Politecnico di Torino

Master Degree Course in Mechatronic Engineering

Master Degree Thesis

A.A. 2021/2022

July 2022

Improving generalization of LSTM NNs with data augmentation using the capacitive sensor data

Supervisors:

Prof. Luciano Lavagno

Prof. Mihai Lazarescu

Candidate:

Lei Yuping

Summary

Machine learning is a general term for a class of algorithms that attempt to extract the implied laws from a large amount of historical data and use them for prediction or classification. More specifically, machine learning can be seen as finding a function where the input is sample data and the output is the desired result, except that this function is so complex that it is less convenient to express formally. It is important to note that the goal of machine learning is to make the learned function work well for the "new sample", not just to perform well on the training sample. The ability to apply the learned function to new samples is called generalization. Neural network is a machine learning model, in this thesis work, it will be used to predict the trajectory of a person in a 3m * 3m room.

We used four capacitive sensors operating in loading mode for indoor human localization, each one with a sensing plate installed at chest level in the center of a wall of the room. These sensors provide reading five times per second. The ground truth was obtained using four ultrasound anchors that can localize a mobile tag with ± 2 cm accuracy at 15Hz. When a person wears the tag and walks around the room, the ultrasound anchors record the “reference” trajectory. At the same time, the capacitive sensors also records position-related data.

We conducted four experiments at different times to obtain four data sets. Different environmental conditions (temperature, humidity and so on) affected differently each data set measurements. Each experiment has a different walking velocity and a different trajectory, but they all lasted 30 minutes.

To improve the reliability of the tracking system, we process the capacitive sensor data using a neural network (NN). The NN I used is a 2 layers LSTM with 8 cells per hidden layer. However, the generalization ability of the neural network is not so good because we don't have sufficient experimental data. To solve this problem, I used data augmentation technique. Data augmentation is a technique that artificially extends the training data set by allowing limited data to produce more equivalent data. It is an effective means to overcome the shortage of training data. Common data

augmentation methods based on image processing techniques include flipping, cropping, rotation, translation, noise injection, etc. In my case, I used the method of noise injection.

In this thesis, I chose two types of noise to add to the training set: white Gaussian noise and pink noise. This is because white Gaussian noise is commonly found in the environment, while pink noise simulates the drift of a capacitive sensor. In addition, since noise is usually a mixture of these two types of noise in daily life, this thesis also considers combining these two types of noise together in different ways to form mixed noise. The parameter to be determined for white noise is the variance, while for pink noise it is amplitude.

After data pre-processing and determined the most performing data set and the best train-validation-test sequence permutation of that set (more details in Chapter 3), I began the exploration of data augmentation. In order to determine whether a training result has improved and by how much, I created a baseline for comparison. Using the original training set, repeat the training 30 times. Because the initial weights are randomized for each training, the results of the training will be different. Taking the best among 30 trainings as the baseline. To select the best, the first criterion is that the sum of four test MSEs is low, the second criterion is the standard deviation of them is low.

After build the baseline, I started to improve the NN training using data augmentation. I added noise to the original training set via MATLAB. Before adding noise, I rescale the data, and after adding the noise, I rescale again so that the noisy data is between 0 and 1, to improve the learning ability of the neural network. After exploration for optimization (Section 3.4), I decided to create 6 training sets (5 noisy sets + 1 original set), each training set becomes one batch for batch training, with one batch at a time, to preserve the proper temporal sequence of the experimental data. In white and pink noise part, I started from a parameter (variance for white Gaussian noise, amplitude for pink noise) value lower than the smallest signal, around 10 ppm or below, then gradually increase. For each noise parameter, the NN is trained 30 times with the same data set. I pick the best among these 30 independent trainings for comparison.

After that, combining white and pink noise together in different ways to form mixed noise. Here are the steps:

- First, selecting the two best-performing variance values in the white noise experiment, fixing them, and then varying the amplitude of the pink noise (around the best two amplitudes of the

pink noise). Each combination of parameters is trained 30 times and pick the best.

- Second, selecting the two best-performing amplitude values in the pink noise experiment, fixing them, and then varying the variance of the white noise (around the best two variances value of the white noise). Each combination of parameters is trained 30 times and pick the best.

After all these experiments, I have come to this conclusion: the most significant improvement was achieved when using mixed noise. The improvement of sum of test MSEs respect to the baseline is 9.91%, the improvement of standard deviation is 12.55% (Table 1). I achieved the goal of improving the generalization of NN.

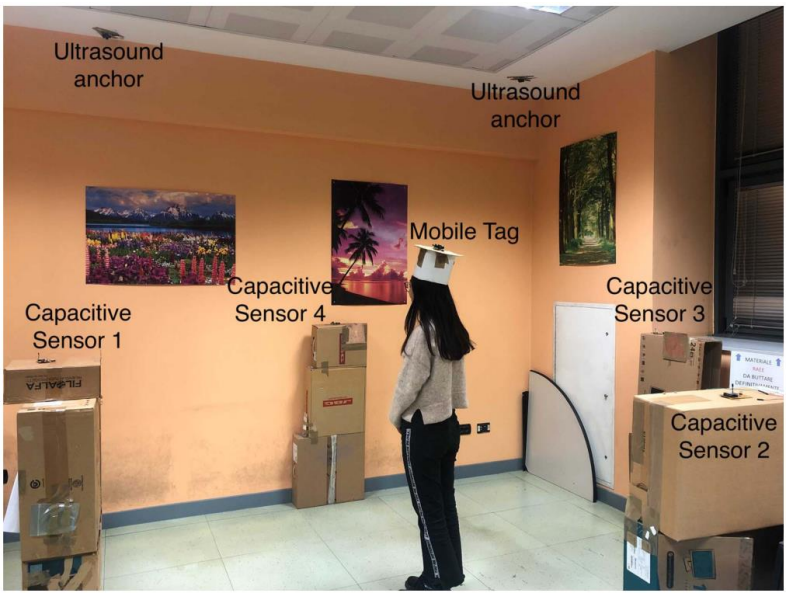


Figure 1 Virtual room used for movement tracking experiments [1]

Table 1 Comparison of the best results for white noise (variance = 5.00E-06), pink noise (amplitude = 5.00E-05) and mixed noise (variance = 5.00E-05, amplitude = 1.00E-05)

	Test MSE A $[m^2]$	Test MSE B $[m^2]$	Test MSE C $[m^2]$	Test MSE D $[m^2]$	Improvement of Standard Deviation of test MSEs $[m^2]$	Improvement of Sum of test MSEs $[m^2]$
Baseline	0.0526	0.3098	0.2486	0.1798	0%	0%
White Noise	0.0509	0.2917	0.2412	0.1362	2.36%	8.95%
Pink Noise	0.0502	0.2997	0.2635	0.1358	-4.52%	5.25%
Mixed Noise	0.0612	0.2692	0.2441	0.1379	12.55%	9.91%

Contents

List of Tables

List of Figures

Chapter 1

Indoor person localization

1.1 Review of localization techniques

Low-cost, low-maintenance, accurate indoor detection and localization of persons has an increasingly important role to play today. Not only can it provide services for people's convenience, but it can also play a role in the field of health care. For example, in the area of health care, studies have shown that by 2050, the number of elderly people worldwide is estimated to be close to 2.1 billion, more than double the number in 2015 [2]. With so many elderly people, it is vital to monitor the trajectory of the elderly, especially those living alone. Through the monitoring of trajectories, the system can determine whether the elderly have abnormal conditions, so that timely alerts can be issued and the elderly can receive timely help.

Over the years, many indoor positioning techniques have been proposed. For example, video or imaging cameras, ultrasonic sensors, infrared sensors, etc. They have different drawbacks.

Camera: it captures the image or video, and the system processes the image or video to calculate the location of the person. It requires high computational, networking and energy resources, a direct line of sight, and adequate lightning, which increase the installation complexity and system cost. It also raises significant privacy concern.

Ultrasonic sensor: consist of two parts, a transmitter and a receiver. Usually, they are placed side by side as much as possible. The transmitter emits ultrasonic in a certain direction and starts timing at the same time as the moment of emission. The ultrasonic travel through the air and return immediately when it hits an obstacle on the way, and the receiver stops timing as soon as it receives the reflected ultrasonic. The distance of the object can be calculated by measuring the time it takes for the ultrasonic to be reflected by the object. Ultrasonic systems require the user to carry a tag and long-term exposure to ultrasonic noise can cause harmful health effects.

Infrared sensor: The human body temperature is generally at 36.5 degrees, it emits infrared rays with a wavelength of about 10um. The infrared sensors work by detecting infrared rays of about 10um emitted by the human body. However, they are sensitive also to non-human heat sources, which increase the chance of false detection.

Capacitive sensing is also used for human detection, localization and identification. It is exactly what we used in our experiments. I describe it in more detail in Section 1.2.

1.2 Capacitive sensors for indoor localization

Capacitive sensing is a technology that can be used to detect and track conductive and non-conductive objects. Capacitive sensors use capacitive transducers that can operate in three different modes (or configurations): transmit, shunt, and load. The first two modes are introduced by Zimmerman et al. [3] in 1995. They can be used for indoor human detection. However, the transducer operating of both models require at least two galvanically coupled plates, which significantly increases the complexity and cost of the installation. Thankfully, the load operation mode uses the human body as a constant-potential plate (Figure 1.1) and requires just one-plate transducers, which is a good solution to the previously mentioned problem. Some of their advantages, such as low cost, easy installation, no major privacy concerns, etc., give us a reason to choose them.

The capacitance depends primarily on the geometry of the plate, dielectric properties of the system , distance between human body and the plate. We indirectly measure the changes in the capacitance of the sensor by measuring the free running frequency of an astable multivibrator, which repeatedly charges and discharges the capacitor of the sensor.

Our indoor localization experiments are conducted in a 3m * 3m room. We used four capacitive sensors operating in loading mode for indoor human localization, each one with a sensing plate installed at chest level in the center of a wall of the room (Figure 1.2). These sensors provide reading five times per second. The ground truth was obtained using four ultrasound anchors that can localize a mobile tag with $\pm$2 cm accuracy at 15Hz. When a person wears the tag and walks around the room, the ultrasound anchors record the “reference” trajectory. At the same time, the capacitive sensors also record position-related data.

To investigate the measurement of capacitive sensors under different conditions, we conducted four

experiments at different times to obtain four data sets. Different environmental conditions (temperature, humidity and so on) affected differently each data set measurements. Each experiment has a different walking speed and a different trajectory, but they all lasted 30 minutes.

