Politecnico di Torino

Master's Degree Thesis in Communications and Computer Networks Engineering

Data Augmentation for Long Short-Term Memory Neural Networks in Trajectory Prediction of Indoor Person Localization from Infrared Sensors

Supervisors:
Prof. Mihai Lazarescu
Prof. Luciano Lavagno

Candidate:
NUERAILI TUERHONGJIANG

March 2022

Summary

There are many indoor personnel tracking and localization applications using infrared thermal sensors, such as health monitoring and indoor security surveillance. This method is the best suitable for tagless localization of human bodies, adapting to different users and different scenarios. Since Infrared sensors are more affordable than other sensors used to detect and track people and their activities, infrared sensors are currently widely used in the Internet of Things.

This thesis evaluates the long short-term memory neural networks for predicting data from a low-resolution 16-pixel thermopile sensor data for indoor localization and tracking, improving robustness and reliability by adding unrelated noise to the sensor data. This noise addition is a type of data augmentation that results in the production of more data. Currently, the long short-term memory neural networks have been proposed for their excellent performance in a variety of tasks such as speech recognition and machine translation. By training neural network models on significantly more data, data augmentation improves the ability of the models to generalize what have learned to the new data and the unseen data.

In this thesis, the model results obtained with augmented data are compared to the model baseline initially generated without augmentation techniques. Data augmentation noise is then added in varying amounts compared to the infrared sensor signal variance between human presence and absence. There are four independent datasets, one of which was used mainly for training and partly for testing, and the other three only for testing. The average mean square error across all sets is used to calculate the generalization quality, while the learning curves are used to determine the learning quality. The noise amplitude is varied, trying to find the optimal improvement over the baseline for generalization and learning quality. The inference performance and learning quality change differently for each set with the noise amplitude such as one set shows a noticeable improvement while the others do not. To avoid false conclusions, the regions of interest with lower mean square error values would be examined in detail. If the overall performance of the model improves for some noise amplitudes, the refined ranges would be examined to see if the model has better generalization for all sets.

To test the generalization ability of the model's inference. Initially, the data was augmented by solely adding white noise which is frequently obtained from various sources, followed by augmenting the data with only brown noise which is characteristic of low-frequency variations. The amplitude of each noise comprised 0.01 % to 163.84 % of the signal variance between person presence and person absence. The performance of the model is found to be improved best in the range [11.6 %; 36.1 %] of the white noise range and [0.1 %; 0.59 %] of the brown noise amplitude. Then, when two types of noises are augmented together by combination, the best improvement can be found when the white noise amplitude is 1.28 % and the brown noise amplitude is 0.29 %.

Contents

Acknowledgements

The first I would like to express my warm gratitude is my family, which motivated me to complete my master's studies successfully.
Moreover, I would like to appreciate the help from Prof. Mihai Lazarescu and Prof. Luciano Lavagno for giving me the opportunity and patiently guiding me to finish my thesis when I lost my way.
At last, I would like to thank all my friends who have helped me and encouraged me in Italy over the past three years.
My best to you all.
Grazie.

List of Tables

List of Figures

Acronyms

Adam	adaptive movement estimation
ANN	artificial neural network
Bi-RNN	bidirectional recurrent neural network
DSE	design space exploration
FOV	field of view
IR	infrared
LSTM	long short-term memory
MSE	mean square error
NLP	natural language processing
NN	neural network
pdf	probability density function
ReLU	rectified linear activation function
RNN	recurrent neural network
tanh	hyperbolic tangent

Chapter 1

Introduction

1.1 Neural Network

Artificial intelligence is playing an increasingly important role in modern life. Whether it is a connected car on the street or a smartphone that people use every day, sophisticated artificial intelligence is behind it all. And the underlying logic of such powerful artificial intelligence is the Neural Network (NN).

In an NN, a neuron is its most basic structure. The most basic neuron model is one that contains input, output and computational functions. The input can be likened to the dendrites of a human brain neuron, and the output can be likened to the axon of a human brain neuron. Figure 1.1 below shows a typical neuron model of a neural network. The connection lines are the main components of the neurons, and each line has a corresponding weight. The ultimate goal of training a NN is to use an algorithm to adjust the value of the weights in the NN so that the prediction effect can be optimal.

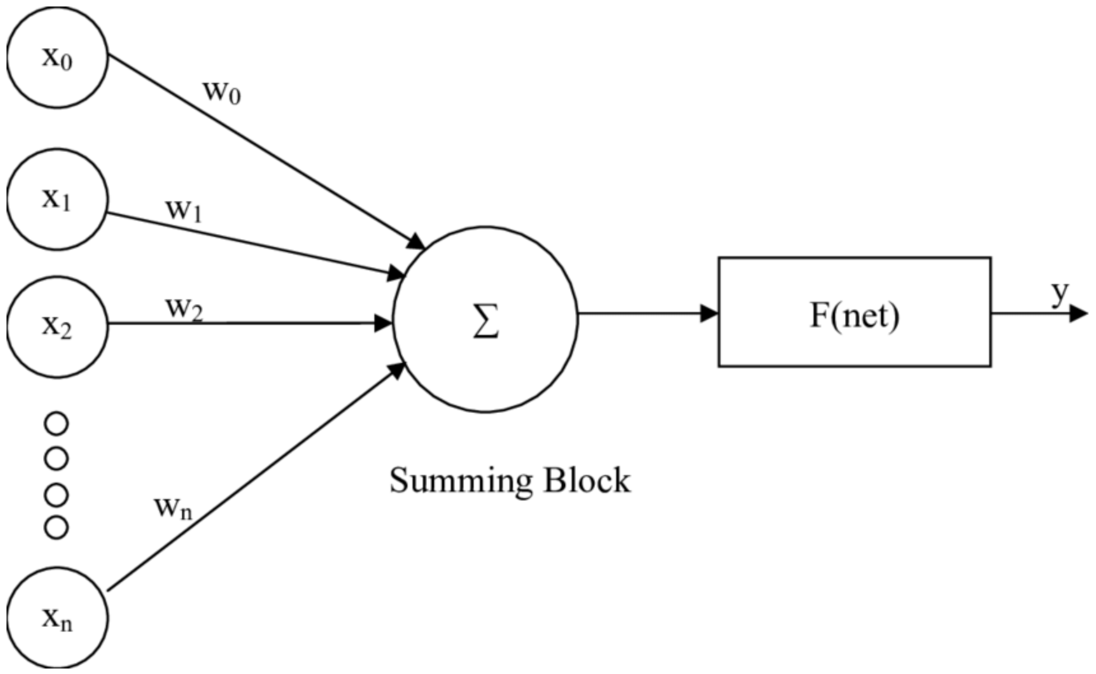


Figure 1.1: Basic neuron model contains input, output and computational functions.

During training, the input is a data set that has been collected or created. In the field of artificial intelligence, datasets can be images, audio, or even text. The datasets used in this thesis were collected from experiments conducted in the laboratory. There are also some open source datasets, such as MNIST and MS-COCO, that researchers can use to improve the performance of their models. In general, the more data, the better the predictions. However, if the number of training samples is significant, too few network layers and insufficient feature training will lead to inadequate training. Therefore, a very important premise is that the feature extraction ability of the network cannot be too low, which depends on the capacity of the NN.

To help the NN learn the rules hidden behind the data, it is necessary to improve the generalization ability of the NN, so that the trained NN can also give correct output for data other than the dataset with the same rules. In order to improve the quality of the generalization ability, the phenomena of overfitting and underfitting have to be avoided. Underfitting occurs when the NN cannot achieve enough low error on the training set. In other words, the model complexity is low, the model performs poorly on the training set and cannot learn the rules behind the data. Overfitting occurs when the gap between the training error and the test error is too large. This means the model complexity is higher than it needs to be and the model performs well on the training set but poorly on the test set. The model mistakenly learns the properties or characteristics of the training set that are not relevant to the test set. The model does not understand the rules behind the data and has a poor ability to generalize. To solve these phenomena, data augmentation and control of model complexity can be attempted. Or reduce the number of eigenvalues, delete redundant eigenvalues, and manually select to retain specific eigenvalues. In machine learning, eigenvalues can be used in several ways to analyze and transform data. A common technique that uses eigenvalues is Principal Component Analysis, which is a method of reducing the dimensionality of a dataset by finding the eigenvalues of the covariance matrix of the dataset. Eigenvalues and data augmentation techniques are complementary techniques that can be used to improve the performance of NN models. Eigenvalues can help to extract useful features from the data, while data augmentation techniques can help to increase the diversity of the training data. Depending on the specific problem and available resources, one or both of these techniques may be applicable and useful in a given machine learning task.

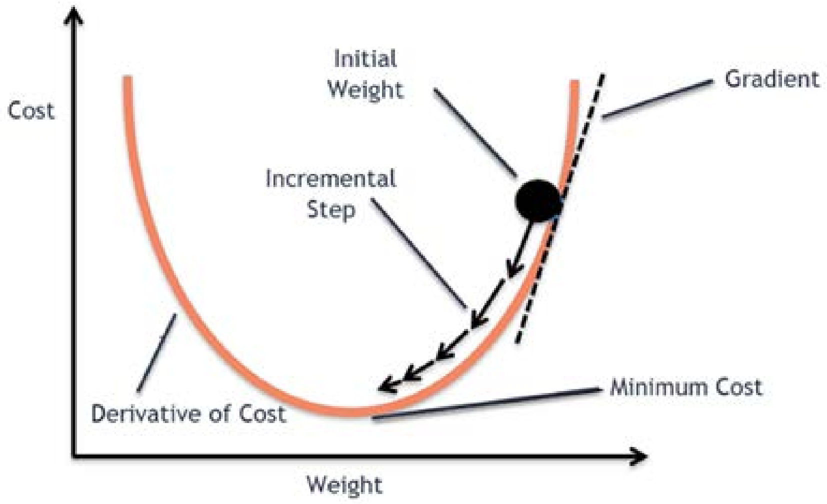


Figure 1.2: Diagram of gradient descent **[1].**

The gradient based learning method is an optimization algorithm that is used to minimize the function result by iteratively shifting in the direction of the steepest descent as indicated by the negative of the gradient. Such as the cost function, which quantifies the error between the predicted value and the expected value and expresses it as a single actual number to measure the performance of the NN model on given data. In machine learning, gradient descent is commonly used to update the parameters of the model. Parameters usually refer to coefficients in linear regression and weights in NN. Gradient descent is an iterative optimization algorithm that efficiently solves for the local minima of a function. To achieve this, the algorithm iteratively performs a simple procedure. First compute the gradient, which is the current slope, and then update the parameters by moving in the direction of the negative gradient. The size of the moving step is determined by the learning rate, which controls how much moving in the direction. If the learning rate is too small, the algorithm may take a long time to converge. While if the learning rate is too large, the algorithm may overshoot the goal.

In an artificial neural network (ANN), the activation function of a neuron node is responsible for transforming the summed weighted input of the node into the activation of the node or output for that input. It is particularly important for nonlinear functions. If the activation function is not used, the output of each layer is a linear function of the input of the upper layer. No matter how many layers there are in the NN, the output is still a linear combination of the input. This is the most basic case of a perceptron. When the activation function is used, the NN can approach any nonlinear function, so the NN can be applied to many nonlinear models. This makes it easy for the model to generalize or adapt to different types of data and to discriminate the output.

As shown in below Table 1.1, here are some commonly used activation functions. In ANN, the sigmoid function is often used as the threshold function of the NN since its single increase, which maps variables between 0 and 1. The hyperbolic tangent (tanh) is one of the hyperbolic functions. In mathematics, the tanh is derived from the basic hyperbolic sine and hyperbolic cosine. The rectified linear activation function (ReLU) is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero.

Table 1.1: Examples of activation functions.

sigmoid	$f(x) = \frac{1}{(1 + e^{-x})}$
tanh	$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$
ReLU	$f(x) = \begin{cases} x & \text{for } x > 0 \\ 0 & \text{for } x \leq 0 \end{cases}$

1.2 Indoor Person Localization

Indoor person localization and activity detection are used in a growing number of smart space applications for ongoing support and safety monitoring. Assisted living applications, for example, can reduce the cost of assistance while improving the safety and quality of care that is becoming increasingly important as the proportion of elderly people grows.

The IEEE 802.11, Bluetooth, radio frequency identification, ultra-wideband radio, visible light communication, audible and ultrasound acoustic signals, wearable or portable devices. These techniques mentioned above can all be used in indoor person localization **[2]**.

However, indoor localization is needed in situations where people may not be carrying or wearing a device that the localization system can detect, such as in some smart home applications or assisted living for the elderly. It should also protect individual privacy, be inexpensive, unobtrusive, easy to install and require little or no maintenance to increase the acceptance and value of the localization system.

Infrared (IR) sensors can be self-contained, simple to install, unobtrusive, privacy-aware, and reasonably priced. IR sensors can detect, recognize, and locate people indoor. But when the room's temperature and humidity change, their sensitivity can be affected sharply. Thermopile sensors usually consist of series-connected thermocouples, which are mainly used to sense IR radiation and are widely used as automatic lighting and motion detectors for security alarms. They are designed to measure the total amount of incident IR flux, not its change. Therefore, they can be used to detect stationary targets. This type of sensors are commonly used for non-contact temperature measurement in household appliances. And because they are very low-resolution thermal cameras, they cannot contain identifiable information, which can protect privacy well **[3]**.

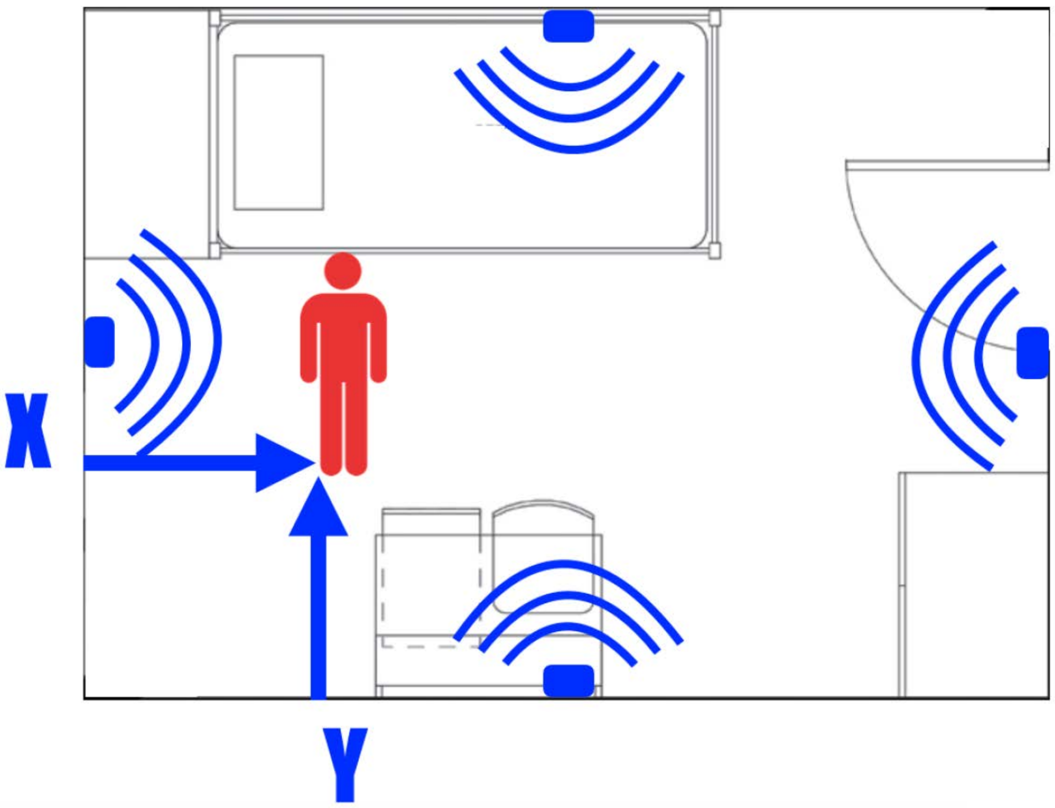


Figure 1.3: Illustration of a virtual room that trace the position of a moving person **[2]**.

