

A Deep Learning Protocol for Health Monitoring of a Rotor-Bearing System from Unprocessed Data

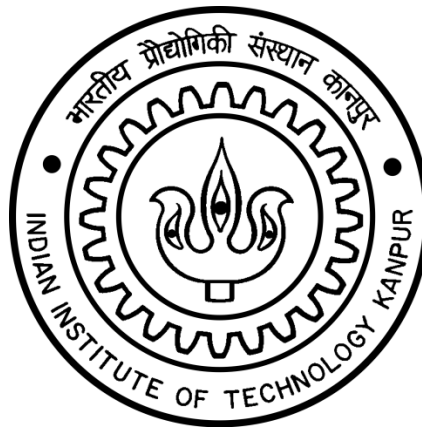
*A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of*

Master of Technology

by

Sonkul Nikhil Avinash

(14807703)



Department of Mechanical Engineering

**Indian Institute of Technology
Kanpur-208016 (India)**

CERTIFICATE

It is certified that the work contained in this thesis entitled, "A Deep Learning Protocol for Health Monitoring of a Rotor-Bearing System from Unprocessed Data", by *Sonkul Nikhil Avinash* (Roll No. 14807703), has been carried out under my supervision and this work has not been submitted elsewhere for a degree.



Nalinakshi S. Vyas
Dr. Nalinakshi S. Vyas 15.05.19.
Professor
Department of Mechanical Engineering
Indian Institute of Technology, Kanpur

ABSTRACT

A Deep Learning protocol is developed for identification of typical faults occurring in rotating machinery. In past Support Vector Machines (SVMs), Clustering, Artificial Neural Networks (ANNs) and other algorithms have been used for this purpose. However these algorithms require the raw time-domain data from the sensors to be first processed and handcrafted into parameters like Fast Fourier Transform (FFT) coefficients, Statistical Moments etc. before being fed as inputs. Their predictions are found to be inadequate with raw time domain data. This is due to the fact that they suffer from the vanishing gradient problem. Deep Neural Network can instead be developed to work directly with such data obtained from the sensors. Convolutional Neural Network (CNN) architecture is commonly used in deep learning algorithms for image recognition. A Deep Learning CNN architecture, which employs the RGB image analogy to map vibration data from sensors on a rotor-bearing system, has been developed in this study, for fault prediction. The developed network is found to effectively recognize all kinds of rotor, bearing, gear, belt-drive faults that were investigated.

*Dedicated to my grandfather
Shri. Ramdas Shankar Sonkul*

ACKNOWLEDGEMENT

I would first like to thank my thesis advisor Dr. Nalinaksh S. Vyas for steering me in the right direction. I would also like to thank Major Jasdeep Singh, whose experimental data is used in the present study. Without his work the thesis objective could not have been achieved. I also like to provide my gratitude to Case Western Reserve University Bearing Data Center for their open source data available online for the research work.

I must express my very profound gratitude to my parents and to my sister for providing me with unfailing support and continuous encouragement throughout my years of study. This work would not have been possible without them.

I thank to all my batch mates, Shivanshu, Shubham and Vineeth for their fruitful discussions throughout my institute tenure. I would like to thank my lab mates Ganesh, Mayank, Om Prakash and Ravi for their stimulating discussion. I appreciate and extend my thanks to Mohsin Sir, Mr. Rohit, Mr. Ramshankar for providing lively working environment and making my stay in laboratory comfortable. Thank you.

Sonkul Nikhil Avinash

CONTENTS

CHAPTER 1 INTRODUCTION.....	1
CHAPTER 2 DEEP LEARNING	4
2.1 Auto-Encoder	6
2.2 Convolutional Neural Network	7
2.3 Long Short Term Memory Networks.....	8
CHAPTER 3 CONVOLUTIONAL NEURAL NETWORK AND PROPOSED METHOD	10
3.1 Convolutional Layer.....	10
3.2 Batch Normalization Layer	13
3.3 ReLU Activation Layer.....	13
3.4 Pooling Layer	14
3.5 Fully Connected Layer	15
3.6 Proposed Method	16
CHAPTER 4 TEST DATA	19
4.1 Sub-Systems and Operational Parameters.....	19
4.2 Types of Faults & Training Datasets.....	21
CHAPTER 5 TRAINING, VALIDATION AND TESTING	26
5.1 Training in Frequency Domain	26
5.2 Training in Time Domain.....	33
5.3 Discussion on Frequency Domain Training.....	39
5.4 Discussion on Time Domain Training	40
5.5 Frequency Domain v/s Time Domain	40

5.6	Effect of Depth and Vanishing Gradient.....	41
CHAPTER 6 VALIDATION WITH OPEN SOURCE DATA		43
6.1	Data Description and Division.....	44
6.2	Network Training and Results for Set C	46
6.3	Network Training and Results for Set D.....	51
6.4	Discussion on results of Set C.....	56
6.5	Discussion on results of Set D	56
6.6	FFT domain v/s Time domain input.....	57
CHAPTER 7 CONCLUSION		58

LIST of FIGURES

Figure 2.1: An Auto-Encoder.....	6
Figure 2.2: On left pre-training of individual hidden layer and Deep Auto-encoder on right	7
Figure 2.3: An LSTM cell.....	8
Figure 3.1: Convolution and ReLU activation layer.....	11
Figure 3.2: Weight matrix of ANN v/s sparse and repetitive weight matrix in CNN.....	12
Figure 3.3: Rectified Linear Unit (ReLU) Activation Function.....	13
Figure 3.4: Average Pooling: Pooling region 2×1	14
Figure 3.5: Block diagram for CNN layers architecture	15
Figure 3.6: A sample CNN architecture for proposed method: Input matrix is made with responses from three sub-systems corresponding to three channels with two sensors response (horizontal & vertical) in each channel.	18
Figure 4.1: The rotor system (Machine fault Simulator) and sensor locations	20
Figure 4.2: A typical response from sensors for particular fault [16]	25
Figure 5.1: Training loss for (Set A) when input is in frequency domain with Avg. Pooling	28
Figure 5.2: Training loss for (Set B) when input is in frequency domain with Avg. Pooling	28
Figure 5.3: Training loss for Set A trained with all sub-systems as input in frequency domain	30
Figure 5.4: Training loss for Set B trained with all sub-systems as input in frequency domain.	30
Figure 5.5: Feature map after dimensionality reduction for (Set A) with Max. Pooling	31
Figure 5.6: Feature map after dimensionality reduction for (Set B) with Max. Pooling	32
Figure 5.7: Training loss for (Set A) when input is in time domain with Avg. Pooling	34
Figure 5.8: Training loss for (Set B) when input is in time domain with Avg. Pooling	34
Figure 5.9: Training loss for (Set A) trained with all sub-systems as input in time domain.....	36
Figure 5.10: Training loss for (Set B) trained with all sub-systems as input in time domain.....	36
Figure 5.11: Feature map after dimensionality reduction for (Set A) with Max. Pooling	37
Figure 5.12: Feature map after dimensionality reduction for (Set B) with Max. Pooling	38
Figure 5.13: Weight Update for Neural Network	42

Figure 5.14: Weight Update for CNN.....	42
Figure 6.1: Setup used for experimentation by [17].....	43
Figure 6.2: Training loss for (Set C) trained with all sensors as input in frequency domain.....	48
Figure 6.3: Training loss for (Set C) trained with all sensors as input in time domain.....	48
Figure 6.4: Feature map after dimensionality reduction for (Set C) with Max. Pooling (FFT)..	49
Figure 6.5: Feature map after dimensionality reduction for (Set C) with Max. Pooling (Time)	50
Figure 6.6: Training loss for (Set D) trained with all sensors as input in frequency domain.....	52
Figure 6.7: Training loss for (Set D) trained with all sensors as input in time domain	53
Figure 6.8 Feature map after dimensionality reduction for (Set D) with Avg. Pooling (FFT) ...	54
Figure 6.9: Feature map after dimensionality reduction for (Set D) with Max. Pooling (Time)	55
Figure 8.1: Frequency Domain	59
Figure 8.2: Frequency Domain	60

LIST of TABLES


Table 3.1: CNN architecture and hyperparameters	17
Table 4.1: Break up of a typical rotor system into its sub-systems	20
Table 4.2: Type of faults in training dataset (Set A)	22
Table 4.3: Type of faults in training dataset (Set B)	23
Table 5.1: Accuracies for (Set A) trained with individual sub-system in frequency domain	27
Table 5.2: Accuracies for (Set B) trained with individual sub-system in frequency domain	27
Table 5.3: Accuracies for (Set A) trained with all sub-systems as input in frequency domain ..	29
Table 5.4: Accuracies for (Set B) trained with all sub-systems as input in frequency domain...	29
Table 5.5: Accuracies for (Set A) trained with individual sub-system in time domain	33
Table 5.6: Accuracies for (Set B) trained with individual sub-system in time domain	33
Table 5.7: Accuracies for (Set A) trained with all sub-systems in time domain	35
Table 5.8: Accuracies for (Set B) trained with all sub-systems in time domain	35
Table 5.9: Training with input from individual Sub-Systems and all sub-systems combined....	39
Table 5.10: Effect of depth and pooling function for model with all sensors responses as input	41
Table 6.1: Accuracies for (Set C) trained with all sensors in frequency domain	47
Table 6.2: Accuracies for (Set C) trained with all sensors in time domain	47
Table 6.3: Accuracies for (Set D) trained with all sensors in frequency domain	52
Table 6.4: Accuracies for (Set D) trained with all sensors in time domain	52
Table 6.5: Structure of data downloaded from [17]	45

CHAPTER 1

INTRODUCTION

Condition monitoring of machinery is continuously expanding area of engineering activity. The machine condition monitoring market was valued at USD 2.21 Billion in 2017 and is expected to reach USD 3.50 Billion by 2024 [1]. Cheap sensors, high computational power and increased runtime of industries due to escalating demand are all fuelling research and development of fast and robust algorithms to address this issue. The Fourth Industrial Revolution (*I4*) aims to combine different branches of technology so that we can run our businesses and organizations with maximum efficiency. Nature is the best example of a system or organization that runs at optimum efficiency. Bio-mimicry is the imitation of the models, systems, and elements of nature for the purpose of solving complex human problems. Recent progress in machine learning is highly inspired from bio-systems. Artificial Neural Networks (ANN) is a branch of machine learning, which learns complex tasks using nature-inspired models. ANN originated as a model to investigate how learning happens in brain. As complexity of problem increased, different models like Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) networks were introduced, which were inspired from animal visual cortex and long-term, short-term memory cells, respectively.

Rotor Machinery constitute a large part of the industrial assets and can be a very complex system within itself. Complexity and production capabilities of these systems are increasing due to technological advancements. To meet the increasing demand, machines are expected to run continuously for a long time without uncertain casualties. To avoid unexpected downtime, operation and maintenance teams are gradually shifting their preference from corrective to preventive to predictive maintenance practices. Condition monitoring is primarily

used in predictive maintenance. A  needs to be detected before it causes critical machine failure and hence fault diagnosis techniques are required to monitor the health of rotor systems.

Vibration signals are widely used in fault identification. Data collection, feature extraction and fault identification are typical steps in fault diagnosis. Raw vibration data contains information about faults but cannot be interpreted by humans. Features from time and frequency domain are first extracted from these raw signals. Statistical parameters, like moments and Fast Fourier Transform coefficients are widely used features. Machine learning techniques have shown their potential to identify faults using these handcrafted features. Fuzzy Logic, Wavelet, Clustering, Decision Tree, Support Vector Machines (SVM) and Artificial Neural Networks (ANN) are some of the techniques used in past for fault identifications ([2], [3], [4], [5], [6], [7]). Statistical moments of raw time domain signal and its derivatives were used as features of ANN by Vyas [8] for fault diagnosis of rotor-bearing system.

Extracting information rich data from raw sensors need humans with industry experience. Efforts to automate the feature extraction are being made in recent studies. Janssens [9] used Discrete Fourier Transform coefficients as input to Convolutional Neural Network (CNN) for bearing fault classification. Features learnt from CNN network combined with time domain features were used by Xie [10] to train SVM model. Guo [11] trained a CNN network with input as continuous wavelet transform scalogram of rotor machinery.

ANNs are not capable of extracting abstract features from raw time domain data due to vanishing gradient. The proposed method directly works upon raw time domain data without manually extracting features.

CNN is chosen as primary model in this study. CNN is emerging as a powerful deep learning technique in the field of image related tasks. CNN preserves topology of input. It has less learning parameters than a neural network of same depth. This makes learning fast for model. Deep Neural Networks have shown their capabilities to learn abstract features from large and multi-dimensional data like images and audio signals to solve complex problems like image classification, language translation, speech recognition etc. Deep Learning architectures are also showing their potential in the field of condition monitoring.

As complexity and size of system increases, vibration from particular location gets damped out while reaching to a farther location. Faults generating from locations farther from sensor location are hard to identify. A multi sensor model is necessary to identify faults for large systems. This issue is considered for developing a model of fault diagnosis in the present study.

The objective of this study is to develop a deep learning model which can classify faults with raw data as input without handcrafted feature. A Multi-channel input Convolution Neural Network (McCNN) is developed for fault identification of a Rotor-Bearing System. Raw time domain vibration signals from different locations are merged into a multi-channel input analogous to a RGB image.

The McCNN model is capable of classifying all faults introduced to a rotor on which experiments were performed, for both time domain and frequency domain input. The model is validated with an open source data from Case Western Reserve University Bearing Data Center Website. The model satisfactorily classified all faults when input was provided as raw time domain data and was successfully able to classify all faults when data was provided in form of FFT coefficients.

The remaining report is organized as follows. Chapter 2 gives introduction to deep learning and its need; details about CNN and proposed model are given in Chapter 3; experiment setup and training data is detailed in Chapter 4; results of network training are discussed in Chapter 5; proposed model is validated with data from different system in Chapter 6 and Chapter 7 concludes the thesis along with discussion on future work.

CHAPTER 2

DEEP LEARNING

Artificial Neural Networks and other Machine learning methods like Clustering, Decision Tree and Support Vector Machines have been employed to solve several complex engineering problems in recent times. These methods need data, which are information rich and not very raw i.e. the data needs to be preprocessed and features need to be handcrafted before they are fed to these algorithms. Vibration based condition monitoring mostly involves preprocessing sensor data in time or frequency domain, which we call handcrafted data..

In past, machine learning methods were applied on handcrafted features for machine health monitoring. Vyas [8] used statistical moments to train Neural Network for fault identification in Rotor-bearing System. Yaguo [4] diagnosed fault and compound fault of locomotive roller bearings with an unsupervised Clustering algorithm. Data was demodulated and both time-domain and frequency-domain features parameters were used to train the model in this study. Mouloud [5] identified bearing defects in rotating machinery using Decision Tree, which was constructed with statistical features and trained with C4.5 algorithm. Jack [7] examined the performance SVM's on to detect faults in rotating machinery using both statistical features and FFT coefficients.

It is difficult to choose between various feature extraction techniques for different tasks.

It is therefore, better to automate the process of feature engineering, using machine learning.

This is known as representation learning. Mapping of not only feature to output but also

mapping of raw data to features is included in representation learning. These learnt features

often give good results compare to hand crafted results. They are robust to new tasks or new systems and require minimum human intervention. As complexity of problem increases, the process of feature extraction becomes more difficult for humans. Features are extracted so that we can separate factors that are causing variations in data, which can avoid corruption of model

due to noisy or unwanted data. Such features sometimes, even though present, in raw data cannot be perceived manually. Deep networks learn these representations or features as expressed in terms of simpler representations. It builds complex concepts out of simpler concepts. For example, in image related tasks an image of a person is represented as combination of simpler features like corners and contour which is itself a representation of simpler representation edges. It is not possible to visualize, in context of vibration data, what these simpler representations are, but the features learnt are surely the one, which cause large variation in data. Factors that enhance usage of deep learning are increasing sizes of datasets models, required accuracy, application complexity and real world impact.

ANNs have proved their importance in past, but they were incapable of learning faults from raw time domain data. The input size of raw time domain data is almost hundred times than the inputs created using feature engineering in traditional approaches as discussed above. It was necessary to have large number of hidden layers in Neural Networks, so that abstract information or patterns can be extracted at each layer to help next layer. However, in practice deep architectures can be particularly hard to learn [12].

ANNs with large number of hidden layers can be called a deep network, but training these networks gives poor results. As the depth of network increases it is more difficult to obtain good generalization. Gradient becomes less informative about the required change in parameters as we move back towards the lower layers. This is called Vanishing Gradient. Backpropagation computes gradients by the chain rule. Traditional activation functions such as sigmoid, the hyperbolic tangent function have gradients in the range (0, 1). Each layer is influenced by all preceding layers while computing gradient and the effect is exponential. As a result the layers near input have very small gradients and hence they fail to learn any abstract information from raw data. Due to small weight gradients weights gets stuck in local minima or plateaus. This results in over fitting and hence deep networks just learn training data but fail to generalize any pattern to get good results on test data.

This problem was solved by proposing different methods of weight initialization, constrained between weights of same layer etc. Auto-encoders, CNN's, LSTM Networks are

some deep architecture methods proposed to overcome this problem. Subsequently, deep learning approaches have been proven to learn complex tasks like image classification, language translation speech recognition etc. These techniques are described briefly, below.

2.1 Auto-Encoder

An auto-encoder is a neural network with a single hidden layer. Auto-encoder employs an unsupervised learning method to encode the input \mathbf{x} into some representation $\mathbf{c}(\mathbf{x})$. It also learns mapping $\mathbf{g}(\mathbf{c}(\mathbf{x}))$ going backwards from representation $\mathbf{c}(\mathbf{x})$ to \mathbf{x} . So the output of the auto-encoder is input itself. The representation $\mathbf{c}(\mathbf{x})$ are features capturing factors of variation in data.

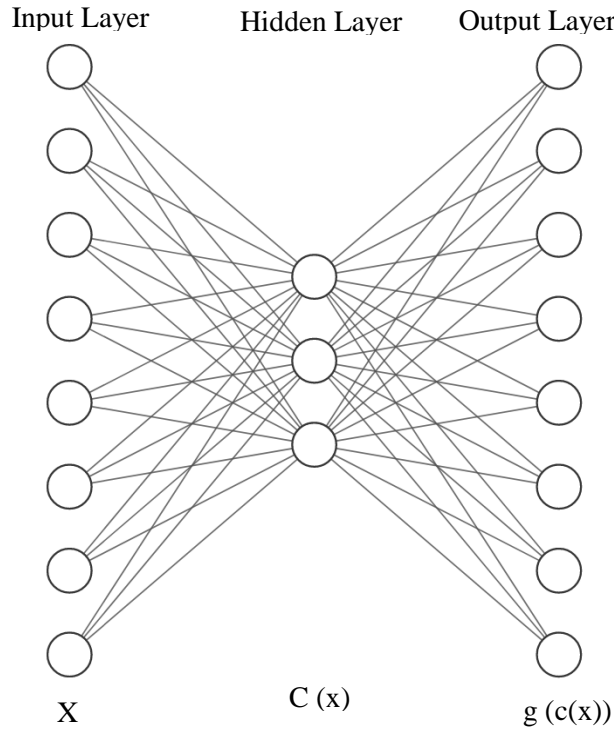


Figure 1: An Auto-Encoder

The unsupervised learning in auto encoder is used to successfully train a deep network and obtain good generalization. The deep auto-encoders use greedy layer-wise unsupervised pre-training to initialize the weights. Each layer of a deep neural network is pre-trained with

output as its input from previous hidden layer and the new encoded representation is used as next hidden layer and also the input to further hidden layer for pre-training.