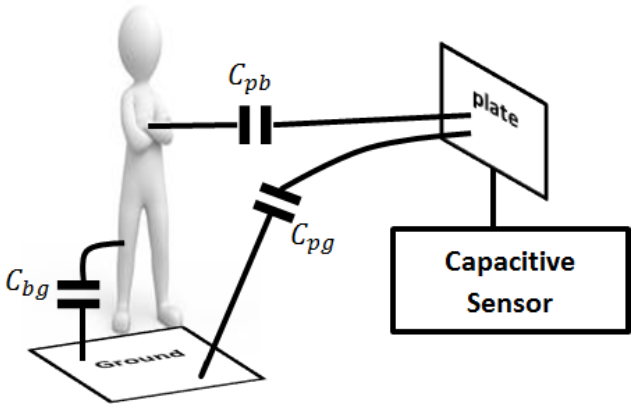


Figure 1.1 Main sensor capacitances in load mode: plate-body (C_{pb}), plate-ground (C_{pg}), and body-ground (C_{bg}) [4]

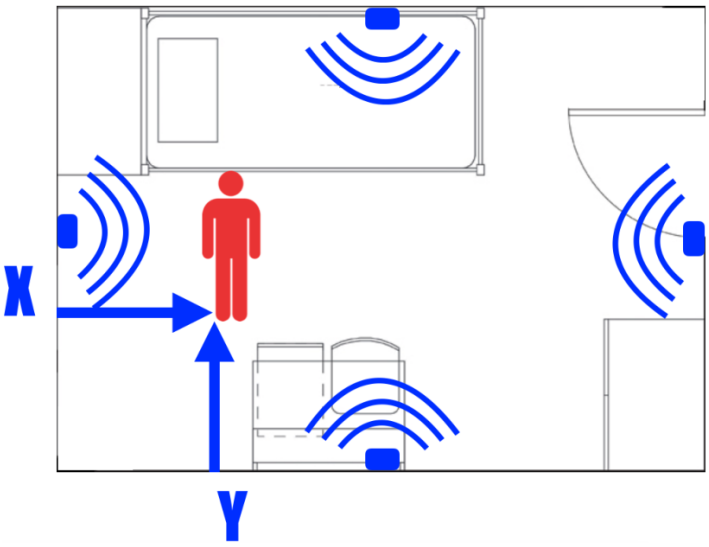


Figure 1.2 Four capacitive sensors centered on the walls of a 3 m×3 m virtual room in the lab trace the position of a person moving in the space [1]

1.3 Capacitive sensor module and data acquisition system

Each sensor has a copper clad plate attached as the external capacitor to a 555 integrated circuit in astable multivibrator configuration, for which the oscillation frequency is:

$$Frequency = \frac{1}{0.7(R_1 + 2R_2)C}$$

where $R_1 = 200k\Omega$ and $R_1 = 560k\Omega$.

We used an Arduino Uno board to measure the frequency, and an XBee 802.15.4 modem to transmit the measurements to a central node for post-processing and person localization. The block diagram of our localization system is shown in Fig. 1.3.

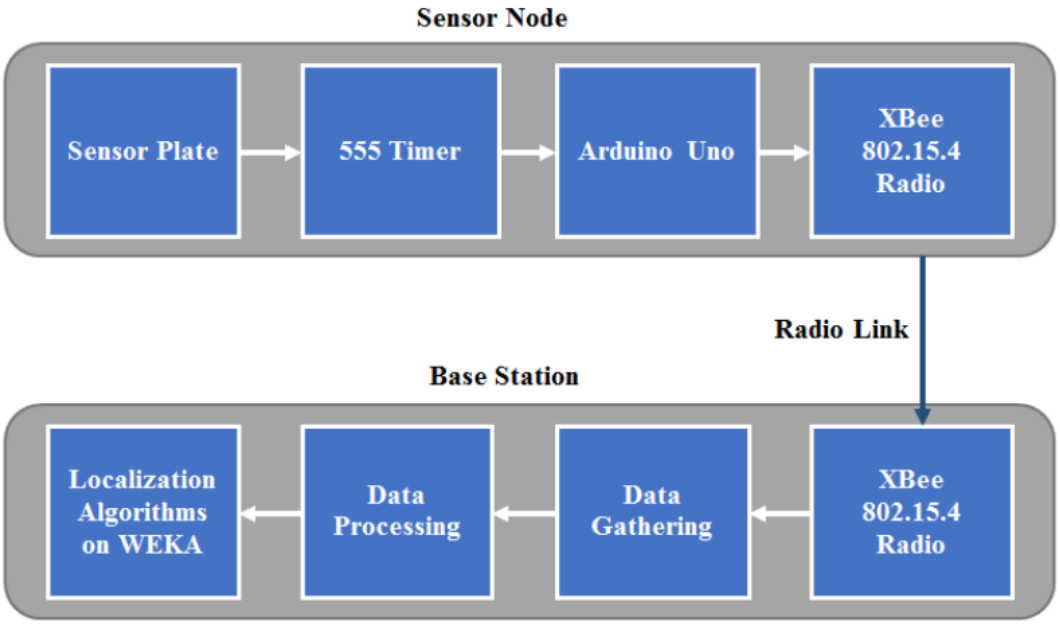


Figure 1.3 Main building blocks of Sensor Node and Base Station. Four Sensor Nodes were connected to a single Base Station [5]

1.4 Neural networks for person localization

To improve the reliability of the tracking system, we process the capacitive sensor data using a neural network. Moreover, we can also filter the data using neural network, instead of using a digital median filter as in [4].

Neural networks can process large amounts of data quickly and accurately, and help solve complex real-time problems. Neural networks can help examine and model complex and nonlinear associations between multiple variables to draw inferences. A neural network trains itself by constantly adjusting the connections between its neurons. This allows the neural network to be continuously improved.

1.4.1 Main NN architectures

In general, the basic structure of neural network models can be divided into two types according to whether the information input is fed back or not: Feedforward Neural Networks and Feedback Neural Networks.

In Feedforward Neural Network, the information entered from the input layer, and the neurons in each layer receive the input from the previous layer and output to the next layer until the output layer. There is no feedback in the transmission of information throughout the network.

Common Feedforward Neural Network include Convolutional Neural Networks (CNN), Fully Connected Neural Networks (FCN), Generative Adversarial Networks (GAN), etc.

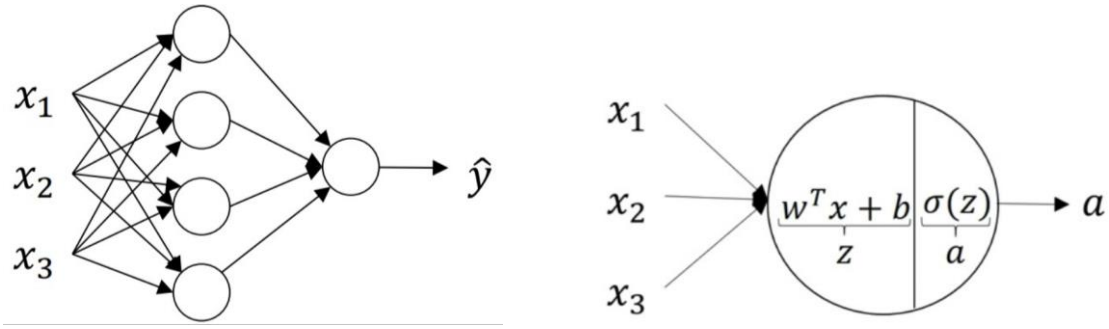
In Feedback Neural Networks, neurons can receive signals not only from other neurons, but also from themselves. Compared with the Feedforward Neural Network, the neurons in the Feedback Neural Network have a memory function and have different states at different moments. The information propagation of Feedback Neural Network can be single or bidirectional.

Common Feedback Neural Networks include Recurrent Neural Networks (RNN), Long Short-Term Memory networks (LSTM), Boltzmann Machines, etc.

This section briefly introduces the following common NN architectures：FCN、CNN、RNN、LSTM.

Fully Connected Neural Networks (FCN): are the most common network structure for deep learning. It has three basic types of layers: input layer, hidden layer, and output layer. Each neuron in

the current layer is connected to each neuron in the previous layer. During each connection, the signal from the previous layer is multiplied by a weight, and a bias is added, then pass by an activation function (Figure 1.4, Figure 1.5), such as sigmoid, tanh, Relu, etc. Complex mapping from input space to output space is achieved by multiple compositions of simple nonlinear functions.



(a) (b)

Figure 1.4 (a) A simple 2-layer Fully Connected Neural Network; (b) Neurons process the input data. [6]

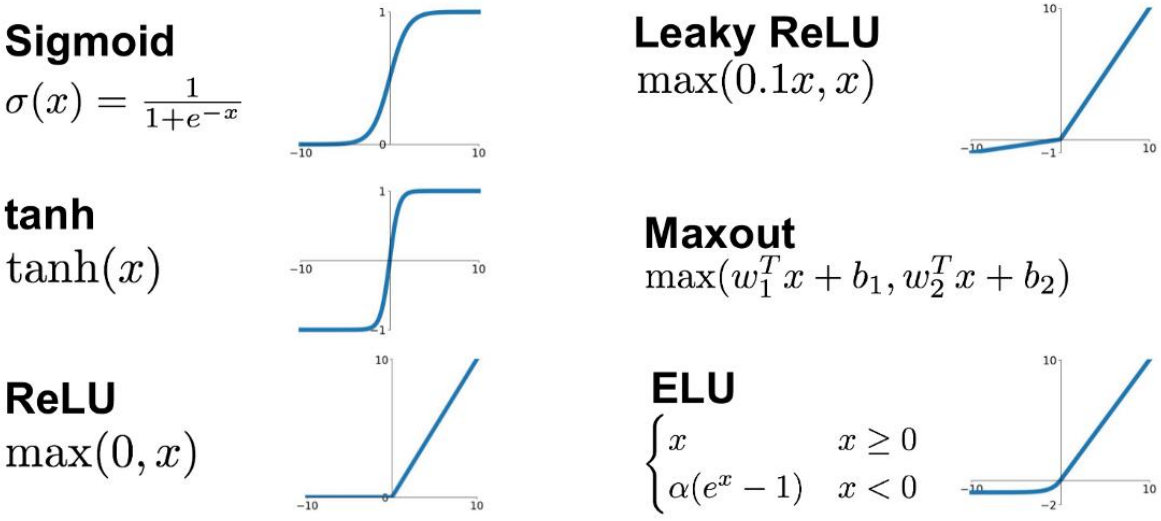


Figure 1.5 Activation functions graph

Convolutional Neural Networks (CNN): Images have very high dimensionality, so training a standard feedforward network to recognize images would require a huge amount of neurons. It is not only very computationally intensive, but can also lead to many problems associated with dimensional disasters in neural networks. Convolutional neural networks provide a solution that uses convolution and pooling layers to reduce the dimensionality of an image. Since the convolutional layer is trainable but has significantly fewer parameters than a standard hidden layer, it is able to highlight the important parts of the image and propagate each important part forward.

