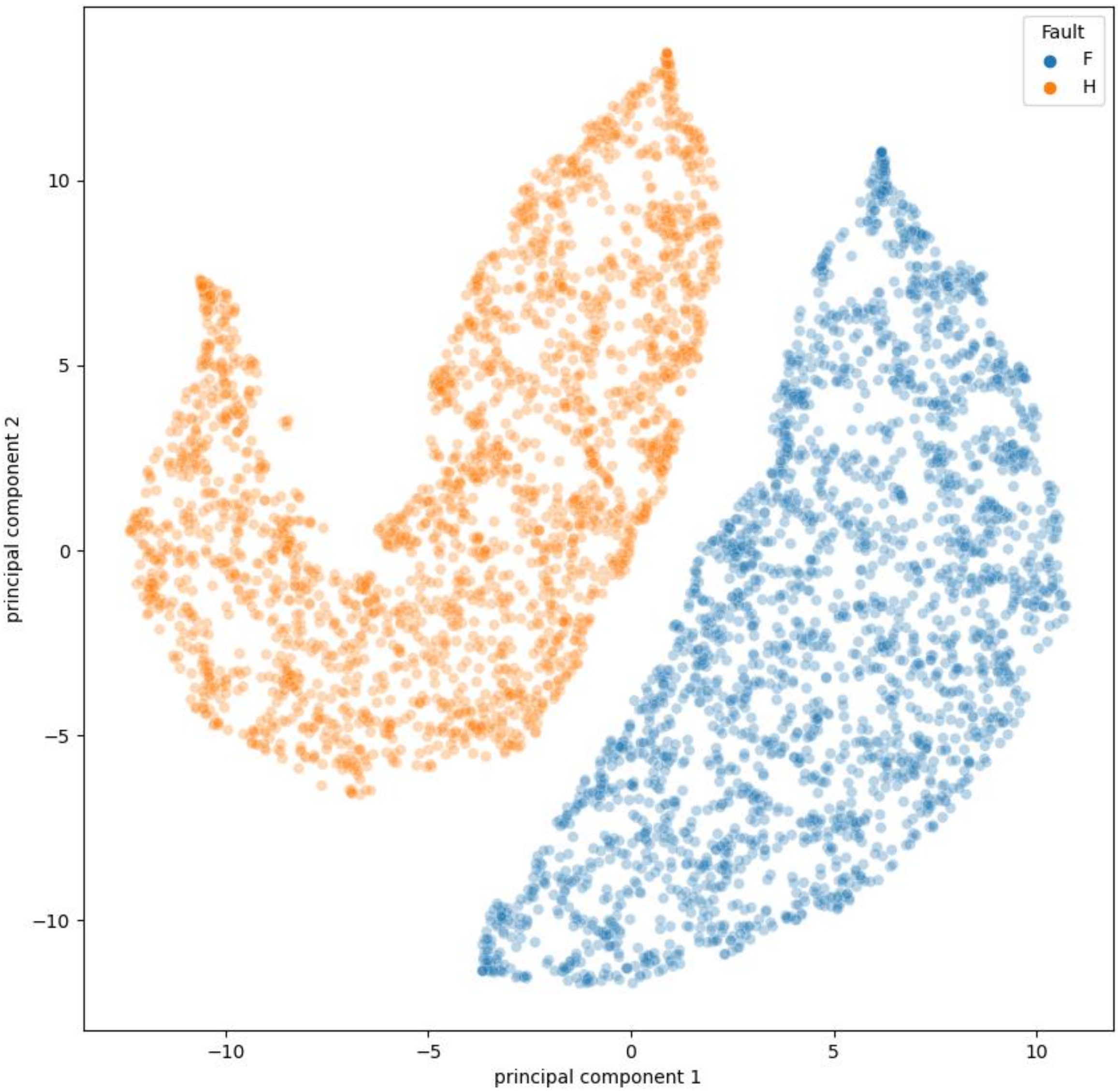
```
dummy_cnn = Model(inputs=cnn_model.input,outputs=cnn_model.layers[5].output)
y_viz = dummy_cnn.predict(X_train)
```

```
157/157 [==============================] - 2s 3ms/step
```