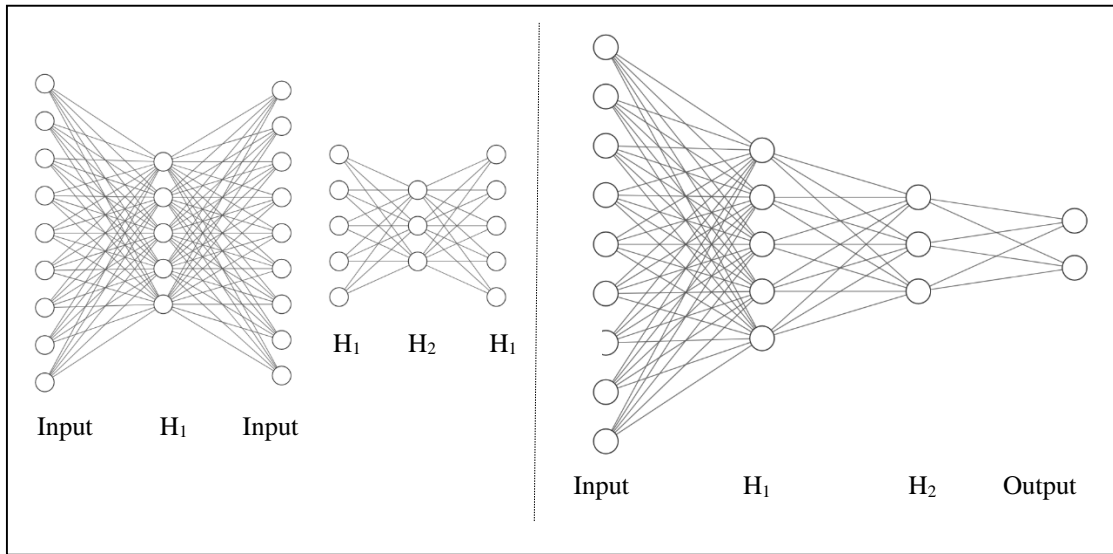


Figure 2: On left pre-training of individual hidden layer and Deep Auto-encoder on right

After the last layer is pre-trained, a supervised layer is put on top, and all the layers are jointly trained with respect to the overall supervised training criterion. In other words, the pre-training was only used to initialize a deep supervised neural network. This pre-training results in better generalization.

2.2 Convolutional Neural Network

Convolutional Neural Networks (CNN) have large number of layers which are not possible to be trained for Fully Connected Neural Network (FCNN). CNNs have two primary types of layers, convolutional layers and subsampling or pooling layers. Each neuron has a spatial structure i.e. each is associated with a fixed two dimensional geometry which is a location in the input layer of that particular layer.

Each layer of CNN is itself a two dimensional structure (unlike 1D in FCNN). And each neuron in this 2D layer is associated with neurons from previous layers within a rectangular field with same set of weights. CNNs have ReLU activation function which avoids

vanishing gradients and enables to add large number of hidden layers to learn complex tasks. More about CNN in Section 3.

2.3 Long Short Term Memory Networks

Recurrent Neural Networks (RNN) are primarily used for time series data and related tasks. LSTMs were developed to deal with the exploding and vanishing gradient problems that can be encountered when training traditional RNNs. LSTM networks are well-suited to classifying, processing and making predictions based on time series data.

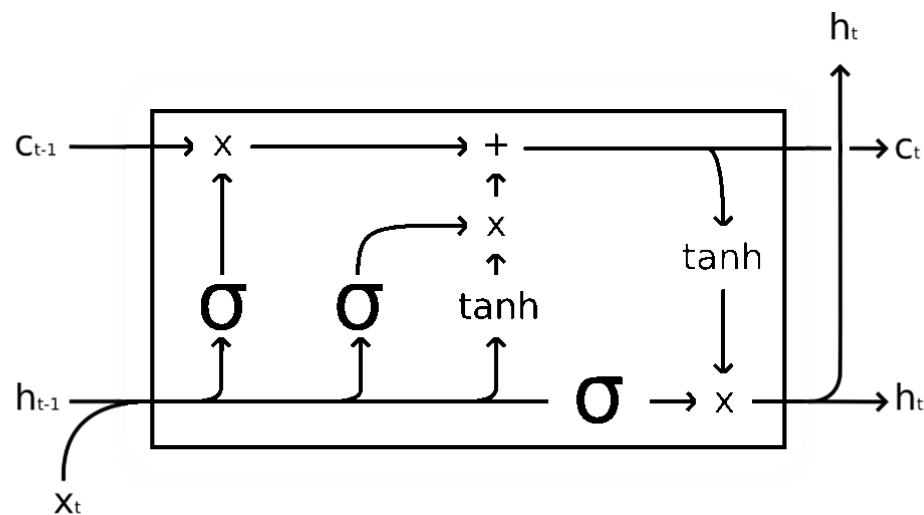


Figure 3: An LSTM cell*

LSTM solves problem of RNN by introducing a memory unit, called the cell as shown in Figure 3. The cell takes three type of inputs. X_t is the input of current time step, h_{t-1} is output from previous cell and C_{t-1} is memory from previous cell. Output of current cell is h_t and memory is C_t . Thus the unit makes decision by considering current input, previous output and previous memory. It also generates its own memory and output.

*https://upload.wikimedia.org/wikipedia/commons/3/3b/The_LSTM_cell.png (wiki-image)

Each cell has regulators called gates, to control flow of information inside the LSTM unit: an input gate, an output gate and a forget gate. The input gate controls the extent to which a new value flows into the cell, the forget gate controls the extent to which a value remains in the cell and the output gate controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit.

CNN is chosen as primary method for present study and a detailed explanation of its working is given in Chapter 3.

CHAPTER 3

CONVOLUTIONAL NEURAL NETWORK AND PROPOSED METHOD

CNNs are specialized kind of neural networks employed for processing multi-dimensional data, for example time-series data as a 1D grid, taking samples at regular time intervals, and image data as a 2D grid of pixels. Convolutional neural networks are widely used in deep networks. CNN was first introduced by LeCun [13]. CNNs are essentially deep neural networks with shared and sparse weights or parameters (Section 3.1). Most of the weights are forced to be zero (making it sparse) and non-zero weights are repeated in each neuron of a hidden layer. They use convolution instead of matrix multiplication in at least one of the layers. CNNs are mostly used in image related tasks like recognition, classification etc. The reason they are proved to be powerful in computer vision is that they preserve topology of input i.e. they accept two dimensional image as input unlike conventional ANN which accepts one dimensional data or features from data. It is capable of capturing spatial information like curves, edges, shapes etc. The spatial information gets lost once it is opened up into 1D which is avoided in CNN.

The building blocks of a CNN are discussed in detail below.

3.1 Convolutional Layer

Convolution is a mathematical operation on two functions. If f and g are two real valued functions with real argument t , convolution is defined as

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau$$

The data generated in real world is mostly of discrete form. For functions or data with two arguments the discrete convolution is modified as

$$(x * w)[i, j] = \sum_m \sum_n x[i + m, j + n] w[m, n]$$

where x is 2D input function and w the weight matrix called kernel. The definition can be generalized for more than two parameter functions. Number of kernels and size of kernel are

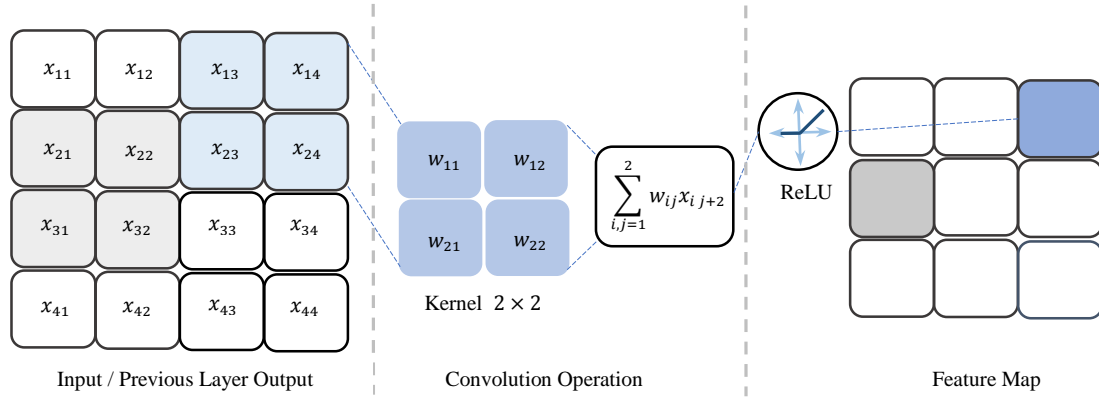


Figure 4: Convolution and ReLU activation layer

the hyperparameters for particular convolutional layer. Refer Figure 4 for visualization of convolution operation.

Discrete Convolution is similar to matrix multiplication with **additional constraints**. Weight matrix has several entries to be equal to other entries called shared parameters. Also many entries of weight matrix are constrained to zero making it a sparse matrix (matrix with most entries equal to zero). The weight matrix is sparse because kernel is usually of very small dimension compared to input matrix. Traditional fully connected ANN has a separate weight for connecting each input entity to each output neuron. CNNs typically have sparse weight matrices. This is achieved by keeping kernel size considerably small compared to input dimension. These kernel parameters are also repeated within weight matrix.

Consider a 2D input of size 4×2 as shown in Figure 5 and a hidden layer with three neurons. For a fully connected neural network (on left of Figure 5) we will require 24 weights or our weight matrix will have 24 entries. These entries are w_{ij} (where $i = 1, 2 \dots 7, 8$ and $j = 1,$

2, 3). Same input can produce a next hidden layer or feature map of size 3 with convolutional operation (on right of Figure 5).

For the first neuron of hidden layer the weights needed for dot product in ANN are

$(w_{11}, w_{21}, w_{31}, w_{41}, w_{51}, w_{61}, w_{71}, w_{81})$ and the weights used in CNN are $(w_1, w_2, w_3, w_4, 0, 0, 0, 0)$.

For second and third neuron in ANN weights are

$(w_{12}, w_{22}, w_{32}, w_{42}, w_{52}, w_{62}, w_{72}, w_{82})$ and $(w_{13}, w_{23}, w_{33}, w_{43}, w_{53}, w_{63}, w_{73}, w_{83})$ respectively.

For CNN weights used for second and third neurons are

$(0, 0, w_1, w_2, w_3, w_4, 0, 0)$ and $(0, 0, 0, 0, w_1, w_2, w_3, w_4)$ respectively.

Figure 5 shows the operation of dot product and weights for third neuron only.

As we compare the weights, we can understand that most of the weights are zero in CNN and are also repeated at each neuron. This explains the forced constraints of sparse and repetitive weight matrix CNN. Here these repetitive weights (w_1, w_2, w_3, w_4) are called kernel.

Due to smaller kernels, we need to store fewer parameters, thus reducing memory requirement and improving the efficiency of the model. CNN provides a means for working with inputs of variable size.

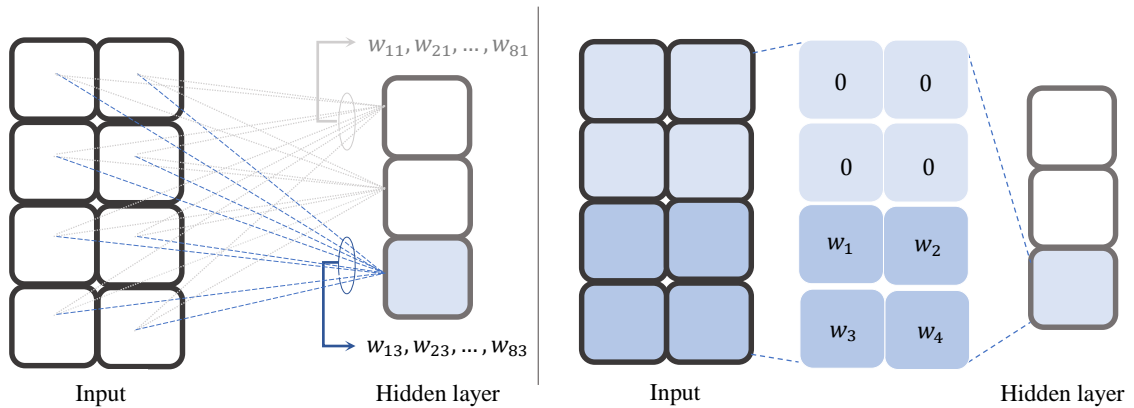


Figure 5: Weight matrix of ANN v/s sparse and repetitive weight matrix in CNN

Dimension of output or feature map after convolution is $D_x - D_K + 1$. D_x is the size of input in particular direction and D_K is size of kernel in same direction. Sometimes rows or columns of zeroes are added at the boundaries of input, called padding. Padding sometimes help to retain the dimension after convolution.

3.2 Batch Normalization Layer

Batch Normalization layer is added between Convolution and ReLU activation layer. It normalizes the feature map after convolution layer firstly by subtracting each of its inputs by mini-batch mean and then dividing by their standard deviation for each channel, and secondly scaling the new obtained featured map by γ and then shifting by β , where γ and β are learnable parameters. Batch Normalization accelerates deep network training [14].

3.3 ReLU Activation Layer

After convolutional layer the obtained output or feature map is passed through an activation layer. Refer Figure 6.

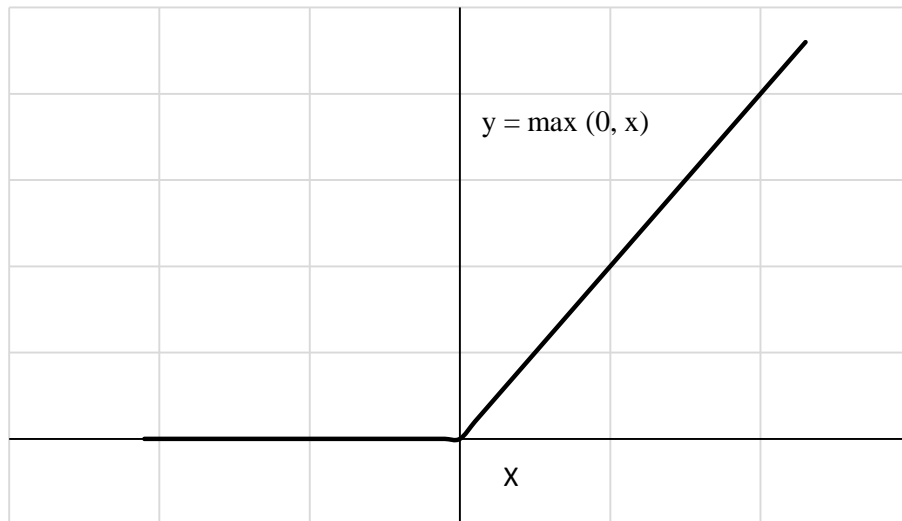


Figure 6: Rectified Linear Unit (ReLU) Activation Function

This layer maps the convoluted output into different space using activation function. The activation function mostly used in CNNs is rectified linear unit (ReLU). ReLU is defined as $\max(0, x)$. The vanishing gradient problem is avoided to some extent by using ReLU activation function. As the derivative of this function is either zero or one, it doesn't bring the gradients far too low as moving from output to input.

3.4 Pooling Layer

After ReLU activation the obtained feature map is passed through pooling layer, which modifies output of layer further. Pooling summarizes the responses over a neighborhood. It reduces the size of output from activation layer. It reduces the computational burden of the network. It also helps reducing memory requirement. Pooling helps to make features invariant to small translations of input. It means if the pattern location in input is varied across dimensions it will still extract it. Max Pooling operation is performed to extract maximum value from a rectangular region. Average Pooling takes the average value from region. The size of the region is the hyperparameter. Refer Figure 7. Other popular pooling functions include the $L2$ norm of a rectangular neighborhood, or a weighted average based on the distance from the central pixel

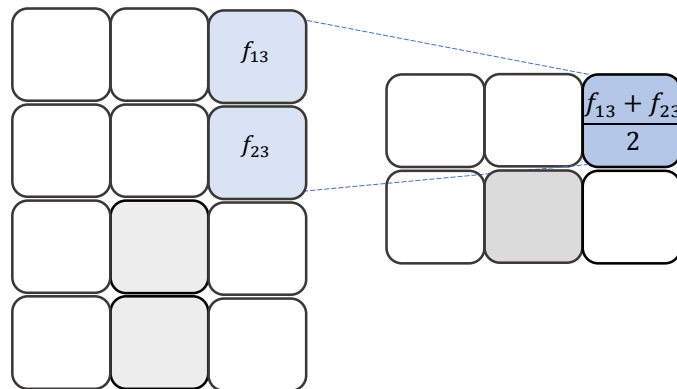


Figure 7: Average Pooling: Pooling region 2×1

Multiple sets of {A-B-C} layers are added depending upon problem complexity (Figure 8). Refer

Figure 9 for a sample network with three sets of A-B-C with Fully Connected Layer at last.

3.5 Fully Connected Layer

The last layer is a Fully Connected (FC) layer. The final output from last pooling layer is opened up into 1D and then connected to this last layer. The number of neurons in this layer are equal to number of classes present in datasets. This layer further transfers the output into softmax function and classifies accordingly. For detailed information on CNN refer [\[15\]](#).

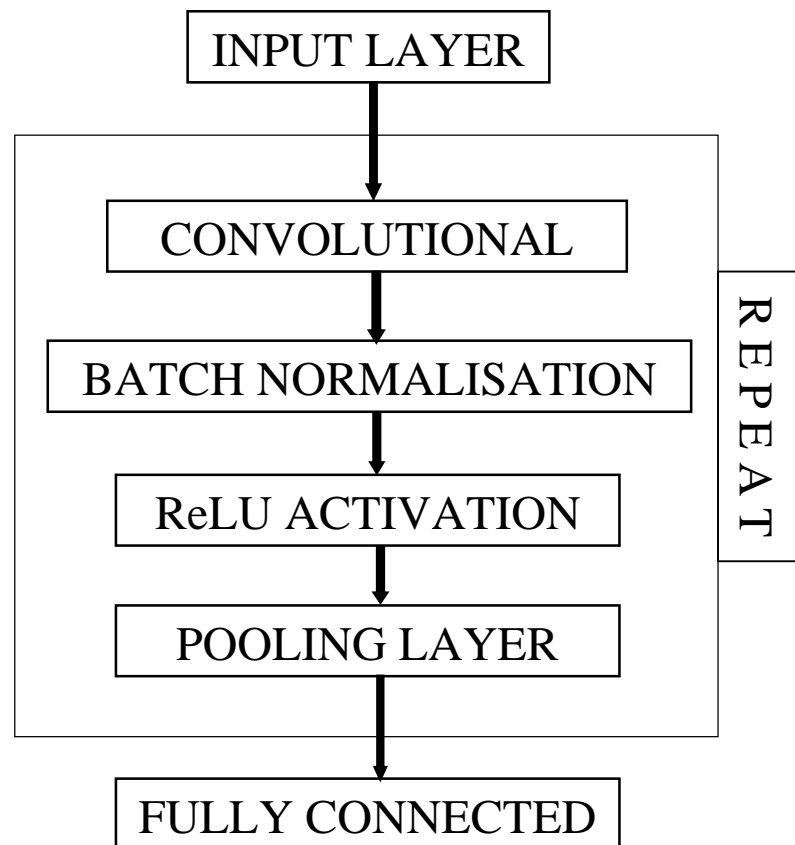


Figure 8: Block diagram for CNN layers architecture

3.6 Proposed Method

In our study a data from a rotor-bearing test set-up is used for prediction of different types of faults that could possibly occur.

The sensor data from accelerometers represent acceleration while proximity sensors collected displacement data. For homogeneity the proximity response is first double differentiated, using second order central difference method to get acceleration response. The new acceleration response of all six sensors is mapped between -1 and 1. In McCNN, a multi-channel input similar to RGB image is constructed from the sensor responses. Each of the two (horizontal and vertical) sensors at particular sub-system comprises a channel and three such sub-systems complete the multi-channel input to CNN. The topology of input is hence $4096 \times 2 \times 3$. Each channel is analogous to an RGB channel of a colored image. Each channel is padded with zero columns on both sides as shown in

Figure 9. This input is passed down to further layers given in Table 3.1.

Network training is performed with stochastic gradient descent algorithm with momentum. The network was run for maximum 800 iterations. Maximum iterations was the stopping criterion for network training. Learning rate was 0.01. Mini-batch size was kept to 64 for (Set A) training and 128 for (Set B). Validation Patience is kept infinite. Data is shuffled after every epoch, i.e. one round of training with all data. Both max pooling and average pooling were used in training for comparison purpose. The loss function used for back propagation is cross entropy loss function. The function is defined as

$$Loss_{ce} = - \sum_{i=1}^N \sum_{j=1}^K Y_{ij} \ln O_{ij}$$

where N is number of samples or mini-batch size, K is number of classes, Y_{ij} is the required output for i^{th} sample belonging to class j (1 if i^{th} sample belongs to class j else 0), O_{ij} is output from softmax function for j^{th} class for i^{th} example.

Beyond this point, unless otherwise mentioned, the pooling operation used in this study is Average Pooling.