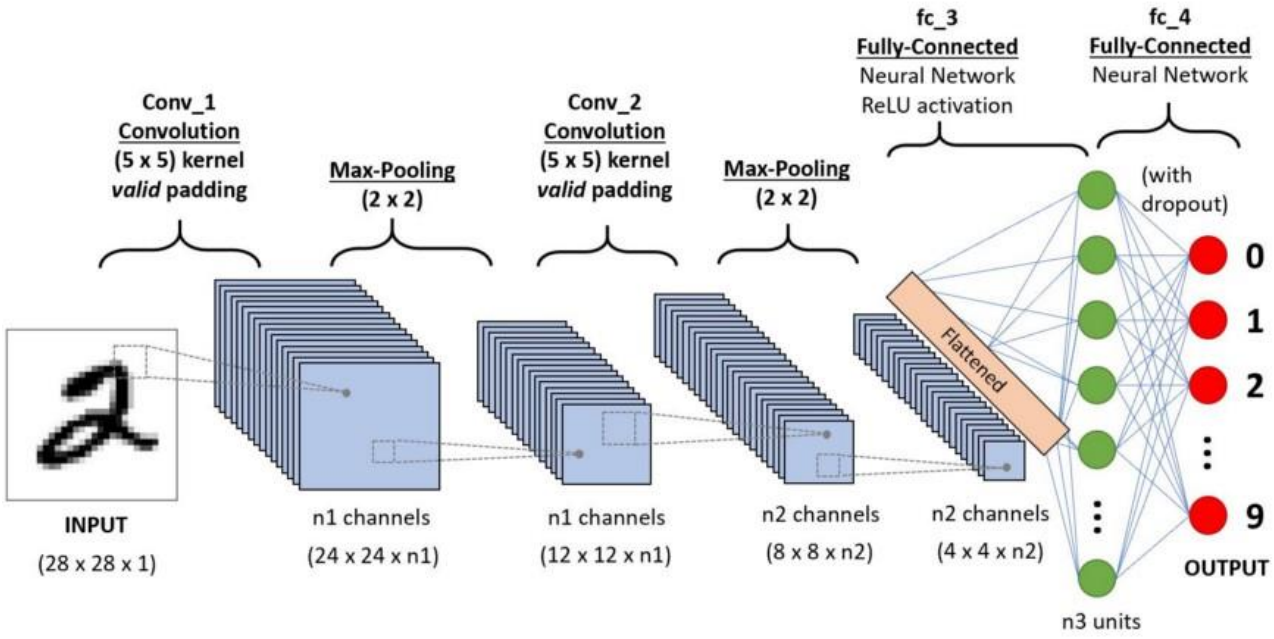
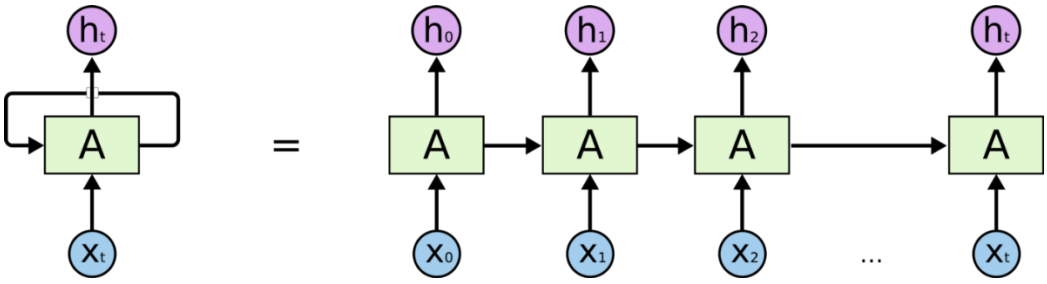


Figure 1.6 Convolutional Neural Networks (CNN). [8]

Recurrent Neural Networks (RNN): are a special type of network that contains loops and self-repetitions. By allowing information to be stored in the network, RNNs uses inferences from previous training to make better inferences about upcoming events. Due to their nature, RNNs are commonly used for exogenous sequential tasks such as generating text word by word or predicting time series data.



Fugire 1.7 Recurrent Neural Networks (RNN). [9]

Long short-term memory (LSTM): is specifically designed to solve the problem of gradient vanishing and explosion that occurs when RNNs learn long contextual information, with memory blocks incorporated in the structure. Each block contains several cyclically connected memory cells and three gates (input, output and forget). The input of information can only interact with neurons through each gate, so these gates learn to open and close intelligently to prevent gradients from exploding or vanishing.

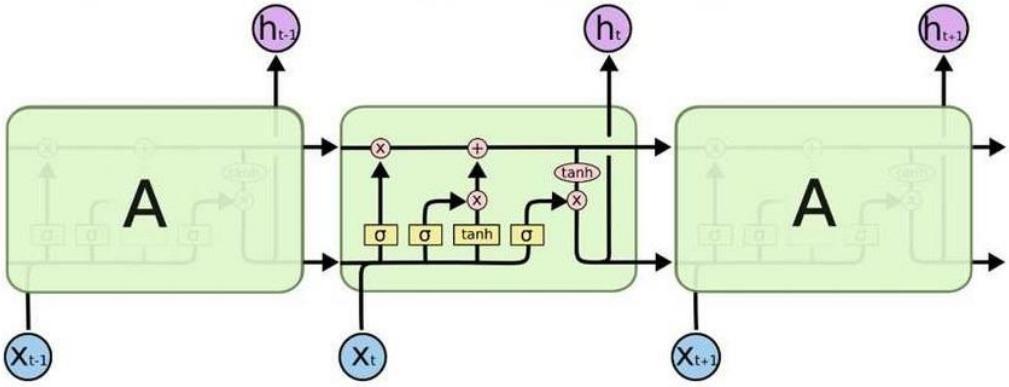


Figure 1.8 Long short-term memory (LSTM). [10]

Chapter 2

Improving generalization of neural network with data augmentation

Whenever we train our neural networks, we need to keep an eye on the generalization of the neural network. Essentially, this refers to how well the model learns from the given data and applies what it learns elsewhere. A neural network with good generalization will be able to produce the correct input-output mapping even when the input data is slightly different from the training samples.

2.1 Problems with small data sets

When training a neural network, it is easy to cause overfitting if the data set is relatively small. That is, the high-dimensional input space corresponding to the small sample size is sparse, and it is difficult for the neural network to learn the mapping relationships from it.

Small data sets pose a problem when training large neural networks.

The first problem is that the network can effectively memorize the training dataset. The model can learn specific input examples and their associated outputs rather than learning a general mapping from inputs to outputs. This will result in a model that performs well on the training dataset and poorly on new data, i.e., poor generalization.

The second problem is that small data sets provide less opportunity to describe the structure of the input space and its relationship to the output. More training data provides a richer description of the problems that the model can learn. Less data means oscillatory and discrete input spaces rather than smooth input spaces, which may make it more difficult for the model to learn feature mappings.

2.2 Introduction of data augmentation

One method to smooth the input space is data augmentation, which facilitates neural network learning. Data Augmentation is a technique that artificially extends the training data set by allowing limited data to produce more equivalent data. It is an effective means to overcome the shortage of training data. Common data augmentation methods based on image processing techniques include flipping, cropping, rotation, translation, noise injection, etc.

In this thesis, I use the method of noise injection. By adding noise during training, the ability of the model to learn mapping rules from the input space can be improved, and the generalization ability and error tolerance of the model can be improved.

This method also has the same effect as regularization, which in turn improves the robustness of the model. As with the weight regularization method, it has been shown that adding noise has a similar effect on the loss function as the penalty term.

Chapter 3

Experimental results

3.1 Methodology

In order to give the neural network good expressiveness while taking less resources, I used 2 layers LSTM with 8 cells per hidden layer. The goal of the thesis is to understand how different types of noise affect the 2layers-LSTM neural network and to try to improve the generalization ability of the neural network by adding noise.

For each noise parameter, the neural network is trained 30 times with the same data set. Because the initial weights are randomized for each training, the results of the training will be different. I pick the best among these 30 independent trainings for comparison. The best training means both good prediction accuracy and good generalization ability. It can be represented by sum of test Mean Square Error (MSE) and standard deviation of four test sets, respectively. To select the best, the first criterion is that the sum of four test MSEs is low, the second criterion is the standard deviation of them is low.

The Mean Square Error is defined as:

$$MSE = \sum_{i=1}^{N} \frac{(y_i - \hat{y}_i)^2}{N}$$

where y is the label, $\hat{y}$ is the output, N is the number of samples. The unity of measurement of MSE is m^2.

In my thesis work, I chose two types of noise to add to the training set: white Gaussian noise and pink noise. This is because white Gaussian noise is commonly found in the environment, while pink noise simulates the drift of a capacitive sensor. In addition, since noise is usually a mixture of these two types of noise in daily life, this thesis also considers combining these two types of noise together in different ways to form mixed noise.

I carried out the following steps:

Step 1: Pre-processing of data. To level occasional gaps and jitter in the experimental data, the capacitive sensor data were resampled at 5 Hz. To improve the learning ability of the neural network, I rescaled the data in the interval [0, 1] for each sensor separately. Set the input window size to 5s, because in [1] it has been proved to be the optimal configuration.

Step 2: Build the neural network. Figure 3.1 shows a typical LSTM structure. As already mentioned, the neural network I use is a 2-layer LSTM with 8 cells per hidden layer. Each layer is followed by a 50% dropout (to avoid over-fitting). Then the output layer. The optimizer I used is AdaMax. Section 3.2 has an introduction to optimizer AdaMax. In the Keras API function fit(), there are two batch generator functions, one is training batch generator, another is validation batch generator. In data augmentation part, there are six training sets, the first one is the original training set, others are applied noise with the same parameters to the original samples. Every batch generator passes to the NN with one data set in each batch for a total of 6 batches per epoch, using the shuffle order of the data sets. Since in our case, we process a "movie" of the person movement, the order of the samples is very important (encode the dynamic movement of the person), so we should never shuffle the samples, we can only shuffle the order of the data sets.

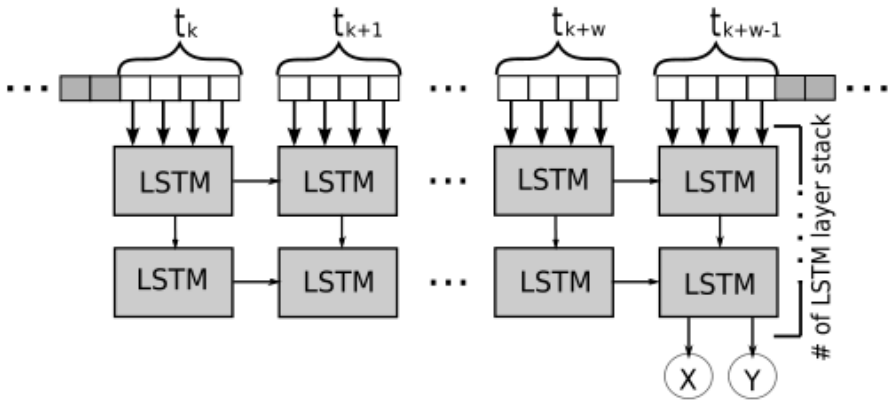


Figure 3.1 Network structure and data access for the Long-Short Term Memory (LSTM) network, in which the LSTM cells in the input layer process the data tuples from the input window. Each window is labelled for training with the person coordinates corresponding to the middle tuple in the window (not shown for readability). [1]

Step 3: Among 4 data sets from different experiments, the training set with the best training result (good prediction accuracy and good generalization ability) is identified. In this process, I trained, validated and tested each of these 4 datasets and also considered all permutations of the training-validation-test sequence. In the end, I selected the first 20% samples of set A as the validation set, the last 60% as the training set, and the rest as the testing set.

