Active Noise Cancellation Using Atrous Scaled Convolution Recurrent Neural Networks

By

Sukhpreet Singh Benipal

A Thesis submitted to the Faculty of Graduate Studies of

The University of Manitoba

in partial fulfilment of the requirements of the degree of

MASTER OF SCIENCE

Department of Civil Engineering

Faculty of Civil Engineering

University of Manitoba

Winnipeg

ABSTRACT

It has been proven that the external environmental noises affect the human mental health and performances of works. To avoid these detrimental effects, different passive and active noise control system had been developed. Moreover, the focus on active noise control has been increased because of availability of efficient circuits and computational power. However, most of the active noise cancellation systems are based on traditional modelling with limited efficiency. However, in this study, I propose a deep learning based active noise cancellation system which can perform well under different environmental noises. It has been shown in this study that the performance of the proposed methodology is superior to traditional and machine learning based models.

ACKNOWLEDGEMENT

I am deeply debited to Prof. Dr. Young-Jin Cha for providing constant assistance and believing in me. I strongly admire his work ethic and his ability to empathize, which helped me in my ups and downs.

I am extremely grateful to my parents who trusted me and are always there for help. The words of encouragement from them always helps me to move forward and keep me grounded. It is because of them that I moved to whole new country to follow my dream and make them proud.

Many thanks to my colleagues and friends for their technical assistance and making this journey even more wonderful.

The research was supported by NSERC Engage grant (Application No.: 533690 – 18), partially an NSERC Discovery Grant (RGPIN-2016-05923), and University of Manitoba Graduate Fellowship.

Table of Contents

List of Tables

List of Figures

Chapter 1. Introduction

Increased environmental noise is detrimental to human health, causing increased blood pressure and long-term memory loss (Stansfeld and Matheson, 2003). Gupta and Ghatak (2011) noted that higher noise levels can cause insomnia, irregular heart rate, and speech-related problems. The main sources of environmental noise include airplanes, mining, electric transformers (Zhao and Liang, 2016), construction sites (Suter et al., 2002), and construction machinery (Kwon and Park, 2016). In order to cancel noise, Kwon and Park (2016) proposed an active noise cancellation system for a construction site. This study shows that the system can cancel the noise generated by heavy equipment (i.e., earth augers and dump trucks).

To avoid the negative effects of excessive noise, researchers have proposed ideas for both passive noise cancellation (Ross and Burdisso, 1999) and active noise cancellation (Gonzalez and Ferrer, 2003). Passive noise cancellation is an isolation of the space from external noises. This isolation can be achieved by using a design that absorbs noise and vibrational energy (Denenberg et al., 1992). Marburg et al. (2002) used structural-acoustic optimization to control environmental noises. Other than structural modification, materials such as soundproofing foam, mass loaded vinyl, and acoustic fabric can be used.

On the other hand, active noise cancellation is the cancellation of external noises by producing anti-noises (Denenberg, 1992). This type of noise cancellation takes advantage of destructive interference (Denenberg, 1992), which cancels the noise waves with waves of the opposite phase. Researchers have proposed different types of active noise cancellation methods, such as adaptive feed-forward active noise cancellation (Elliott and Nelson, 1983) and feedback active noise cancellation for ducts (Tichy and Warnaka, 1983). Each of these methods uses different approaches to generate the anti-noise to cancel external noises.

As an adaptive/hybrid feed-forward system, Elliott and Nelson (1983) proposed a finite impulse response (FIR) filter using a filtered-x least mean square algorithm to generate anti-noises. This algorithm considers the secondary path [i.e., the path traveled by anti-noises from the point of origin (i.e., the noise cancellation speaker) to the error microphone]. The filtered-x least mean square algorithm is the most popular algorithm because of its simplicity. However, the algorithm does not work efficiently in certain environments where noises are nonstationary. Some methods have been proposed to overcome the limitations of this algorithm; for example, Rey and Bitmead (1991) introduced the "importance leakage" concept to calculate the weights of the FIR filter to address the stalling problem of the filtered-x least mean square algorithm. Hesselbach and Hoffmeister (2009) applied a recursive least mean square algorithm and a filtered-x least mean square algorithm to control the vibration noises in a wood machine. Hesselbach and Hoffmeister (2009) claimed that the algorithm could successfully reduce the noise amplitude due to vibration by 30 dB. Chang and Chen (2010) provided the applicability of the adaptive genetic algorithm in the field of active noise cancellation. The author mentions that this algorithm addresses the limitation of the filtered-x least mean square algorithm being stuck on the local minima while converging the weights for the FIR filters.