Table 3.1: CNN architecture and hyperparameters

#	Layer	Kernel Size/ Pooling Region	# of Kernels	Padding
1	Input	-	-	-
2	Convolution 1	40×2	8	Yes
3	Batch Normalization + ReLU	-	-	-
4	Pooling 1	3×1	-	-
5	Convolution 2	20×2	16	Yes
6	Batch Normalization + ReLU	-	-	-
7	Pooling 2	4×1	-	-
8	Convolution 3	4×2	32	No
9	Batch Normalization + ReLU	-	-	-
10	Pooling 3	3×1	-	-
11	Convolution 4	4×2	32	No
12	Batch Normalization + ReLU	-	-	-
13	Pooling 4	2×1	-	-
14	Convolution 5	2×2	32	No
15	Batch Normalization + ReLU	-	-	-
16	Pooling 5	2×1	-	-
17	Convolution 6	2×1	32	No
18	Batch Normalization + ReLU	-	-	-
19	Pooling 6	2×1	-	-
20	Convolution 6	2×1	32	No
21	Batch Normalization + ReLU	-	-	-
22	Pooling 6	2×1	-	-
23	Fully Connected	Neurons = No. of Classes	-	-

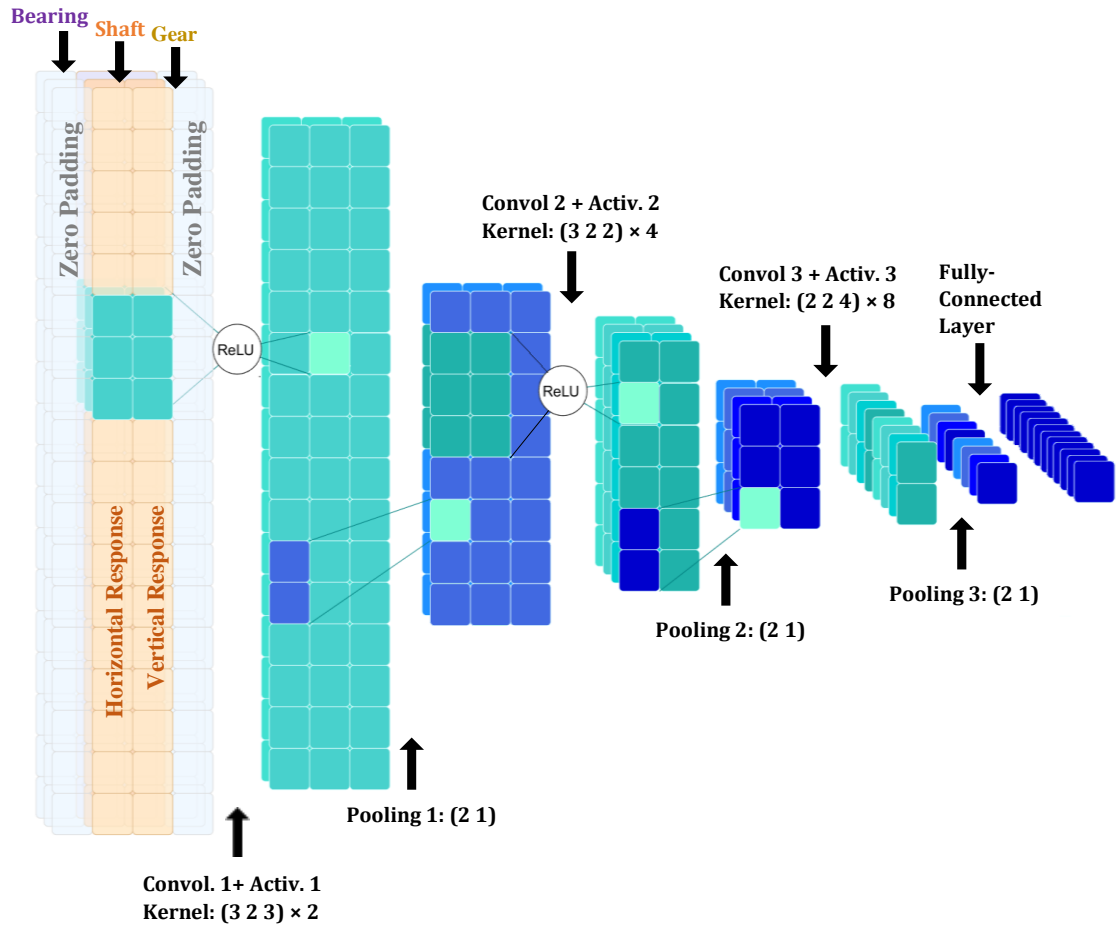


Figure 9: A sample CNN architecture for proposed method: Input matrix is made with responses from three sub-systems corresponding to three channels with two sensors response (horizontal & vertical) in each channel.

CHAPTER 4

TEST DATA

Test data has been taken from experiments performed on a rotor test rig (Machine Fault Simulator Figure 10) by Jasdeep Singh [16]. The set-up, instrumentation and data acquisition carried out by him is briefly described below, for sake of ready reference.

Accelerometers and Proximity pickups were used for collecting vibration signals from different subsystems of the rig. The Machine Fault Simulator consists of a shaft supported in two roller bearings and driven by a DC motor. Operating speed and acceleration of motor are controllable. The rig includes two rotors mounted on the shaft. Rotor disc can be mounted on shaft at desired location. A flexible coupling is used to connect rotor shaft to that of motor to compensate for any misalignment between the bearings as well as preventing any unwanted dynamic effects occurring in the motor being transmitted through the coupling into the rotor itself. At one end of the shaft there is a sheave which is connected to a reciprocating mechanism through a belt drive and a gear-box. The rig has adjustable speed in the range of 0 to 6000 RPM or 0 to 100 Hz.

4.1 Sub-Systems and Operational Parameters

A typical rotor system can be divided into its subsystems according to Table 4.1. Mostly faults occur in rotational category. Hence the locations for sensors are determined from the same. Sensors are placed at (i) Gearbox (ii) Bearing and (iii) Shaft-Rotor Disc. Each location has two sensors oriented mutually perpendicular to each other (horizontal & vertical). Accelerometers were used at (i), (ii) and Proximity pickups at (iii). Data from total six sensors was collected using Data Acquisition Card and LabVIEW software. Vibration data was collected at 2560 samples/sec. For an experiment data is collected for duration of 1.6 sec i.e. 4096 sample points in time domain. The data was collected for operating speed of 40 Hz.

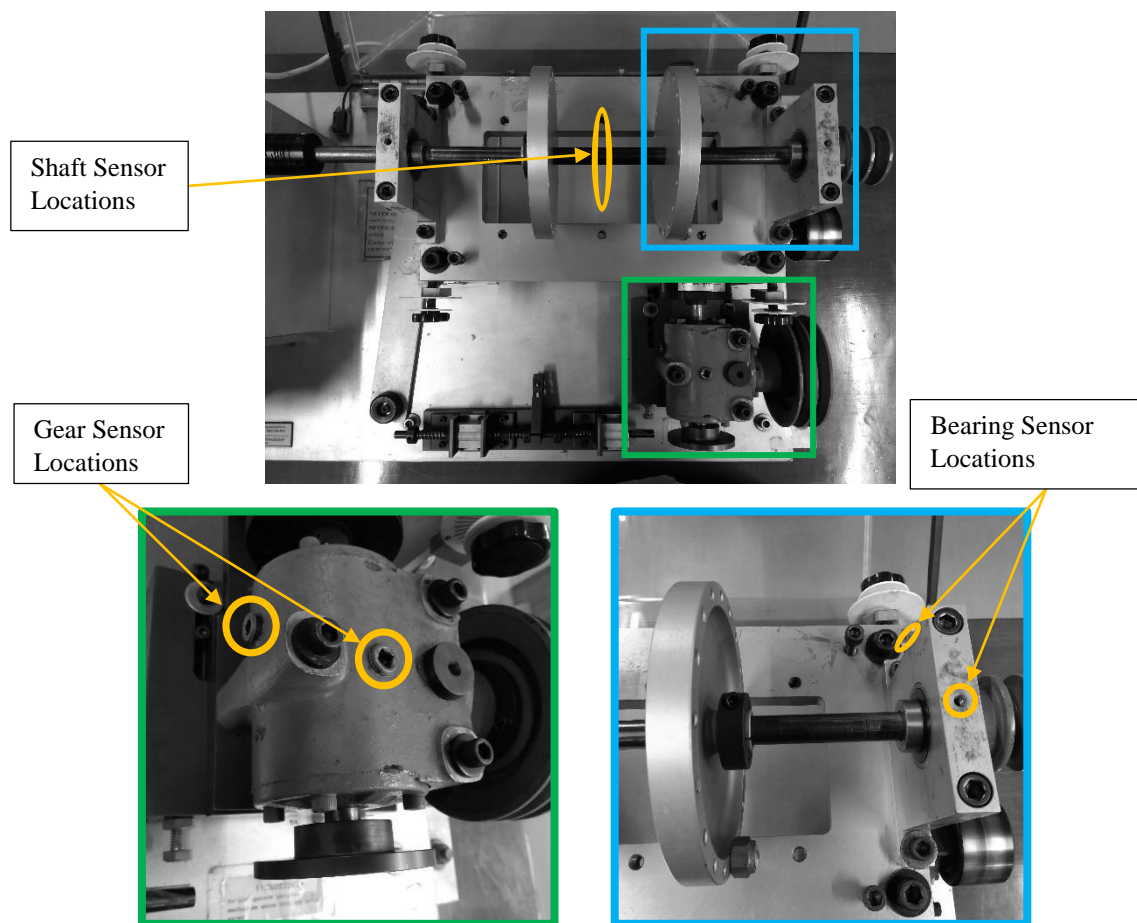


Figure 10: The rotor system (Machine fault Simulator) and sensor locations

Table 4.1: Break up of a typical rotor system into its sub-systems

ROTOR SYSTEM		
<u>ROTATIONAL</u>	<u>STATIONARY</u>	<u>CIRCULATORY</u>
Gearbox [*]	Casing	Oil-Fluid Distribution System
Bearings [*]		
Shaft-Rotor Disc [*]	Foundation	
Motor		
Couplings		

* Sub-systems collecting sensor responses

4.2 Types of Faults & Training Datasets

Variety of faults were introduced by Jasdeep Singh [16] at constant operating frequency of 40 Hz. Typical vibrations sensed for one of the faults introduced in the rotor are shown in Figure 11. The signals are shown in time domain. Twenty such signals were collected for each sensor for the fault introduced. The following faults were introduced in the rotor test rig for creating training data.

<i>No Fault:</i>	A baseline signal for no fault operation is collected.
<i>Unbalance:</i>	An unbalance mass of 6.82 grams was introduced on the right disk.
<i>Eccentric Rotor:</i>	The eccentric disc provided by the manufacturer is used for introducing this fault.
<i>Cocked Rotor:</i>	The right disc was replaced by cocked disc which is bored slightly off center to shaft.
<i>Bearing Defects:</i>	Four type of faults corresponding to bearings were introduced by replacing good with defective one. The faults are Outer Race Defect, Inner Race Effect, Ball Spin Error and Combined Bearing Faults.
<i>Loose Gear:</i>	Good gear was replaced by loose gear for this case.
<i>Missing Tooth:</i>	Pinion with one missing tooth was used for this case.

Two types of training datasets were generated for training purpose. One, with faults restricted to single sub-system and another, with combined faults from different systems. The twelve non-combined faults for dataset (Set A) are described in Table 4.2. Another dataset (Set B) is combination of (Set A) and combined faults. The combined faults have Missing Tooth as the common fault and other faults are - from fault number 5 in Table 4.2 (Loose Gear) to 12 (Combined bearing Fault), making a total of twenty faults as shown in Table 4.3. Set A has a total of 240 and Set B a total of 400 experimental signatures. The datasets are divided as training (60%), validation (20%) and test (20%) datasets.

Raw Data points were mapped between -1 to 1 with following expression.

$$X_{ts} = (X_{ts} - \min_s) / (\max_s - \min_s)$$

where, t represents time [$t = 1, 2, 3 \dots 4096$] and s represents sensor type [$s = 1, 2, \dots, 6$], \min_s and \max_s are minimum and maximum values encountered by sensor s .

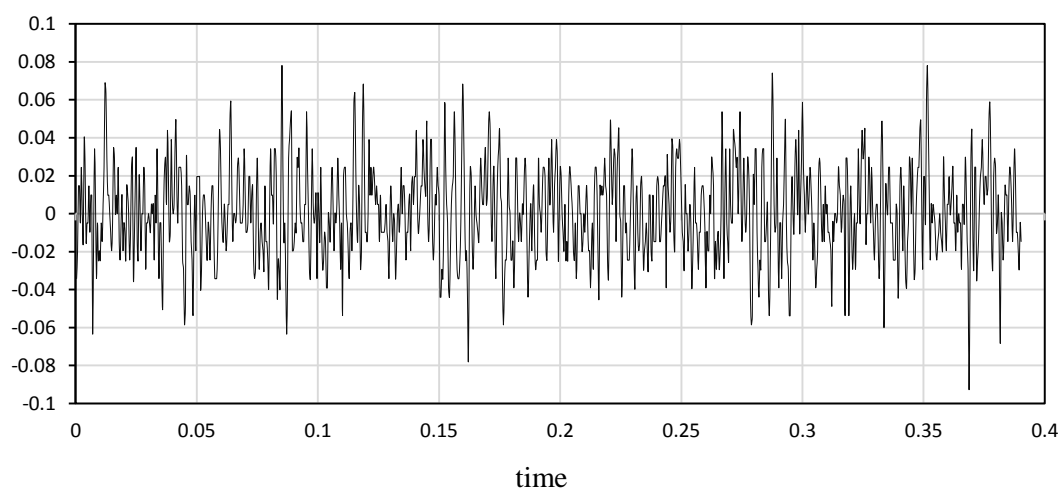
Table 4.2: Type of faults in training dataset (Set A)

#	FAULTS
1	No Fault with Attached Belt
2	Loose Belt
3	Tight Belt
4	Missing Tooth
5	Loose Gear
6	Unbalance 6.82 gram
7	Eccentric Rotor
8	Cocked Rotor
9	Bearing Outer Race Defect
10	Bearing Inner Race Defect
11	Ball Spin Fault
12	Combined Bearing Fault

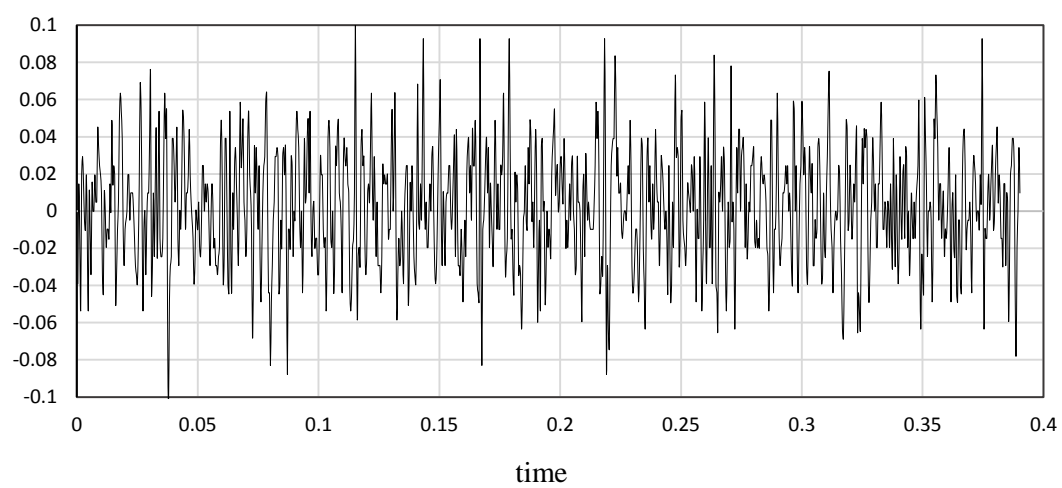
Table 4.3: Type of faults in training dataset (Set B)

#	FAULT	FAULT SOURCE
	<i>Faults From Individual Sub-Systems</i>	
1	No Fault with Attached Belt	No Fault
2	Loose Belt	Belt
3	Tight Belt	
4	Missing Tooth	Gear
5	Loose Gear	
6	Missing Tooth + Loose Gear	
	<i>Combined Faults Along With Missing Tooth</i>	
7	Unbalance 6.82 gram	Shaft + Gear
8	Eccentric Rotor	
9	Cocked Rotor	
10	Bearing Outer Race Defect	Bearing + Gear
11	Bearing Inner Race Defect	
12	Ball Spin Fault	
13	Combined Bearing Fault	
	<i>Faults From Individual Sub-Systems</i>	
14	Unbalance 6.82 gram	Shaft
15	Eccentric Rotor	
16	Cocked Rotor	
17	Bearing Outer Race Defect	Bearing
18	Bearing Inner Race Defect	
19	Ball Spin Fault	
20	Combined Bearing Fault	

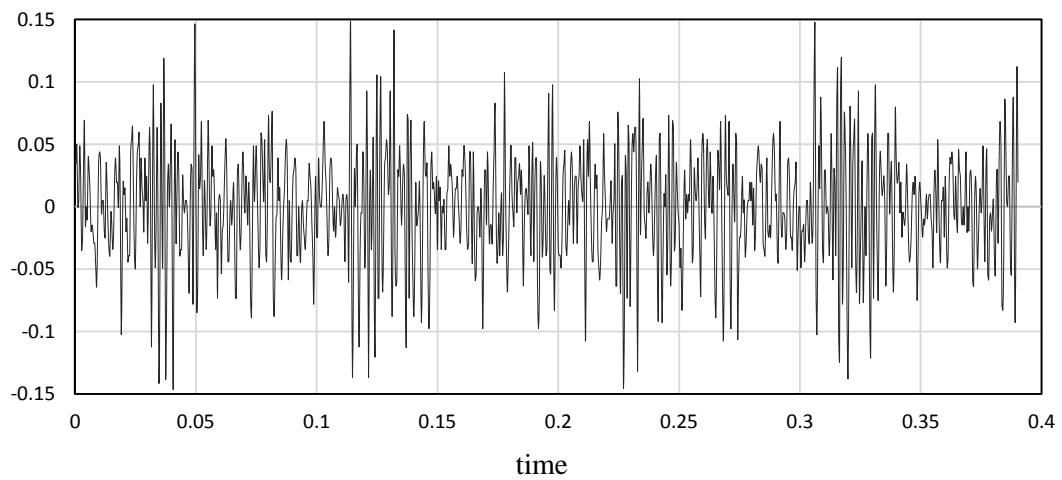
Bearing (Vertical)



Bearing (Horizontal)



Gear (Horizontal)



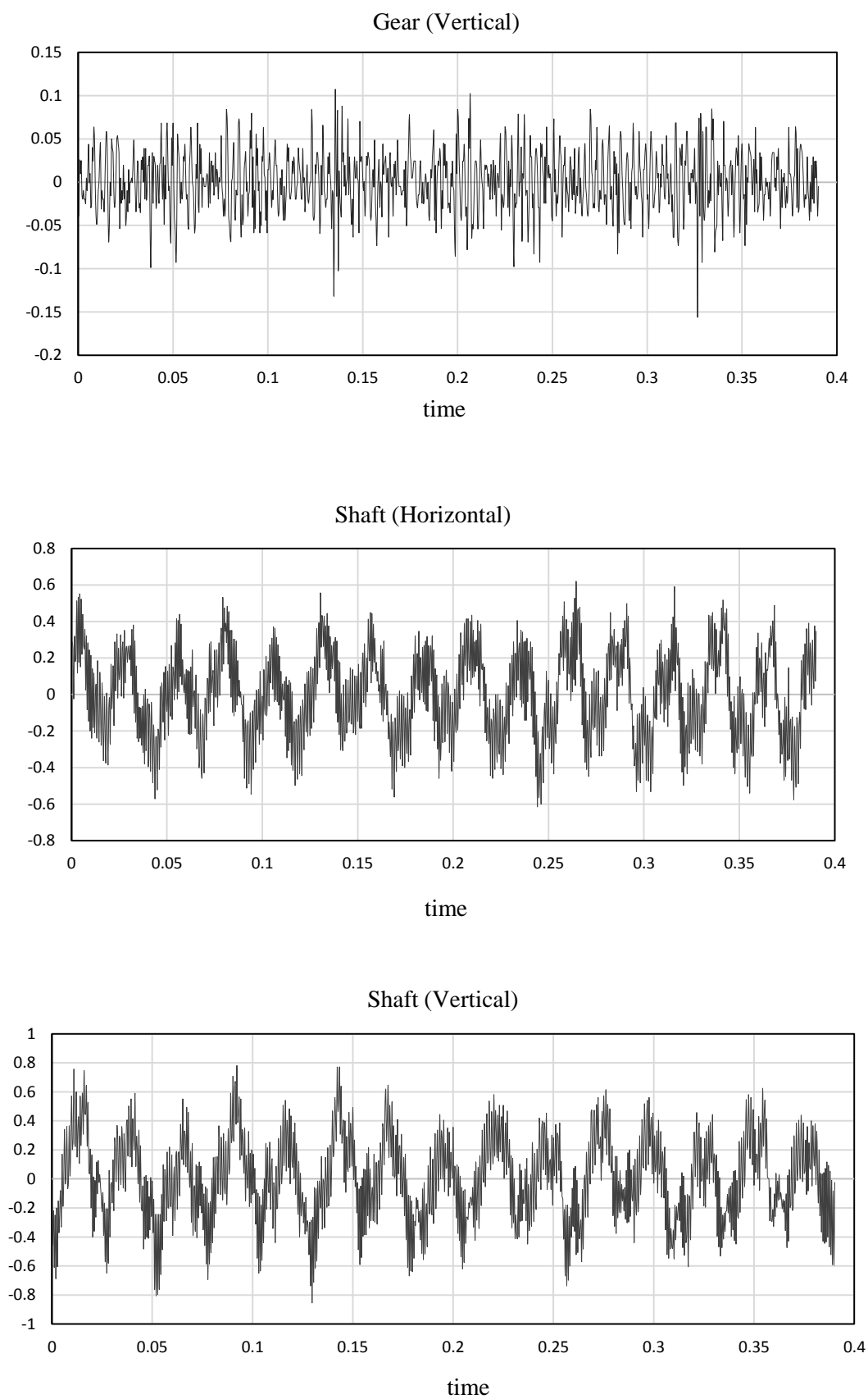


Figure 11: A typical response from sensors for particular fault [16]

CHAPTER 5

TRAINING, VALIDATION AND TESTING

Training of McCNN is performed both on Set A and Set B. Various observations noticed during training are discussed below. The proposed method classified all training, validation and test dataset successfully for both Set A (individual faults) and Set B (combined faults). The training is performed both on time domain and frequency domain response. The results for both are presented below and compared successively.