Step 4: Obtain a comparison baseline by taking the best training result of at least 30 trainings. All subsequent data augmentation experiments will be compared to it to determine if and how much improvement there is. I will talk more about this in Section 3.3.

Step 5: Add various types of noise（white, pink, mix）, one at a time, on the samples of the training sequence of set A and check the effect on the validation and testing results under the same conditions as baseline. More details in Section 3.4.

The flow diagram of processing steps is shown in Figure 3.2.

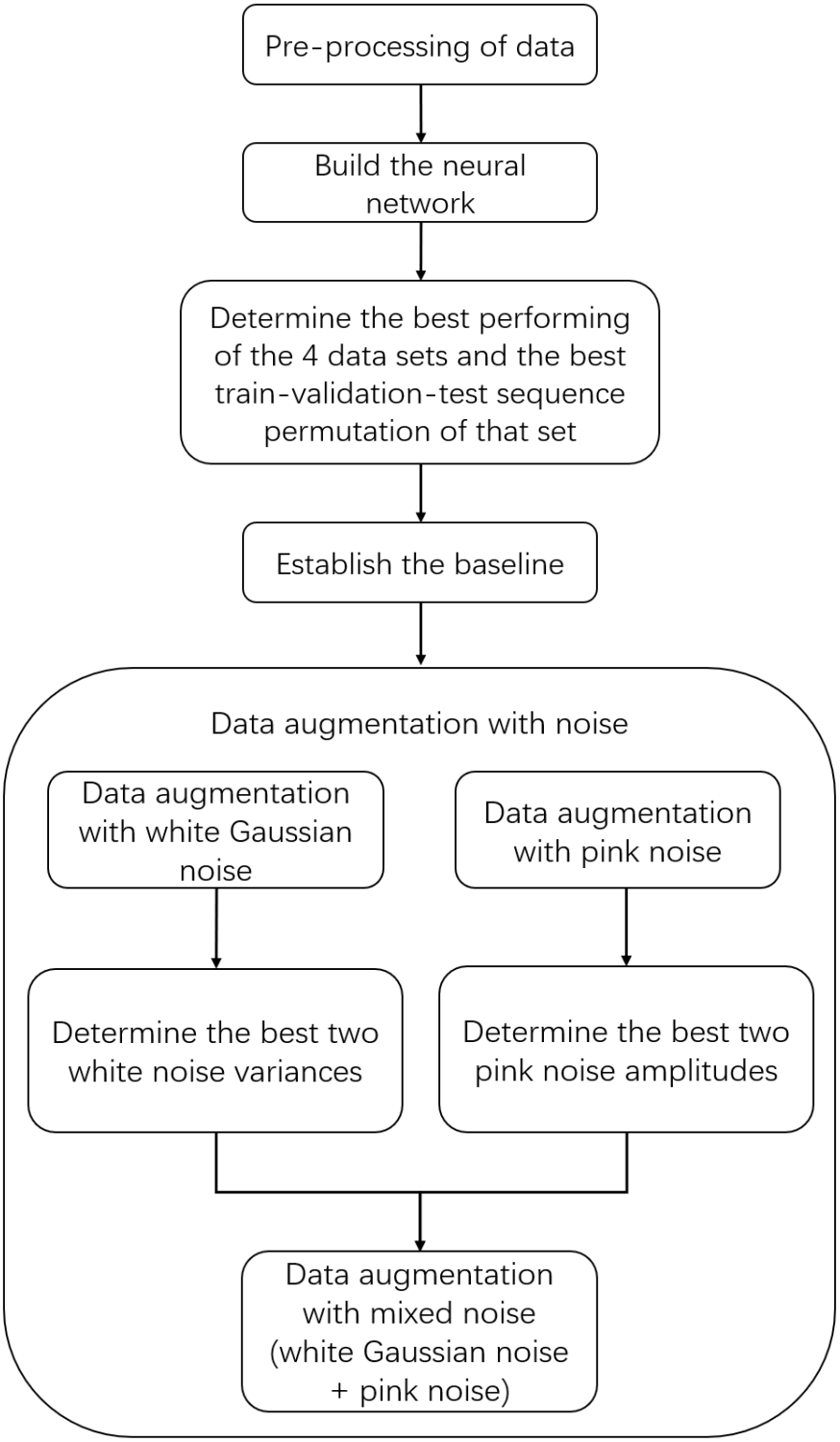
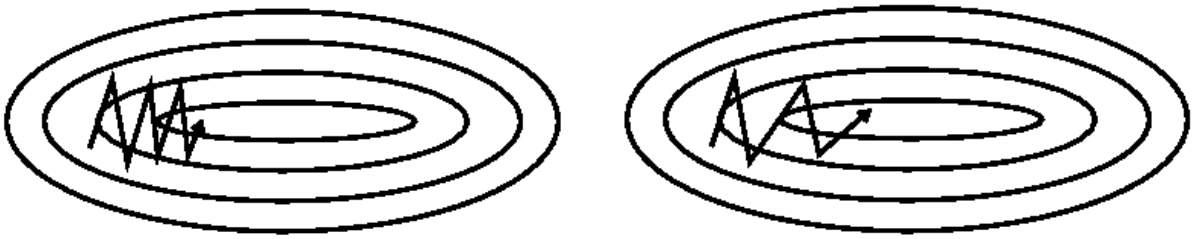


Figure 3.2 Flow diagram of processing steps.

3.2 Optimizer AdaMax

Before introducing AdaMax, I briefly introduce two other algorithms: Momentum and RMSprop.

Momentum is an optimization of the Stochastic Gradient Descent (SGD) algorithm. It introduces momentum on the basis of SGD. From a physical point of view, the introduction of momentum can add two main advantages over the SGD. First, when encountering a local optimum, it is possible to punch out the local optimum under the action of momentum. Second, the SGD algorithm is completely determined by the gradient, which may lead to serious shocks in the process of finding the optimal solution and slow down the speed, however in the case of momentum, the direction of motion is determined by both momentum and gradient, which can make the oscillation weaken and move faster to the optimal solution (Figure 3.3).



(a) (b)

Figure 3.3 (a) SGD without Momentum; (b) SGD with Momentum. [11]

Here is the implementation details of Momentum.

On iteration t:

Compute dW, db on the current mini-batch

$v_{dW} = \beta_1 v_{dW} + (1 - \beta_1) dW$

$v_{db} = \beta_1 v_{db} + (1 - \beta_1) db$

$W = W - \alpha v_{dW}$

$b = b - \alpha v_{db}$

where W is weight, dW is gradient of weight, b is bias, db is gradient of bias, α is a hyperparameter (learning rate), β_1 is a hyperparameter (represents the effect of past gradients on the current gradient).

RMSprop, the full name is root mean square prop. It can also eliminate oscillations in gradient descent, and allows the use of a larger learning rate, thus speeding up algorithm learning.

Here is the implementation details of RMSprop.

On iteration t:

Compute dW, db on the current mini-batch

$S_{dW} = \beta_2 S_{dW} + (1 - \beta_2) dW^2$

$S_{db} = \beta_2 S_{db} + (1 - \beta_2) db^2$

$W = W - \alpha \frac{dW}{\sqrt{S_{dW}} + \varepsilon}$

$b = b - \alpha \frac{db}{\sqrt{S_{db}} + \varepsilon}$

where W is weight, dW is gradient of weight, b is bias, db is gradient of bias, α is a hyperparameter (learning rate), β_2 is a hyperparameter (represents the effect of past gradients on the current gradient), ε is a small number to avoid a denominator of 0.

The Adam optimization algorithm is basically a combination of Momentum and RMSprop. It is a common learning algorithm that has been shown to be effective for different neural networks. AdaMax is a variant of Adam based on infinity norm. Sometime AdaMax is better than Adam, especially in models with embedding. In my case, I used AdaMax. These two algorithms were first proposed by Diederik P. Kingma and Jimmy Lei Ba in [12]. The algorithms are shown in their paper (Figure 3.4, Figure 3.5).

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_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)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

Figure 3.4 Algorithm of Adam. [12]

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: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_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 θ_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 θ_t (Resulting parameters)

Figure 3.5 Algorithm of AdaMax. [12]

3.3 Establish baseline

In order to determine whether a training result has improved and by how much, we need to create a baseline for comparison. The last 60% of the samples of set A are used as the training set, the first 20% samples as the validation set, and the rest as the testing set. Repeat the training 30 times. Taking the best among 30 trainings. The best result is shown in Table 3.1. The MSE values show that the test set B, C and D contributes a lot to the sum of MSE, especially B and C, while the contribution of A is very small. It can also be seen from Figure 3.6 that the prediction accuracy of A is relatively high, the prediction curve follows reference curve well, while B, C and D are not. Especially test set B and D, sometimes they go in the opposite direction of the ground truth.

Table 3.1 Baseline result

	Test MSE A $[m^2]$	Test MSE B $[m^2]$	Test MSE C $[m^2]$	Test MSE D $[m^2]$	Standard Deviation of test MSEs $[m^2]$	Sum of test MSEs $[m^2]$
Baseline	0.0526	0.3098	0.2486	0.1798	0.1103	0.7908