The use of different monitoring solutions could improve response times or improve the prediction of dangerous events indoors. Traditional high-resolution cameras offer good image quality and the ability to cover large areas, but their acceptance in residential environments is not so good due to privacy concerns. Therefore, multiple unobtrusive IR sensors distributed throughout the house are an alternative to regular cameras, providing valuable information for predicting or detecting common dangerous events. However, the classification accuracy is limited and machine learning is needed as an aid to improve the classification accuracy of the sensors. **[4]**. Data augmentation can further improve machine learning performance and accuracy.

In this thesis, how data augmentation processing can improve the accuracy of NN is researched. Relatively simple IR sensors, which are known to be more susceptible to environmental noise, were tested to better compare the effectiveness of the data processing on the overall accuracy. The long short-term memory (LSTM) neural networks are optimized through design space exploration (DSE). The IR sensor is used to gather data while a person walks arbitrarily in a 3m × 3m experimental room, then compare the predicted position and trajectory line by NN with the ground truth location acquired using an accurate ultrasound localization device. The main contribution of this thesis is neural network-based data augmentation for indoor person localization and tracking using small IR sensors.

Chapter 2

Tools and Methodology

2.1 Long Short-Term Memory Neural Network

The data processing paradigm modeled after the biological human brain is known as an ANN. An ANN is trained for a specific task, such as pattern recognition or data classification, which used a learning process. Although there are different NN topologies, this thesis focuses on LSTM NN. LSTM is a novel recurrent neural network (RNN) architecture combined with an appropriate gradient-based learning methodology. The LSTM aims to overcome the error backflow problem. It can bridge time intervals even in the presence of noisy or incompressible input sequences, without losing short-time capability **[5]**.

An RNN is a type of ANN which connections between nodes form a graph along a temporal sequence. This enables it to display temporal dynamic behavior. So RNN allows knowledge to be kept in the network, it can use reasoning from previous training to generate better, more educated predictions about upcoming events. However, RNN has significant short-term memory issues. When the data to be processed is too lengthy, the impact of the input data at the early stage on the ultimate judgment result is extremely little, even if the information is critical. Since RNN contain memory, parameter sharing, and Turing completeness, it has some advantages in learning nonlinear sequence characteristics. Natural language processing (NLP) applications in RNN include speech recognition, language modeling, and machine translation.

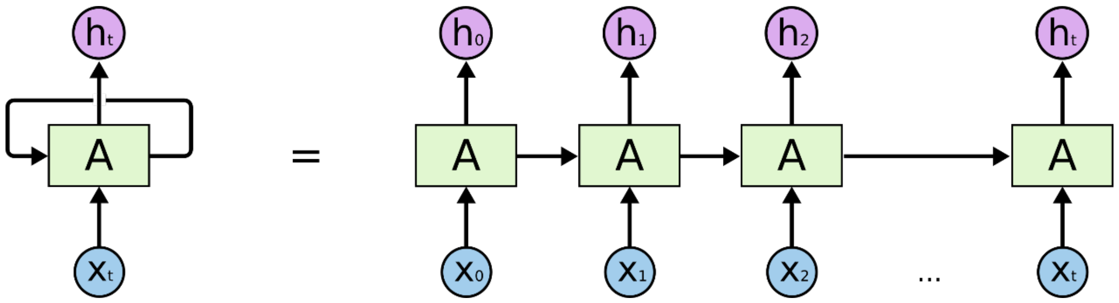


Figure 2.1: Illustration of recurrent neural network **[6].**

RNN research began in the 1980s and evolved into one of the deep learning algorithms in the early 2000s. The typical RNN are bidirectional RNN (Bi-RNN) and

LSTM NN. All RNNs have the structure of several repeating modules. Although the repeating module of LSTM also has a chain-like topology, however it is structured differently. There are multiple functional layers, and these layers interact in very different ways. It is designed to be used to record additional information, which means the memory mentioned above.

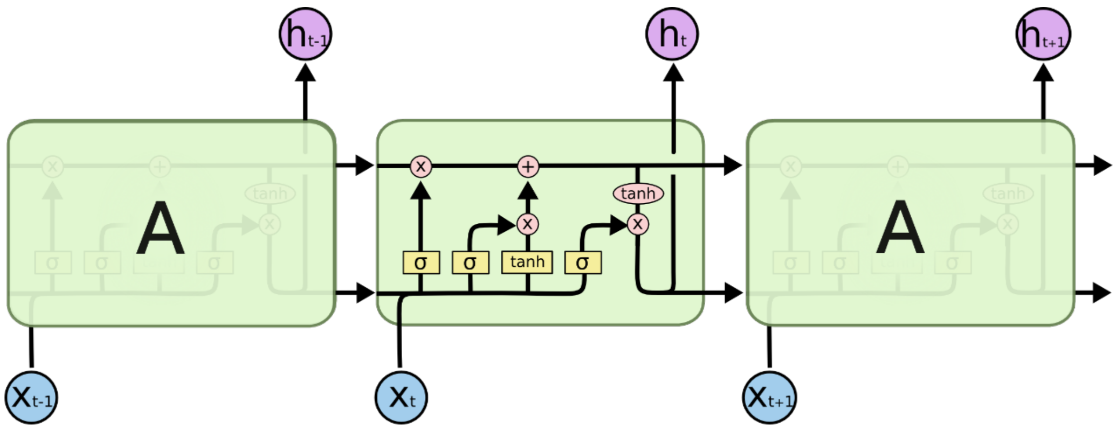


Figure 2.2: Illustration of the LSTM NN **[6].**

The continuous line that runs through the top of the Figure 2.2 and represents the cell state is the core trick of LSTM. The cell state resembles a factory line in certain ways. With only a few tiny linear interactions, it proceeds directly down the entire chain which makes information can very simply continue to flow along. The LSTM NN can modify the cell state by removing or adding information, which is carefully controlled by gates. The gates in an LSTM NN allow the network to selectively forget or remember information over time, which is particularly useful for modeling sequential data. There are three gates, input gate, output gate and forget gate. By using these gates, the LSTM network can learn to selectively store and retrieve information over long sequences. Information can pass through gates on an entirely optional basis. This consist of the pointwise computations and a layer of sigmoid or tanh. These three gates serve to secure and manage the cell state in an LSTM NN.

Choosing what information from the previous cell state to discard is the first phase in the standard LSTM NN procedure. The forget gate, a sigmoid layer with the concatenation of input and previous output, can make this happen. To every number in the previous cell state, it outputs a number between 0 and 1 after considering the module's input. In the output, 1 denotes total retention of the information, while the notes complete deletion of the information. The next step is to select the new information that will be kept in the cell state. Two components make up this, first one is called the input gate that has the content as same as the forget gate while another one is a tanh layer. The input gate determines which values would be updated. The tanh layer provides a vector of potential new values by concatenation of input and previous output to be added to the cell state. The following procedure is to update the cell state. The module only needs to perform pointwise computations since the preceding phases have already decided what to do. The module pointwise multiplies the previous cell state value by the output of the forget gate. This procedure can drop the information which previously determined to forget. Then, pointwise multiply the output of the input gate with the previous new candidate values that produced by the tanh layer. This will produce the new candidate values that scaling by the amount we chose to update each state value. The updated cell state can be obtained by pointwise addition of the new candidate values and the previously obtained result of the previous cell state with output by forget gate. Lastly, the module must choose what will be output by the output gate. This output will be based on the cell state and concatenation of input and previous output, but will be filtered by a tanh layer. The sigmoid layer is used to determine which parts of the cell state will be output. The module put the cell state through the tanh layer to transform the values to be between -1 and 1 then pointwise multiply it by the output of the sigmoid gate, so that the module only output the parts that decided to. The repetition of the above steps is the main operation of LSTM NN.

2.2 Data Augmentation

Data augmentation is an invaluable technique utilized in machine learning to address the issue of limited training datasets by generating additional equivalent data. This approach is highly effective in overcoming the challenge of insufficient data, and it has become a popular tool across a variety of fields in machine learning. Specifically, in computational vision, data augmentation algorithms can be broadly categorized into two distinct categories. The first category pertains to data augmentation techniques that leverage fundamental image processing methods, including but not limited to flipping, rotation, and noise injection. These techniques have proven to be quite successful, as they allow models to accurately classify data even when noise is introduced or certain parts of an image are cropped. The second category of data augmentation algorithms is based on deep learning and includes techniques such as kernel filters and random erasing. These approaches leverage the power of deep learning to generate new, high-quality data points that can be added to a limited training dataset. By utilizing both traditional image processing techniques and deep learning algorithms, data augmentation has emerged as a crucial tool for any machine learning practitioner looking to optimize their model's performance.

NN performs well in many scenarios, but these models often require large amounts of data to avoid overfitting. Unfortunately, enormous amounts of data are not available for many scenarios, such as medical image analysis. The existence of data augmentation technology can solve this problem brilliantly. Data augmentation techniques increase the size and quality of training datasets so that can be used to build learning models. In the field of computer vision, generating augmented images is easier than other methods.

By introducing noise to inputs during training can smooth out the input space and make it simpler to learn. Expanding noise increases the size of the training dataset. Random noise is added to the input variables each time while the training sample is exposed to the model, making them distinctive each time. This makes it as a simple data augmentation method.

Since training samples are constantly changing due to the addition of noise, the network is less able to memorize them. This leads to smaller network weights and a more robust network with lower generalization error. The noise means that it is as though new samples are being drawn from the domain in the vicinity of known samples, smoothing the structure of the input space. This smoothing could make it easier for the network to learn the mapping function, resulting in better and faster learning **[7]**.

The addition of Gaussian noise to input variables is the most typical form of noise used during training of the NN model. Only during training, the noise is introduced. When the model is evaluated or used to make predictions based on the dataset, no noise is added. In this thesis, gaussian white noise and brown noise are mainly discussed.

A type of signal noise with a probability density function (PDF) equal to that of the normal distribution, also known as the Gaussian distribution, is referred to as Gaussian White Noise in the theory of signal processing. The range of values that the noise can take is followed by Gaussian-distributed, with a mean of zero and a standard deviation of 1. Wideband Gaussian noise from various natural sources can interfere with communication channels in telecommunications and computer networks.

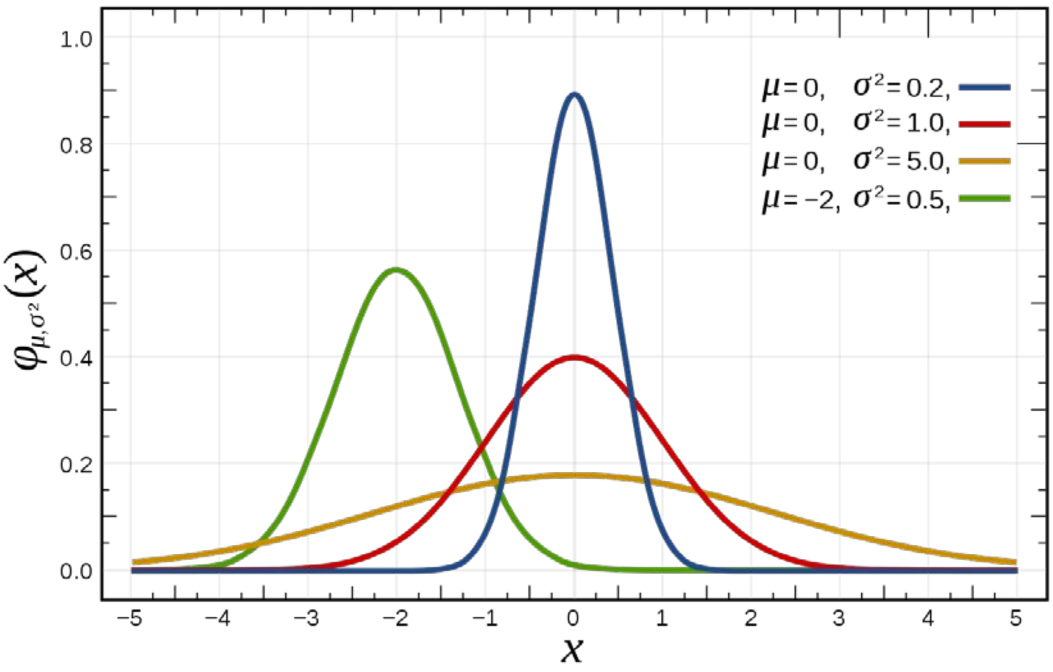


Figure 2.3: Plot of normal distribution with different parameters **[8]**.

Brown noise is a type of signal produced by Brownian motion. It is used to simulate certain physical phenomena, such as turbulence or the behavior of a chaotic system, and can be used to study the properties of these phenomena. The power spectral density of white noise is flat. And since Brownian motion is obtained as the integral of a white noise signal, it can be derived that the power spectral density of brown noise is inversely proportional to the square of the frequency. Meaning that brown noise has greater energy at lower frequencies.

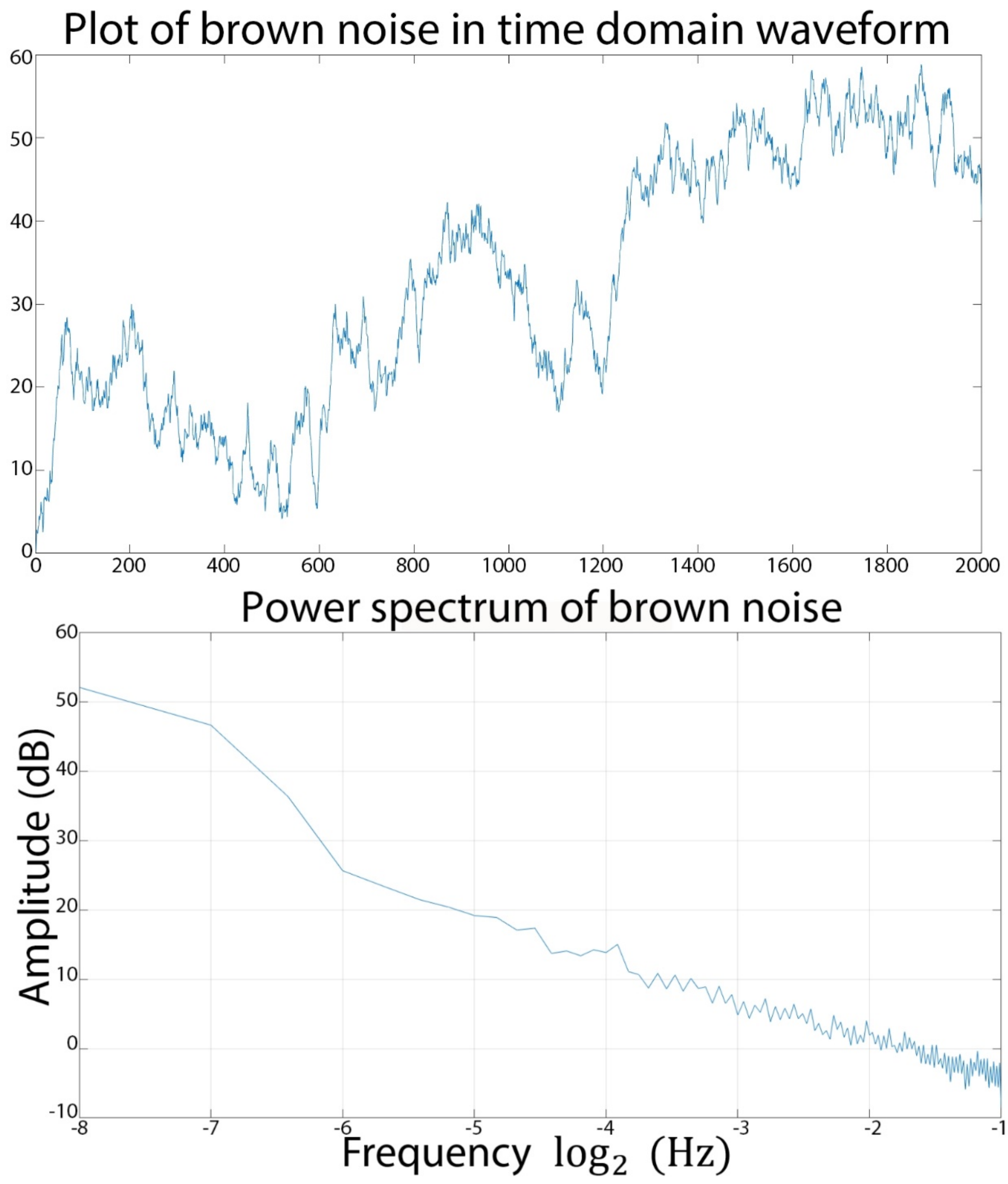


Figure 2.4: Plot of brown noise and power spectrum of brown noise.