5.1 Training in Frequency Domain

The raw time-domain data is first converted into frequency domain using Fast Fourier Transform (FFT). The FFT data is used as input to CNN architecture presented in Section 3.6. Training was performed with sensor-responses taken from single sub-system and all sub-systems.

5.1.1 Model with input from a sub-system

The input layer is modified as $4096 \times 2 \times 1$ for two sensor responses from a single sub-system. The subsystems are numbered as

Sub-System 1: Bearing

Sub-System 2: Gear Box

Sub-System 3: Shaft-Rotor Disc

When trained with individual sub-systems the accuracies for (Set A) are as shown in

Table 5.1 and the training loss is shown in Figure 12.

Accuracies for (Set B) for individual sub-system response as input are shown in Table 5.2 and the training loss plots are shown in Figure 13.

Table 5.1: Accuracies for (Set A) trained with individual sub-system in frequency domain

Network Input	Input Layer Topology	% Accuracy		
		Train	Validation	Test
Bearing	$4096 \times 2 \times 1$	100	100	97.92
Gear	$4096 \times 2 \times 1$	100	100	100
Shaft	$4096 \times 2 \times 1$	100	100	100

Table 5.2: Accuracies for (Set B) trained with individual sub-system in frequency domain

Network Input	Input Layer Topology	% Accuracy		
		Train	Validation	Test
Bearing	$4096 \times 2 \times 1$	100	100	98.75
Gear	$4096 \times 2 \times 1$	100	100	100
Shaft	$4096 \times 2 \times 1$	100	100	98.75

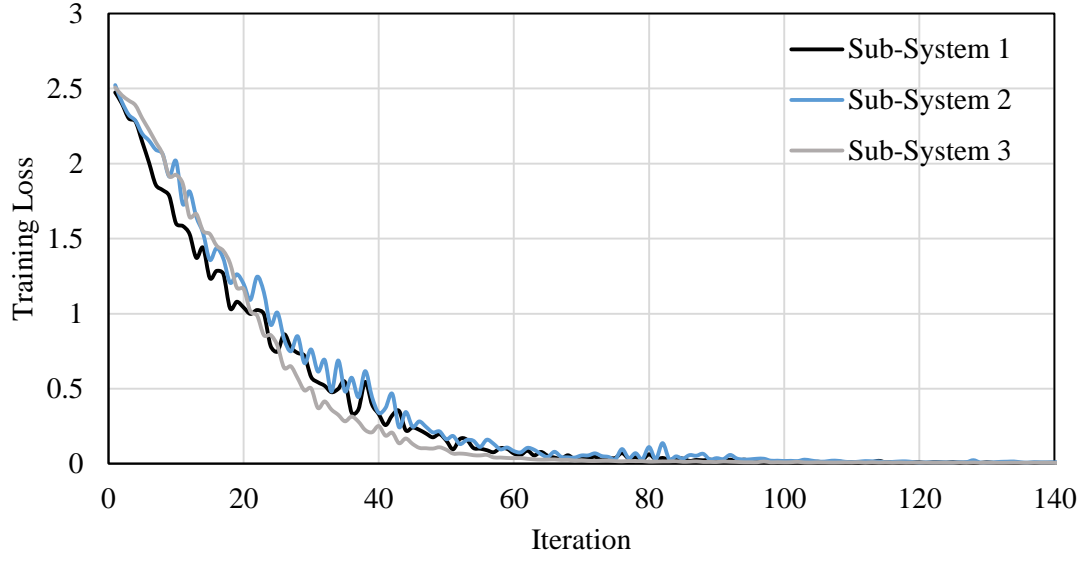


Figure 12: Training loss for (Set A) when input is in frequency domain with Avg. Pooling

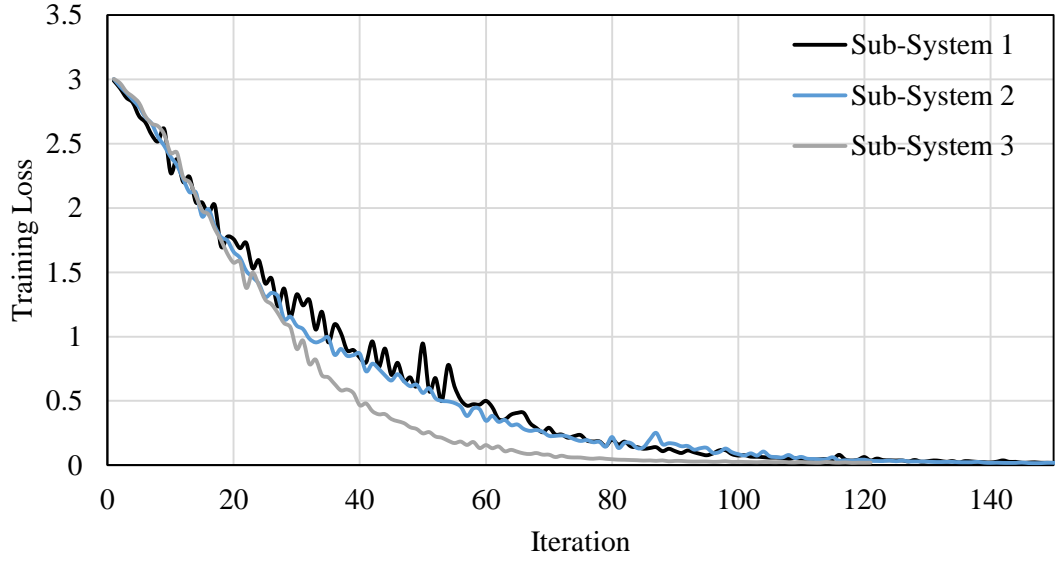


Figure 13: Training loss for (Set B) when input is in frequency domain with Avg. Pooling

5.1.2 Model with input from all sub-systems

The input layer is modified as $4096 \times 2 \times 3$ for two sensor responses from all sub-system (4096 for number of frequencies, 2 is for horizontal & vertical response, 3 for number of sub-systems). The sub-systems are arranged in different channels analogous to RGB channels

of a colored image. When trained with all sub-systems the accuracies for (Set A) are shown in Table 5.3 and for (Set B) are shown in

Table 5.4 for both max. (Maximum) and avg. (average) pooling. When trained for (Set A) training losses for max. and avg. pooling are as shown in Figure 14 and the same for (Set B) is shown in Figure 15.

Table 5.3: Accuracies for (Set A) trained with all sub-systems as input in frequency domain

Pooling Type	Input Layer Topology	% Accuracy		
		Train	Validation	Test
Max. Pooling	$4096 \times 2 \times 3$	100	100	100
Avg. Pooling	$4096 \times 2 \times 3$	100	100	100

Table 5.4: Accuracies for (Set B) trained with all sub-systems as input in frequency domain

Pooling Type	Input Layer Topology	% Accuracy		
		Train	Validation	Test
Max. Pooling	$4096 \times 2 \times 3$	100	100	100
Avg. Pooling	$4096 \times 2 \times 3$	100	100	100

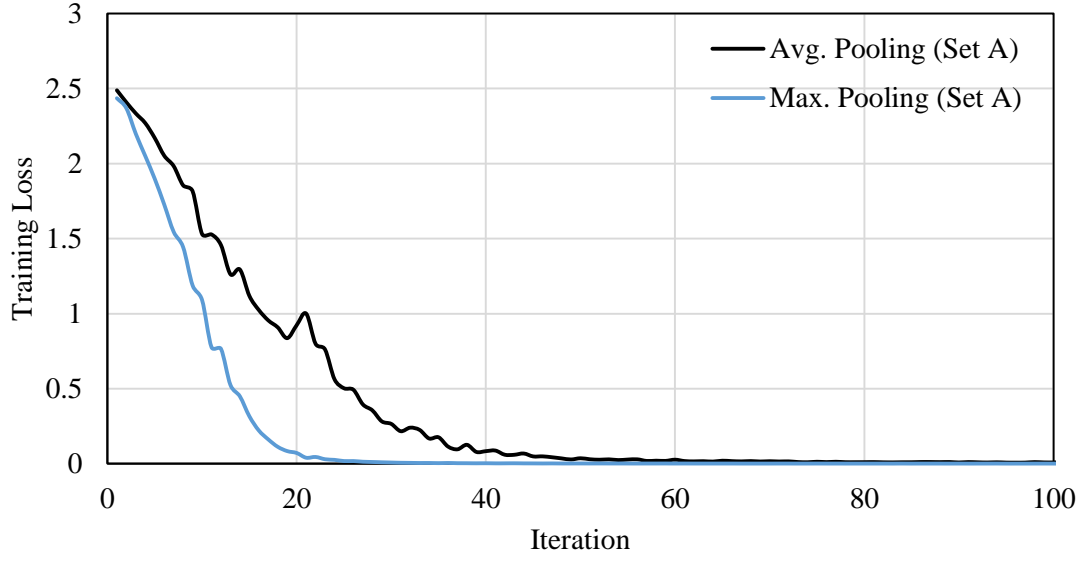


Figure 14: Training loss for (Set A) trained with all sub-systems as input in frequency domain

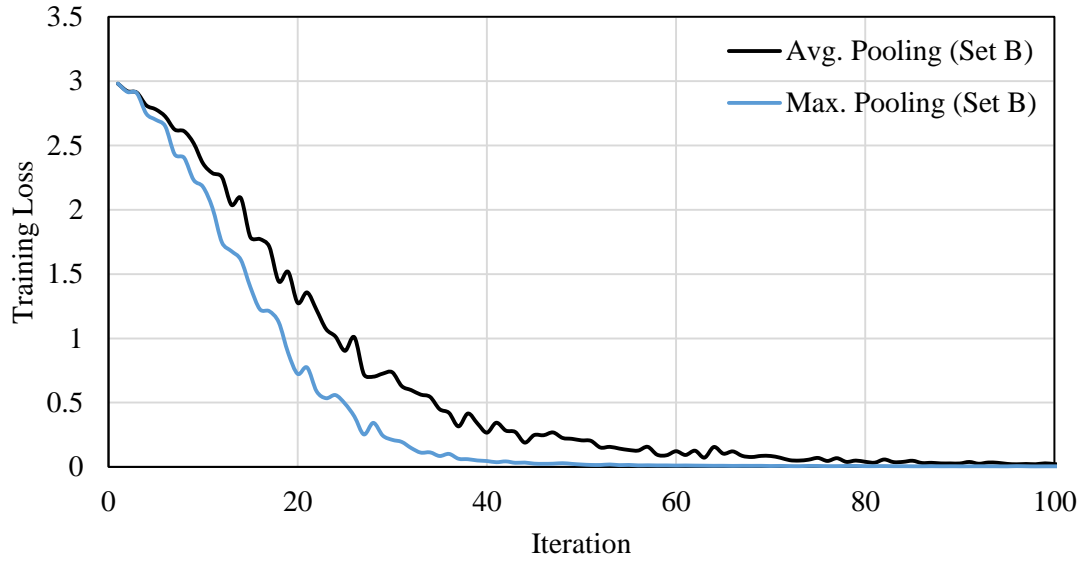


Figure 15: Training loss for (Set B) trained with all sub-systems as input in frequency domain

When feature maps after each ReLU layer are mapped to two dimension by t-SNE dimensionality reduction algorithm, the clusters of different fault data are obtained. The cluster diagram for (Set A) is shown in Figure 16 and for (Set B) in Figure 17.

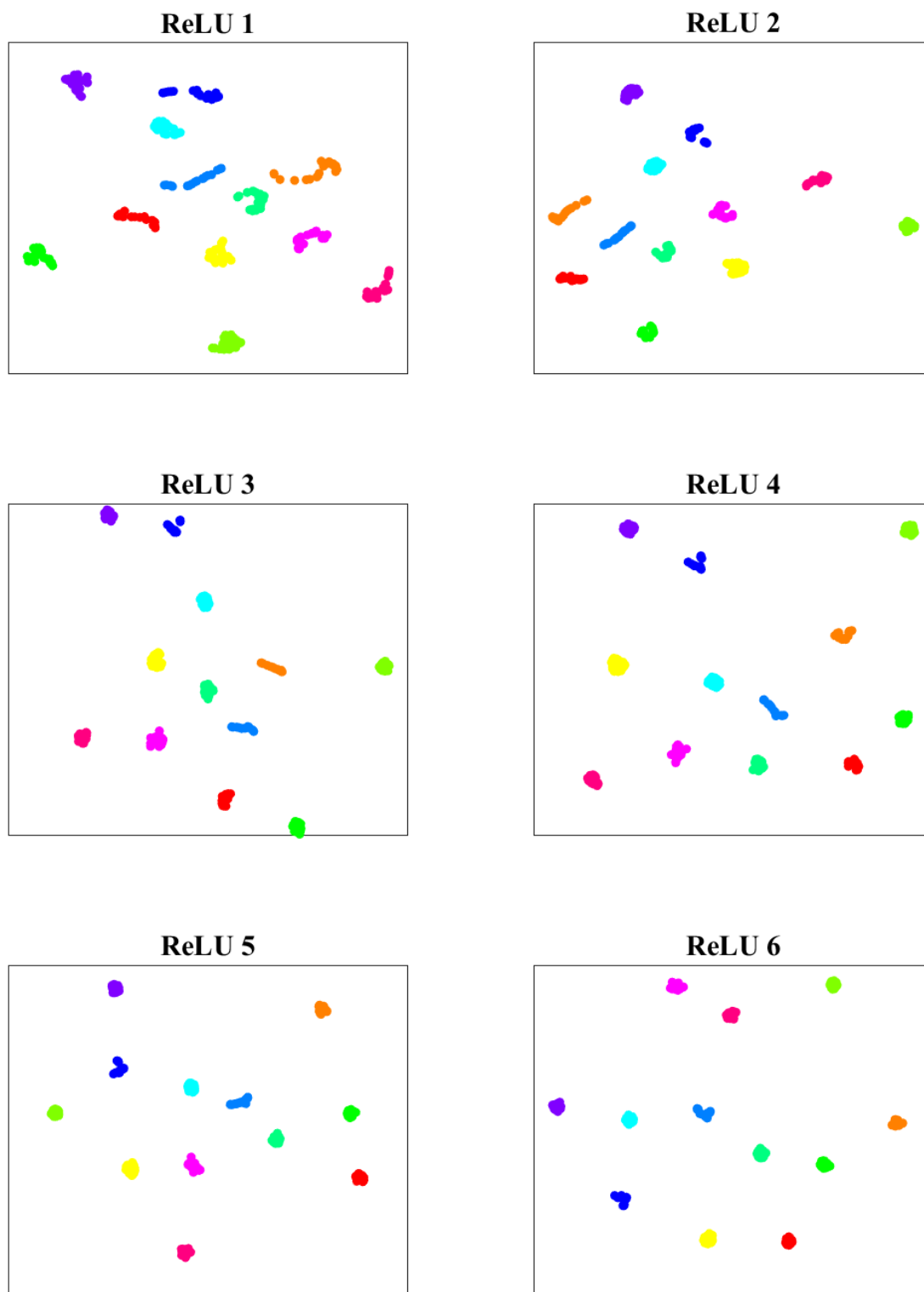


Figure 16: Feature map after dimensionality reduction for (Set A) with Max. Pooling

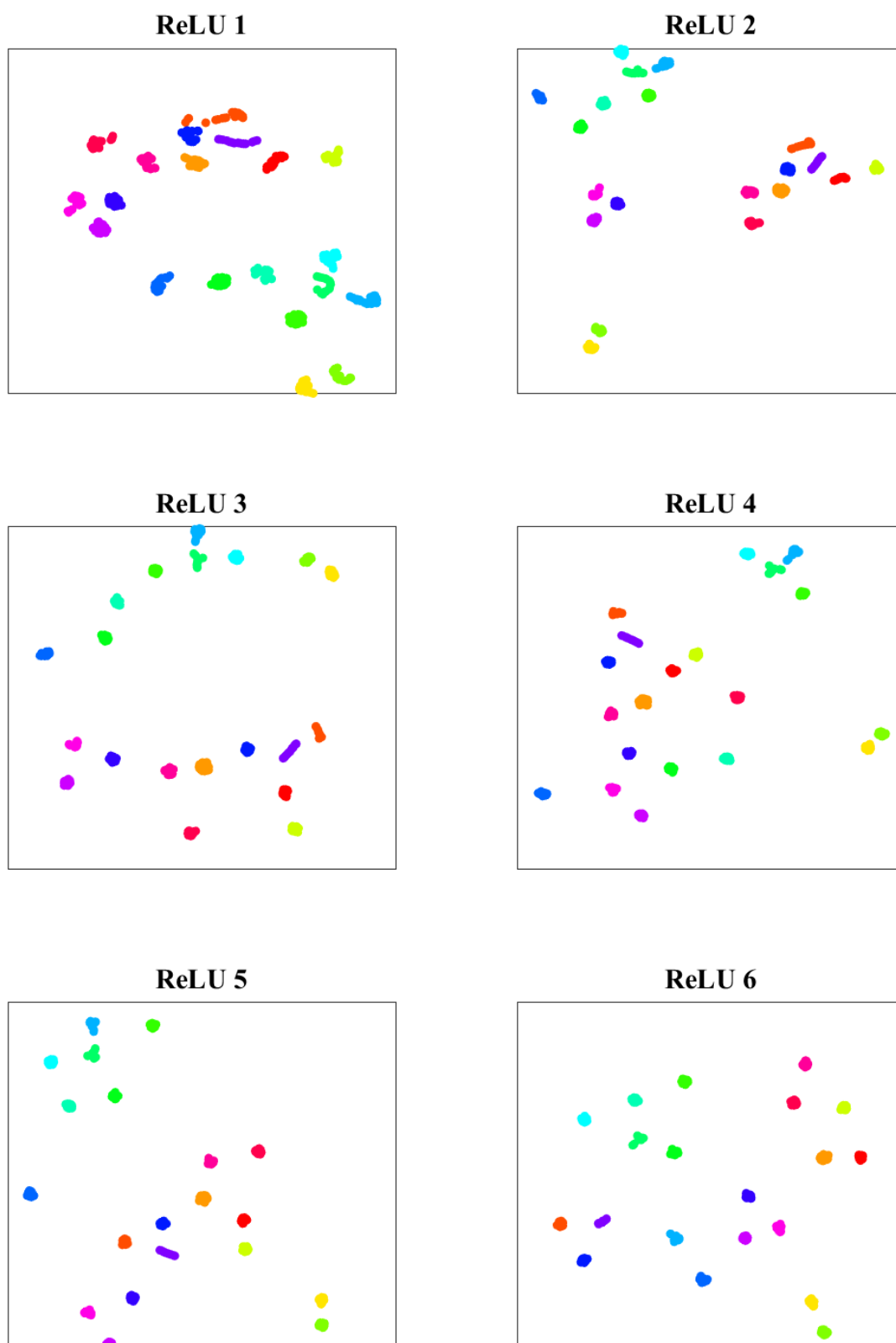


Figure 17: Feature map after dimensionality reduction for (Set B) with Max. Pooling

5.2 Training in Time Domain

The raw time-domain data is directly fed as input to CNN architecture presented in Section 3.6 without any feature engineering. Training was performed with sensor-responses taken from single sub-system and all sub-systems.