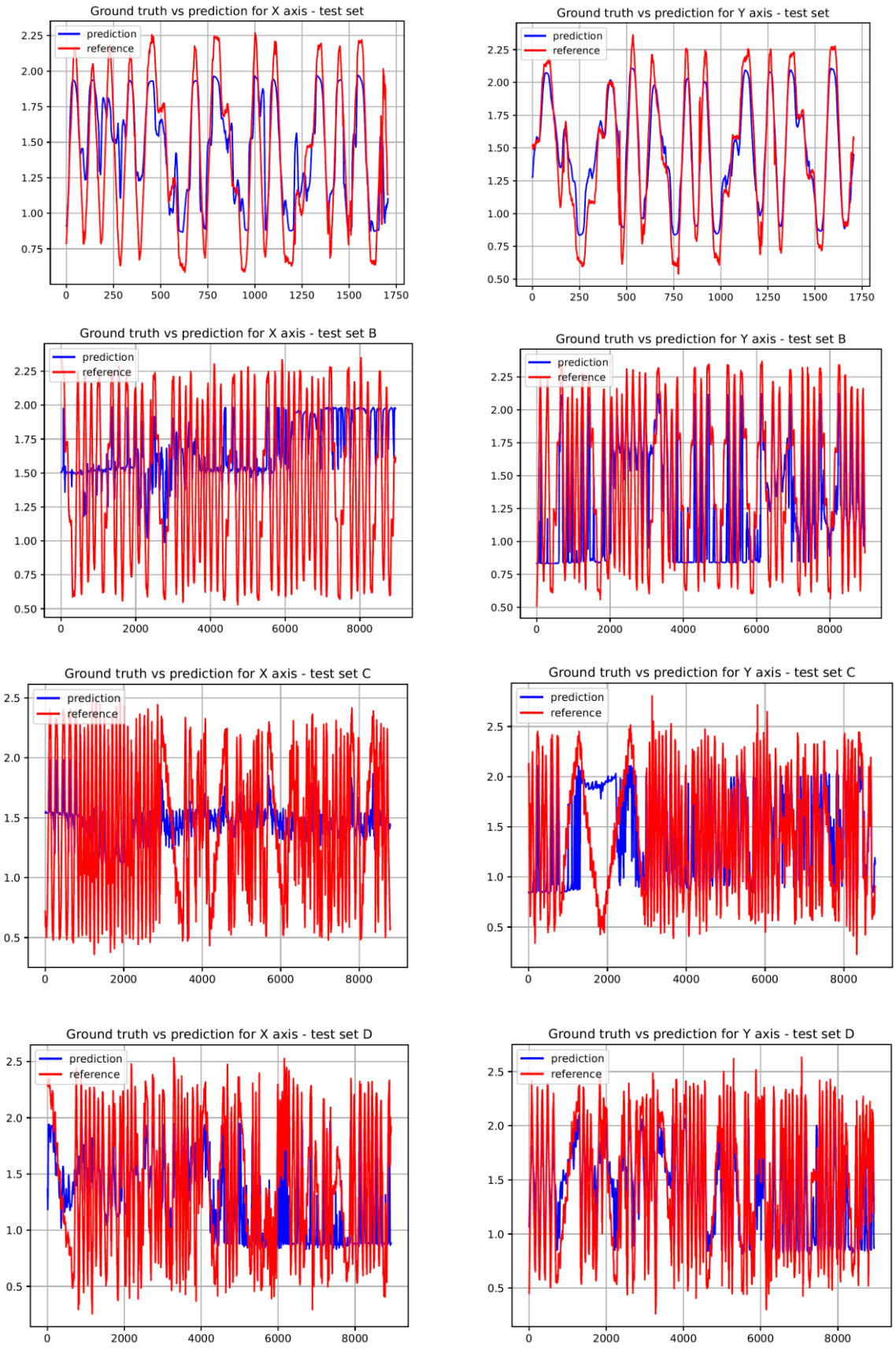


Figure 3.6 Ground truth vs prediction for baseline

3.4 Data augmentation

When training a neural network, it is easy to cause overfitting if the data set is relatively small. The NN can effectively memorize the training dataset. It can learn specific input examples and their associated outputs rather than learning a general mapping from inputs to outputs. This will result in a model that performs well on the training dataset and poorly on new data, i.e., poor generalization. To solve this problem, I used data augmentation technique.

Data augmentation is a technique that artificially extends the training data set by allowing limited data to produce more equivalent data. It is an effective means to overcome the shortage of training data. Noise injection is a method of data augmentation.

I added the noise to the original training set via MATLAB. Before adding the noise, I rescale the data, and after adding the noise, I rescale again so that the noisy data is between 0 and 1, to improve the learning ability of the neural network. But first of all, I should decide how many number of training sets should be created. Taking white Gaussian noise with variance value 5.00E-06 (It is proved to be the best variance value of white Gaussian noise in Section 3.4.1) as an example, I did an experiment to compare the training results of different numbers of training sets. Table 3.2 shows the results. Since the first criterion is the sum of four test MSEs, the second criterion is the standard deviation of them, in the end, I decided to create 6 training sets (5 noisy sets + 1 original set), each training set becomes one batch for batch training. In this way, the training data becomes 6 times more than before.

Table 3.2 Comparison of different number of training sets

Number of training sets	Test MSE A $[m^2]$	Test MSE B $[m^2]$	Test MSE C $[m^2]$	Test MSE D $[m^2]$	Standard Deviation of test MSEs $[m^2]$	Sum of test MSEs $[m^2]$
1(Baseline)	0.0526	0.3098	0.2486	0.1798	0.1103	0.7908
6	0.0509	0.2917	0.2412	0.1362	0.1077	0.7200
10	0.0566	0.2924	0.2392	0.1507	0.1035	0.7389
15	0.0534	0.2902	0.2701	0.1394	0.1121	0.7531
20	0.0535	0.3103	0.2370	0.1342	0.1129	0.7350

3.4.1 Data augmentation with white Gaussian noise

White Gaussian noise (WGN) is a basic noise model used in information theory to mimic the effect of many random processes that occur in nature. White refers to the idea that it has uniform power across the frequency band for the information system. It is an analogy to the color white which has uniform emissions at all frequencies in the visible spectrum. Gaussian because it has a normal distribution in the time domain with an average time domain value of zero [13].

For white noise, mean value is 0 and mean power equals variance. The higher the variance, the higher the power of the noise, i.e. the stronger the noise. So the parameter to be decided is the variance. Variance equal to 0 means no noise is added. Starting from a variance value (1.00E-07) lower than the smallest signal, around 10 ppm or below and then gradually increase. As with baseline, the training was repeated 30 times for each variance value, and then the best training was selected.

At the beginning, to explore the trend of training results with variance values, I chose the following variance: 1.00E-07, 5.00E-07, 1.00E-06, 5.00E-06, 1.00E-05, 5.00E-05. By comparing the results (Table 3.3) I found that 5.00E-06 was the best. So next I continued to explore around 5.00E-06. The variance value I chose was: 2.5E-06, 4.00E-06, 6.00E-06, 7.50E-06. Finally I found that 5.00E-06 was still the best and 7.50E-06 was also good. They are the best two variance values.

To make the data more intuitive, I present the data in the form of plots (Figure 3.7). It can be seen from the plots, when the variance is too small, the improvement of the neural network is quite limited. When the variance reaches 5.00E-05, the training results are worse instead, both standard deviation and sum of test MSEs increase.

From the table and plots, we can know that all variance values except 1.00E-06 and 5.00E-05 can make the sum of MSE decrease. However, for standard deviation, only 1.00E-07, 4.00E-06, 5.00E-06 and 7.50E-06 decrease, while the others increase instead, which indicates that the generalization ability does not improve. When the variance is equal to 5.00E-06, both sum of MSE and standard deviation are the lowest, which indicates that both prediction accuracy and generalization ability have improved. Indeed, as can be seen from the table, when the variance is 5.00E-06, the MSE of all four test sets decreases, especially set D decreased a lot.

Figure 3.8 shows the ground truth and prediction for the best white Gaussian noise found. For the X axis of test set B, there is a significant improvement in the interval from about the 4000th sample

to the 6000th sample. Another significant improvement appears in the Y axis of test set D, starting from about the 6000th sample to the end. Indeed, it can be seen from Table 3.3, B and D show more improvement than A and C.

Table 3.3 White Gaussian noise result

Variance	Test MSE A $[m^2]$	Test MSE B $[m^2]$	Test MSE C $[m^2]$	Test MSE D $[m^2]$	Standard Deviation of test MSEs $[m^2]$	Sum of test MSEs $[m^2]$
0(Baseline)	0.0526	0.3098	0.2486	0.1798	0.1103	0.7908
1.00E-07	0.0492	0.2777	0.2755	0.1759	0.1079	0.7782
5.00E-07	0.0513	0.2796	0.2855	0.1643	0.1110	0.7807
1.00E-06	0.0459	0.2942	0.2864	0.1696	0.1169	0.7961
2.50E-06	0.0474	0.3176	0.2457	0.1333	0.1195	0.7440
4.00E-06	0.0580	0.2527	0.3040	0.1451	0.1101	0.7598
5.00E-06	**0.0509**	**0.2917**	**0.2412**	**0.1362**	**0.1077**	**0.7200**
6.00E-06	0.0537	0.3036	0.2877	0.1370	0.1207	0.7820
7.50E-06	0.0531	0.2842	0.2583	0.1352	0.1081	0.7308
1.00E-05	0.0544	0.2885	0.2663	0.1363	0.1107	0.7455
5.00E-05	0.0454	0.3355	0.2780	0.1468	0.1306	0.8058

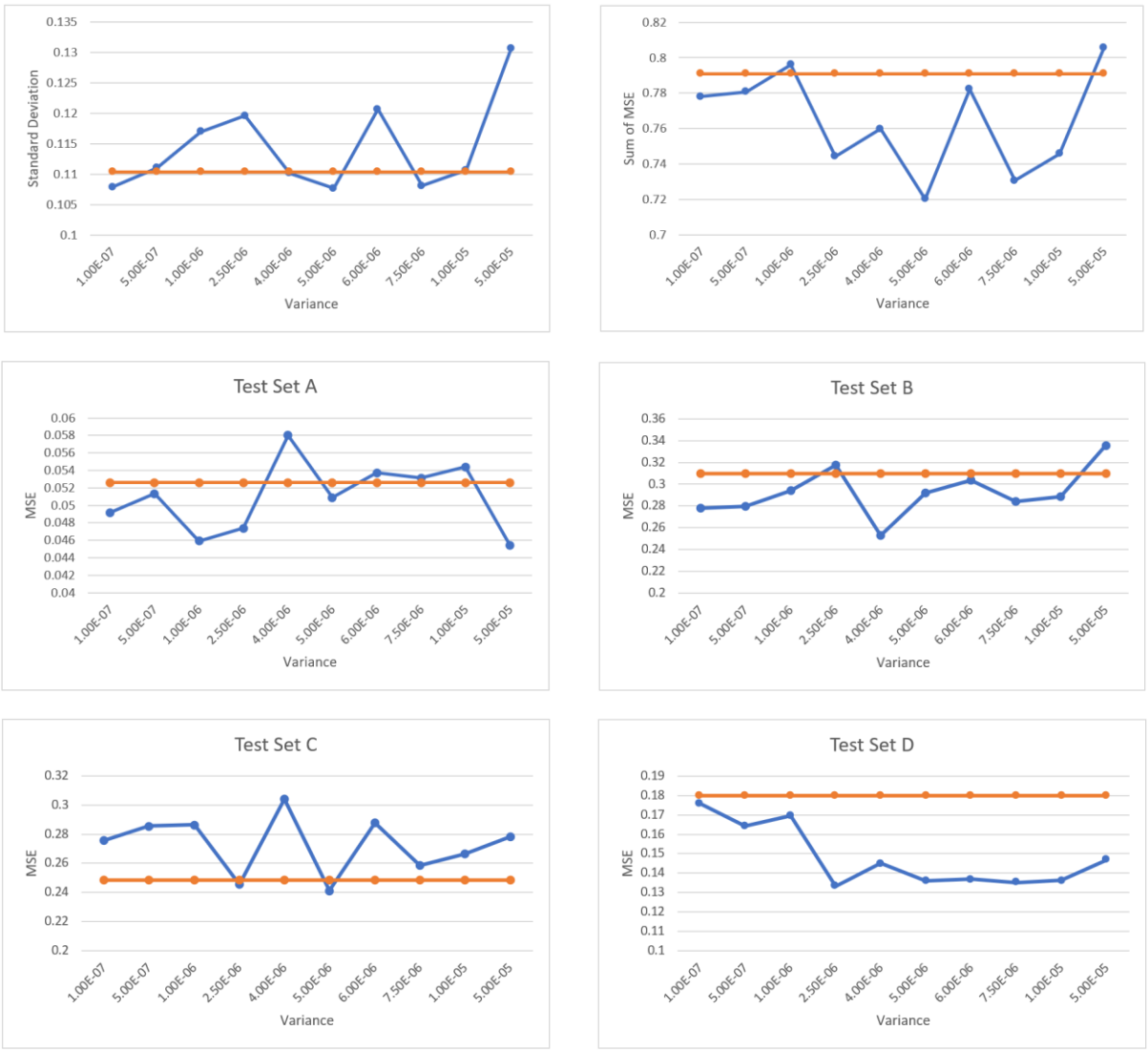


Figure 3.7 Dependency of test MSEs on variance of the white Gaussian noise used for data augmentation