Even though the aforementioned algorithms are simple, they are ineffective in cancelling complex noises. Traditional machine learning methods were used to overcome this drawback. Chen and Chiueh (1996) proposed multilayer perceptron (MLP) based active noise cancellation. Na and Chae (1997) applied the Elman recurrent neural network for active noise cancellation. Na and Chae (1997) showed that Elman recurrent neural network with 22.35 dB noise attenuation performed better than MLP (20.83 dB noise attenuation) and filtered LMS (14.35 noise attenuation). These traditional machine learning-based approaches also only work well for noises coming from a specific environment; they are not adaptive to different noises that were not used to train the model, and they only work well in narrow-band noises.

Introduction of deep learning have opened up the possibility of addressing the limitations of traditional machine learning-based noise cancellation methods. It is possible to train machines to learn very complex nonstationary patterns and behaviors since big data and computational power are available (Deng and Yu, 2014). Deep learning models have been successfully deployed for damage-detection tasks using computer vision images (Cha et al., 2017; Cha et al., 2018) and one-dimensional time series data (Cai and Pipattanasomporn, 2019; Wang and Cha, 2020). There are some deep learning-based active noise cancellation methods, but the performance must be improved for them to be robust methods that can be used in versatile environments.

1.1 Problem definition

In past few decades, researchers have shown the importance of noise cancellation and how it can improve the cognitive behaviour of human beings. Moreover, because of restriction on structural and economical limitation, the focus shifts towards the active noise cancellation. Because of the complex acoustic behaviours of the environmental noises, the traditional models have limitations to work well under those conditions. To avoid the negative effect on the people and to cancel out unwanted noises efficiently, a new active noise cancellation method is needed.

1.2 Research objective

The main objective of this study is to create a deep learning network that can produce anti-noises effectively and efficiently for the noises coming from the vehicles, construction cites and airplane cockpits. Due to highly nonlinear and nonstationary characteristics of the noises, possibility of the implication of a new deep learning network is explored in this study. The model performance must surpass the previous algorithms used in the field of active noise cancellation. Moreover, specific objectives of this thesis are as follows:

1. Developing a deep learning network to create the efficient anti-noise model.
2. Study the behaviour of the model when is deployed in the new environmental noises.
3. Fine tuning of the hyper-parameters of the proposed approach, so that it can work well in many types of noises.
4. Comparative studies with existing machine learning and deep learning-based approaches.

1.3 Methodology

In order to generate effective anti-noises, a deep learning network as one of active noise cancellation methods is proposed. The role of deep learning is to produce anti-noises to cancel unwanted noises in many different environmental conditions. The proposed deep learning network is based on the feature extraction using a one-dimensional (1D) atrous convolution, casual convolution layers, recurrent neural network, etc. Exponential rates of 1D atrous convolutional layers are used to determine the atrous rate for each layer. These features then were fed into a recurrent unit which then is followed by dense layers. Different activation functions are used after the convolution layers. This deep architecture of the model enables to learn more invariant features from the data set and provides the more detailed spatiotemporal information. A detailed explanation about the proposed deep learning network is given in the Chapter 3.

1.4 Thesis organization

Following are the chapters including:

- Chapter 1 includes the introduction to the problem statement along with brief on the past studies done in the same field. Research objective and short methodology descriptions are also included in this chapter.

- Chapter 2 includes the review of the literature for active noise cancellation. Detailed explanation of active noise cancellation systems and algorithms is provided in this chapter along with the drawbacks of those methods.

- Chapter 3 provides detailed explanation of the proposed method. Details of the different components of the deep learning network are explained.

- Chapter 4 describes detailed information on the database providing for different noises.

- Chapter 5 describes the training and testing of the proposed method using various noises. Hyper parameters of the method are explained in this chapter.

- Chapter 6 provides comparative studies of the performances of the proposed method and the models proposed by other researchers.