In this thesis, the order of data should be maintained since the data collected are time-series data. Time-series data is a sequence of data that is taken over a period of time. It is used in machine learning to make predictions about future values, identify patterns and trends, and detect anomalies. Time-series data can be used to develop predictive models and to better understand the underlying causes of a system's behavior. In short, time-series data can provide valuable insights into the future that can be used to make a better prediction. RNN is ideal for processing sequential data due to its properties.

Determining the noise amplitude level is crucial for improving model generalization. The signal variations of the sensor detecting either a person or no person could be related to a reasonable range of noise. Inside the sensor, 16 pixels are transformed into temperature values representing a person's presence. Firstly, start by analyzing the sensor single-pixel variations. The highest point denotes the presence of a person, whereas the lowest point denotes their absence. The difference between these two magnitudes will be the foundation for searching noise amounts. In order to determine the noise amplitude, the same method was implemented on both white noise and brown noise. An insight into combining noises to produce augmented sets comes from the independent exploration of two noises to enhance model generalization quality. All systems naturally experience various perturbations. Using a combination of white and brown noises is the next method of augmented data generation in this thesis. Without explicitly mentioning noise levels, this section mainly focused on the various augmentation techniques used to solve the problem. After many training sessions, reasonable noise levels will be established. In the section that follows, acceptable noise levels will be discussed.

Once the noise amplitude level is determined, data augmentation can be achieved by adding noise to the corresponding training set to create a new training set. For each of the noise amplitudes, the noise was applied only to the training segment and the random generator was initialized with a different value each time. The augmented training set would then be stored as a batch, preserving the original order of the segment's samples. A total of 10 augmented data were generated, increasing the original training data by ten times. The final training data for each amplitude should contain ten training batches with noise and one original training segment without augmentation, and the samples within a batch should not be randomized to avoid destroying continuity. Test and validation batches are unchanged. During training, only one of the eleven batches was randomly selected per epoch.

2.3 Experimental Setup

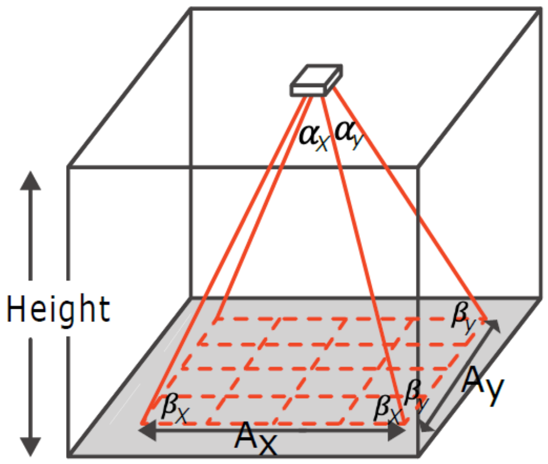


Figure 2.5: Conceptual illustration of the experimental space. α show the view angles for X and Y directions respectively, while Height is the height the sensor was placed. A show the area covered by the FOV **[2]**.

A 4 × 4 pixels Omron D6T-44L-06 thermopile infrared sensor **[9]** with a temperature resolution of 0.06 ℃ and accuracy ± 1.5 ℃ was used to monitor the 3 × 3 m experiment space. It is installed on the ceiling, 3.05 m above the floor, with a 2.48 × 2.57 m field of view (FOV) at floor level. The person's ground truth location with an ultrasound-based tag of the Marvelmind Starter Set HW v4.9 **[10]**, with ± 2 cm accuracy, 15 measurements per second was collected. Determining experimentally that the average accuracy of the ground truth system in the environment is ± 3.9 cm by measuring the localization accuracy acquiring four times per second for 5 s the location of a person standing on each of 16 central predefined locations inside the 3 × 3 m experiment room space.

During the actual location tracking experiment, a person walked for 30 minutes along an arbitrary and irregular path in the space, with variable speed, and collected synchronous readings at 5 Hz from both the IR sensor and the ground truth system. Four sets of 9000 tuples were collected, each made of 16 thermal sensor readings and two coordinates from the ground truth system relative to the room space **[11]**, the ground truth trajectory plot of all sets is shown in below Figure 2.6. These four sets were named as set A, set B. set C and set D. These datasets are collected on different days, according to different movement patterns and in different environmental conditions. These datasets not only simulate the trajectories of random walking in the room but also simulate the trajectories of people walking around in the room with furniture.

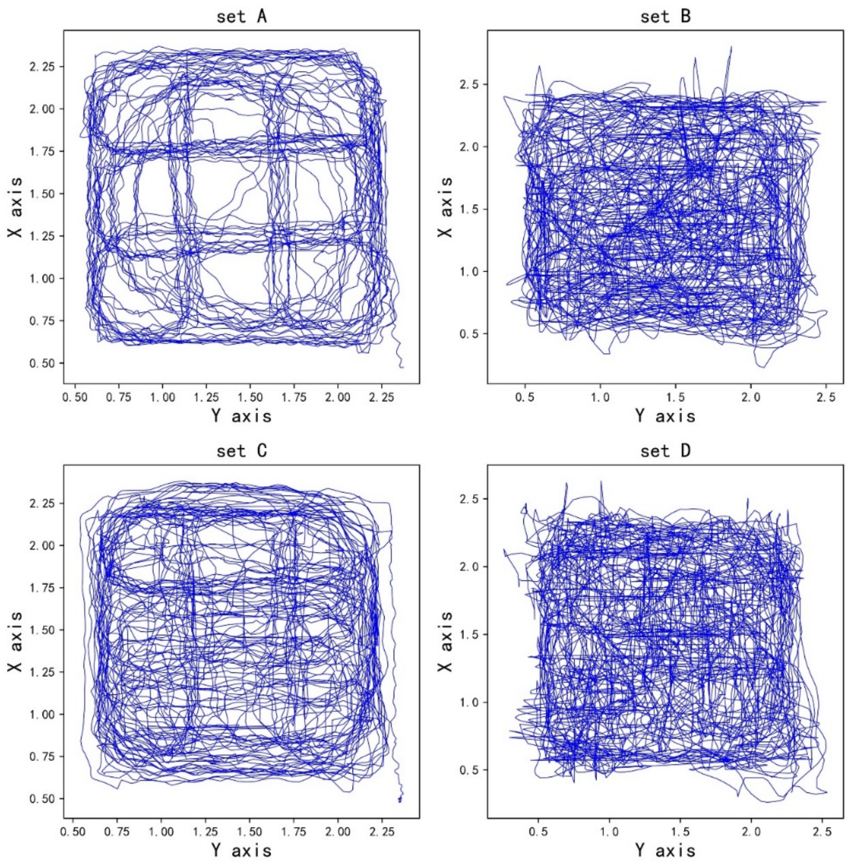
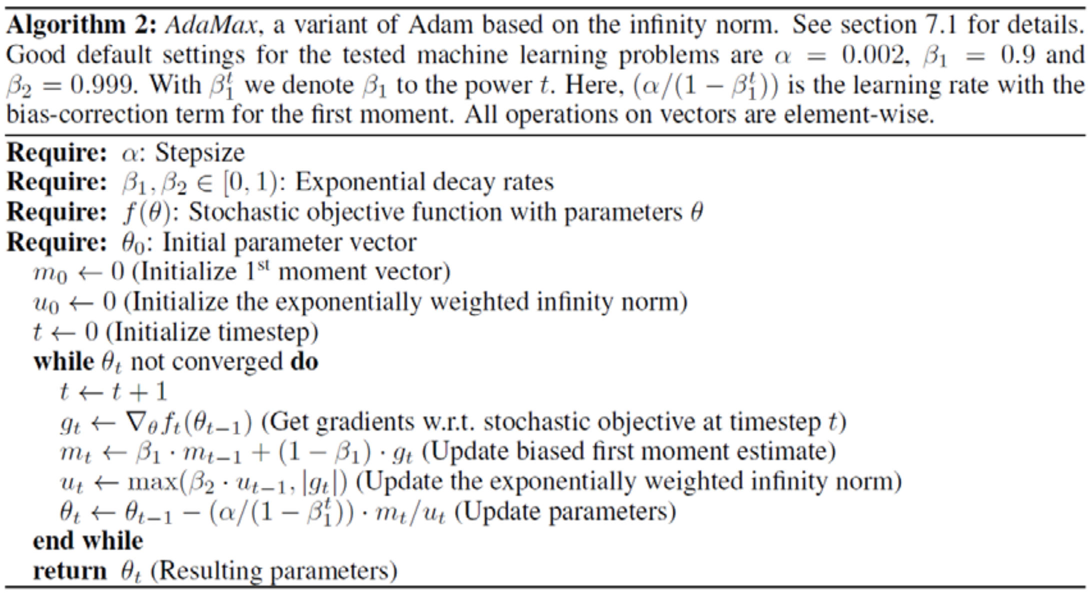


Figure 2.6: Ground truth trajectory plot of all sets.

Since the IR sensor takes 16-pixel, dynamic images, the LSTM NN architecture is considered in this thesis. General data is the time series organization of each frame. The next step is to work on the model input data after the appropriate architecture for the model has been chosen. The NN can be considered a function that receives input and produces an appropriate response. Just like all other functions, it has a domain. The values must be normalized before supplying them to the NN to ensure the values are within the domain. As with all functions, the outcome is not guaranteed to be appropriate if the arguments are beyond the domain. All input data are rescaled in the range [0; 1]. The initial IR sensor datasets have been normalized and fitted to the model input domain.

The NN architecture does not always ensure the correct model fit. A regularization technique named dropout was incorporated into our model to prevent overfitting. Dropout is a training method in which some neurons are ignored at random, and it is a straightforward and effective regularization technique for machine learning. This means that randomly chosen weight updates are not applied to the neuron on the backward pass, and their contribution to the activation of downstream neurons is temporally removed on the forward pass. Neuron weights within a NN find their way in the network as it learns. Neuronal weights are customized for particular characteristics, resulting in some specialization. This specialization is necessary for neighboring neurons. But if it goes too far, it could create a fragile model that is overly specialized on the training set. A neuron's dependence on the context during training is a complex co-adaptation **[12]**.

As described previously, one drawback of gradient descent is that it only uses one step size for all input variables. Extensions to gradient descent, such as the adaptive movement estimation (Adam) algorithm, use a different step size for each input variable, but this may cause the step size to drop quickly to extremely low values. Gradient descent with a generalization to the infinite norm known as AdaMax may produce a more effective optimization on some problems than Adam. It is an expansion of the gradient descent optimization algorithm in a broader sense. For each parameter in the optimization problem, AdaMax typically automatically adapts a different step size **[13]**. The optimizer of LSTM NN used in this thesis is AdaMax, and its algorithm is shown in Figure 2.7 below.



Algorithm 2: *AdaMax*, a variant of Adam based on the infinity norm. See section 7.1 for details. Good default settings for the tested machine learning problems are $\alpha = 0.002$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. With β_1^t we denote β_1 to the power t. Here, $(\alpha/(1-\beta_1^t))$ is the learning rate with the bias-correction term for the first moment. All operations on vectors are element-wise.

```
Require: $\alpha$: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates
Require: $f(\theta)$: Stochastic objective function with parameters $\theta$
Require: $\theta_0$: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $u_0 \leftarrow 0$ (Initialize the exponentially weighted infinity norm)
  $t \leftarrow 0$ (Initialize timestep)
  while $\theta_t$ not converged do
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep $t$)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $u_t \leftarrow \max(\beta_2 \cdot u_{t-1}, |g_t|)$ (Update the exponentially weighted infinity norm)
    $\theta_t \leftarrow \theta_{t-1} - (\alpha/(1 - \beta_1^t)) \cdot m_t/u_t$ (Update parameters)
  end while
  return $\theta_t$ (Resulting parameters)
```

Figure 2.7: AdaMax optimizer algorithm procedure **[13]**.

Before performing data augmentation, a baseline needs to be established, and then the final optimization effect can be known. As the experimental dataset used in this thesis is derived from previous research **[11]**, the step before optimization on the baseline is to rebuild the results of the previous research. This previous work varies the LSTM layers (1, 2, and 3) and LSTM units from range 2 to 64, in powers of two, and the input window width (1 s and 3 s). Then 36 types of combination parameters and 144 different situations finally can be gathered. Training using a continuous series of samples from 60 % data of the dataset. Then followed by validation on another contiguous sequence from 20 % of the dataset. Finally testing on the remaining contiguous 20 % data of the dataset, processing on all datasets. The LSTM NN model was trained 30 times for each parameter because the training procedure is stochastic. 1000 is chosen as the epoch number, and the sampling frequency of NN is 5 which means the model will process 5 data in 1 second, and 15 data in 3 seconds. Since 60 % contiguous sequence data is used, so here are 6 permutations for each set. The training set can be at the beginning, middle or end of the set. All permutation is considered and tested for avoiding unrepresentative validation problem and getting improved accuracy. Set A, set B and set C share the same order of dataset. These three datasets used the first 20 % data as validation dataset, the next 60 % of the data as training dataset when last 20 % data as testing dataset. Set D used the first 60 % data as the training dataset, the next 20 % data as testing dataset when last 20 % data as the validation dataset. Here are the resulting values from the NN model without data augmented for each parameter corresponding mean square error (MSE) metrics for set A, set B, set C and set D are given in the Table 2.1.