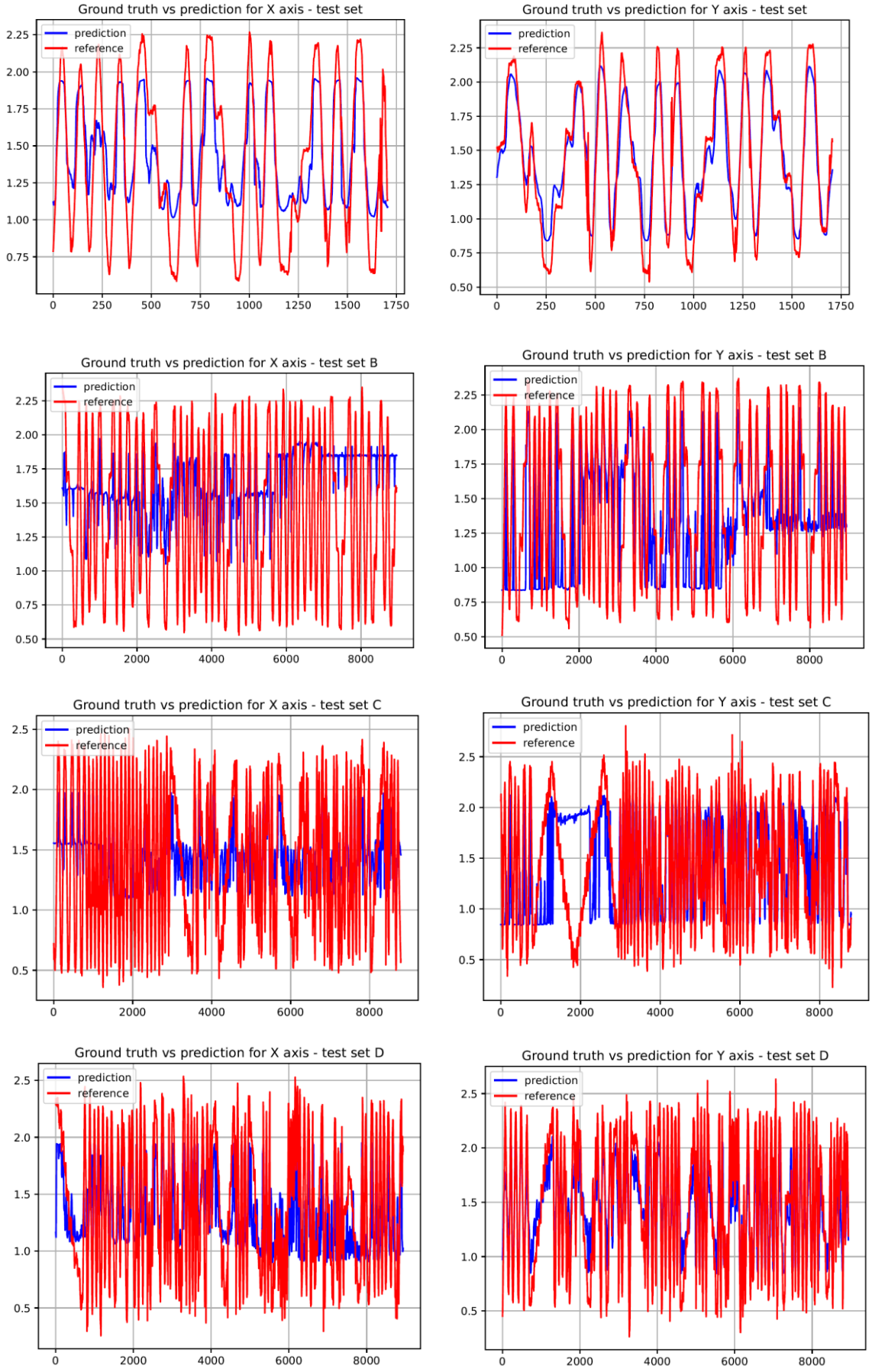


Figure 3.8 Ground truth vs prediction for the best white Gaussian noise (variance = 5.00E-06)

3.4.2 Data augmentation with pink noise

Pink noise is a signal or process with a frequency spectrum such that the power spectral density (power per frequency interval) is inversely proportional to the frequency of the signal [14]. Pink noise has higher amplitude for lower frequencies, so it emulates the drift for the sensors. In this experiment, the parameter to be decided is the amplitude of pink noise. Repeat training 30 times and pick the best.

Starting from an amplitude value lower than the smallest signal, around 10 ppm or below, then gradually increase. At the beginning, to explore the trend of training results with amplitude values, I chose the following amplitudes: 1.00E-06, 5.00E-06, 1.00E-05, 5.00E-05, 1.00E-04, 5.00E-04, 0.001. From Table 3.4 it can be seen, although the lowest sum of test MSEs (0.7375 m^2) was found when amplitude = 1.00E-05, the test MSE decreased for only two test sets (C and D). However even though the sum of MSE is not the lowest when amplitude = 5.00E-05, there are three test MSEs that decrease. Therefore, among all these amplitude values, it can be considered that 5.00E-05 performs the best. Next I continued to explore around 5.00E-05. The amplitude value I chose was: 2.5E-05, 7.50E-05. But the conclusion remains the same.

Again, I present the data in the form of plots (Figure 3.9). It can be seen from the plots, When the amplitude is too small or too big, the improvement of the neural network is not good. For all amplitude values, the standard deviation increased, sum of MSE decreased (except for 2.50E-05), but not significantly.

Figure 3.10 shows the ground truth and prediction for the best pink noise (amplitude = 5.00E-05) found. Consistent with the data in Table 3.4, only test set D has a significant improvement, starting from about the 6000th sample to the end.

Table 3.4 Pink noise result

Amplitude	Test MSE A $[m^2]$	Test MSE B $[m^2]$	Test MSE C $[m^2]$	Test MSE D $[m^2]$	Standard Deviation of test MSEs $[m^2]$	Sum of test MSEs $[m^2]$
0(Baseline)	0.0526	0.3098	0.2486	0.1798	0.1103	0.7908
1.00E-06	0.0476	0.2736	0.3108	0.1554	0.1195	0.7874
5.00E-06	0.0450	0.3472	0.2440	0.1510	0.1291	0.7871
1.00E-05	0.0532	0.3173	0.2300	0.1370	0.1143	0.7375
2.50E-05	0.0467	0.2903	0.2951	0.1714	0.1176	0.8034
5.00E-05	**0.0502**	**0.2997**	**0.2635**	**0.1358**	**0.1153**	**0.7493**
7.50E-05	0.0444	0.2678	0.3038	0.1603	0.1169	0.7763
1.00E-04	0.0515	0.3293	0.2283	0.1383	0.1193	0.7473
5.00E-04	0.0464	0.2909	0.2829	0.1445	0.1176	0.7647
0.001	0.0523	0.3048	0.2745	0.1588	0.1155	0.7904

Figure 3.9 Dependency of test MSEs on amplitude of the pink noise used for data augmentation

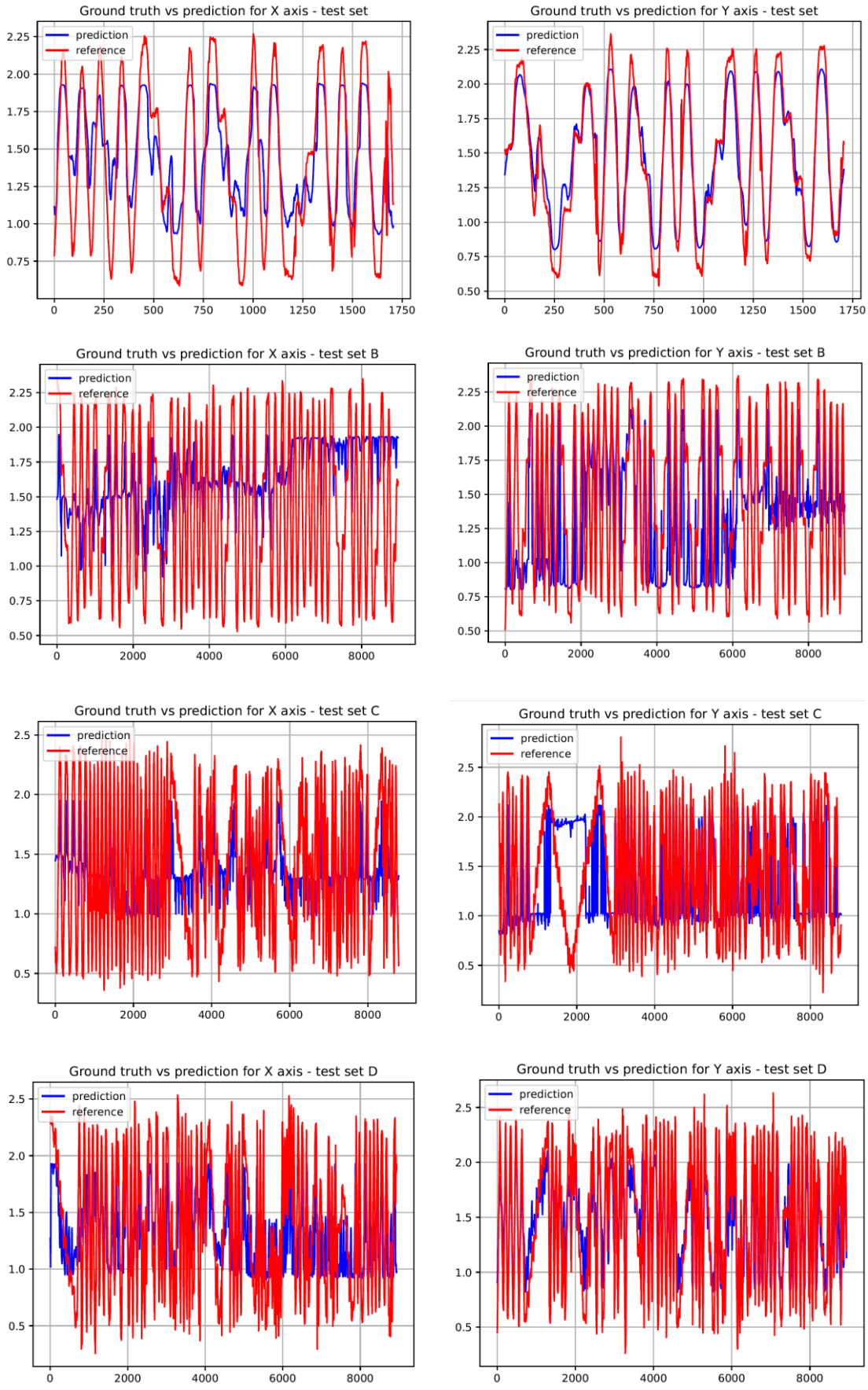


Figure 3.10 Ground truth vs prediction for the best pink noise (amplitude = 5.00E-05)

3.4.3 Data augmentation with mixed noise

In real life, it is often the case that different kinds of noise appear at the same time. To better simulate this situation, I mixed white noise and pink noise to see how it would affect the results. In the previous experiments, I have found the best-performing white and pink noise, and I will use the results to conduct experiments with mixed noise. Here are my steps:

- First, selecting the two best-performing variance values (5.00E-06 and 7.50E-06) in the white noise experiment, fixing them, and then varying the amplitude of the pink noise (around the best two amplitudes of the pink noise 1.00E-05 and 5.00E-05). Each combination of parameters is trained 30 times and pick the best.
- Second, selecting the two best-performing amplitude values (1.00E-05 and 5.00E-05) in the pink noise experiment, fixing them, and then varying the variance of the white noise (around the best two variances of the white noise 5.00E-06 and 7.50E-06). Each combination of parameters is trained 30 times and pick the best.