5.2.1 Model with input from a sub-system

The input layer is modified as $4096 \times 2 \times 1$, for two sensor responses from a single sub-system. When trained with individual sub-systems, the accuracies for (Set A) are obtained as shown in Table 5.5 and the training loss is as shown in Figure 18. Accuracies for (Set B) for individual sub-system response as input are shown in Table 5.6 and the training loss plots are shown in Figure 19.

Table 5.5: Accuracies for (Set A) trained with individual sub-system in time domain

Network Input	Input Layer Topology	% Accuracy		
		Train	Validation	Test
Bearing	$4096 \times 2 \times 1$	100	100	100
Gear	$4096 \times 2 \times 1$	100	100	100
Shaft	$4096 \times 2 \times 1$	100	100	100

Table 5.6: Accuracies for (Set B) trained with individual sub-system in time domain

Network Input	Input Layer Topology	% Accuracy		
		Train	Validation	Test
Bearing	$4096 \times 2 \times 1$	100	100	100
Gear	$4096 \times 2 \times 1$	100	100	100
Shaft	$4096 \times 2 \times 1$	100	100	100

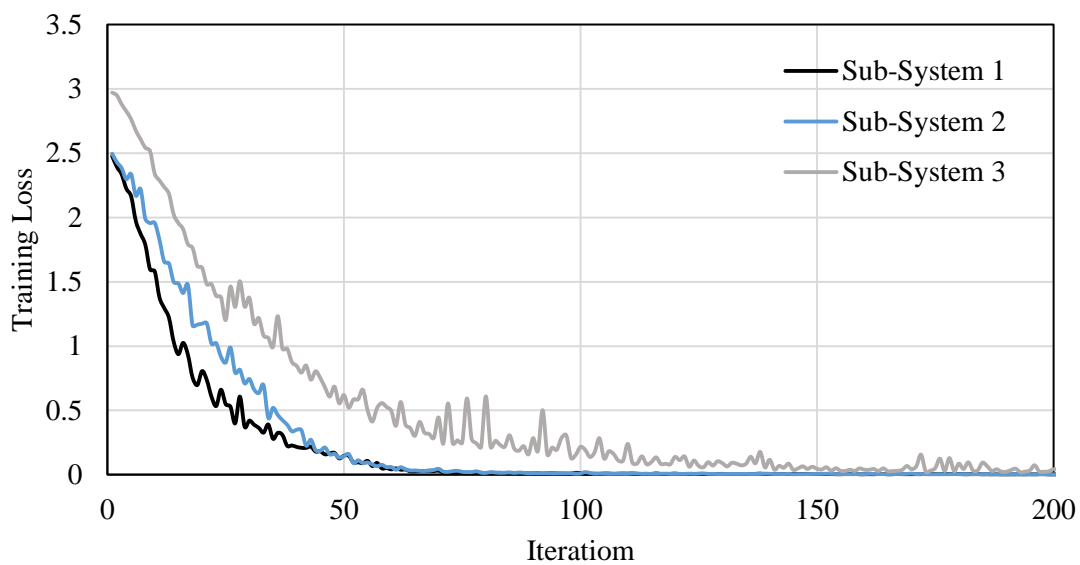


Figure 18: Training loss for (Set A) when input is in time domain with Avg. Pooling

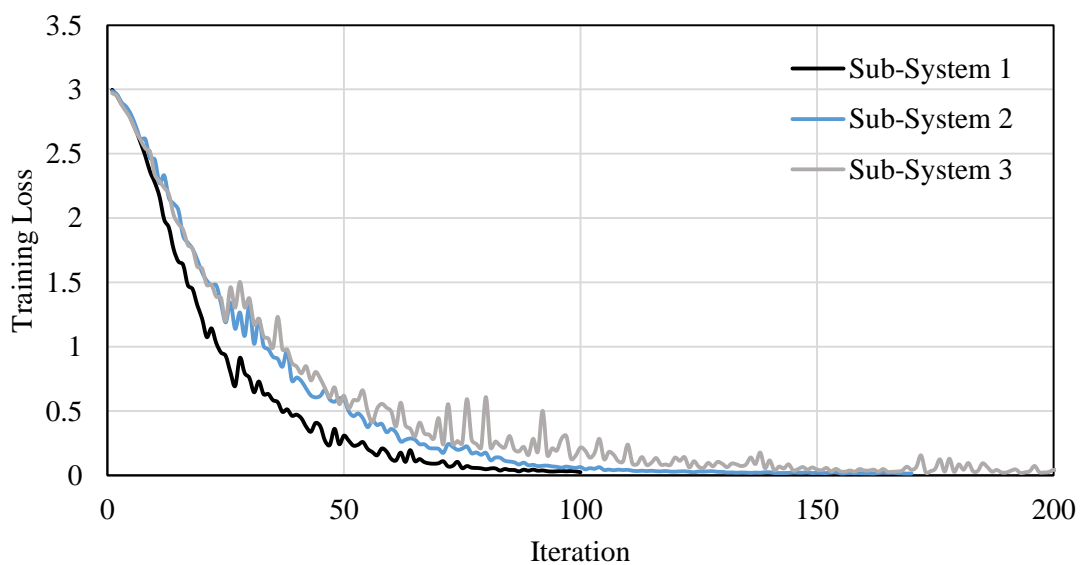


Figure 19: Training loss for (Set B) when input is in time domain with Avg. Pooling

5.2.2 Model with input from all sub-systems

The input layer is modified as $4096 \times 2 \times 3$ for two sensor responses from all sub-systems. The sub-systems are arranged in different channels analogous to RGB channels of a colored image. When trained for (Set A) training losses for max. (Maximum) and avg. (average) pooling are as shown in Figure 20 and the same for (Set B) is shown in Figure 21

When trained with all sub-systems the accuracies obtained for (Set A) are shown in Table 5.7 and those for (Set B) are shown in Table 5.8.

Table 5.7: Accuracies for (Set A) trained with all sub-systems in time domain

Pooling Type	Input Layer Topology	% Accuracy		
		Train	Validation	Test
Max. Pooling	$4096 \times 2 \times 3$	100	100	100
Avg. Pooling	$4096 \times 2 \times 3$	100	100	100

Table 5.8: Accuracies for (Set B) trained with all sub-systems in time domain

Pooling Type	Input Layer Topology	% Accuracy		
		Train	Validation	Test
Max. Pooling	$4096 \times 2 \times 3$	100	100	100
Avg. Pooling	$4096 \times 2 \times 3$	100	100	100

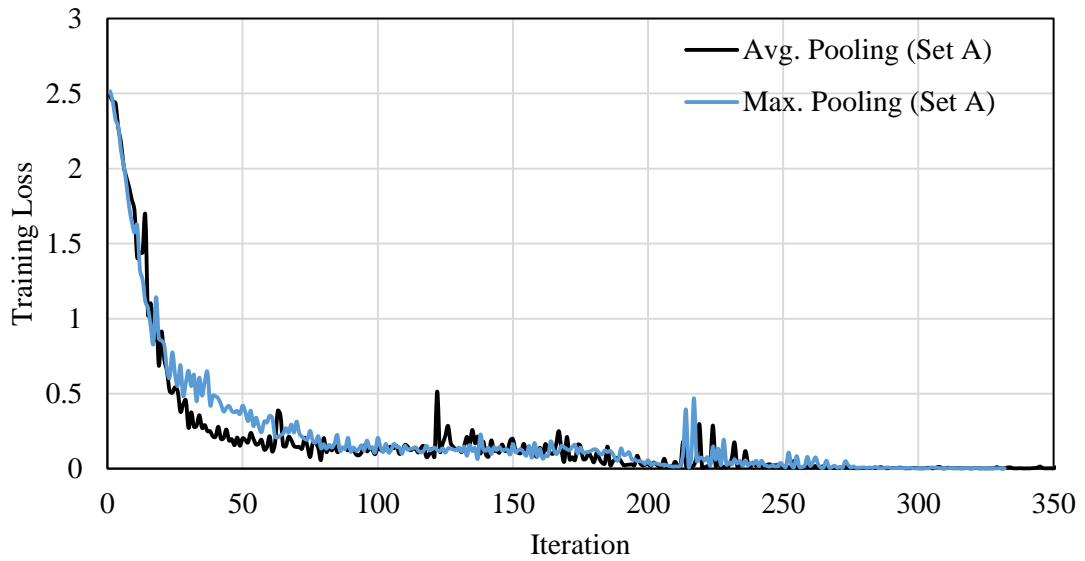


Figure 20: Training loss for (Set A) trained with all sub-systems as input in time domain

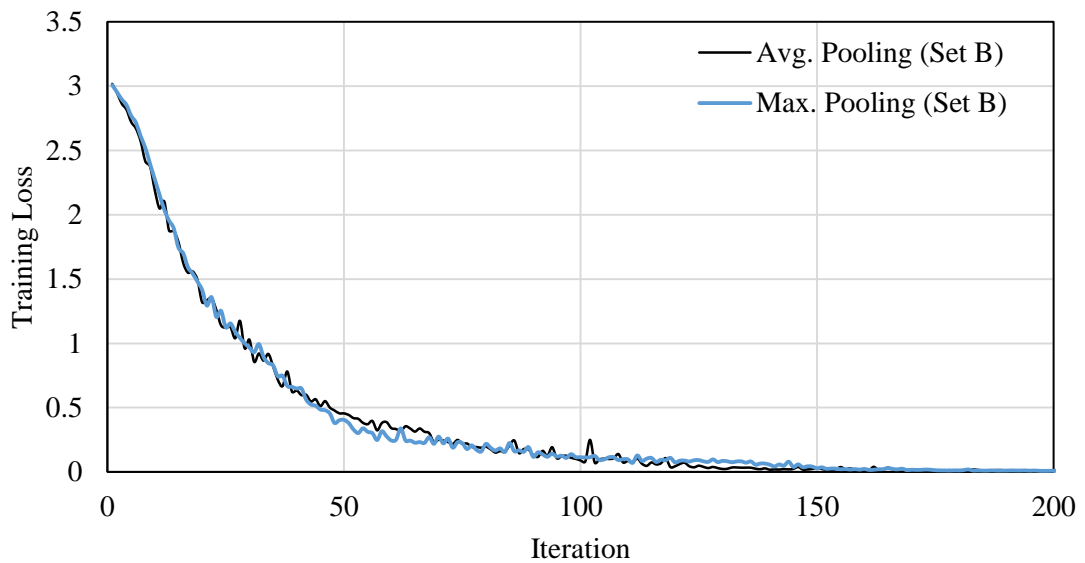


Figure 21: Training loss for (Set B) trained with all sub-systems as input in time domain

After dimensionality reduction of feature maps after each ReLU layer, clusters of different fault data are plotted. The cluster diagram for (Set A) is shown in Figure 22 and for (Set B) in Figure 23.

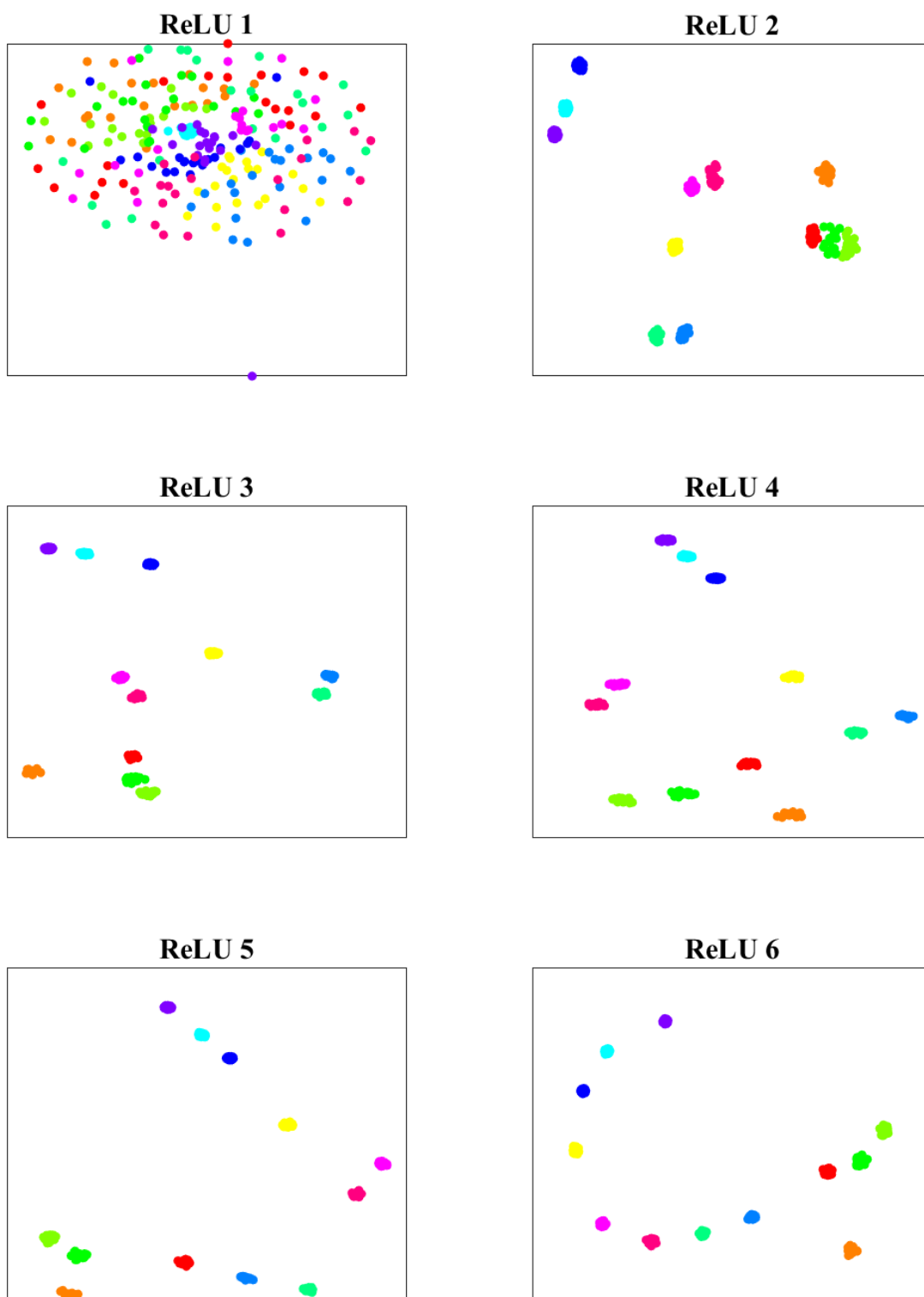


Figure 22: Feature map after dimensionality reduction for (Set A) with Max. Pooling

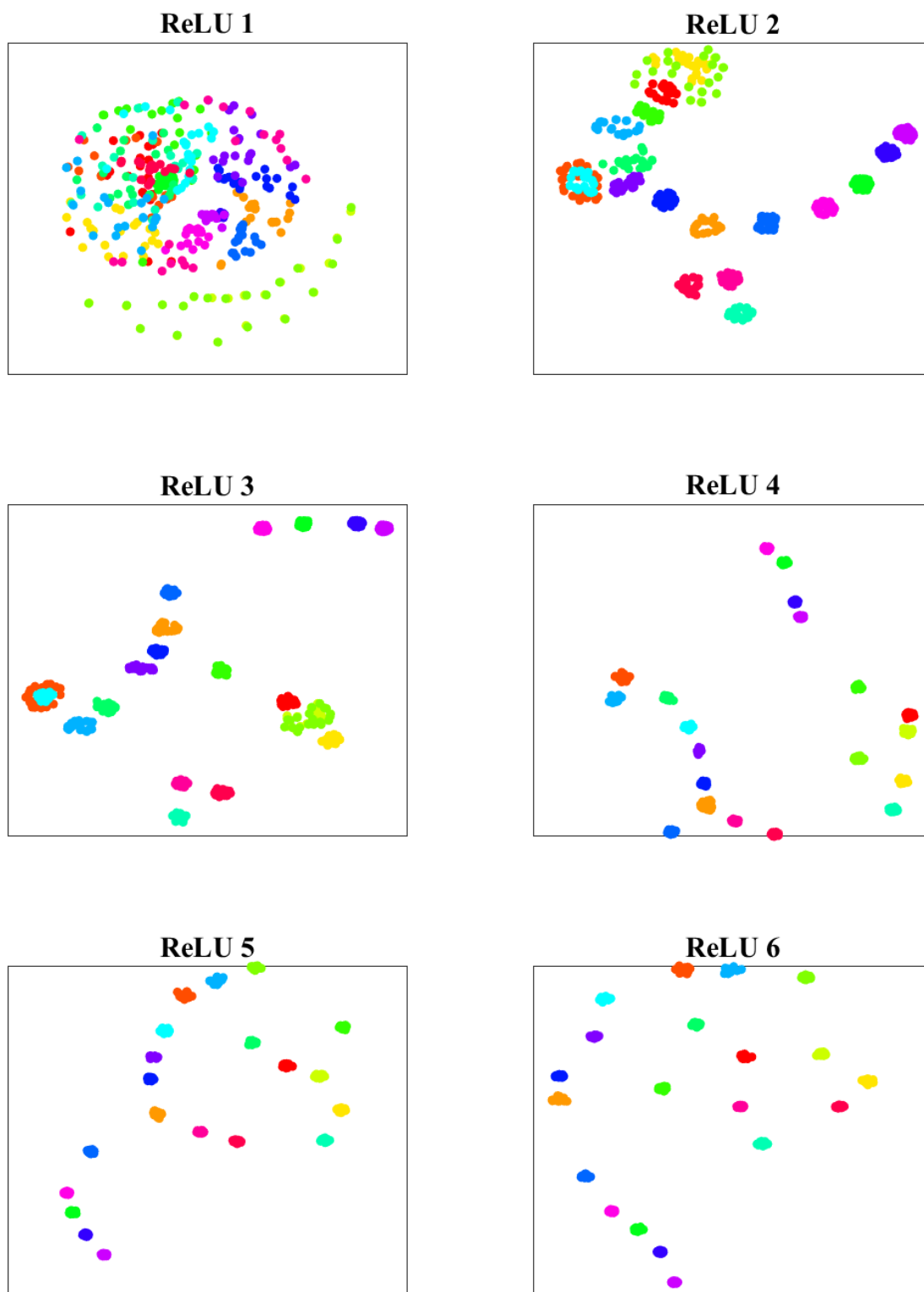


Figure 23: Feature map after dimensionality reduction for (Set B) with Max. Pooling

5.3 Discussion on Frequency Domain Training

When sub-system 1 i.e. bearing sensor response alone is used for training, all faults were not classified in both (Set A) and (Set B). The model employing sub-system 3 i.e. shaft-sensors response as input was not able to classify all faults for (Set B). This is possibly explained by the fact that some fault signals may have damped out before reaching the bearing or shaft.

To study this effect in further detail an experiment is carried out. For 6 combinations of Convolutional, Batch Normalization, ReLU, pooling layers, a CNN network is trained for 800 iterations for same batch size in time domain. The results are shown in

Table 5.9. It clearly shows that individual sub-system responses as input perform poorly in combined faults case unlike all sub-systems as input case. Set B contains more faults and combined faults, which increases complexity of task; hence this effect can be clearly observed in (Set B) compared with (Set A). For a fairly large and complex system, we need sensor response from multiple critical locations. Fault diagnosis for such systems could be performed by adding raw time domain sensor responses of different sub-systems, as different channels, into McCNN input layer.