In this thesis, MSE value and learning efficiency are used to judge the performance of the model as researchers usually do. In this metric, a lower value indicates better model performance. As can be discovered from the MSE value in Table 2.1, the value results of set B and set D are more obvious similar while the value results of set A and set C are slightly similar. Meanwhile. This may be due to the similarity of the trajectory, as shown in Figure 2.6 above. The trajectories of set A and set C are slightly similar. Meanwhile, the trajectories of set B and set D are more random and chaotic. It also can be observed that when the value of LSTM units is 2 or 4, the MSE value of the model is higher. There is also a very good result when the input window width is 3 seconds. The optimal result of set B happens when the number of LSTM units is equal to 32. Else happens when the number of LSTM units is 64. The main optimal results are concentrated on 2 layers. At the same time, as the number of LSTM units increases, the average value of MSE decreases, which means that the prediction of the NN is getting better. However, the overall trend is getting smoother, and the increase in the number of LSTM units has less impact on MSE. It can observe in Figure 2.8 that is shown the inference of the trajectory coordinates for the chosen combination of parameters compare to the ground truth when the input window width is 3 seconds with 2 LSTM layers. The left side is results for 64 units when there is 2 units result on right. As expected from MSE value results, the predicted trajectories shown left side which represent the best result of 64 units are much better than the right part that represent the result of 2 units, and this result is shown on all sets.

In the learning curve which is shown in Figure 2.9 below, it can be seen that as the number of LSTM units increases, the effect of model underfitting decreases and the result of prediction is better. In the first 200 epochs, the training loss is decreasing rapidly and continues to decrease slowly at the end of the plot. Through the analysis, it can be obtained that the model is capable of further learning and possible further improvements which is the main exploration in the next step.

Table 2.1: Model training results of different combination parameters. For each parameter corresponding MSE metrics for set A, set B, set C and set D are shown.

best result of each combination parameters in set A							
input window width	LSTM layer	LSTM units					
		2	4	8	16	32	64
1 s	1	0.0761	0.0353	0.0159	0.0083	0.0054	0.0041
	2	0.1075	0.0483	0.0248	0.0105	0.0048	**0.0032**
	3	0.1435	0.0808	0.0290	0.0141	0.0060	0.0036
3 s	1	0.0760	0.0337	0.0136	0.0062	0.0036	0.0024
	2	0.1040	0.0447	0.0202	0.0067	0.0025	**0.0018**
	3	0.1273	0.0761	0.0232	0.0079	0.0029	0.0020

best result of each combination parameters in set B							
input window width	LSTM layer	LSTM units					
		2	4	8	16	32	64
1 s	1	0.1186	0.0718	0.0486	0.0409	0.0393	0.0387
	2	0.1578	0.0919	0.0598	0.0429	0.0345	0.0366
	3	0.2437	0.1402	0.0643	0.0476	0.0366	**0.0338**
3 s	1	0.1199	0.0727	0.0479	0.0394	0.0368	0.0350
	2	0.1627	0.0851	0.0551	0.0386	**0.0320**	0.0339
	3	0.2414	0.0610	0.0610	0.0445	0.0335	0.0338

best result of each combination parameters in set C							
input window width	LSTM layer	LSTM units					
		2	4	8	16	32	64
1 s	1	0.0803	0.0421	0.0214	0.0158	0.0128	0.0136
	2	0.1172	0.0553	0.0326	0.0181	0.0105	**0.0089**
	3	0.1513	0.0807	0.0366	0.0210	0.0133	0.0093
3 s	1	0.0866	0.0433	0.0227	0.0144	0.0113	0.0133
	2	0.1087	0.0544	0.0281	0.0163	0.0108	**0.0070**
	3	0.1476	0.0824	0.0333	0.0184	0.0100	0.0070

best result of each combination parameters in set D							
input window width	LSTM layer	LSTM units					
		2	4	8	16	32	64
1 s	1	0.0970	0.0643	0.0491	0.0444	0.0416	0.0409
	2	0.1121	0.0669	0.0512	0.0431	0.0403	0.0396
	3	0.1402	0.0834	0.0535	0.0457	0.0403	**0.0396**
3 s	1	0.0978	0.0645	0.0494	0.0438	0.0411	0.0408
	2	0.1154	0.0676	0.0504	0.0412	**0.0392**	0.0400
	3	0.1356	0.0816	0.0537	0.0433	0.0396	0.0398

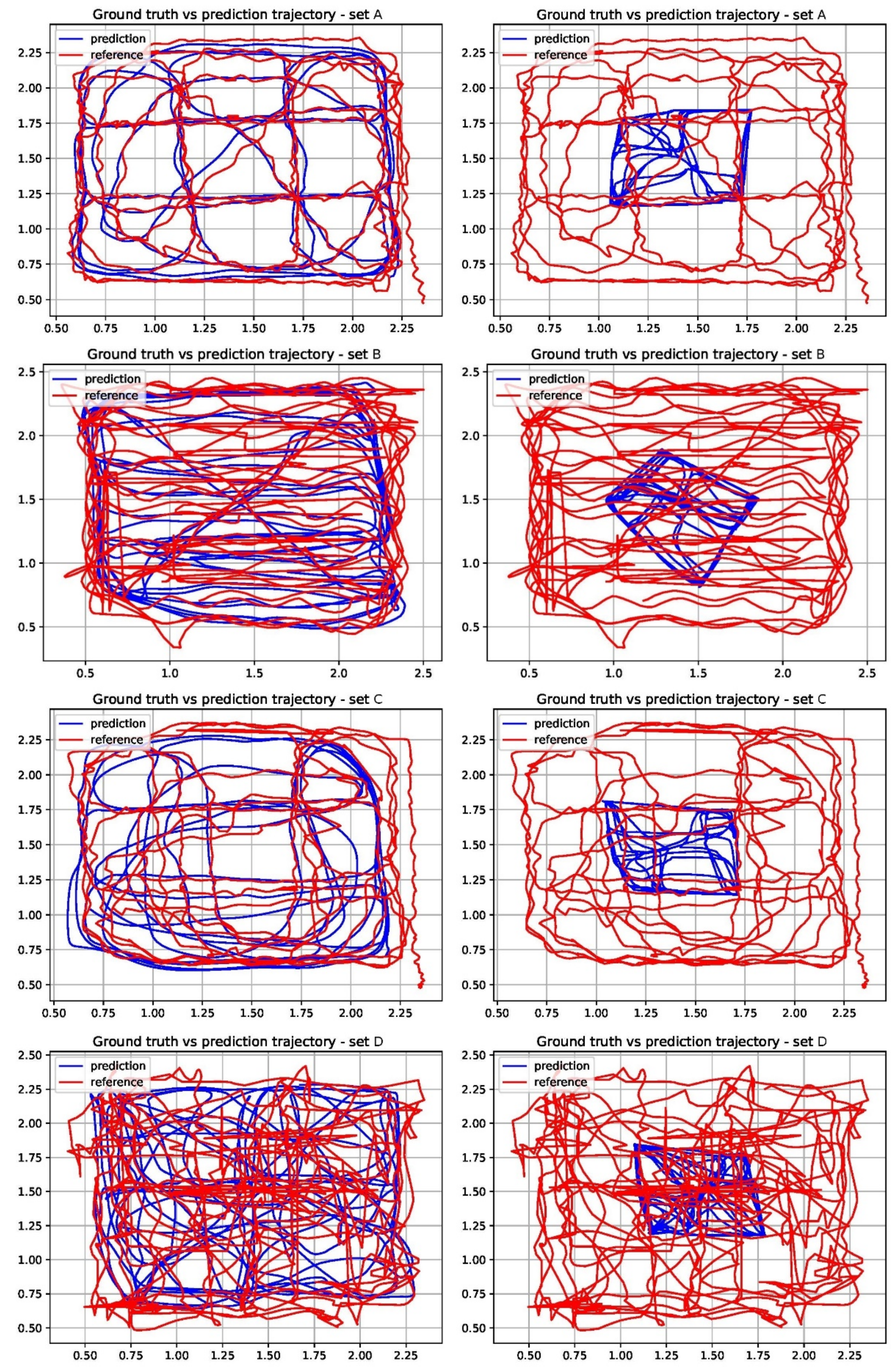


Figure 2.8: Ground truth compare with prediction in all sets, results of 64 units on left while results of 2 units on right. The input window width is 3 seconds with 2 LSTM layers.

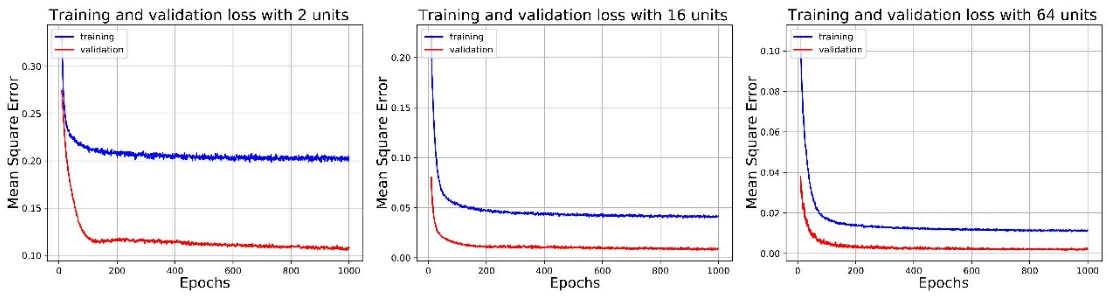


Figure 2.9: The learning curve of set A in the different number of units when the input window width is 3 seconds with 2 LSTM layers.

Chapter 3
Experimental Results

3.1 Baseline

After the discussion in the previous section, it is necessary to select a model with a stable learning efficiency as a baseline to proceed the next step which is adding noise. The selected model should have the highest possible learning efficiency and the best possible ability to avoid overfitting interference. The number of epochs was tried to increase to 2000 epochs and 3000 epochs when the number of LSTM units was expanded to 40 units or 48 units.

Relevant rules can be found after testing the model with different parameters. Compared to the results of 2000 epochs, the results of 1000 epochs show that the model cannot produce results that satisfy the learning efficiency, while the results of 3000 epochs are more likely to be overfitting. After comparing the results of different numbers of LSTM units, it can be concluded that a more fitting learning curve result can be obtained under the condition of 40 units, which can be verified by Figure 2.9 and Figure 3.1. Finally, after filtering the results from the different parameters of each set, the set B was chosen since other sets are easier to obtain unrepresentative validation problem or cannot be improved. Training using a continuous series of samples from the mid 60 % data of set B. Then followed by validation on another contiguous sequence from the first 20 % data of set B. Finally testing on the remaining contiguous 20 % data of set B, then testing by the entire other datasets. Using 40 LSTM units with 3 seconds window width and 2000 epochs. Repeat training 10 times instead of 30 times and pick the best result. And here are the baseline MSE value results and learning curve plot as shown below.

Table 3.1: Test MSE results of baseline.

dataset	set A	set B	set C	set D
test MSE	0.0589	0.0332	0.1962	0.1828

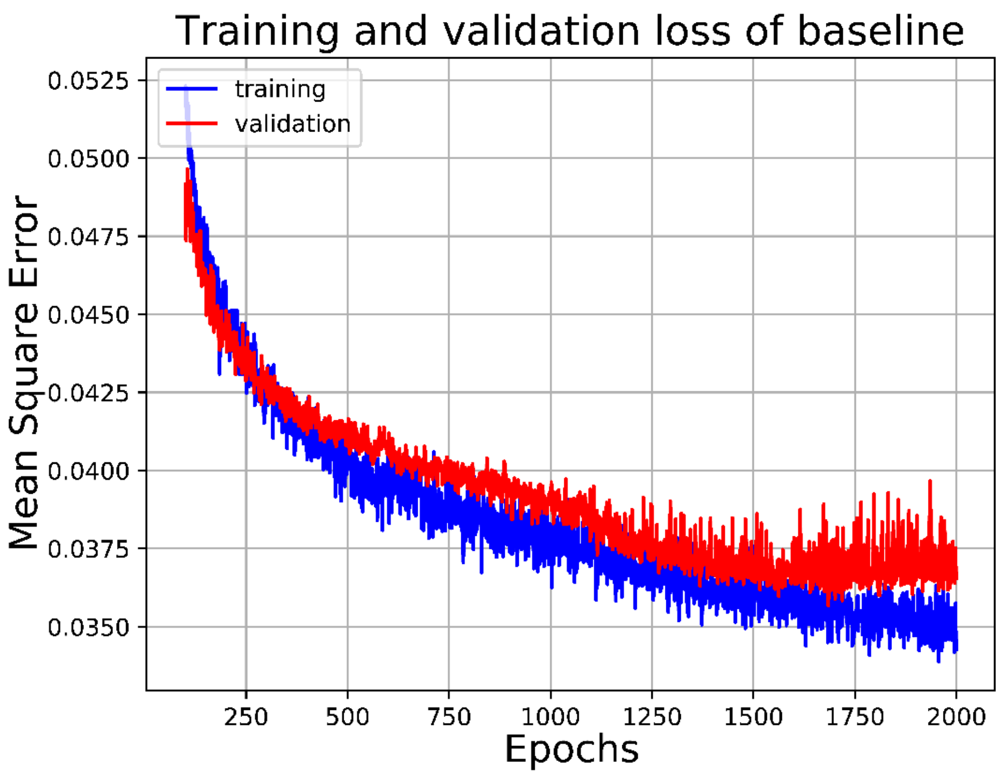


Figure 3.1: Learning curve plot of baseline. Using 2 LSTM layers with 40 LSTM units when 3 seconds window width and 2000 epochs in set B. Using the first 20 % data as validation dataset, the mid 60 % data as training dataset, the last 20 % data as testing dataset.

3.2 White Noise Augmentation

The goal of creating augmented datasets is to achieve better generalization than model training without augmentation and look for the regularization effect of noise on the training. As previously stated, noise amplitude is linked to signal amplitude. The variance of the sensor pixel outputs is used to estimate the signal change.

In the training data of set B, the temperature signal amplitude is equal to 4.9 ℃. The model's behavior is analyzed with a fraction of the noise amplitudes ranging from 0.01 % to 163.84 %. The noise level increases geometrically, with powers of 2, the main goal is to extend the DSE till an increasing trend for the losses can be observed then may need to refine the search around the possible inflection point. For each of the white noise amplitudes, random white noise was applied only to the training segment of set B and the random generator was initialized with a different value each time. Then the augmented training set would save as a batch that maintaining the original order of the samples of the segment. This step would repeat 10 times. The final training data for each amplitude should contain 10 training batches with noise and one original training segment without augmented, and the samples within one batch should not be randomized. Through the training, only one batch was selected randomly per epoch out of the 11 batches. Train the model the same as the previous baseline procedure that

2000 epochs and pick the training with the best validation result. The next step is to test the model for sets A, B, C, and D and compare the results with the baselines numerically in the table and plots. As shown in the Table 3.2 and the Figure 3.2 below.