Table 3.5 shows the results of mixing different pink noise when the variance value of white noise is 5.00E-06. As can be seen from Figure 3.11, the variation of standard deviation is not stable, while the sum of test MSEs consistently improves in the interval [0.000001, 0.0001]. The neural network performs best when the amplitude of pink noise is equal to 1.00E-05, showing low standard deviation (0.1078 m^2) and low sum of test MSEs (0.7309 m^2).

Table 3.5 Results when the variance of white noise is fixed at 5.00E-06

Pink Noise Amplitude	Test MSE A $[m^2]$	Test MSE B $[m^2]$	Test MSE C $[m^2]$	Test MSE D $[m^2]$	Standard Deviation of test MSEs $[m^2]$	Sum of test MSEs $[m^2]$
Baseline	0.0526	0.3098	0.2486	0.1798	0.1103	0.7908
1.00E-06	0.0518	0.2892	0.2853	0.1353	0.1170	0.7615
2.50E-06	0.0489	0.2998	0.2926	0.1384	0.1226	0.7797
5.00E-06	0.0578	0.3022	0.2521	0.1495	0.1089	0.7615
7.50E-06	0.0574	0.3222	0.2626	0.1427	0.1189	0.7849
1.00E-05	**0.0491**	**0.2807**	**0.2586**	**0.1425**	**0.1078**	**0.7309**
2.50E-05	0.0511	0.3116	0.2370	0.1540	0.1119	0.7537
5.00E-05	0.0492	0.2975	0.2599	0.1405	0.1136	0.7471
7.50E-05	0.0539	0.2951	0.2495	0.1459	0.1080	0.7444
0.0001	0.0478	0.2894	0.2786	0.1548	0.1143	0.7706

Figure 3.11 Dependency of test MSEs when changing the amount of pink noise for fixed white noise variance 5.00E-06

Table 3.6 shows the results of mixing different pink noise when the variance value of white noise is 7.50E-06. Similar to the case where the white noise variance is 5.00E-06, the variation of standard deviation is not stable, while the sum of test MSEs consistently improves in the interval (Figure 3.12). The neural network performs well when the amplitude is equal to 1.00E-05 and 7.50E-05. I choose the result with an amplitude of 7.50E-05 as the best. Because compared with 1.00E-05, while the sum of test MSEs of 7.50E-05 is 0.075% higher, the standard deviation is 3% lower.

Table 3.6 Results when the variance of white noise is fixed at 7.50E-06

Pink Noise Amplitude	Test MSE A $[m^2]$	Test MSE B $[m^2]$	Test MSE C $[m^2]$	Test MSE D $[m^2]$	Standard Deviation of test MSEs $[m^2]$	Sum of test MSEs $[m^2]$
Baseline	0.0526	0.3098	0.2486	0.1798	0.1103	0.7908
1.00E-06	0.0588	0.3258	0.2350	0.1344	0.1166	0.7540
2.50E-06	0.0570	0.2882	0.2894	0.1437	0.1144	0.7783
5.00E-06	0.0523	0.2919	0.2706	0.1456	0.1123	0.7605
7.50E-06	0.0653	0.3106	0.2518	0.1481	0.1090	0.7758
1.00E-05	0.0552	0.2831	0.2391	0.1414	0.1020	0.7188
2.50E-05	0.0530	0.2941	0.2725	0.1433	0.1134	0.7629
5.00E-05	0.0584	0.2960	0.2850	0.1402	0.1154	0.7796
7.50E-05	**0.0549**	**0.2728**	**0.2430**	**0.1488**	**0.0987**	**0.7194**
0.0001	0.0549	0.3187	0.2570	0.1519	0.1164	0.7826
0.0005	0.0481	0.2959	0.2650	0.1418	0.1144	0.7508

Figure 3.12 Dependency of test MSEs when changing the amount of pink noise for fixed white noise variance 7.50E-06.

Table 3.7 shows the results of mixing different white noise when the amplitude of pink noise is 1.00E-05. Figure 3.13 shows the plots. We can see that both standard deviation and sum of test MSEs start to rise sharply starting from a variance of 7.50E-05. It can be assumed that the high noise is not conducive to the training of the neural network. The neural network performs best when the variance of white noise is equal to 5.00E-05, showing lowest standard deviation (0.0965 m^2) and lowest sum of test MSEs (0.7124 m^2).

Table 3.7 Results when the amplitude of pink noise is fixed at 1.00E-05

White Noise Variance	Test MSE A $[m^2]$	Test MSE B $[m^2]$	Test MSE C $[m^2]$	Test MSE D $[m^2]$	Standard Deviation of test MSEs $[m^2]$	Sum of test MSEs $[m^2]$
	0.0526	0.3098	0.2486	0.1798	0.1103	0.7908
2.50E-07	0.0553	0.2986	0.2545	0.1483	0.1093	0.7567
5.00E-07	0.0487	0.2789	0.2864	0.1604	0.1125	0.7744
7.50E-07	0.0538	0.3110	0.2509	0.1422	0.1143	0.7580
1.00E-06	0.0467	0.2867	0.2767	0.1324	0.1164	0.7426
2.50E-06	0.0526	0.2918	0.2668	0.1440	0.1114	0.7552
5.00E-06	0.0491	0.2807	0.2586	0.1425	0.1078	0.7309
7.50E-06	0.0486	0.2887	0.2637	0.1466	0.1111	0.7476
1.00E-05	0.0491	0.3055	0.2754	0.1470	0.1187	0.7769
2.50E-05	0.0689	0.3022	0.2532	0.1466	0.1050	0.7709
5.00E-05	**0.0612**	**0.2692**	**0.2441**	**0.1379**	**0.0965**	**0.7124**
7.50E-05	0.0537	0.3371	0.2826	0.1377	0.1302	0.8110
1.00E-04	0.0607	0.3051	0.2906	0.1400	0.1187	0.7965
2.50E-04	0.0567	0.3270	0.3311	0.1316	0.1390	0.8463

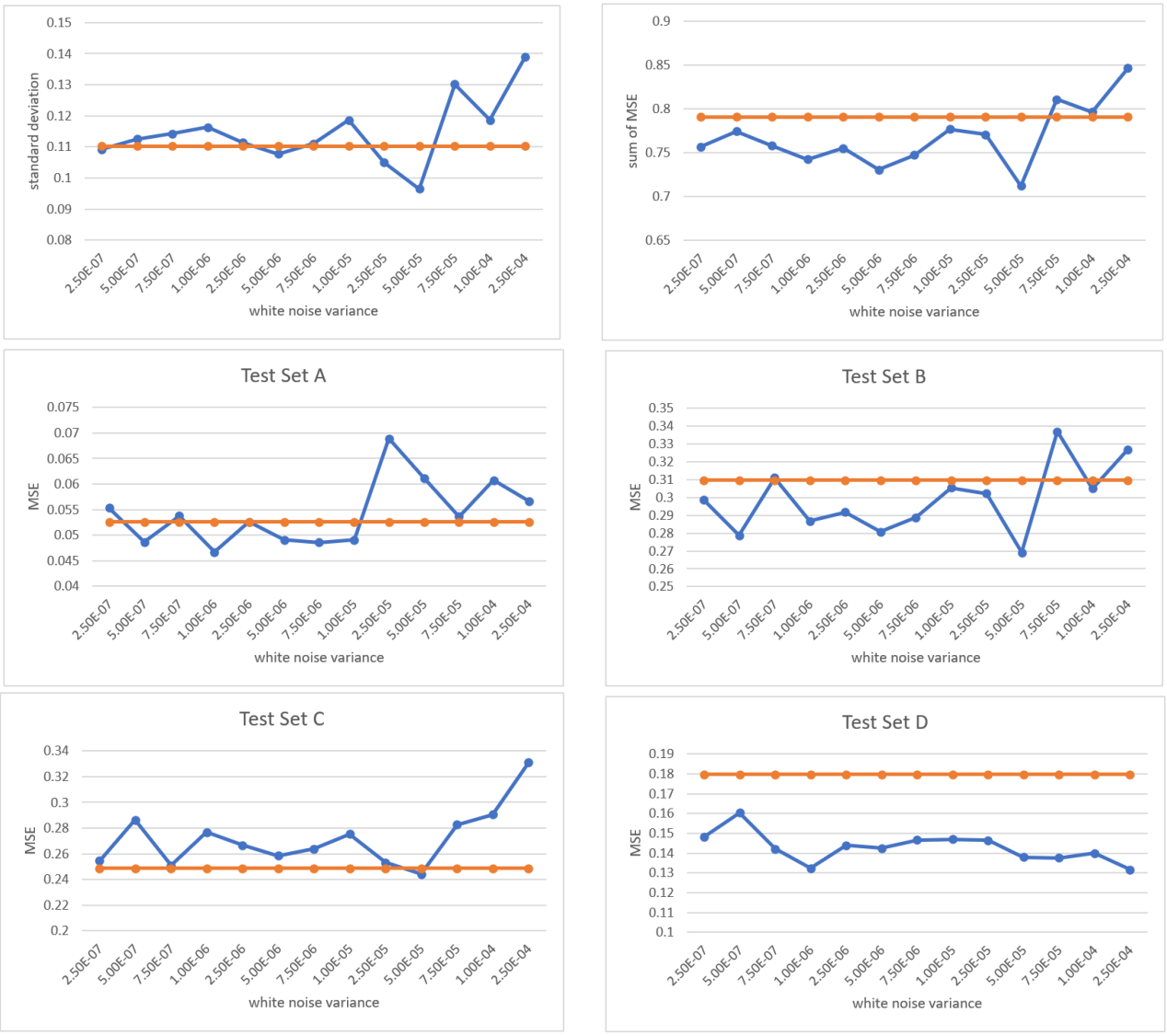


Figure 3.13 Dependency of test MSEs when changing the amount of white noise for fixed pink noise amplitude 1.00E-05.