Table 5.9: Training with input from individual Sub-Systems and all sub-systems combined

Network Input	Input Layer Topology	% Error (Set A)			% Error (Set B)		
		Train	Validation	Test	Train	Validation	Test
Bearing	$4096 \times 2 \times 1$	0	0	0	0	7.5	3.75
Gear	$4096 \times 2 \times 1$	0	0	0	0	1.25	1.25
Shaft	$4096 \times 2 \times 1$	0	0	0	0	0	1.25
All sub-system	$4096 \times 2 \times 3$	0	0	0	0	0	0

When all sub-system responses were used as input, the model successfully classified all faults in both (Set A) and (Set B).

Both pooling function were performing identically in case of individual sub-system input, but max pooling was faster in case of all sub-system as input.

As discussed in Section 1, several efforts to automate feature extraction have been made using deep learning. The feature map obtained from last pooling layer (prior to fully connected layer) was used to train traditional machine learning algorithms and all faults were successfully identified. McCNN is learning information rich features for selected task. This observation is supported by t-SNE Clustering figures, in which features obtained after each ReLU activation layer were reduced to two dimensions using t-SNE dimensionality reduction algorithm. The features at last layer show each fault data points clustered more compact manner and distinctly separate from other faults. The compactness between intrafault and separation between interfault points increases as we move toward output layer, inferring that each layer is assisting next level to learn better representation.

5.4 Discussion on Time Domain Training

All models were capable of classifying faults of both data type successfully. Behavior of pooling functions in both (Set A) and (Set B) was equivalent.

5.5 Frequency Domain v/s Time Domain

Training was faster in frequency domain as compared to time domain, which can be seen from the training loss plots. However, the main objective of training without feature engineering is obtained on small cost of training iterations. Also, FFT coefficients are features, which are well defined for every input node.

5.6 Effect of Depth and Vanishing Gradient

It is evident from the Table 5.10 that accuracy increases as we increase the number of layers, i.e. the depth of network. Deep networks are necessary to learn complex functions or tasks. Average Pooling network required lesser number of layers compared to Max Pooling to achieve zero error on test data.

Table 5.10: Effect of depth and pooling function for model with all sensors responses as input

Network Depth	Set A (Test Accuracy %)		Set B (Test Accuracy %)	
	Max Pooling	Average Pooling	Max Pooling	Average Pooling
Layer [1 to 4] + 23	0	39	5	33
Layer [1 to 7] + 23	50	29	23	16
Layer [1 to 10] + 23	81	95	97	92
Layer [1 to 13] + 23	95	100	98	98
Layer [1 to 16] + 23	100	100	98	100
Layer [1 to 19] + 23	-	-	100	100

It can be readily seen from Figure 24 and Figure 25 that traditional Neural Network shows the effect of vanishing gradient, while in McCNN the weight updation of layers distant from output layer are of nearly same order compare to proximate layers. Complex systems can be diagnosed in McCNN with significant depth without facing vanishing gradient.

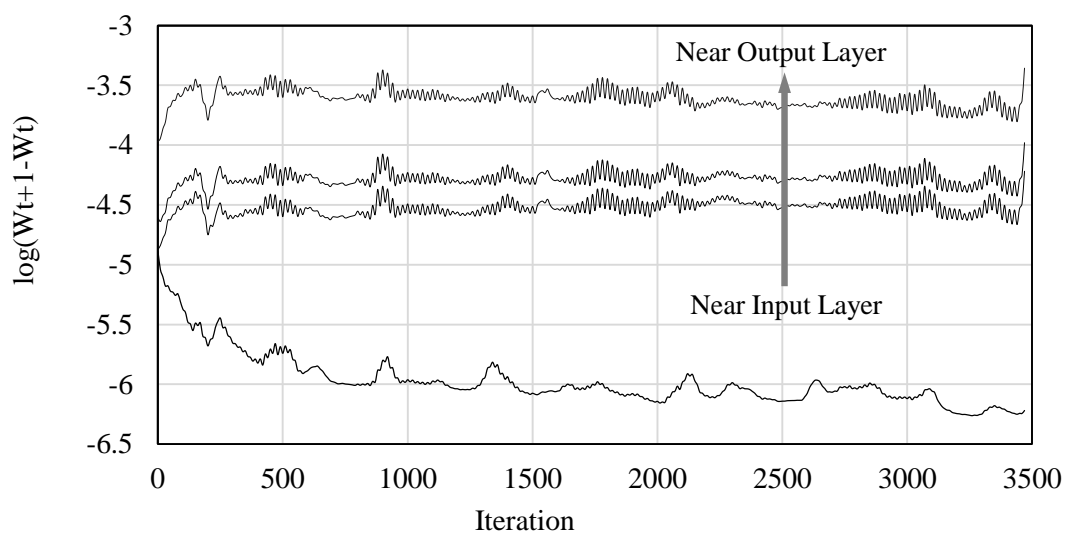


Figure 24: Weight Update for Neural Network

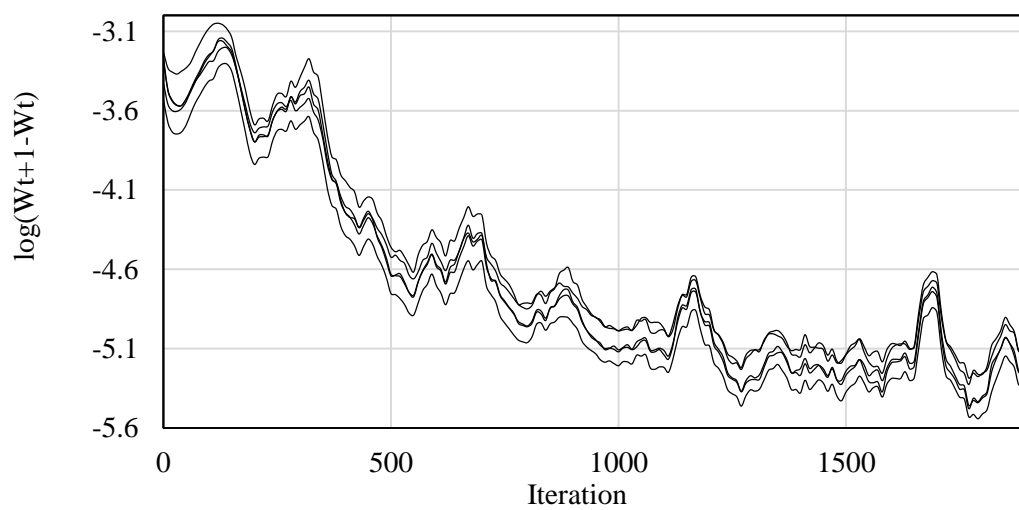


Figure 25: Weight Update for CNN

CHAPTER 6

VALIDATION WITH OPEN SOURCE DATA

The trained model is validated with an open source experimental data, also. The data is available at Case Western Reserve University Bearing Data Center (BDC) website [17]. The experiments are performed for collecting data of both normal and faulty ball bearings. Faults, ranging from 0.007 inches to 0.040 inches in diameter are generated by electro-discharge machining. Faults are introduced in inner raceway, rolling element (i.e. ball) and outer raceway. Motor loads of 0 to 3 horsepower (motor speeds of 1797 to 1720 rpm) are used to collect data. Accelerometers are used to collect data, which were attached to housing with magnetic bases. The sensors are located at both drive end and fan end of motor housing. The data from accelerometer is collected at 12000 samples per second. The setup for experiment is shown in Figure 26. A sample data from both sensors in time and FFT domain is given in Appendix.

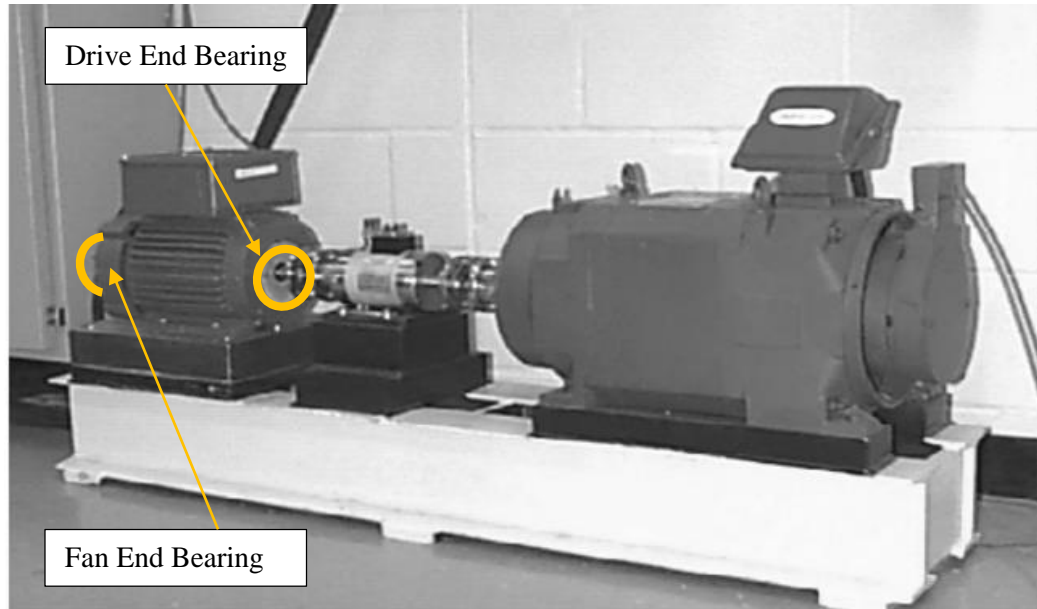


Figure 26: Setup used for experimentation by [17]

6.1 Data Description and Division

The data structure is as shown in Table 6.1. There are 3 types of faults, namely ball fault, inner race fault and outer race fault. All 3 types of faults are introduced at both fault locations i.e. drive end and fan end.

First objective was chosen as, to classify these faults along with baseline condition. Each data file corresponding to the Table 6.1 has certain number of sample points. Sample points equivalent to 10 seconds are considered for the training and testing. There are 78 files or experiments considered for validating the model. The test dataset constitutes 30% of the total data. The remained 70% dataset is further divided as - 80% for training and 20% for validation. Each data file consisting of 10-second data, is further split into 5 files, consisting of 2-second data, in each i.e. 24000 sample points in time domain. This can be labelled as data Set C.

The second objective is to check motor speed invariance of McCNN. The data of 1797 rpm motor speed or zero load is chosen as test data. The remaining data for 1772 rpm, 1750 rpm and 1730 rpm for motor load 1, 2, 3 HP respectively, is further divided into 80% for training and 20% for validation. This can be labelled as data Set D.

Table 6.1: Structure of data downloaded from [17]

Fault Location	Fault Type	Motor Speed (rpm)	Fault Dia. (milli inch)
No Fault (Baseline)	-	1797*	-
		1772*	
		1750*	
		1730*	

Drive End Fault	Ball Fault	1797	(7, 14, 21, 28) *
		1772	(7, 14, 21, 28) *
		1750	(7, 14, 21, 28) *
		1730	(7, 14, 21, 28) *
	Inner Race Fault	1797	(7, 14, 21, 28) *
		1772	(7, 14, 21, 28) *
		1750	(7, 14, 21, 28) *
		1730	(7, 14, 21, 28) *
	Outer Race fault	1797	(7, 14, 21) *
		1772	(7, 14, 21) *
		1750	(7, 14, 21) *
		1730	(7, 14, 21) *

Fan End Fault	Ball Fault	1797	(7, 14, 21) *
		1772	(7, 14, 21) *
		1750	(7, 14, 21) *
		1730	(7, 14, 21) *
	Inner Race Fault	1797	(7, 14, 21) *
		1772	(7, 14, 21) *
		1750	(7, 14, 21) *
		1730	(7, 14, 21) *
	Outer Race fault	1797	(7, 14, 21) *
		1772	7*
		1750	7*
		1730	7*

* Represents a data file downloaded from Bearing Data Center (total 78 files)

6.2 Network Training and Results for Set C

The network used in Section 3.6 is used for the training, with following modification according to system needs:

- *Input layer topology is changed to $(24000 \times 1 \times 2)$ according to $(\text{Sample points} \times \text{sensors per sub-system} \times \text{number of sub-systems})$.*
- *The output layer has four neurons for three fault conditions and one no fault condition.*
- *Mini-Batch size is changed to 115 or 55 (mini-batch size around half or one fourth the size of training data has shown good results throughout study) depending upon good results.*

Training for both average and max pooling is performed. Both frequency domain and time domain inputs are considered while training.

6.2.1 Training in Frequency Domain

The raw time-domain data is first converted into frequency domain using Fast Fourier Transform (FFT). The FFT data is used as input to CNN architecture presented in Section 3.6 with modifications described above. Training was performed with sensor-responses taken from all sub-systems i.e. drive end bearing and fan end bearing.

The accuracies obtained for (Set C) are shown in Table 6.2. The training loss is shown in Figure 27. Both max. and avg. pooling is used for comparison purpose. The clusters of feature maps after each ReLU layer, when dimensionally reduced to 2D, using t-SNE dimensionality reduction technique, yield the information shown in Figure 29.

6.2.2 Training in Time Domain

The raw time domain data is directly fed to input layer without feature engineering. The accuracies for (Set C) are shown in Table 6.3. The training loss is shown in Figure 28. The clusters of feature maps after each ReLU layer when dimensionally reduced to 2D using t-SNE dimensionality reduction technique, is visualized in Figure 30.

Table 6.2: Accuracies for (Set C) trained with all sensors in frequency domain

Pooling	Train Accuracy (%)	Validation Accuracy (%)	Test Accuracy (%)
Max Pooling	100	100	100
Average Pooling	100	100	100

Table 6.3: Accuracies for (Set C) trained with all sensors in time domain

Pooling	Train Accuracy (%)	Validation Accuracy (%)	Test Accuracy (%)
Max Pooling	100	100	99.13
Average Pooling	100	100	96.52

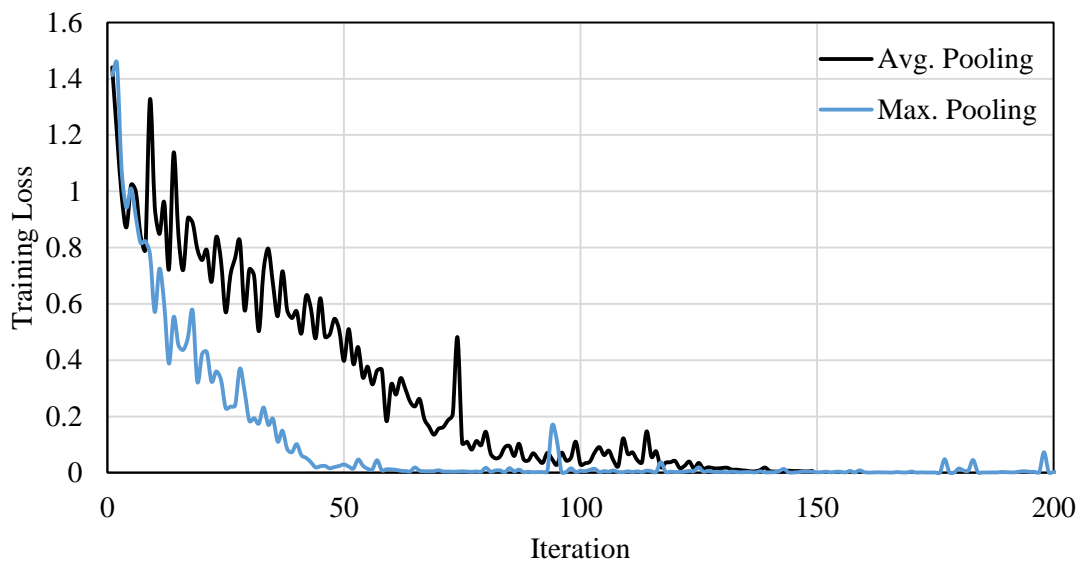


Figure 27: Training loss for (Set C) trained with all sensors as input in frequency domain

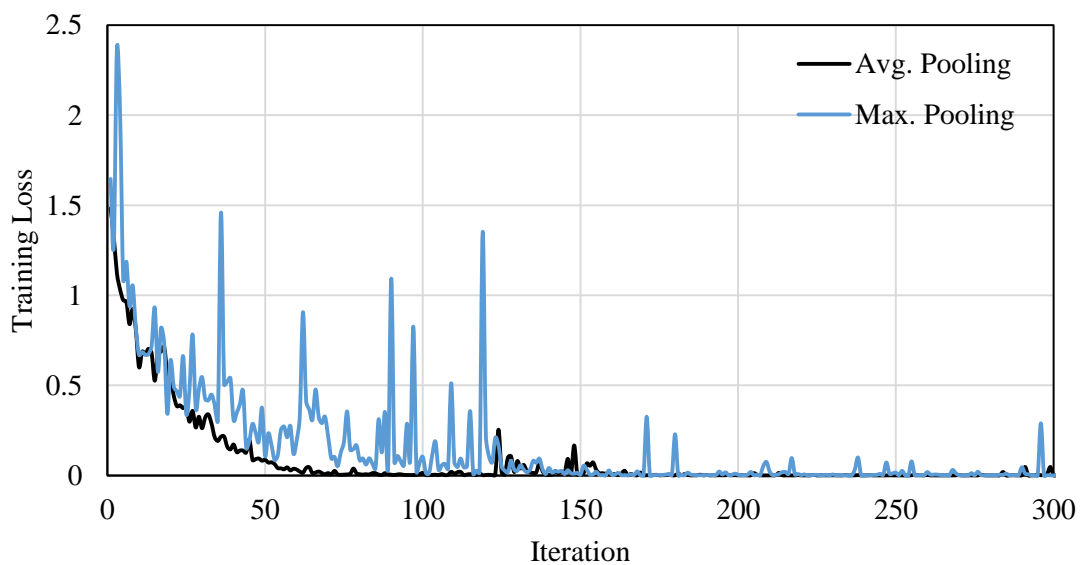


Figure 28: Training loss for (Set C) trained with all sensors as input in time domain

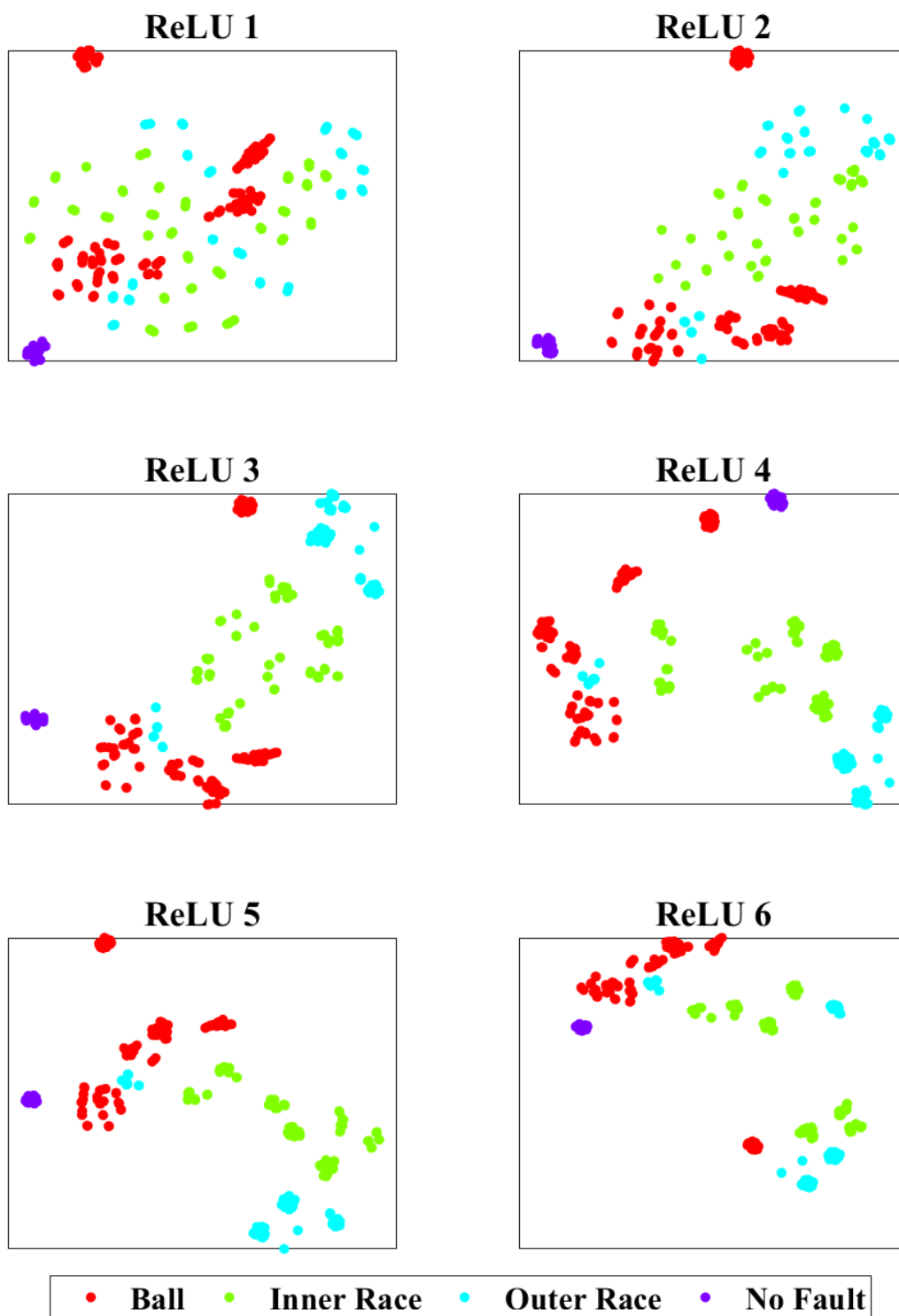


Figure 29: Feature map after dimensionality reduction for (Set C) with Max. Pooling (FFT)