Table 3.2: Augmented results of different white noise amplitudes that compared with the baseline in the amplitude range [0.01 %; 163.84 %].

white noise amplitude	set A	set B	set C	set D	average
baseline	**0.0589**	**0.0332**	**0.1962**	**0.1828**	**0.1178**
0.01 %	0.0568	0.0372	0.2101	0.2005	0.1262
0.02 %	0.0390	0.0385	0.1780	0.1346	0.0975
0.04 %	0.0478	0.0373	0.2084	0.2413	0.1337
0.08 %	0.0419	0.0398	0.1983	0.1425	0.1056
0.16 %	0.0424	0.0390	0.1558	0.1213	0.0896
0.32 %	0.0488	0.0378	0.1785	0.1499	0.1038
0.64 %	0.0339	0.0382	0.1800	0.1773	0.1074
1.28 %	0.0360	0.0389	0.1498	0.1170	0.0854
2.56 %	0.0341	0.0411	0.1427	0.1019	0.0800
5.12 %	0.0225	0.0411	0.1210	0.0887	0.0683
10.24 %	0.0190	0.0419	0.0723	0.0508	0.0460
20.48 %	0.0102	0.0439	0.0460	0.0628	0.0407
40.96 %	0.0136	0.0477	0.0738	0.0702	0.0513
81.92 %	0.0141	0.0471	0.0518	0.0560	0.0423
163.84 %	0.0145	0.0448	0.1272	0.0910	0.0694

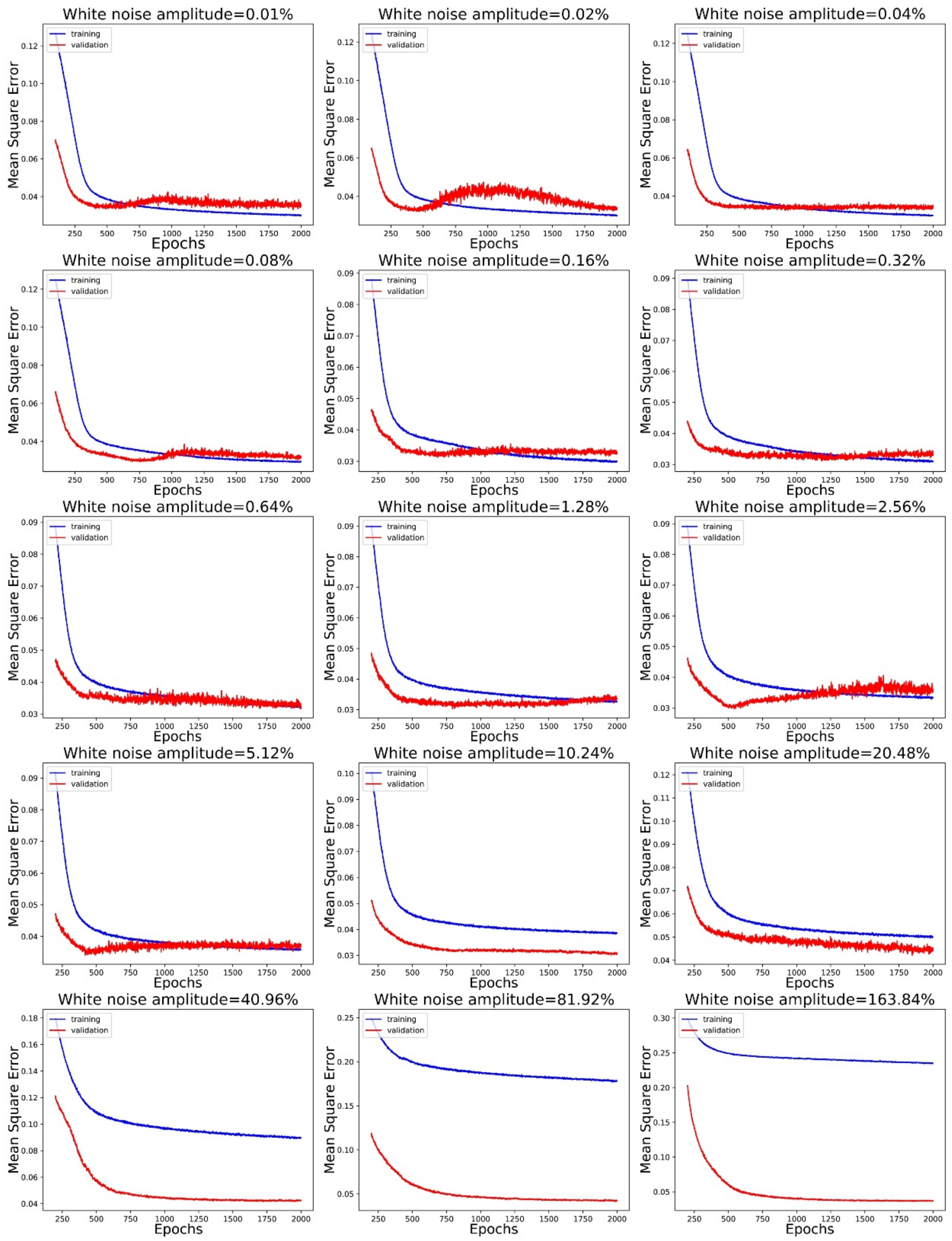


Figure 3.2: Learning curve results of augmented results by different white noise amplitudes in amplitude range [0.01 %; 163.84 %].

The MSE value of the four sets were calculated as a measure of generalization capability. General performance equals to average of augmented results. Main purpose is to find out the objective laws of optimization and the best optimization results. Thus, the most critical norm in the Table 3.2 is the comparison of the average MSE values of the model results obtained by augmented with the baseline average. Each set's results need to be further examined separately. The model would have enhanced generalization for all sets if at least the average value shows improvements in these amplitudes.

Through the numerical analysis of the MSE of the Table 3.2, the average value shows a downward trend as the amplitude increases which can prove the model was optimized. When analyzing the obtained learning curves in Figure 3.2, it can be seen that an increasing trend for the loss is observed at the result of 40.96 % white noise amplitude. So this part for the second exploration which is shown in the Table 3.3 and the Figure 3.3 below need to be focused on. In the second exploration, the model with the same previous procedure but in 10 additional subdivided amplitude in a geometric series was processed. The related plot of model performance that can visually assess the model output to identify the most promising parameter is also listed below.

Table 3.3: Augmented results of different white noise amplitudes compared with the baseline in the refined amplitude range [11.6 %; 36.1 %].

white noise amplitude	set A	set B	set C	set D	average
baseline	**0.0589**	**0.0332**	**0.1962**	**0.1828**	**0.1178**
11.60 %	0.0191	0.0425	0.0613	0.0495	0.0431
13.20 %	0.0178	0.0420	0.0619	0.0846	0.0516
14.90 %	0.0123	0.0431	0.0612	0.0426	0.0398
17.00 %	0.0124	0.0438	0.0458	0.0499	**0.0380**
19.20 %	0.0088	0.0434	0.0591	0.0555	0.0417
20.48 %	0.0102	0.0439	0.0460	0.0628	0.0407
21.80 %	0.0092	0.0434	0.0635	0.0472	0.0408
24.70 %	0.0104	0.0444	0.0598	0.0501	0.0412
28.10 %	0.0118	0.0457	0.0626	0.0789	0.0498
31.80 %	0.0118	0.0456	0.0577	0.0825	0.0494
36.10 %	0.0118	0.0459	0.0850	0.0599	0.0507

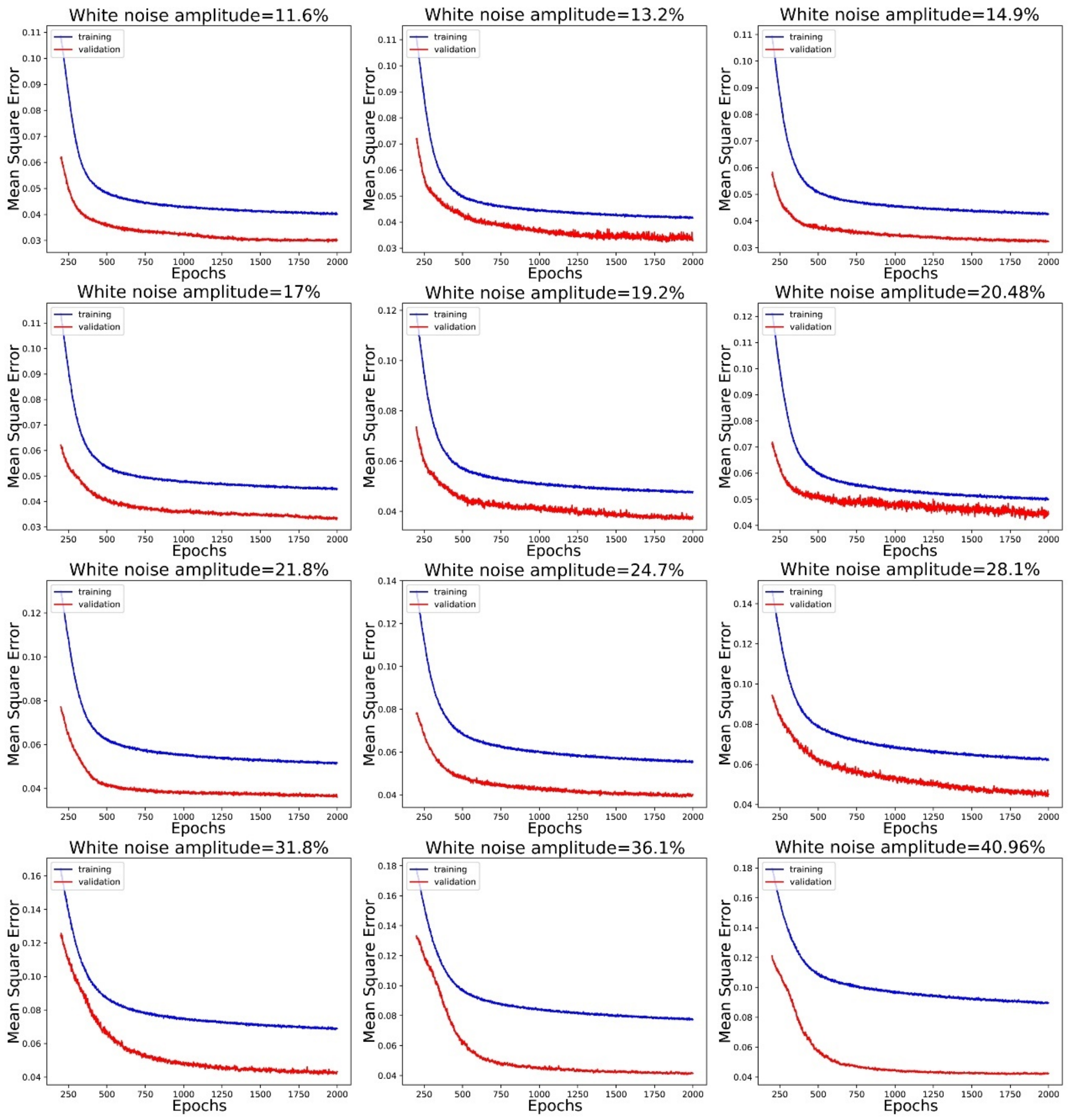


Figure 3.3: Learning curve results of augmented results by different white noise amplitudes in refined amplitude range [11.6 %; 40.96 %].

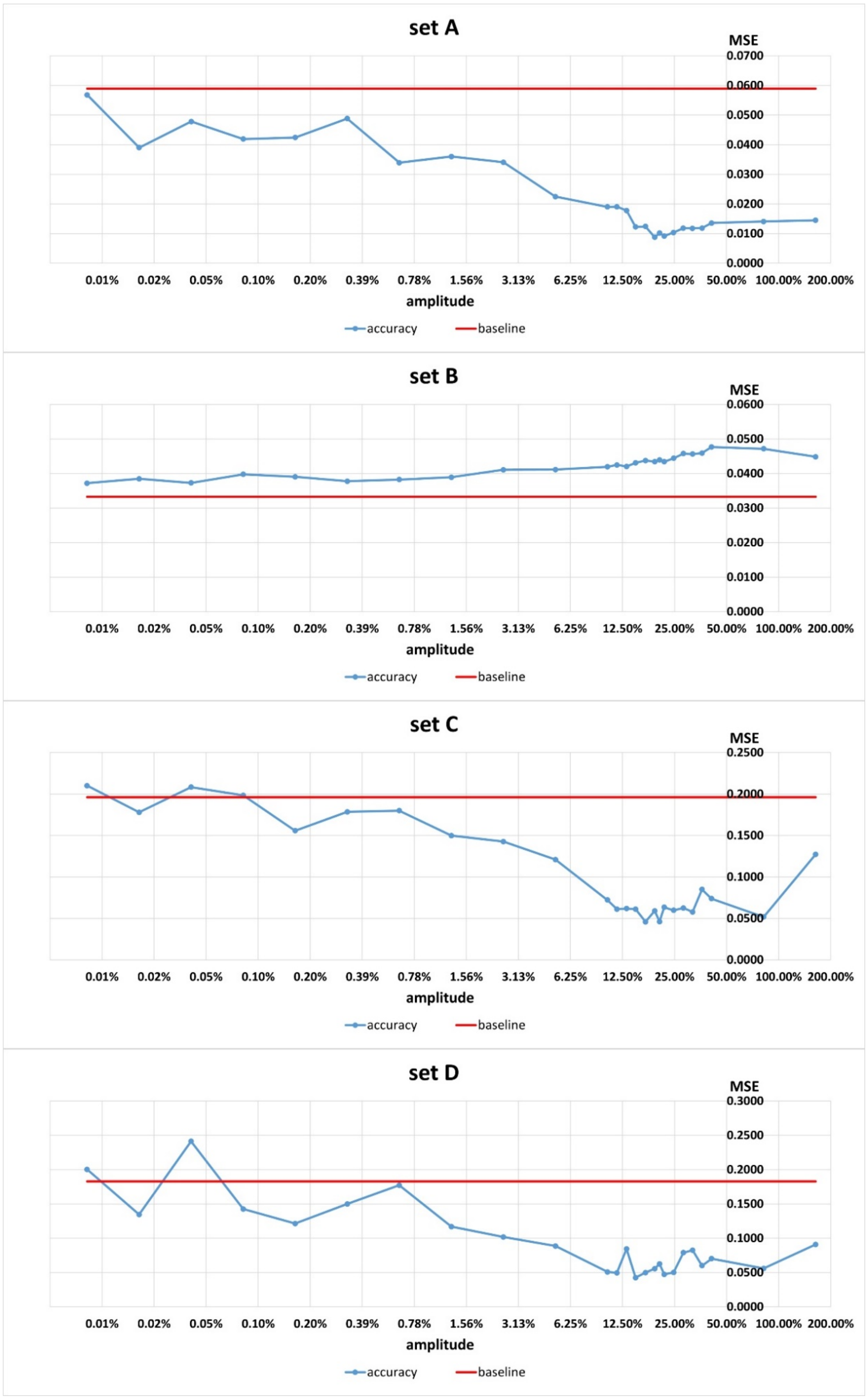


Figure 3.4: Plot of the training results obtained by augmentation with white noise amplitude range [0.01 %; 163.84 %] for all sets.

From Figure 3.4 above, the results of each set compared with the baseline can be observed conveniently. For set A, improvement has been obtained at all white noise amplitudes. The results before 1.28 % white noise amplitude are smoother, and then it will be improved faster. Smooth stabilization is obtained after 24.7 % white noise amplitude. By examining the plot of set B, all the results are not optimized. This trend is contrary to the decreasing trends of the MSE for all the other sets. But it also can be seen that the increase of the MSE of the set B with the increase of the noise is relatively small compared to the large reductions of the MSE with the increase of the noise for the other sets. Comparing the results of 25 % white noise amplitude, the optimization results of other sets are several times that of set B. Set C and set D can be bundled concurrently to discuss since the plot of these two sets acquired a similar trend. Less than ideal results would be acquired before 0.08 % white noise amplitude. But similar to the results of set A, these two sets can also be optimized after 1.28 % white noise amplitude.