- Chapter 7 describes the summary, and conclusion of the study. Limitation and future scope are also discussed in this chapter.

Chapter 2. Literature Review

Environmental noise can adversely affect human performance, making people prone to more mistakes (Nassiri and Monazam, 2013). To control environmental noise, many studies have been carried out, including active noise control and passive noise control. Active noise control includes secondary sound sources, such as speakers, which take advantage of destructive interference to control noise in the surrounding environment (Kuo and Morgan, 1999). These secondary sound sources produce sound waves or vibrations having the same amplitudes but opposite phases as that of the noises to cancel them out. On the other hand, passive noise control systems include noise absorbers, dampers, barriers, or similar items. However, these external materials make the structures heavy and hinder the working space (Liu and Lee, 2006). To reduce costs and to meet other structure-related constraints, such as weights and dimensions, active noise cancellation is preferred. This section covers an extensive review of active noise control systems and algorithms.

2.1 Active noise cancellation systems

A typical active noise cancellation (ANC) system consists of the primary noise source and an anti-noise signal generator, such as a speaker, along with noise measurement sensors, such as microphones. The system produces an anti-noise signal by inputting the noise from noise measurement sensors to a digital controller and then to an anti-noise generator such as a speaker, actuator, or other device (Kuo and Morgan, 1999). The selection of sensors to capture the noise plays a critical role in the efficacy of these system.

The suitability of the noise measurement sensors depends upon the nature of the primary noise source. Non-periodic and nonstationary noises are usually measured by microphones, while harmonic and repetitive noises are measured by tachometers (Kuo et al., 2000). Active noise cancellation systems can be broadly classified into feed-forward ANC systems, feedback ANC systems, and hybrid ANC systems. A detailed explanation of these systems is given in the following sections.

2.1.1 Feed-forward ANC system

A feed-forward ANC system takes advantage of the reference signal coming directly from the primary noise measuring sensor placed near the primary noise source. Feed-forward ANC systems can be differentiated into broadband and narrow-band ANC systems. The broadband feed-forward ANC system consists of a primary noise sensor (e.g., a microphone), an anti-noise signal generator (e.g., a speaker), and a digital ANC controller. The reference sensor is placed near the primary noise source, as shown in Figure 2-1. G(z) and S(z) are the z-transforms that change discrete time signals to a complex frequency-domain representation. When the primary noise signal x(n) travels through the environment G(z), the environment modifies the incoming signal x(n). The path followed by the signal from the primary noise source to the primary noise sensor is known as the primary path. Similarly, when the anti-noise signal travels through the environment S(z), it is modified. This path from the anti-noise generator to the error sensor is called the secondary path. Figure 2-1 shows a feed-forward ANC system. The signal x(n) of primary noise passes through environment G(z) which changes it into p(n). The anti-noise signal y(n) passes through S(z), which converts it into h(n). The error e(n) is the remaining noise that the user hears. This feed-forward approach cannot address the actual remaining error noises, and it only works well in known environments.

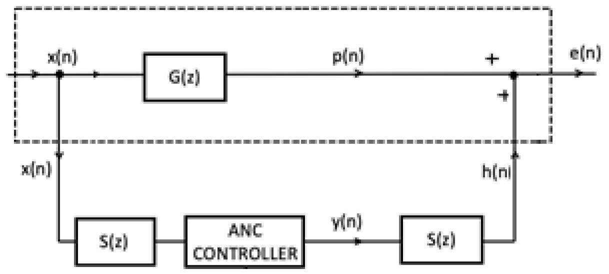


Figure 2-1: Broadband feed forward ANC system

However, a narrow-band feed-forward ANC system is useful for harmonic or periodic noises, which are easy to predict in future noise patterns. In this type of system, the primary noise sensor is replaced by a non-acoustic sensor (i.e., a tachometer) (Kuo et al., 2000). The anti-noise signal is generated based on the electrical signal coming from the non-acoustic sensor. The availability of only driving frequency is what limits the applicability of this system to periodic noises.