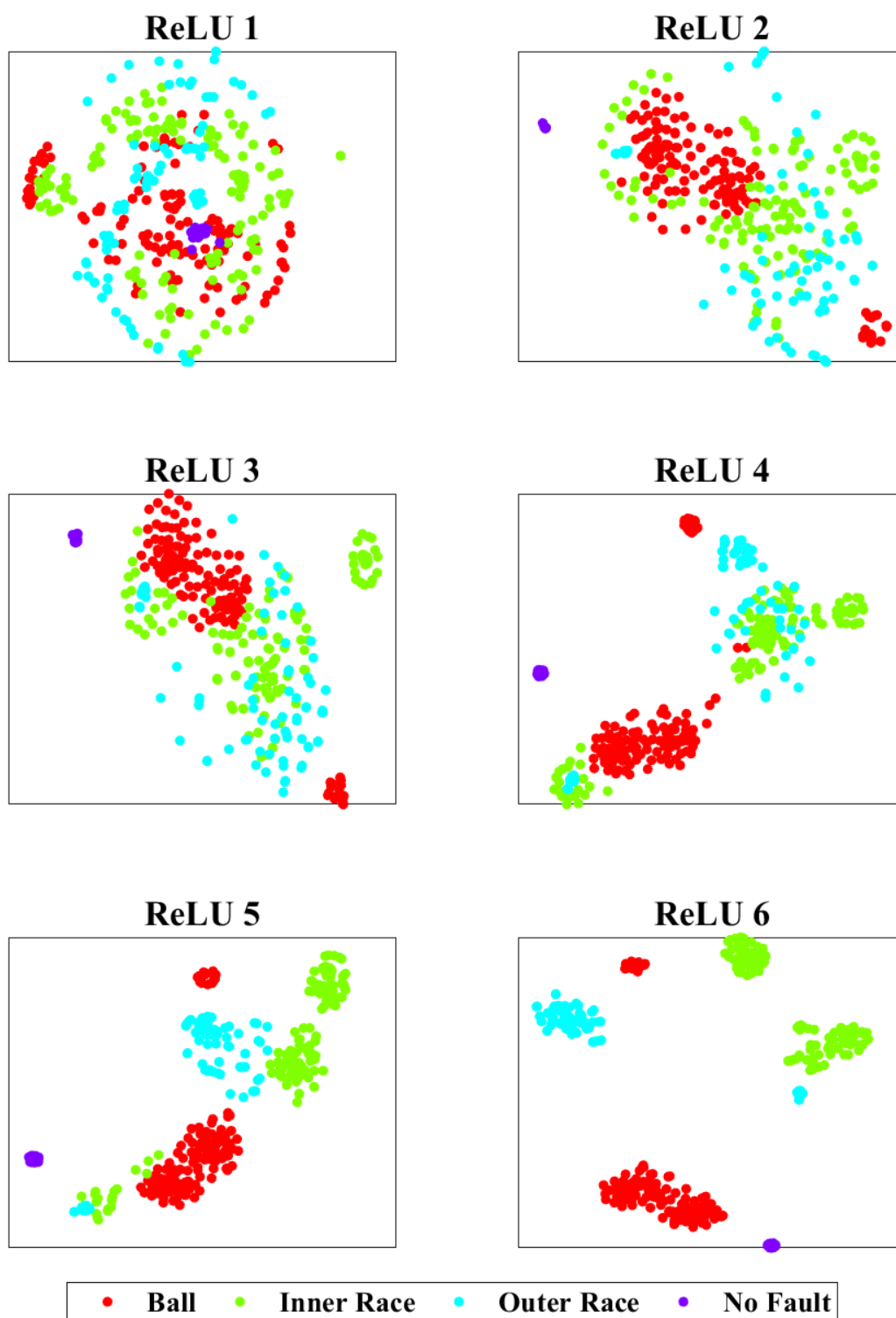


Figure 30: Feature map after dimensionality reduction for (Set C) with Max. Pooling (Time)

6.3 Network Training and Results for Set D

Same modifications as given in Section 6.2 are made to network and the network is trained for both frequency domain and time domain input.

6.3.1 Training in Frequency Domain

The raw time-domain data is first converted into frequency domain using Fast Fourier Transform (FFT). The accuracies for (Set D) are shown in Table 6.4. The training loss is shown in Figure 31. Both max. and avg. pooling is used for comparison purpose. The clusters of feature maps after each ReLU layer when dimensionally reduced to 2D using t-SNE dimensionality reduction technique, is visualized in Figure 33.

6.3.2 Training in Time Domain

The raw time domain data is directly fed to input layer. The accuracies for (Set D) are shown in Table 6.5. The training loss is shown in Figure 32. The clusters of feature maps after each ReLU layer when dimensionally reduced to 2D using t-SNE dimensionality reduction technique, is visualized in Figure 34.

Table 6.4: Accuracies for (Set D) trained with all sensors in frequency domain

Pooling	Train Accuracy (%)	Validation Accuracy (%)	Test Accuracy (%)
Max Pooling	100	100	84.76
Average Pooling	100	100	95.24

Table 6.5: Accuracies for (Set D) trained with all sensors in time domain

Pooling	Train Accuracy (%)	Validation Accuracy (%)	Test Accuracy (%)
Max Pooling	100	100	98.1
Average Pooling	100	100	91.43

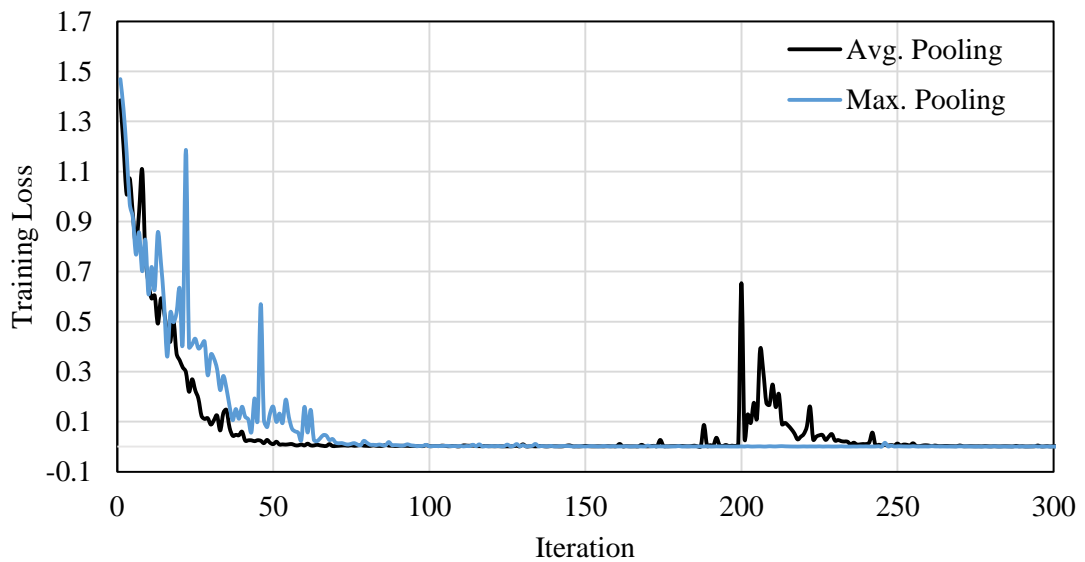


Figure 31: Training loss for (Set D) trained with all sensors as input in frequency domain

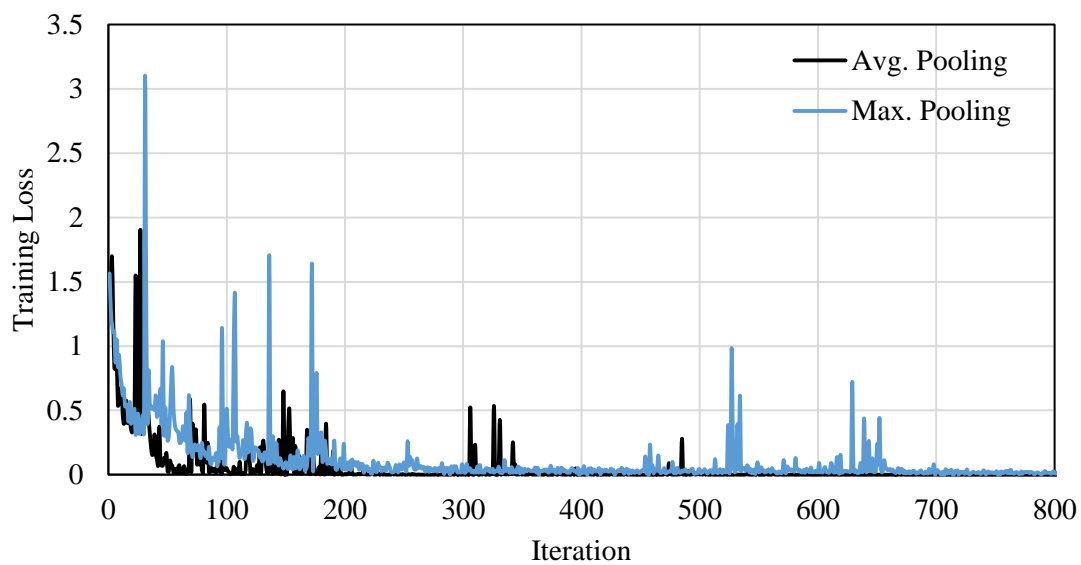


Figure 32: Training loss for (Set D) trained with all sensors as input in time domain

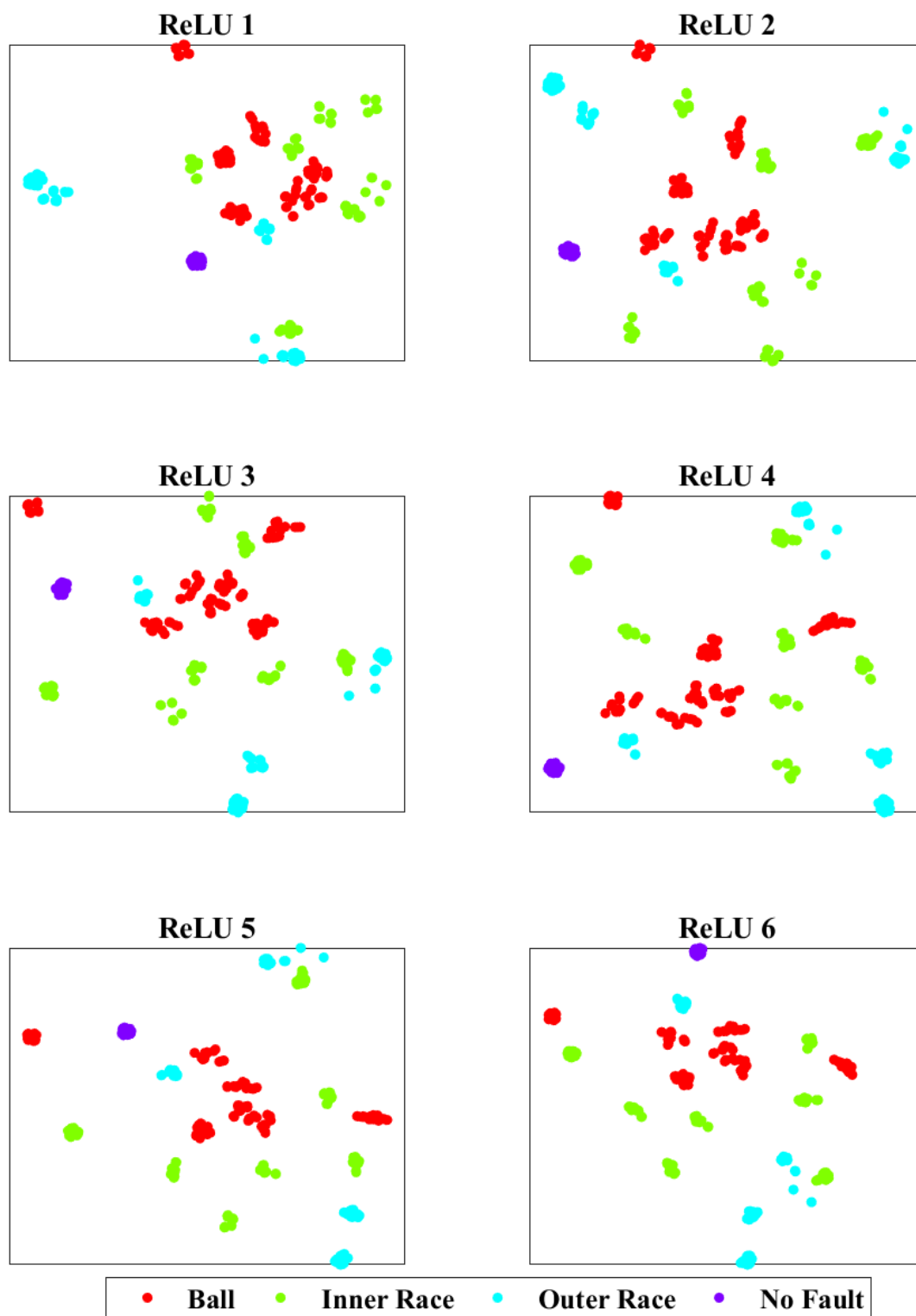


Figure 33 Feature map after dimensionality reduction for (Set D) with Avg. Pooling (FFT)

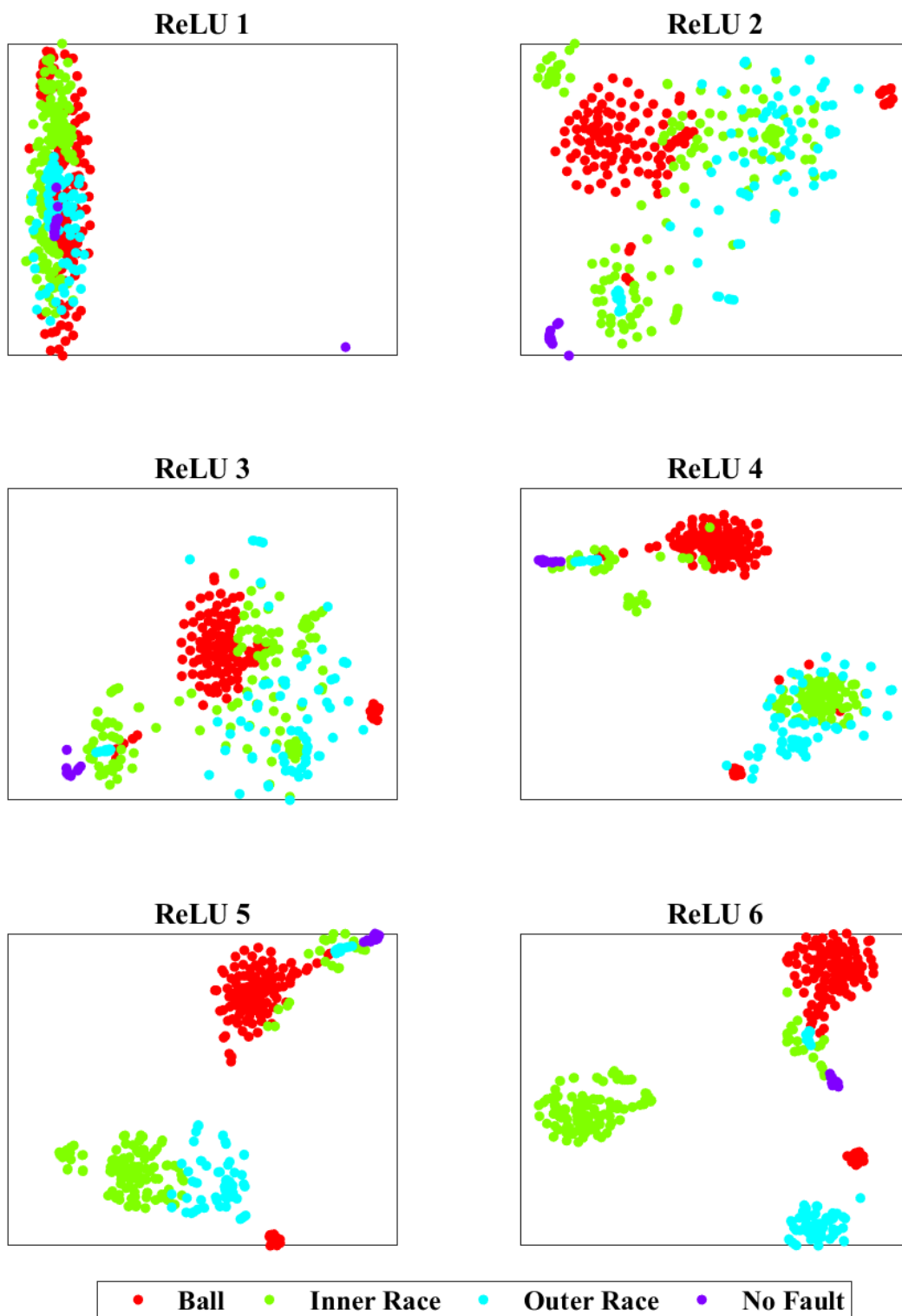


Figure 34: Feature map after dimensionality reduction for (Set D) with Max. Pooling (Time)

6.4 Discussion on results of Set C

The proposed method was capable of classifying particular type of fault (say ball fault) independent of its location (drive end or fan end), independent of motor speed (1797, 1772, 1750, 1730 rpm) and independent of intensity of fault (fault dia. 7, 14, 21, 28 milli inches) satisfactorily. This results shows robustness of model to fulfill the task of fault classification.

- FFT domain input gives good result compare to time domain input.
- Max. Pooling is faster for FFT domain and avg. pooling is good for time domain input
- Frequency domain as input model's feature map clusters are not distant and compact compare to time domain as input model's feature map cluster.

6.5 Discussion on results of Set D

The motor speed invariance is proved to be true after getting satisfactory results from training of Set D. Model trained with 3 motor speeds and tested with fourth motor speed gave good results.

- Time domain as input model with max. pooling gave best accuracy.
- FFT domain input model results good with avg. pooling and time domain input model results good with max. pooling.
- Max. pooling is faster for FFT domain and avg. pooling is faster for time domain input
- Frequency domain as input model's feature map clusters are not distant and compact compare to time domain as input model's feature map cluster.

6.6 FFT domain v/s Time domain input

Time domain model gives clusters more compact and distinct results, while FFT domain input model gave smaller clusters, larger than the number of classes. This is due to the fact that when time domain data obtained from the sensors is processed to obtain FFTs, the Fourier Transform algorithm itself introduces computational inaccuracies (i.e. computational noise), which gets reflected in the final fault classification results.

CHAPTER 7

CONCLUSION

Convolutional Neural Network (CNN) has been developed in this study for fault identification of a rotor-bearing system using. The need to extract features like statistical moments or FFTs for training, is eliminated in the present study. Raw time domain sensor response is used without any preprocessing or feature engineering for constructing input layer. A universal method of fault diagnosis for big complex systems with distant and multiple subsystems is presented, which merges sensor responses from demanding subsystems into distinct channels of input matrix to form the input layer of Multi-channel CNN (McCNN) architecture. Patterns in raw time domain data which are extremely difficult to visualize are perceived by the McCNN network. Manual extraction features results in losing valuable raw response information, which is avoided in the present study.