Now that conclusion can be made based on the Figure 3.5 below and the related information above, a plot is created to evaluate the overall improvement of the model. The Figure 3.5 shows the noise amplitudes on the horizontal axis and the average of the MSE values for all sets on the vertical axis. The red line represents the model MSE baseline without augmented data. The overall behavior of the model exhibits a progressive reduction in MSE as the noise amplitude is increased. Beginning with few MSE results higher than the baseline, the model has made general progress after 0.08 % white noise amplitude. The model is actually optimized can be inferred. The improved average MSE across all sets serves as a measure of the generalization quality of the model. The most important white noise amplitude range is [11.6 %; 36.1 %] when white noise has a regularization effect in the learning curves starting at 10 % amplitude or so. In the last process, the white noise and brown noise would add together to the dataset. Since there is an optimum around 17 % white noise amplitude, so the 17 % white noise amplitude is chosen as the composite amplitude. Unable to locate a suitable amplitude range as a possible option for optimization for set B. However, given sets A, C, and D are at least operating satisfactorily, this can be accepted as a credible result providing greater model generalization quality.

To simplify the visual analysis of the results. Here is the relative optimize value figure of the loss function compared to the baseline in each set shown in Figure 3.6. It can be observed that the best optimization improvement occurs in set A when 19.2 % white noise amplitude, an increase of 85 %. In contrast, the least optimization occurs in set D when 13.2 % white noise amplitude, an improvement of 54 %.

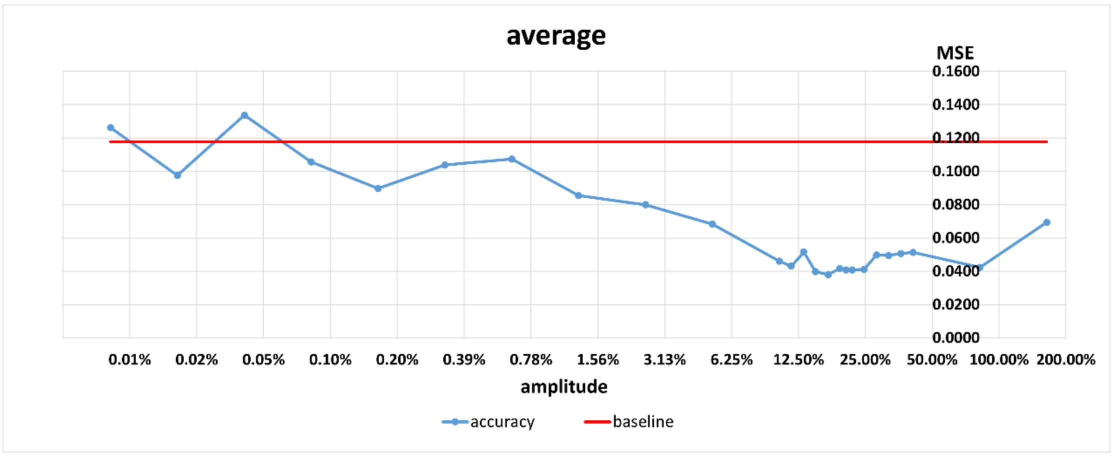


Figure 3.5: Plot of the model average generalization results with white noise augmentation.

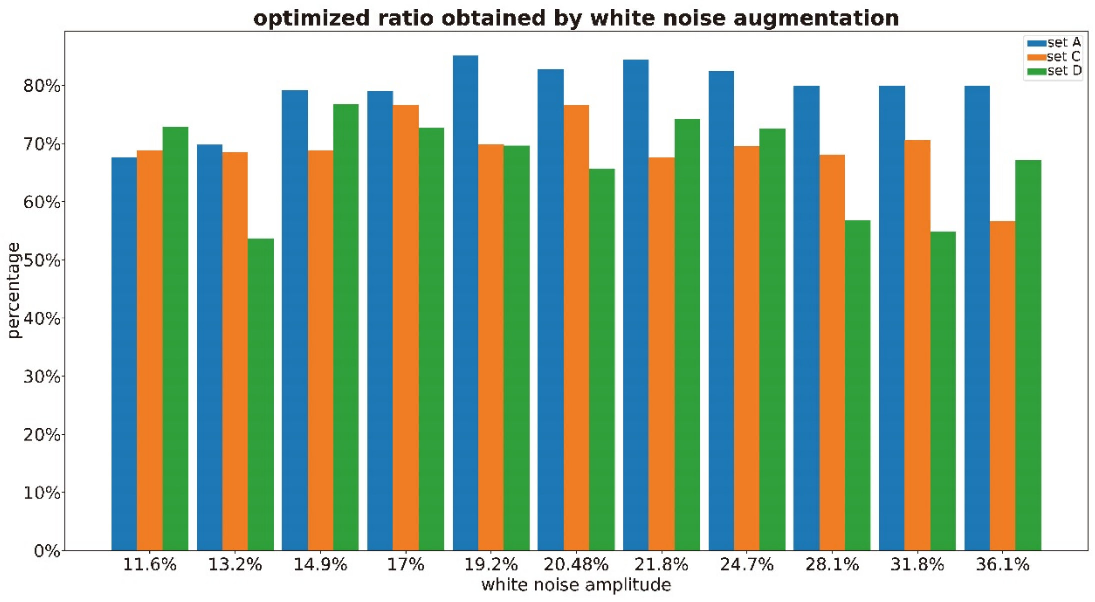


Figure 3.6: Relative optimize ratio in each optimized set by refined amplitude range of white noise.

3.3 Brown Noise Augmentation

The model was trained using augmented data produced by white noise in the previous section and is characterized by appropriate input data and architecture. As a second way to produce augmented training sets, the brown noise amplitude would be investigated. As discussed about white noise. By adding brown noise to augmented sets, the improvement of the model's generalization is the desired result. As mentioned in the previous section, the first idea regarding noise parameters relates to the signal variation between person detection and without detection. The maximum and minimum sensor pixel output variance values acquired from the sensor are used to calculate the signal deviation. The IR sensor's 16 pixels are all converted to the temperature inside the sensor.

Since the training NN model is a stochastic process. For the objective experimental procedure, the model's behavior was examined as same as Section 3.2’s procedure. The same amplitude range and model frame will be used, the only difference is that the added noise was switched from white noise to brown noise.

The model is trained with 11 batches for each noise amplitude in the range, and the best model is selected based on the minimum loss average of all trainings. The best training outcomes are presented in the Table 3.4 below, also as the plot in the Figure 3.7.

Table 3.4: Augmented results of different brown noise amplitudes that compared with the baseline in the amplitude range [0.01 %; 163.84 %].

brown noise amplitude	set A	set B	set C	set D	average
baseline	**0.0589**	**0.0332**	**0.1962**	**0.1828**	**0.1178**
0.01 %	0.0368	0.0378	0.1909	0.1620	0.1068
0.02 %	0.0435	0.0374	0.1457	0.1573	0.0960
0.04 %	0.0277	0.0391	0.0926	0.1020	0.0654
0.08 %	0.0196	0.0414	0.0855	0.0668	0.0534
0.16 %	0.0149	0.0431	0.0495	0.0479	0.0389
0.32 %	0.0117	0.0444	0.0568	0.0415	0.0386
0.64 %	0.0168	0.0477	0.0660	0.0493	0.0449
1.28 %	0.0254	0.0574	0.0565	0.0681	0.0519
2.56 %	0.0319	0.0567	0.0891	0.0827	0.0651
5.12 %	0.0354	0.0539	0.0962	0.0816	0.0668
10.24 %	0.0271	0.0490	0.2562	0.1286	0.1152
20.48 %	0.0268	0.0493	0.1649	0.0973	0.0846
40.96 %	0.0234	0.0483	0.1196	0.0949	0.0715
81.92 %	0.0302	0.0488	0.1509	0.1010	0.0827
163.84 %	0.0222	0.0490	0.1708	0.1123	0.0886

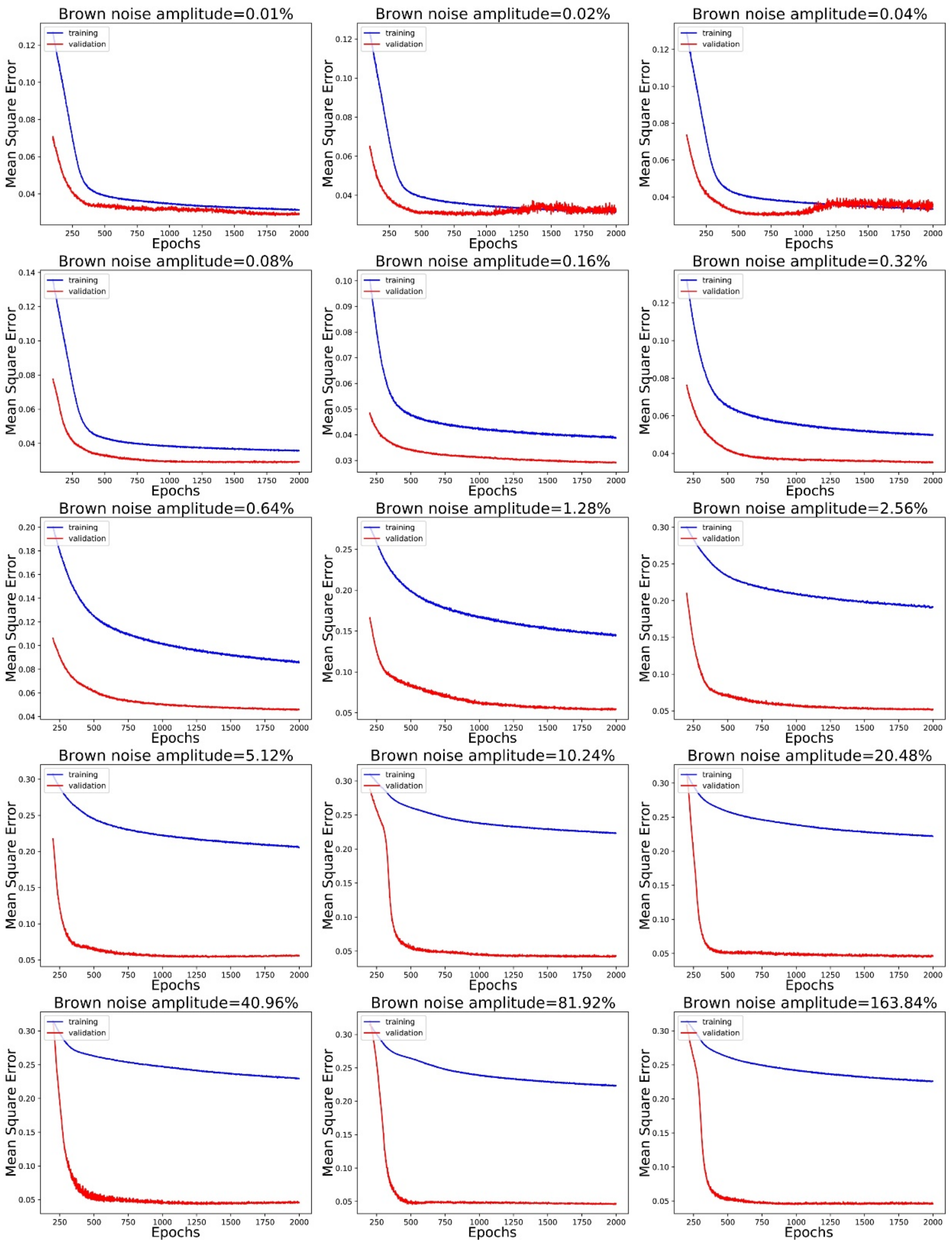


Figure 3.7: Learning curve results of augmented results by different brown noise amplitudes in amplitude range [0.01 %; 163.84 %].

Discuss the results from Table 3.4 and Figure 3.7 together, the resulting trend of brown noise is similar to that of white noise. Except for set B, different levels of optimization can be observed. The training regularization effect is clear from 0.08 % noise. Also from the learning curves can be observed that higher noise levels reduce the training accuracy but leave the validation accuracy largely unchanged, especially up to [0.16 %; 0.32 %] noise. In order to try to get better results, it is necessary to accomplish further subdivision experiments on the brown noise amplitude in the range [0.08 %;

0.64 %]. The related results are shown below.

Table 3.5: Augmented results of different brown noise amplitudes that compared with the baseline in the refined amplitude range [0.1 %; 0.59 %].

brown noise amplitude	set A	set B	set C	set D	average
baseline	**0.0589**	**0.0332**	**0.1962**	**0.1828**	**0.1178**
0.10 %	0.0260	0.0412	0.1153	0.0701	0.0632
0.12 %	0.0226	0.0420	0.0296	0.0428	0.0343
0.14 %	0.0240	0.0421	0.0766	0.0484	0.0478
0.16 %	0.0149	0.0431	0.0495	0.0479	0.0389
0.17 %	0.0118	0.0425	0.0435	0.0450	0.0357
0.20 %	0.0201	0.0434	0.0589	0.0474	0.0424
0.24 %	0.0111	0.0441	0.1183	0.0456	0.0548
0.29 %	0.0109	0.0432	0.0246	0.0416	**0.0301**
0.32 %	0.0117	0.0444	0.0568	0.0415	0.0386
0.34 %	0.0108	0.0447	0.0409	0.0426	0.0347
0.41 %	0.0144	0.0460	0.0166	0.0464	0.0309
0.50 %	0.0129	0.0455	0.0200	0.0445	0.0307
0.59 %	0.0195	0.0498	0.0296	0.0461	0.0363

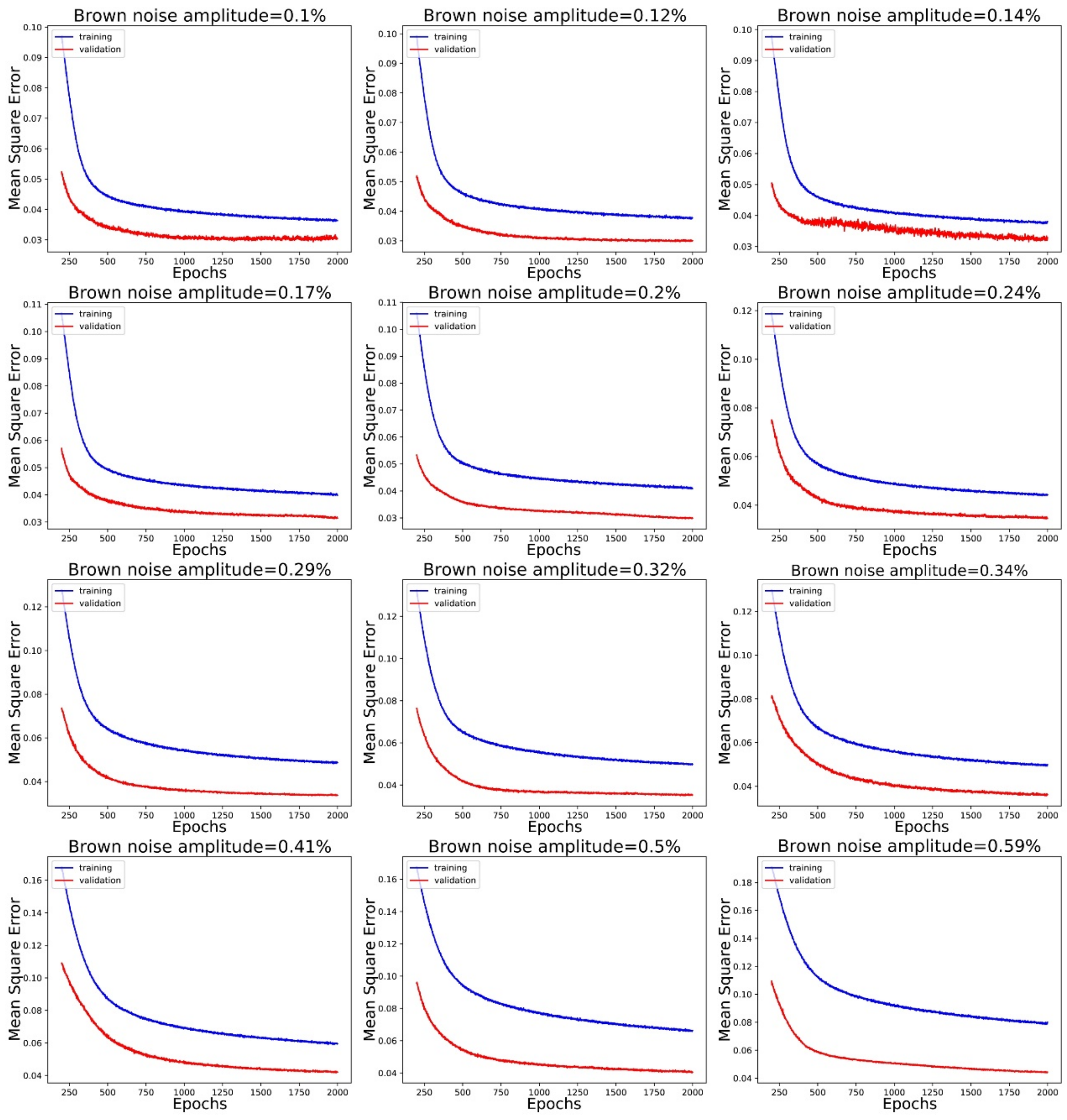


Figure 3.8: Learning curve results of augmented results by different brown noise amplitudes in refined amplitude range [0.1 %; 0.59 %].