Table 3.8 shows the results of mixing different white noise when the amplitude value of pink noise is 5.00E-05. From the table and plots (Figure 3.14), It can be seen that the standard deviation of test MSEs is higher than the baseline in the whole interval, and the sum of test MSEs is improved in the interval [2.50E-07, 1.00E-05] (except for 1.00E-06), however the improvement is not so much. In this case, the best result is when the white noise variance is equal to 1.00E-05.

Table 3.8 Results when the amplitude of pink noise is fixed at 5.00E-05

White Noise Variance	Test MSE A $[m^2]$	Test MSE B $[m^2]$	Test MSE C $[m^2]$	Test MSE D $[m^2]$	Standard Deviation of test MSEs $[m^2]$	Sum of test MSEs $[m^2]$
Baseline	0.0526	0.3098	0.2486	0.1798	0.1103	0.7908
2.50E-07	0.0500	0.2830	0.2865	0.1663	0.1125	0.7857
5.00E-07	0.0478	0.3027	0.2683	0.1585	0.1154	0.7773
7.50E-07	0.0605	0.3353	0.2378	0.1412	0.1190	0.7747
1.00E-06	0.0542	0.3131	0.2744	0.1622	0.1169	0.8038
2.50E-06	0.0482	0.3421	0.2537	0.1350	0.1294	0.7791
5.00E-06	0.0580	0.3022	0.2710	0.1422	0.1137	0.7734
7.50E-06	0.0522	0.3141	0.2853	0.1394	0.1236	0.7909
1.00E-05	**0.0521**	**0.3120**	**0.2492**	**0.1582**	**0.1131**	**0.7716**
2.50E-05	0.0556	0.3109	0.2950	0.1299	0.1253	0.7914
5.00E-05	0.0491	0.3064	0.2862	0.1517	0.1209	0.7934

Figure 3.14 Dependency of test MSEs when changing the amount of white noise for fixed pink noise amplitude 5.00E-05.

As a summary of this section, the best performing combination of white noise and pink noise in the experiments is variance 5.00E-05 plus amplitude 1.00E-05 (Table 3.9)

Figure 3.15 shows the ground truth and prediction for the best mix noise found. Although for test set A, NN still has accurate predictions, the ragged look of both X and Y is more pronounced compared to baseline. For test set B, even though the case of going in the opposite direction to ground truth still exists, numerically it is better than baseline. For test set D, same as before, there is a significant improvement from about the 6000th sample to the end.

Table 3.9 The best results for different noise combinations

	Test MSE A $[m^2]$	Test MSE B $[m^2]$	Test MSE C $[m^2]$	Test MSE D $[m^2]$	Standard Deviation of test MSEs $[m^2]$	Sum of test MSEs $[m^2]$
var5e-6 + ampl1e-5	0.0491	0.2807	0.2586	0.1425	0.1078	0.7309
var7.5e-6 + ampl7.5e-5	0.0549	0.2728	0.2430	0.1488	0.0987	0.7194
var5e-5 + ampl1e-5	**0.0612**	**0.2692**	**0.2441**	**0.1379**	**0.0965**	**0.7124**
var1e-5 + ampl5e-5	0.0521	0.3120	0.2492	0.1582	0.1131	0.7716

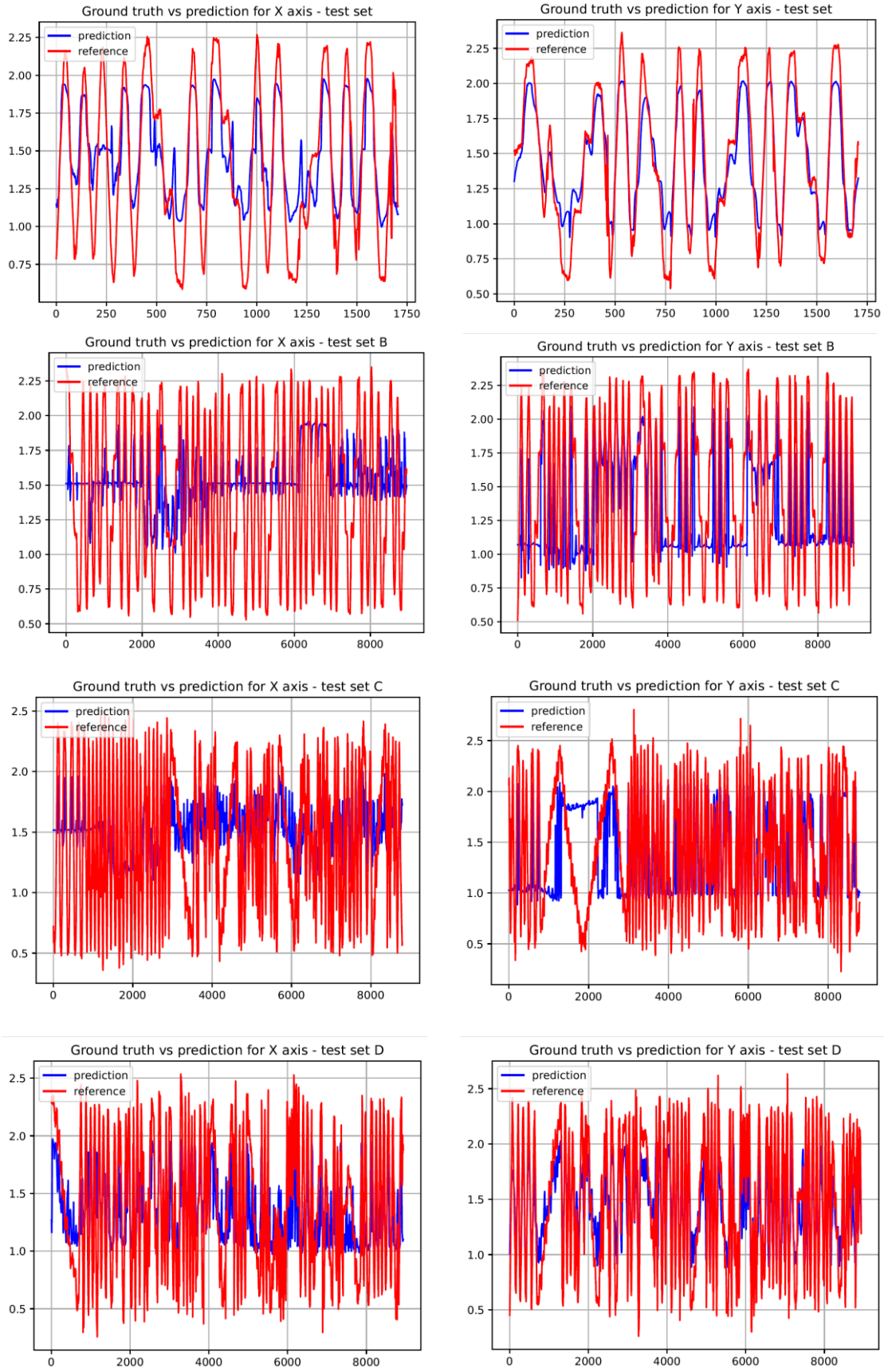


Figure 3.15 Ground truth vs prediction for the best mix noise (variance = 5.00E-05, amplitude = 1.00E-05)

3.4.4 Conclusion

Table 4.1 shows the improvement of the three noises relative to the baseline in terms of standard deviation and sum of four test MSEs, expressed as a percentage. It shows that, the pink noise does not perform so well, especially the standard deviation is a negative value, indicating that the prediction accuracy of the neural network vary more widely for different data sets. White noise and mixed noise made more improvements on the neural network. Especially the mixed noise, both sum of test MSEs and standard deviation are the most improved among the three types of noise. This indicates that the mixed noise improves both the prediction accuracy and generalization ability of the neural network.

In my thesis work, the performance of the neural network is improved by applying data augmentation technique. By adding noise to the training set, the neural network's ability to apply the learned functions to new samples is improved

Table 4.1 Comparison of the best results for white noise (variance = 5.00E-06), pink noise (amplitude = 5.00E-05) and mixed noise (variance = 5.00E-05, amplitude = 1.00E-05)

	Test MSE A $[m^2]$	Test MSE B $[m^2]$	Test MSE C $[m^2]$	Test MSE D $[m^2]$	Improvement of Standard Deviation of test MSEs $[m^2]$	Improvement of Sum of test MSEs $[m^2]$
Baseline	0.0526	0.3098	0.2486	0.1798	0%	0%
White Noise	0.0509	0.2917	0.2412	0.1362	2.36%	8.95%
Pink Noise	0.0502	0.2997	0.2635	0.1358	-4.52%	5.25%
Mixed Noise	0.0612	0.2692	0.2441	0.1379	12.55%	9.91%

Bibliography

[1] Osama Bin Tariq, Mihai Teodor Lazarescu, and Luciano Lavagno, “Neural Networks for Indoor Human Activity Reconstructions,” IEEE Sensors Journal, vol. 20, no. 22, November 15, 2020.

[2] World Population Ageing 2019 Highlights, United Nations, New York, NY , USA, 2019.

[3] Zimmerman, T.G.; Smith, J.R.; Paradiso, J.A.; Allport, D.; Gershenfeld, N. Applying electric field sensing to human-computer interfaces. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Denver, CO, USA, 7–11 May 1995; ACM Press/Addison-Wesley Publishing Co.: New York, NY , USA, 1995; pp. 280–287.

[4] Alireza Ramezani Akhmareh, Mihai Teodor Lazarescu, Osama Bin Tariq and Luciano Lavagno, “A Tagless Indoor Localization System Based on Capacitive Sensing Technology,” Sensors, 16(9):1448, 2016.

[5] Osama Bin Tariq, Mihai Teodor Lazarescu, Javed Iqbal, and Luciano Lavagno, “Performance of Machine Learning Classifiers for Indoor Person Localization With Capacitive Sensors,” IEEE Access, vol. 5, pp. 12913–12926, 2017.

[6] Andrew Ng, "Neural networks and deep learning," https://www.coursera.org/learn/neural-networks-deep-learning.

[7] “What is an activation function? What are commonly used activation functions?” https://www.datasciencepreparation.com/blog/articles/what-is-an-activation-function-what-are-commonly-used-activation-functions/

[8] Sumit Saha, "A Comprehensive Guide to Convolutional Neural Networks," https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[9] Abid Ali Awan, "Recurrent Neural Network Tutorial (RNN)," https://www.datacamp.com/tutorial/tutorial-for-recurrent-neural-network

[10] SuperDataScience Team, "Recurrent Neural Networks (RNN) - Long Short Term Memory (LSTM)," https://www.superdatascience.com/blogs/recurrent-neural-networks-rnn-long-short-term-memory-lstm

[11] Sebastian Ruder, "An overview of gradient descent optimization algorithms," https://ruder.io/optimizing-gradient-descent/index.html#fn5

[12] Diederik P. Kingma and Jimmy Lei Ba, "Adam: A Method for Stochastic Optimization," arXiv:1412.6980v9 [cs.LG],30 Jan, 2017.

[13] Wikipedia, "Additive white Gaussian noise," https://en.wikipedia.org/wiki/Additive_white_Gaussian_noise

[14] Wikipedia, "Pink noise," https://en.wikipedia.org/wiki/Pink_noise