The proposed model is also validated with data from an open-source available online. Results from validation show invariance of the proposed model for operating speed of rotor system. Also, the capability of CNN to extract abstract features is witnessed. Time domain raw data gives better fault classification in comparison to frequency domain data.

In our study combined faults are treated as completely different category of fault. This has been a limitation. The output of softmax function gives the probabilities of presence of individual faults. While, these probabilities are very high in the case of single faults, for a combined fault the training results were not found to be good. Attempt should be made in future studies to create a model that can predict combined faults on the basis of training provided for independent isolated faults.

0.0071	0.9782	0.0147	1	0.9997	0.0003	1E-05	0	0.013	0.9564	0.0306	0	0.0415	0.8498	0.1087	0	0.0493	0.9065	0.0443	0	0.0222	0.9206	0.0573	0	0.033	0.9134	0.0536	
0	1	1	1	0	1	1	1	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	0	1	1	
0.011	0.9713	0.0176	1	0.0073	0.9754	0.0173	0	0.9946	0.0051	0.0003	0	0.999	0.0009	8E-05	0	0.0121	0.9453	0.0426	0	0.1013	0.8078	0.091	0	0.0198	0.9436	0.0366	
0	1	1	1	0	1	1	1	0	1	1	1	1	1	1	0	1	1	1	0	1	1	1	0	1	1	0	
0.0132	0.9452	0.0416	1	0.0134	0.9653	0.0212	0	0.0129	0.9623	0.0248	0	0.9943	0.0054	0.0002	0	0.8384	0.1227	0.0389	0	0.0481	0.8441	0.1078	0	0.0858	0.8381	0.0761	
1	1	1	0	0	1	1	1	0	0	1	1	0	1	1	1	1	1	1	0	1	1	1	0	1	1	0	
0.0297	0.935	0.0353	0	0.0149	0.9554	0.0297	1	0.0353	0.9116	0.0532	1	0.0165	0.9547	0.0288	1	0.9947	0.0051	0.0002	0	0.9842	0.0136	0.0022	0	0.0145	0.9405	0.045	
1	1	1	0	1	1	1	0	0	1	1	1	0	1	1	1	0	0	1	1	1	1	0	1	1	1	0	
0.055	0.8388	0.1061	0	0.0871	0.8549	0.058	0	0.0148	0.9493	0.0359	0	0.129	0.7004	0.1706	0	0.0114	0.9672	0.0214	0	0.9906	0.0088	0.0006	0	0.7915	0.1398	0.0687	
1	1	1	0	1	1	1	0	1	1	1	0	0	1	1	1	0	0	1	1	1	1	1	0	1	1	0	
0.6017	0.3147	0.0836	0	0.0258	0.8942	0.08	0	0.0734	0.8667	0.0599	0	0.0151	0.939	0.0459	0	0.0881	0.8094	0.1025	0	0.0149	0.9568	0.0283	0	0.9927	0.0066	0.0007	
1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	0	1	1	1	0	0	1	1	0	0	1	1
0.9944	0.0052	0.0004	1	0.9369	0.0605	0.0027	0	0.0181	0.9042	0.0777	0	0.0552	0.8893	0.0555	0	0.0128	0.954	0.0333	0	0.0286	0.9341	0.0373	0	0.02	0.9417	0.0383	
0	1	1	1	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	0	1	1	1	0	0	1	1
0.006	0.979	0.015	1	0.9993	0.0007	5E-05	0	0.9945	0.0049	0.0006	0	0.012	0.9284	0.0596	0	0.0338	0.9344	0.0318	0	0.0226	0.9264	0.051	0	0.0371	0.9063	0.0566	
0	1	1	1	0	1	1	1	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	0	1	1	
0.0094	0.9726	0.018	1	0.0077	0.9761	0.0162	0	0.9879	0.0121	5E-05	0	0.9949	0.0041	0.001	0	0.0146	0.9326	0.0528	0	0.0814	0.84	0.0787	0	0.0281	0.9188	0.053	
0	1	1	1	0	0	1	1	0	0	1	1	1	1	1	0	1	1	1	0	1	1	1	0	1	1	0	
0.0112	0.9646	0.0242	1	0.0193	0.9504	0.0303	0	0.0095	0.9696	0.0209	0	0.9949	0.0047	0.0003	0	0.8978	0.1009	0.0013	0	0.0188	0.9218	0.0594	0	0.0339	0.9262	0.0399	
1	1	1	0	0	1	1	1	0	0	1	1	0	0	1	1	1	1	0	0	1	1	0	0	1	1	0	
0.041	0.9182	0.0408	0	0.0156	0.9514	0.033	0	0.0396	0.898	0.0624	0	0.0107	0.9683	0.021	0	0.9871	0.0126	0.0003	0	0.9981	0.0018	0.0002	0	0.0234	0.9212	0.0554	
1	1	1	0	1	1	1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	1	1	0	0	1	0	
0.0426	0.8553	0.1021	0	0.0272	0.937	0.0359	0	0.0204	0.9336	0.0459	0	0.0484	0.8788	0.0728	0	0.0097	0.9734	0.0169	0	0.9988	0.0012	9E-05	0	0.9982	0.0015	0.0003	
1	1	1	0	1	1	1	0	1	1	1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	1	0	
0.9782	0.0165	0.0053	0	0.0221	0.9006	0.0773	0	0.0498	0.8892	0.061	0	0.0209	0.9279	0.0512	0	0.0603	0.8544	0.0852	0	0.0092	0.9706	0.0202	0	0.9854	0.014	0.0006	
1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	0	0	1	1	0	0	1	1	0	0	1	1
0.998	0.002	9E-05	1	0.8973	0.0955	0.0072	0	0.0683	0.8054	0.1262	0	0.0722	0.8688	0.0591	0	0.0108	0.9617	0.0275	0	0.0666	0.8262	0.1072	0	0.0134	0.9604	0.0262	
0	1	1	1	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	0	1	1	1	0	0	1	1
0.0081	0.9731	0.0187	1	0.9739	0.0259	0.0001	0	0.0242	0.9355	0.0403	0	0.0083	0.9582	0.0335	0	0.0369	0.9251	0.038	0	0.0343	0.9069	0.0589	0	0.0733	0.817	0.1098	
0	1	1	1	0	0	1	1	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	0	1	1	1
0.0191	0.9496	0.0313	1	0.0117	0.9634	0.0249	1	0.9985	0.0014	5E-05	0	0.9725	0.0251	0.0024	0	0.0115	0.9303	0.0582	0	0.043	0.9113	0.0457	0	0.0182	0.9474	0.0344	
0	1	1	1	0	0	1	1	0	0	1	1	1	1	1	0	1	1	1	0	1	1	1	0	1	1	0	
0.0128	0.9623	0.0249	1	0.0412	0.893	0.0658	0	0.0131	0.9612	0.0257	0	0.9992	0.0007	8E-05	0	0.9993	0.0006	0.0001	0	0.0195	0.9153	0.0652	0	0.1508	0.7301	0.1191	
1	1	1	0	0	1	1	1	0	0	1	1	0	0	1	1	1	1	1	0	1	1	1	0	1	1	0	
0.0616	0.8865	0.0519	0	0.0156	0.9465	0.0379	1	0.0898	0.7679	0.1423	0	0.0132	0.9636	0.0232	0	0.9986	0.0013	1E-04	0	0.9622	0.0357	0.0021	0	0.0085	0.9579	0.0336	
1	1	1	0	1	1	1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	1	1	0	1	1	0	
0.0669	0.8125	0.1206	0	0.048	0.9027	0.0493	0	0.0215	0.9309	0.0476	0	0.0739	0.8201	0.1059	0	0.0171	0.9519	0.0309	0	0.9976	0.0022	0.0002	0	0.9982	0.0016	0.0002	
1	1	1	0	1	1	1	0	1	1	1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	1	0	
0.7076	0.2145	0.0779	0	0.0507	0.8153	0.134	0	0.0534	0.8921	0.0545	0	0.0111	0.9559	0.033	0	0.0633	0.8598	0.0769	0	0.0141	0.9574	0.0285	0	0.9885	0.0111	0.0004	

Figure 35: Combined Fault: Desired Target and Softmax Values (Frequency Domain)

Figure 36: Combined Fault: Desired Target and Softmax Values (Time Domain)

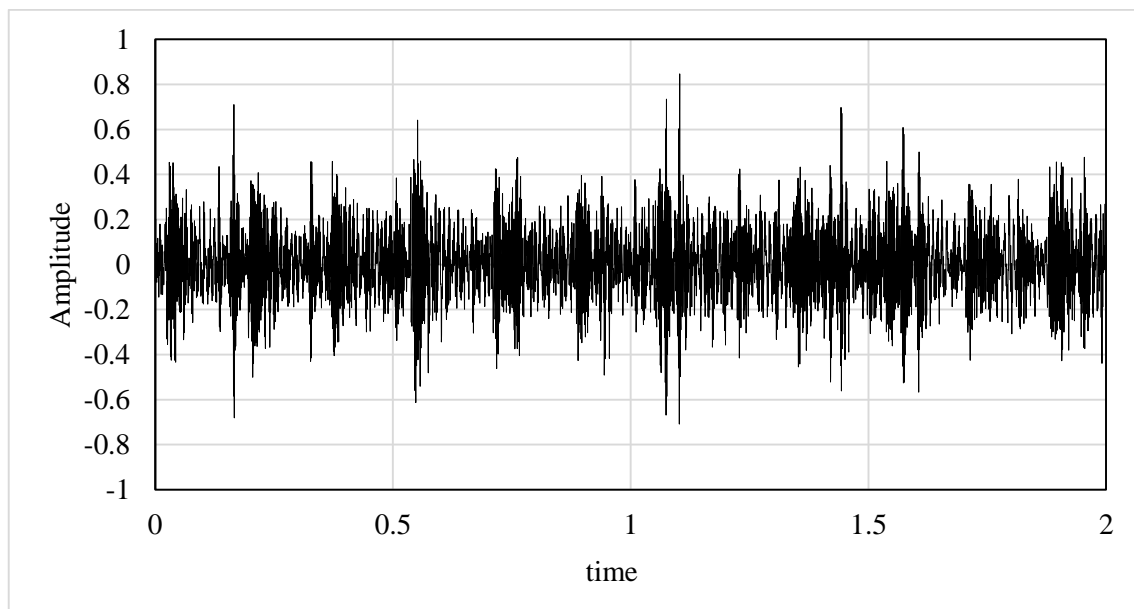
REFERENCES

- [1] “Machine Condition Monitoring Market.” [Online]. Available: <https://www.marketsandmarkets.com/Market-Reports/machine-health-monitoring-market-29627363.html>.
- [2] M. Systems, “Objective Machinery Fault Diagnosis Using Fuzzy Logic,” *Mech. Syst. Signal Process.*, vol. 12, pp. 855–862, 1998.
- [3] R. Yan, R. X. Gao, and X. Chen, “Wavelets for fault diagnosis of rotary machines: A review with applications,” *Signal Processing*, vol. 96, no. PART A, pp. 1–15, 2014.
- [4] Y. Lei, Z. He, Y. Zi, and X. Chen, “New clustering algorithm-based fault diagnosis using compensation distance evaluation technique,” *Mech. Syst. Signal Process.*, vol. 22, no. 2, pp. 419–435, 2008.
- [5] M. Boumahdi, J. P. Dron, S. Rechak, and O. Cousinard, “On the extraction of rules in the identification of bearing defects in rotating machinery using decision tree,” *Expert Syst. Appl.*, vol. 37, no. 8, pp. 5887–5894, 2010.
- [6] C. T. Yiakopoulos, K. C. Gryllias, and I. A. Antoniadis, “Rolling element bearing fault detection in industrial environments based on a K-means clustering approach,” *Expert Syst. Appl.*, vol. 38, no. 3, pp. 2888–2911, 2011.
- [7] L. B. Jack and A. K. Nandi, “Support vector machines for detection and characterization of rolling element bearing faults,” *Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci.*, vol. 215, no. 9, pp. 1065–1074, 2001.
- [8] N. S. Vyas and D. Satishkumar, “Artificial neural network design for fault identification

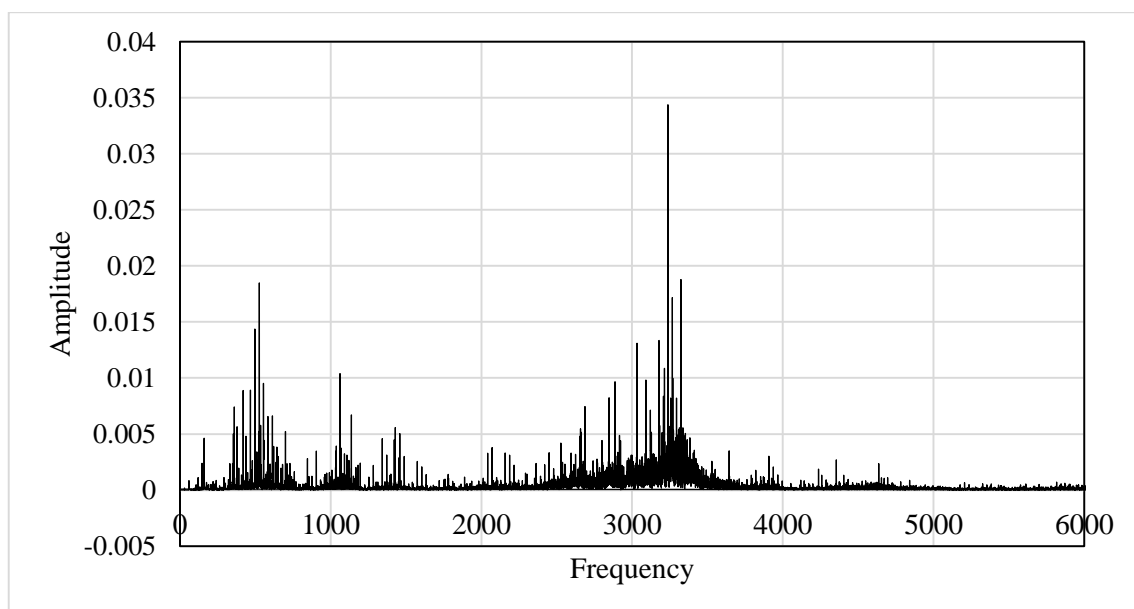
- in a rotor-bearing system,” *Mech. Mach. Theory*, vol. 36, no. 2, pp. 157–175, 2001.
- [9] O. Janssens *et al.*, “Convolutional Neural Network Based Fault Detection for Rotating Machinery,” *J. Sound Vib.*, vol. 377, pp. 331–345, 2016.
 - [10] Y. Xie and T. Zhang, “Fault Diagnosis for Rotating Machinery Based on Convolutional Neural Network and Empirical Mode Decomposition,” *Shock Vib.*, vol. 2017, pp. 1–12, 2017.
 - [11] S. Guo, T. Yang, W. Gao, and C. Zhang, “A novel fault diagnosis method for rotating machinery based on a convolutional neural network,” *Sensors (Switzerland)*, vol. 18, no. 5, 2018.
 - [12] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” *Proc. Thirteen. Int. Conf. Artif. Intell. Stat.*, vol. 9, pp. 249–256, 2010.
 - [13] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-Based Learning Applied to Document Recognition,” *proc. IEEE*, 1998.
 - [14] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” 2015.
 - [15] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
 - [16] J. Singh, “Condition Monitoring of Geared Rotor Systems using Neural Networks,” Indian Institute of Technology Kanpur, 2006.
 - [17] “Case Western Reserve University Bearing Data Center Website.” [Online]. Available: <http://csegroups.case.edu/bearingdatacenter/pages/download-data-file>.

APPENDIX

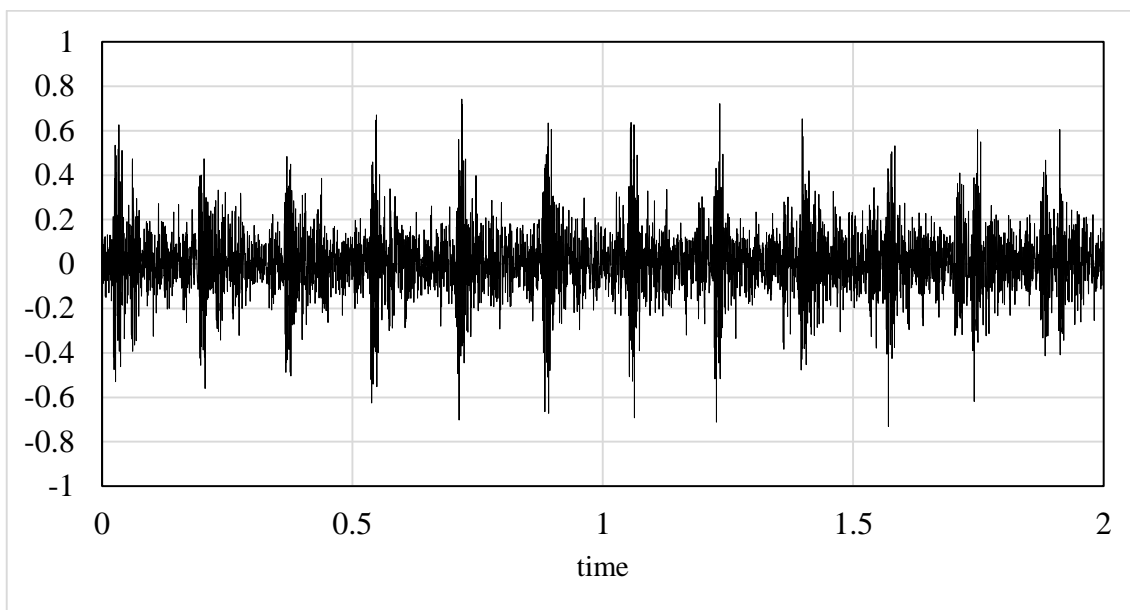
A typical sensor reading from [17] data.



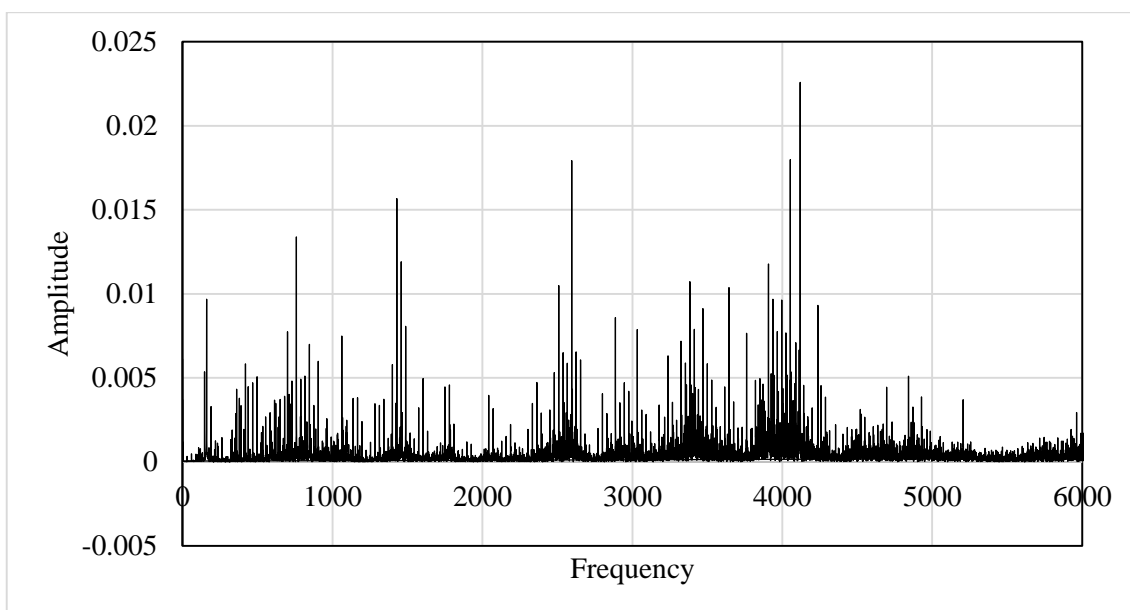
Time Domain response of fan end bearing



FFT response of fan end bearing



Time Domain response of drive end bearing



FFT response of drive end bearing