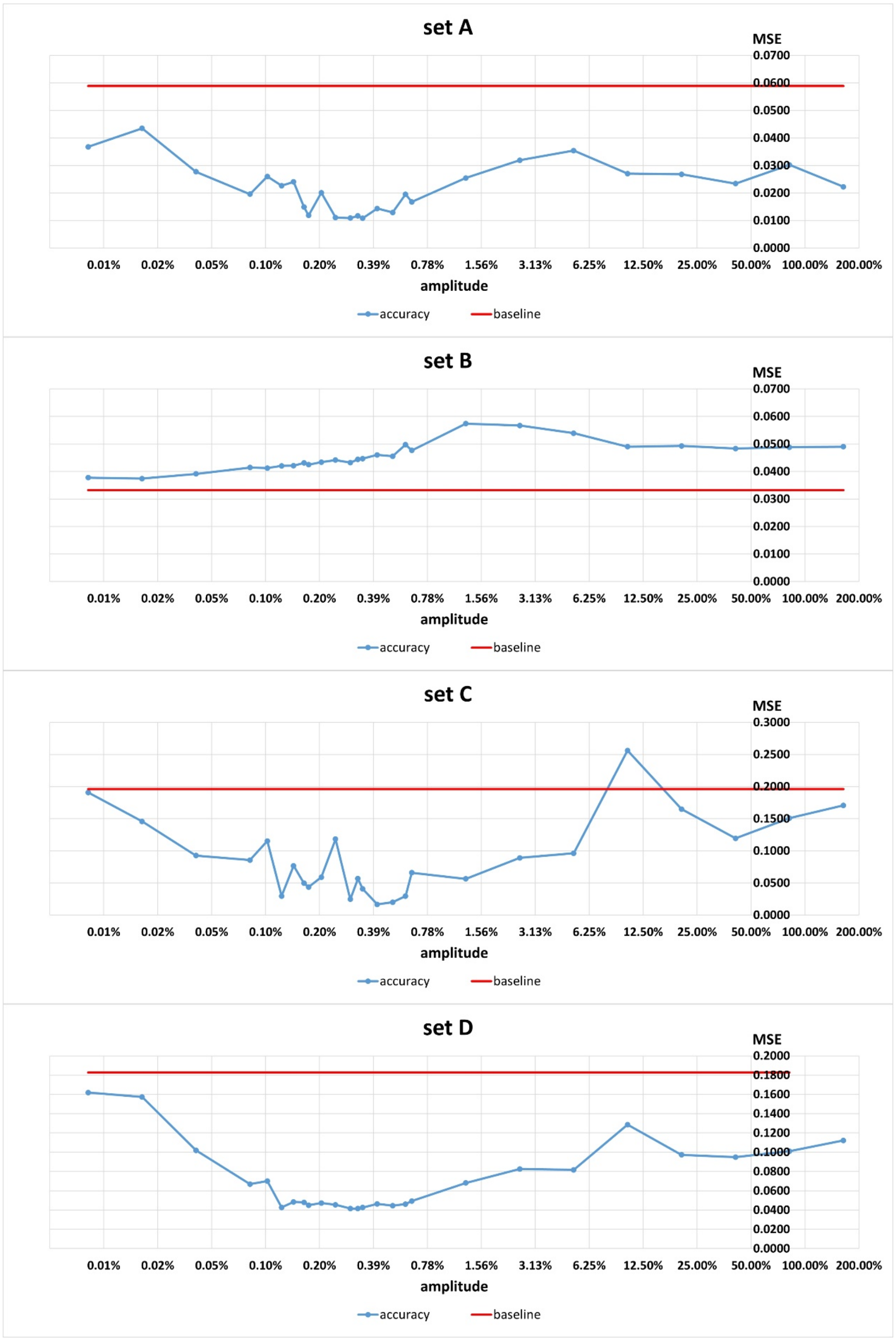


Figure 3.9: Plot of the training results obtained by augmentation with brown noise amplitude range [0.01 %; 163.84 %] for all sets.

From Figure 3.9 above, the results of each set compared with the baseline can be observed conveniently. Through observation, a similar trend to those obtained in section 3.1 can be obtained. The accuracy plots of the refined analysis for sets A, C, and D show an optimum around [0.3 %; 0.5 %]. Compared to those, set B is basically unchanged. For set A, improvement has been obtained at all brown noise amplitudes. Compared with other sets, the result of set A is smoother. By examining the plot of set B, all the results are still not optimized. This trend is contrary to the decreasing trends of the MSE for all the other sets. This is similar to the result of Section 3.2. Set C and set D can be bundled concurrently to discuss since the plot of these two sets acquired a similar trend. For example, both can be seen there is a sudden increase occurs at 10.24 % brown noise amplitude. The result of set C on 10.24 % is even worse than the baseline, but only this one unsatisfactory value. Generally, these two sets can also be optimized on all brown amplitudes. From Figure 3.8, the loss curves show the regularization effects of higher noise levels as happened for the white noise. Also showed that for noise levels beyond 0.35 % or so training for more than 2000 epochs would improve the accuracy since both training and validation curves are clearly improving at the end of the 2000 epochs.

Now that conclusion can be made based on the Figure 3.10 below and the related information above, a plot is created to evaluate the overall improvement of the model. Before the 0.64 % brown noise amplitude, the model is properly optimized as the amplitude increases. The value of MSE decreases accordingly. But after the result of 1.28 % brown noise amplitude, the overall MSE value of the model began to climb but was still lower than the baseline. The strange point occurred at the result of 10.24 % brown noise amplitude, the MSE value was the closest to the baseline of the average result but still below it. The model is actually optimized that can be inferred as same as white noise. The best improve brown noise amplitude range is [0.1 %; 0.59 %] when brown noise has a regularization effect in the learning curves starting at 0.32 % amplitude or so. Since there is an optimum around 0.29 % brown noise amplitude, so the 0.29 % brown noise amplitude is chosen as the composite amplitude.

To simplify the visual analysis of the results. Here is the relative optimize value figure of the loss function compared to the baseline in each set shown in Figure 3.11. It can be observed that the best optimization improvement occurs in set C when 0.41 % brown noise amplitude, an increase of 92 %. In contrast, the least optimization occurs also in set C when 0.24 % brown noise amplitude, an improvement of 40 %.

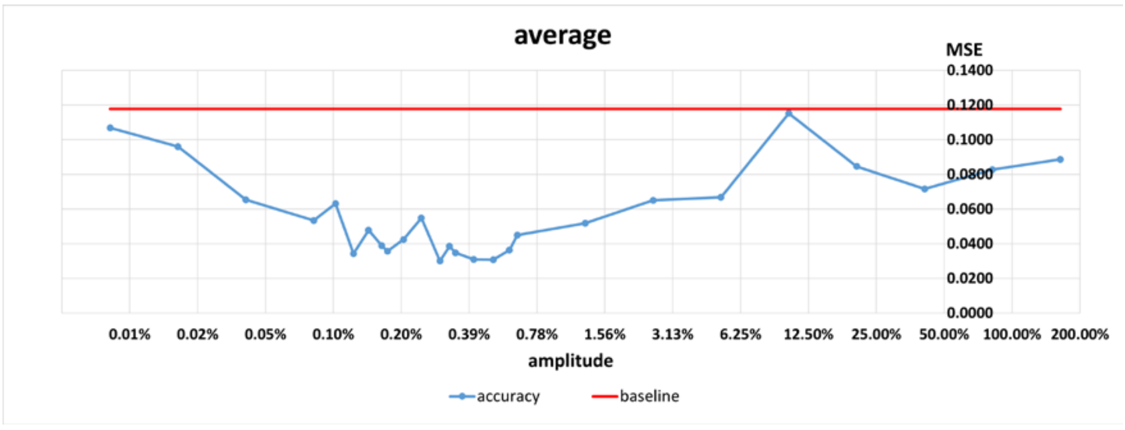


Figure 3.10: Plot of the model average generalization results with brown noise augmentation.

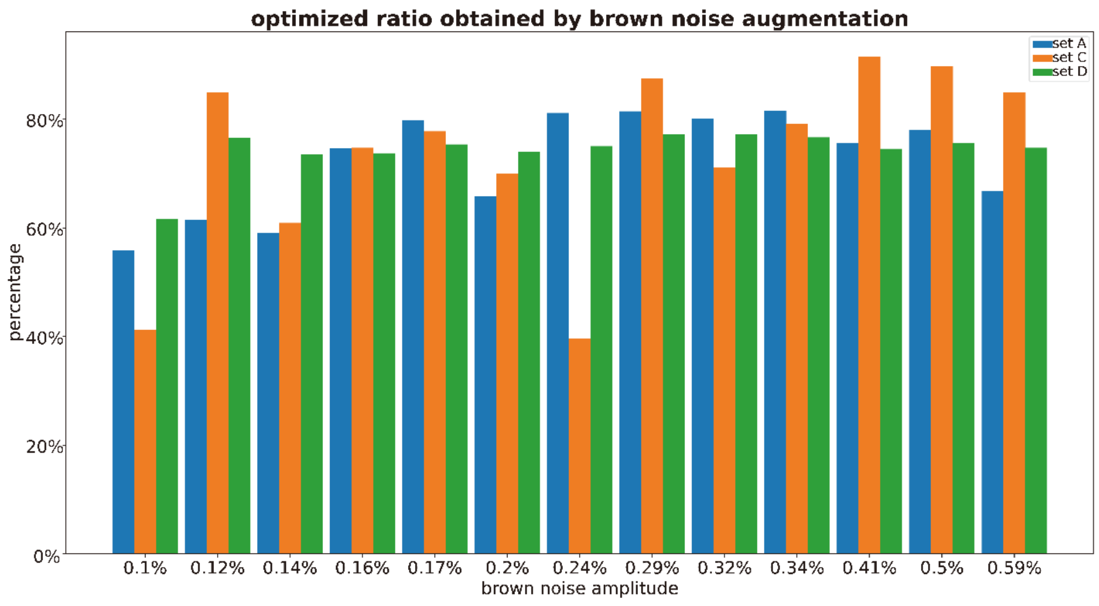


Figure 3.11: Relative optimize ratio in each optimized set by refined amplitude range of brown noise.

3.2 Combined Noise Augmentation

Various noises have an impact on all systems at once. White noise and brown noise effects on model behavior are discussed individually in the previous section. The impacts of brown and white noise fusions were investigated to optimize the NN model. By the outcomes of previous sections. For white noise, the model MSE is described shown in figures and plots that as same as previous sections using two parameters. The white noise amplitude and the brown noise amplitude. Firstly, the white and brown noise amplitude are divided as follows: white noise amplitude was chosen to be 17 %, which was the maximum noise quantity improving the generalization of the model as all, and brown noise amplitudes are considered to be between 0.08 % and 163.84 %. The model is trained with 11 batches in each combined parameter, and the best epoch result is selected after 2000 epochs.

Overall, The optimization results are not as good as the previous sections, but there are still a few optimization results worth discussing. In the results of set A, the ideal value is reached when the brown noise amplitude is 0.16 %, and MSE optimized by 16 %. After 0.16 % brown noise, there is no observable optimization. Except for the range mentioned, other brown noise amplitudes have worse MSE. In the previous sections, there is not any optimized result in set B. This characteristic has been retained, there has been no optimization in this section. The number of optimization results obtained by set C is only two that happened on 0.08 % and 0.32 % brown noise amplitude. Moreover, the result of 0.32 % brown noise amplitude in set C has the highest relative optimization ratio among all sets, reaching 63 %. This also makes this result the best on average. The trend of set D is similar to set C which the optimized results are concentrated before 0.64 % brown noise amplitude. The best relative ratio of optimization is 16 % occurred at 0.16 % brown noise amplitude. It can be seen in the Table 3.6 below that the model average performance has few improved throughout the range before 0.64 % by the impacts of set A, set C and set D. The best relative optimization ratio of average MSE value is 19 % occurred at 0.32 % brown noise amplitude by the impact by the result of set C. Figure 3.12 and Figure 3.13 below show the more intuitive way of visualizing the overall trend of the results. It can be seen that the trends of set A and set B are slightly similar, while the trends of set C and set D are slightly similar.