2.1.2 Feedback ANC system

Feedback ANC systems are different from feed-forward systems in the way they use error noise measuring sensors, as shown in Figure 2-2. The error, which is the sum of the primary noise signal and the anti-noise, is inputted into an ANC controller to generate anti-noises to cancel the error noises. The anti-noise signal is generated from the error signal e(n), which is captured using a microphone. This is done with a single microphone, making the system more compact. The advantage of this approach is that the ANC controller can adjust to the remaining error noises if we can develop an advanced ANC controller. Furthermore, we do not need clear environmental information regarding the primary and secondary paths.

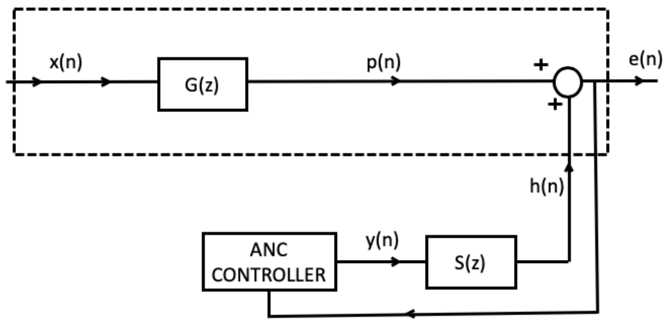


Figure 2-2: Feedback ANC system

2.1.3 Hybrid ANC system

A hybrid ANC system is the integration of feed-forward and feedback ANC systems. Therefore, it uses a primary noise sensor and an error noise sensor together, as shown in Figure 2-3. It is the most efficient method of ANC, but it requires two sensors. Digital filters are traditionally used as feed-forward ANC controllers, receiving the primary noise signals and converting them into anti-noise signals based on the filter coefficients. These filter coefficients are calculated using different adaptive algorithms, such as the least mean square. Infinite impulse response (IIR) (Shynk et al., 1989) filters and FIR (NagaJyothi and SriDevi, 2017) filters are the most popular digital filters.

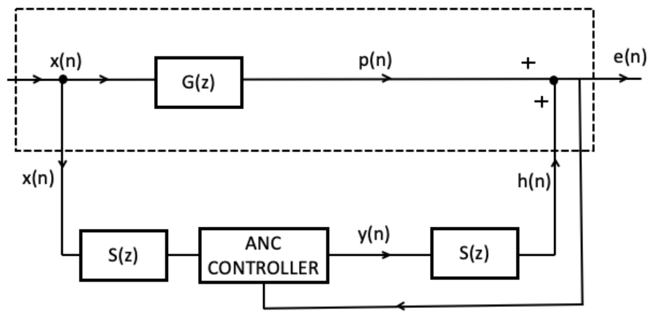


Figure 2-3: Hybrid ANC system

In the past few decades, adaptive algorithms have been used in feedback and hybrid ANC controllers. The idea behind almost every adaptive method is to minimize the error by tuning the weight/filter coefficients of the controller. The most simplest form of adaptive method being used in the field is the least mean square (LMS) algorithm. There are some traditional adaptive algorithms. The filtered-X LMS algorithm (Bjarnason et al., 1995) was developed to consider that there are some components of the system that cause frequency and phase distortion of the anti-noise signal from the adaptive filter to the error microphone. To counteract the effect of the secondary path, an inverse filter is introduced between the reference signal and the LMS algorithm (Kuo and Morgan, 1996; Douglas et al., 1999; Larsson et al., 2011). This algorithm is most commonly used in the application of ANC because of its simplicity and lower memory consumption.

A leaky filtered-X LMS algorithm addresses the instability caused by finite arithmetic precision due to very small updates (Tobias and Seara, 2002; Cartes and Ray, 2002). The problem is corrected by eliminating a small part of the weight updates by implementing leakage with the gradient descent algorithm.

2.2 Machine learning-based ANC controllers

Chen and Chiueh (1996) provide the applicability of multi-layered neural networks in the field of ANC. A multi-layered neural network uses stacked layers of neurons to approximate the function. Each layer of neurons is connected to another layer by a set of weighted connections. Along with these weighted connections, a bias is added to improve the prediction of the network. The set of weights is adjusted using back-propagation through a training process. Updating the set of weights depends on the cost function used to evaluate the performance of the multi-layered neural network. A stochastic gradient descent algorithm (Ruder et al., 2016) can be used to optimize the network parameters. The advantage of the algorithm is that it keeps moving in the direction of a negative gradient.