In [19]:

```
from sklearn.manifold import TSNE

X_t_sne = TSNE(n_components=2, learning_rate='auto',verbose=1, perplexity=40, n_iter=300).f

tSNEdf = pd.DataFrame(data = X_t_sne, columns = ['principal component 1', 'principal compon

tSNEdf['Fault']=inv_Transform_result(y_train)


# Plot the PC-1 and PC-2
fig, ax = plt.subplots(figsize=(10,10))
sns.scatterplot(x=tSNEdf['principal component 1'],y=tSNEdf['principal component 2'],hue='Fa
    data=tSNEdf,
    legend="full",
    alpha=0.3)
plt.show()
```

```
[t-SNE] Computing 121 nearest neighbors...
[t-SNE] Indexed 4996 samples in 0.001s...

C:\Users\prakash\anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:780:
FutureWarning: The default initialization in TSNE will change from 'random'
to 'pca' in 1.2.
  warnings.warn(

[t-SNE] Computed neighbors for 4996 samples in 0.590s...
[t-SNE] Computed conditional probabilities for sample 1000 / 4996
[t-SNE] Computed conditional probabilities for sample 2000 / 4996
[t-SNE] Computed conditional probabilities for sample 3000 / 4996
[t-SNE] Computed conditional probabilities for sample 4000 / 4996
[t-SNE] Computed conditional probabilities for sample 4996 / 4996
[t-SNE] Mean sigma: 0.563384
[t-SNE] KL divergence after 250 iterations with early exaggeration: 63.76034
5
[t-SNE] KL divergence after 300 iterations: 1.746305
```