Table 3.6: Training results obtained by augmentation of fixed 17 % white noise amplitude with brown noise in amplitude range [0.08 %; 163.84 %] for all sets.

brown noise amplitude	set A	set B	set C	set D	average
17 % white noise amplitude	**0.0124**	**0.0438**	**0.0458**	**0.0499**	**0.0380**
0.08 %	0.0109	0.0443	0.0456	0.0453	0.0365
0.16 %	0.0104	0.0442	0.0525	0.0415	0.0372
0.32 %	0.0164	0.0466	0.0170	0.0426	**0.0307**
0.64 %	0.0183	0.0514	0.0655	0.0530	0.0470
1.28 %	0.0249	0.0548	0.0684	0.0645	0.0532
2.56 %	0.0337	0.0570	0.0911	0.0780	0.0650
5.12 %	0.0300	0.0552	0.1089	0.0988	0.0732
10.24 %	0.0268	0.0518	0.2635	0.1589	0.1253
20.48 %	0.0296	0.0464	0.1320	0.0836	0.0729
40.96 %	0.0188	0.0478	0.1306	0.1047	0.0755
81.92 %	0.0246	0.0470	0.1656	0.1046	0.0854
163.84 %	0.0217	0.0490	0.1694	0.1177	0.0895

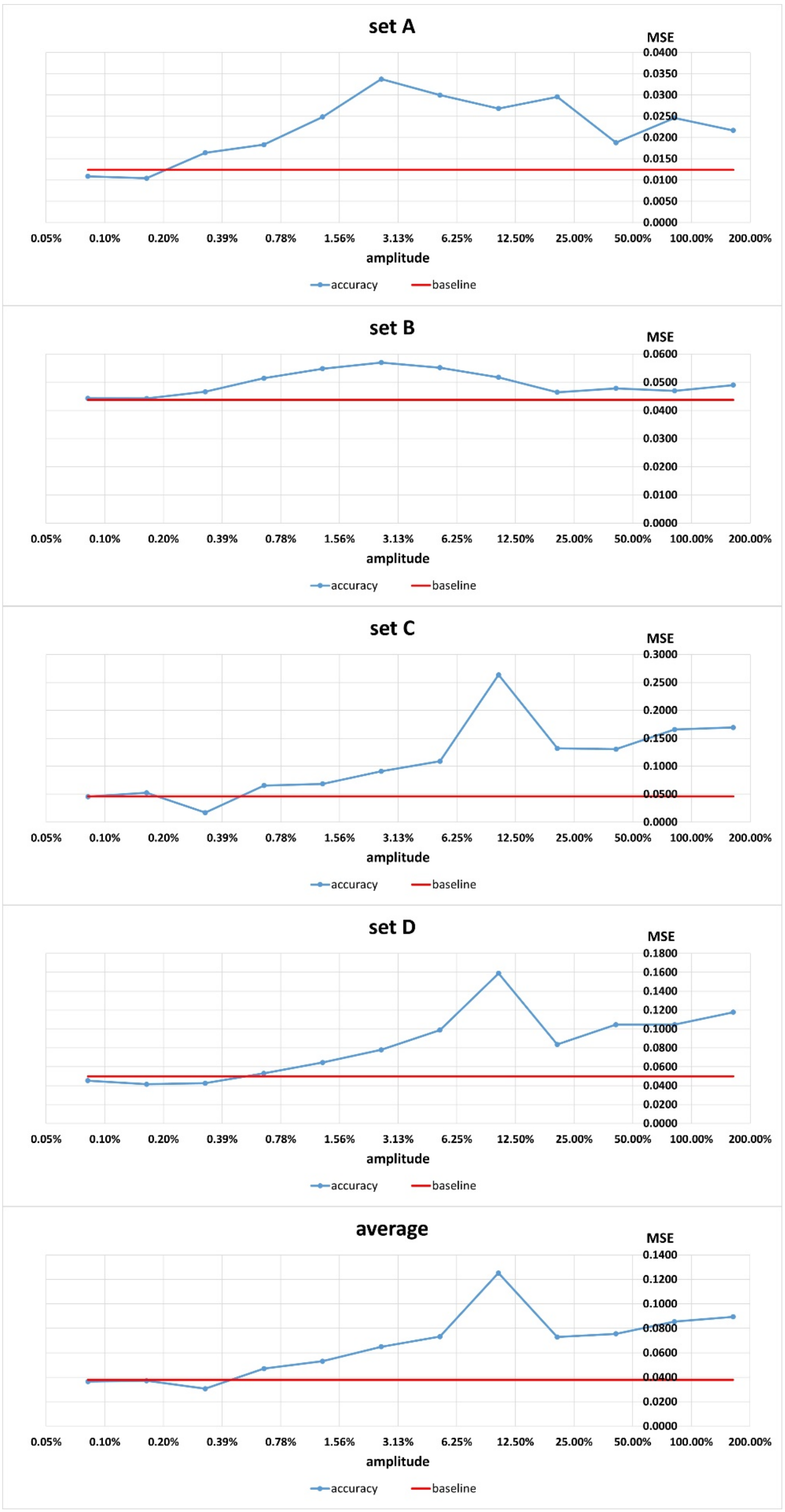


Figure 3.12: Plot of the training results obtained by combined noise augmentation of fixed 17 % white noise amplitude with brown noise in amplitude range [0.08 %; 163.84 %] for all sets.

Second, the white and brown noise amplitude are divided as follows: brown noise amplitude is chosen to be 0.29 %, which is the best in the DSE for brown noise, and white noise amplitudes are between 0.08 % and 163.84 %.

The result of fixed 0.29 % with different levels of white noise amplitude is worse than the result of the previous combination noise. There is no optimization in set A. Only one in set B, set C and average results. Moreover, the best relative optimization ratio also occurs at 1.28 % white noise in set C, reaching 17 %. A relatively acceptable result occurs in set D, there is observable optimization in the white noise amplitude range [0.08 %; 1.28 %] but with really small improvement.

Table 3.7: Training results obtained by augmentation of fixed 0.29 % brown noise amplitude with white noise in amplitude range [0.08 %; 163.84 %] for all sets.

white noise amplitude	set A	set B	set C	set D	average
0.29 % brown noise amplitude	**0.0109**	**0.0432**	**0.0246**	**0.0416**	**0.0301**
0.08 %	0.0121	0.0439	0.0248	0.0404	0.0303
0.16 %	0.0113	0.0440	0.0429	0.0403	0.0346
0.32 %	0.0111	0.0432	0.0306	0.0399	0.0312
0.64 %	0.0114	0.0434	0.0312	0.0395	0.0314
1.28 %	0.0114	0.0436	0.0204	0.0404	**0.0289**
2.56 %	0.0112	0.0437	0.0362	0.0404	0.0328
5.12 %	0.0121	0.0438	0.0932	0.0611	0.0525
10.24 %	0.0122	0.0444	0.0728	0.0501	0.0449
20.48 %	0.0144	0.0476	0.0447	0.0473	0.0385
40.96 %	0.0194	0.0504	0.0317	0.0548	0.0391
81.92 %	0.0147	0.0482	0.0368	0.0559	0.0389
163.84 %	0.0146	0.0470	0.0565	0.0790	0.0493

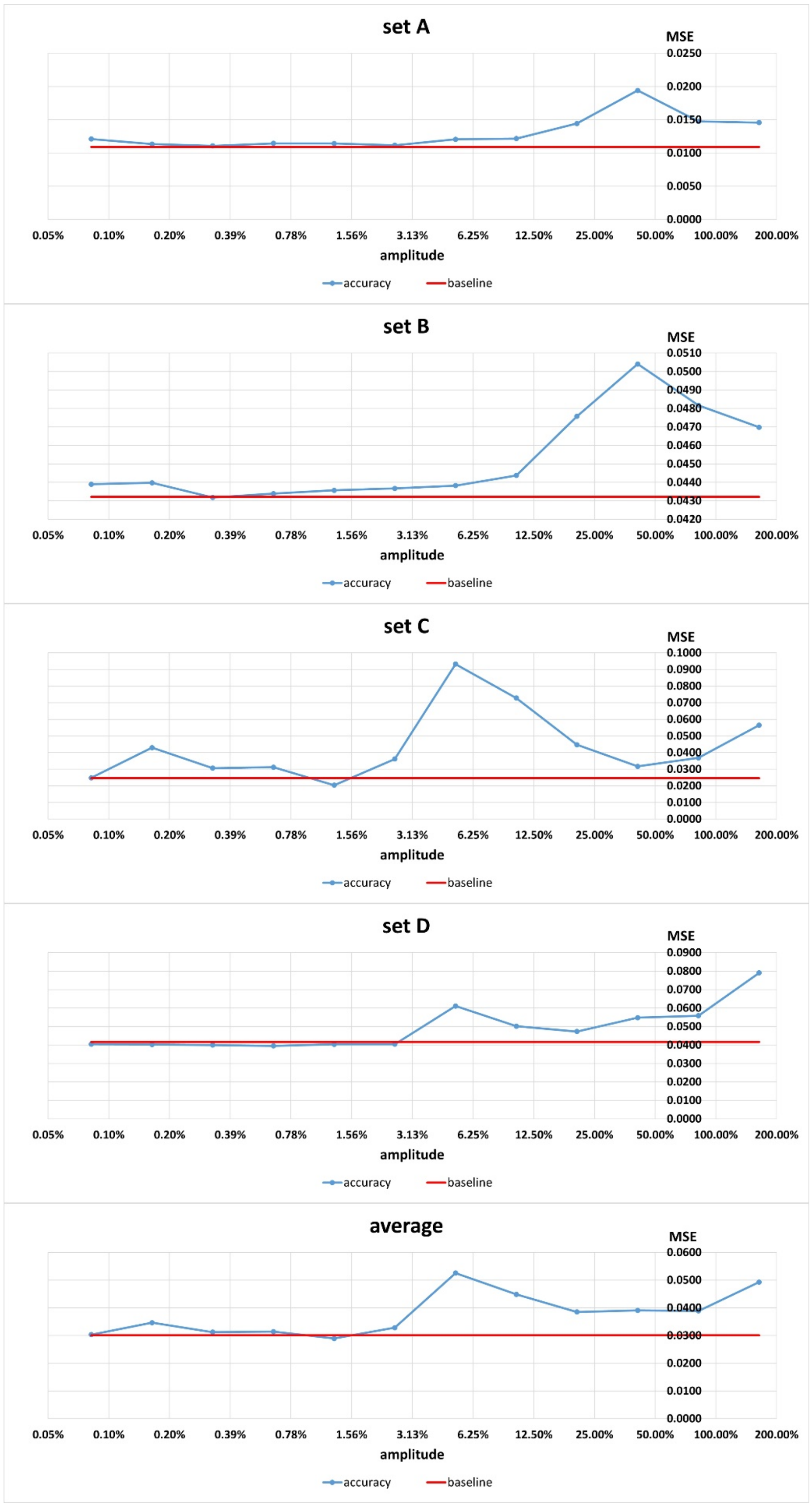


Figure 3.13: Plot of the training results obtained by combined noise augmentation of fixed 0.29 % brown noise amplitude with white noise in amplitude range [0.08 %; 163.84 %] for all sets.

Finally, the individual results from previous Table 3.6 and Table 3.7 can be discussed to evaluate the model's overall generalization performance in combination results. The baseline is set as the results of each best result in all set's MSEs in previous Section 3.2 and Section 3.3, and overall results are acquired as the average of all four sets. All the NN model procedure in Chapter 3 is basically similar.

Most MSE values are worse than the baseline in Section 3.3, various from the results of other Sections in Chapter 3. It demonstrates that, generally speaking, after the combination of brown noise and white noise as the procedure used in this thesis, the result only has a few positive effects on the model. The model's best generalization of the overall average is found at the fixed 0.29 % brown noise amplitude with 1.28 % white noise amplitude, where the lowest MSE value in average of four sets occurs. The MSE value for this parameter is 0.0289.

In comparison, only adding white noise or brown noise to the original data as the procedure used in this thesis can get the more ideal generalization performance. From the average value of relative optimization ratio in refined range's optimization. 62 % in white noise and 66 % in brown noise can be observed, a little bit close but brown noise is slightly better.

Chapter 4
Conclusion

The implementation of machine learning techniques in numerous fields is simplifying the resolution of complicated issues. Numerous industries, including those in healthcare, transportation, security, and industry, use various machine learning algorithms. While NN typically produces satisfactory results, augmentation may be used in some circumstances to improve the model's poor regularization. In this thesis, data augmentation strategies were investigated as a means of enhancing model performance by actually increasing the quantity of the input data by noise. The majority of augmentation methods used in computer vision are used on images. The most common approaches include random rotation, zooming, cropping, and flipping. The resolution of each frame produced by the sensor is relatively low since a 16 pixel IR sensor is used to study human indoor localization in a constrained 3x3m area. Per each sensor frame, the corresponding person location's X and Y coordinates are also gathered. A total of 30 minutes of person tracking at 5 Hz results in approximately 9000 tuples of experimental results, each of the original data is made up of the outputs of 16 IR sensor values and their related X and Y positions. Four independent sets of experimental data produced under various circumstances and at various room temperatures have been gathered. The NN is trained using the second set of experiments, and the remaining three sets are employed to evaluate the model regularization and generalization. Because sensor data could be impacted by environmental noise, data augmentation by noise is used to enhance the overall model inference.
The first sort of noise is white noise, which is a typical noise that affects everything around us. The second type of noise, brown noise, is a more stochastic noise. In order to get successful outcomes, various white and brown noise levels were investigated. Both noise amplitude ranges between [0.01 %; 163.84 %], the noise level increases geometrically with powers of 2, are examined to verify the generalization of the model inference. Figures in the Section 3.2 shows that the range [11.6 %; 36.1 %] with at least three sets of improvements is corresponds to the model behaving overall the best. Considering overall performance, the 17 % white noise amplitude can be seen as the best choice to investigate further. The same amplitudes range and the same model procedure for brown noise are used. As illustrated in the Section 3.3, the refined and most promising amplitude range is [0.1 %; 0.59 %]. The generalization and regularization of the model can be improved in general. In order to illustrate the

model properties for white and brown noise amplitudes in combination, the investigation of two noise combinations is completed in the Section 3.4. First is fixed 17 % white noise amplitude with the brown noise amplitude range [0.08 %; 163.84 %] when the next one is fixed 0.29 % brown noise amplitude with the white noise amplitude range [0.08 %; 163.84 %]. Compared with adding only one kind of noise, the result of combination noise in the Section 3.4 has only been optimized by a few.

Bibliography

[1] A. Ng, Machine Learning, Coursera, 2022.

[2] O. B. Tariq, M. T. Lazarescu and L. Lavagno, " Machine Learning Techniques for Device Free Indoor Person Tracking," Ph.D. dissertation, DET, Politecnico di Torino, Turin, TO, 2021.

[3] F. Alam, N. Faulkner and B. Parr, "Device-Free Localization: A Review of Non-RF Techniques for Unobtrusive Indoor Positioning," in IEEE Internet of Things Journal, vol. 8, no. 6, pp. 4228-4249, 15 March 15, 2021, doi: 10.1109/JIOT.2020.3030174.

[4] C. Kowalski, K. Blohm, S. Weiss, M. Pfingsthon, P. Gliesche, A. Hein, "Multi Low-resolution Infrared Sensor Setup for Privacy-preserving Unobtrusive Indoor Localization," Conference: 5th International Conference on Information and Communication Technologies for Ageing Well and e-Health, 2019, pp. 183-188, doi: 10.5220/0007694601830188.

[5] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," in Neural Computation, vol. 9, no. 8, pp. 1735-1780, 15 Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[6] C. Olah (2015, Aug). Understanding LSTM Networks [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[7] G. An, "The Effects of Adding Noise During Backpropagation Training on a Generalization Performance," in Neural Computation, vol. 8, no. 3, pp. 643-674, April 1996, doi: 10.1162/neco.1996.8.3.643.

[8] Wikipedia, Normal Distribution. [Online]. Available: https://en.wikipedia.org/wiki/Normal_distribution

[9] D6t MEMS Thermal Sensors, Accessed: Dec. 1, 2020. [Online]. Available: https://components.omron.com/us-en/products/sensors/D6T?partNumber=D6T

[10] Starter Set HW v4.9, Accessed: Dec. 10, 2019. [Online]. Available: https://marvelmind.com/product/starter-set-hw-v4-9/

[11] O. B. Tariq, M. T. Lazarescu and L. Lavagno, "Neural Networks for Indoor Person Tracking With Infrared Sensors," in IEEE Sensors Letters, vol. 5, no. 1, pp. 1-4, Jan. 2021, Art no. 6000204, doi: 10.1109/LSENS.2021.3049706.

[12] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," in The journal of machine learning research, 15(1), 15 June, 2014.

[13] D. Kingma, J. Lei, "Adam: A method for stochastic optimization," in International Conference on Learning Representations, 2015, arXiv:1412.6980.