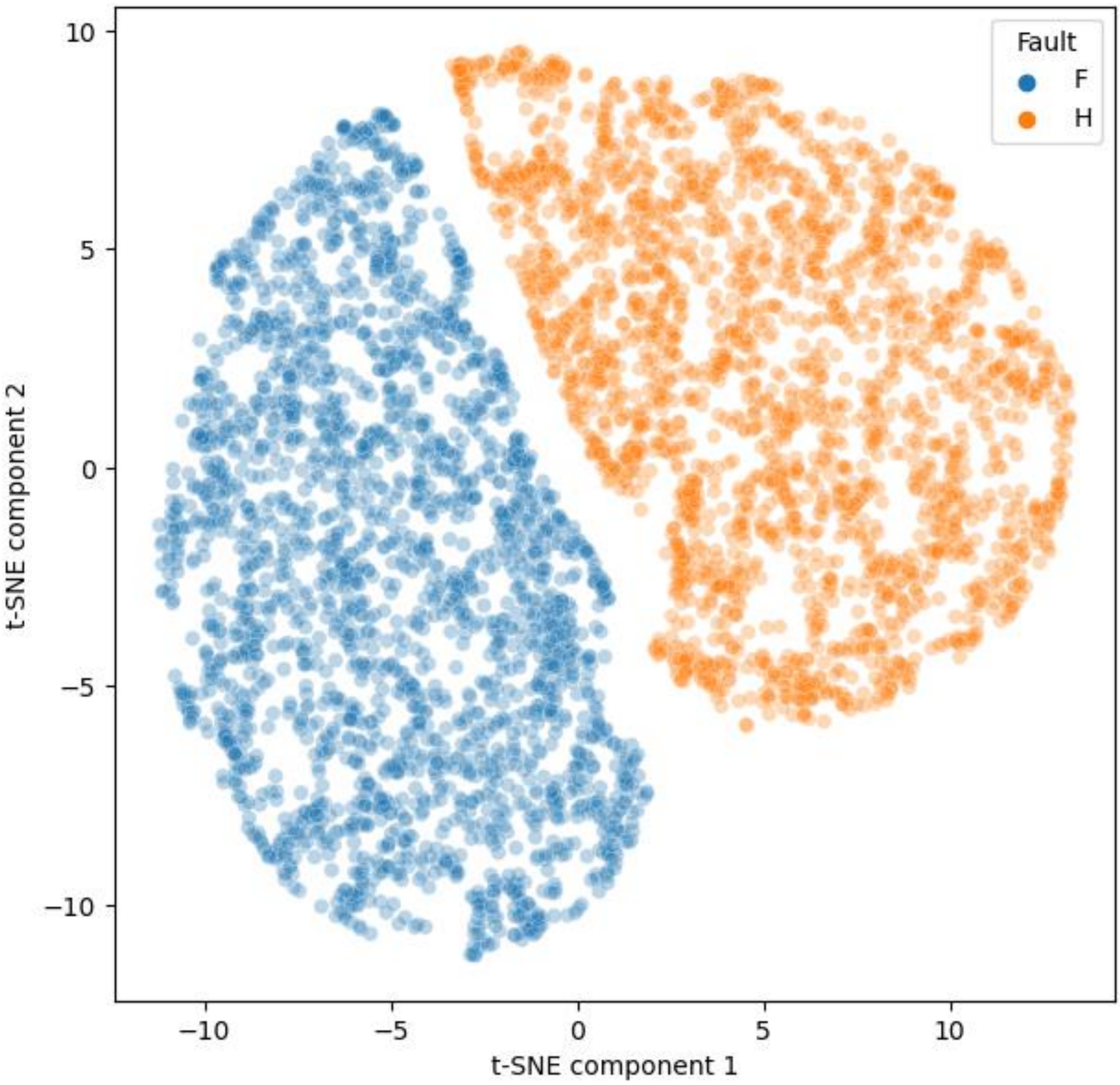


In [20]:

```
dummy_cnn = Model(inputs=cnn_model.input,outputs=cnn_model.layers[4].output)
y_viz = dummy_cnn.predict(X_train)
```

```
from sklearn.manifold import TSNE

X_t_sne = TSNE(n_components=2, learning_rate='auto',verbose=1, perplexity=40, n_iter=300).f

tSNEdf = pd.DataFrame(data = X_t_sne, columns = ['t-SNE component 1', 't-SNE component 2'])

tSNEdf['Fault']=inv_Transform_result(y_train)


# Plot the PC-1 and PC-2
fig, ax = plt.subplots(figsize=(7,7))
sns.scatterplot(x=tSNEdf['t-SNE component 1'],y=tSNEdf['t-SNE component 2'],hue='Fault',
    data=tSNEdf,
    legend="full",
    alpha=0.3)
plt.show()
```

```
157/157 [==============================] - 2s 4ms/step
[t-SNE] Computing 121 nearest neighbors...
[t-SNE] Indexed 4996 samples in 0.005s...

C:\Users\prakash\anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:780:
FutureWarning: The default initialization in TSNE will change from 'random'
to 'pca' in 1.2.
  warnings.warn(

[t-SNE] Computed neighbors for 4996 samples in 0.765s...
[t-SNE] Computed conditional probabilities for sample 1000 / 4996
[t-SNE] Computed conditional probabilities for sample 2000 / 4996
[t-SNE] Computed conditional probabilities for sample 3000 / 4996
[t-SNE] Computed conditional probabilities for sample 4000 / 4996
[t-SNE] Computed conditional probabilities for sample 4996 / 4996
[t-SNE] Mean sigma: 0.905886
[t-SNE] KL divergence after 250 iterations with early exaggeration: 68.12220
0
[t-SNE] KL divergence after 300 iterations: 1.900491
```


Fault
F
H
10
5
0
−5
−10
t-SNE component 2
−10
−5
0
5
10
t-SNE component 1

5.CONCLUSION

In order to identify the failure patterns of the gearbox, a deep learning method based on CNN for the vibration data has been suggested. The current CNN approach uses accelerometer-measured vibration data to detect and categorise gearbox defects. Different methodologies were used in the gearbox defect diagnostic studies to assess the suggested CNN method. The findings demonstrate that, when compared to competing approaches, the current method has the best performance for diagnosing gearbox faults. This kind of classifiers might assist with industrial system maintenance procedures, helping to save costs and ensure a continuous production system. With the right tools, online diagnostics could also be carried out.

REFERENCE

- Gearbox Fault Diagnosis | Kaggle
- Gear Faults and Gear Defects - CBM CONNECT®
- Gear Tooth Fault Detection Based on Designed Convolutional Neural Networks | SpringerLink
- Gearbox Fault Identification and Classification with Convolutional Neural Networks (hindawi.com)

Plagiarism Detector .net

Dec 04, 2022

Plagiarism Scan Report

3% Plagiarized

97% Unique

Characters:7537	Words:988
Sentences:35	Speak Time: 8 Min

Excluded URL	None

Plagiarism Detector .net

Dec 04, 2022

Plagiarism Scan Report

0% Plagiarized

100% Unique

Characters:2599	Words:356
Sentences:13	Speak Time: 3 Min

Excluded URL	None