Researchers have shown the efficacy of the Elman recurrent neural network in system identification and temporal prediction (Jaeger et al., 2003; Li and Ho, 2005; Coulibaly and Baldwin, 2005). The Elman recurrent neural network (Elman et al., 1990) consists of weighted looped connections. These internal hidden-layer looped connections, known as the recurrent layer, help the network to possess short-term memory. Therefore, these networks are used to extract the features and patterns from past data to predict the new system. Temporal supports come from the recurrent layers in Elman neural networks, and this support makes these networks more useful in series classification and prediction (Toha and Tokhi, 2008). The number of recurrent layers follows the number of hidden-layer neurons. Each neuron is connected to neurons in another layer using a set of weighted connections. Na and Chae (1997) have shown the application of an Elman recurrent neural network for a single-sensor feedback ANC system. It has been shown that the aforementioned network outperforms the traditional methods. The Elman recurrent neural network achieved 22.35 dB of noise attenuation, whereas the filtered-x LMS algorithm achieved only 14.35 dB, for noise from a moisture removing machine.

Convolution neural networks (CNNs) have been successfully applied in the domain of image-based object recognition (Krizhevsky and Sutskever, 2012; Iandola and Han, 2016) and natural language processing (Britz et al., 2015). Recently, researchers have started to apply these CNN architectures to predict and classify time series data (Zhao and Lu 2017). CNNs use convolutions over time stamps (Fawaz and Forestier, 2019).)

A number of filters can be applied to the input data to change the univariate time data into multi-variate. This multi-variation of the data helps a CNN to extract the features to classify or predict the time steps (Fawaz and Forestier, 2019). A CNN overcomes the limitation of single filter usage in a multilayer perceptron (MLP) network, which limits the ability of an MLP to generalize. In addition to convolution layers, different operations can be performed, such as max pooling, to control the spatial dimensionality (Aloysius and Geetha, 2017). These layers can help the network to learn well with fewer parameters. Toward the end of the network, the max pool layer is flattened and connected with a dense layer to generate the output. Long short-term memory (LSTM) is a one of the types of recurrent neural network (RNN) that includes a memory unit. This type of LSTM network addresses the problem of the vanishing gradient or gradient blow-up present in the Elman recurrent neural network (Hochreiter and Schmidhuber, 1997). Researchers have shown that LSTM works well by remembering long data for prediction problems (Fu and Zhang, 2016; Duan and Yisheng, 2016). An LSTM memory unit consists of a gate mechanism to control the flow of information. Gates can either let the information pass through or stop it, and they consist of pointwise multiplication with sigmoid activation functions. A typical LSTM unit is composed of an input gate, a forget gate, a cell state, and an output gate. The input gate regulates the passing of incoming information to the LSTM unit. The forget gate decides if the LSTM unit needs to be kept or discarded. The cell state stores the important information in the LSTM unit. The output gate determines the next hidden state of the LSTM unit. The past hidden state and current state are passed through an activation function.

Park and Patterson (2019) applied an LSTM for active noise cancellation along with Convolution neural networks, Recurrent neural network and MLP. Park and Patterson (2019) used a simple LSTM unit with 20 input neurons. The results of this study show that the LSTM results closely mimic the results of an MLP, as LSTM is highly dependent on long-term temporal dependencies, but the input is relatively short. Based on these recent studies, a new deep learning-based active noise cancellation algorithm is proposed to improve the performance of the ANC controller. The detailed method is described in Chapter 3.

Chapter 3. Methodology

This chapter includes a detailed explanation of the proposed deep learning-based active noise cancellation system. Section 3.1 explains the schematics of the implementation of the proposed method in the active noise cancellation system. The remaining sections explain the proposed method and initialization of the hyperparameters.

3.1 Overview of proposed method:

In order to effectively cancel noises, a hybrid deep 1D atrous convolutional and recurrent neural network (ACRN) based active noise cancellation (ANC) method is proposed, as shown in Figure 3-1. In Figure 3-1, S(z) is the z-function that represents the secondary environment due to secondary path environments, x(n) is the source of the noises, h(n) is the anti-noise generated by the ACRN, y(n) is the final anti-noise applied, and e(n) is the remaining noise error.

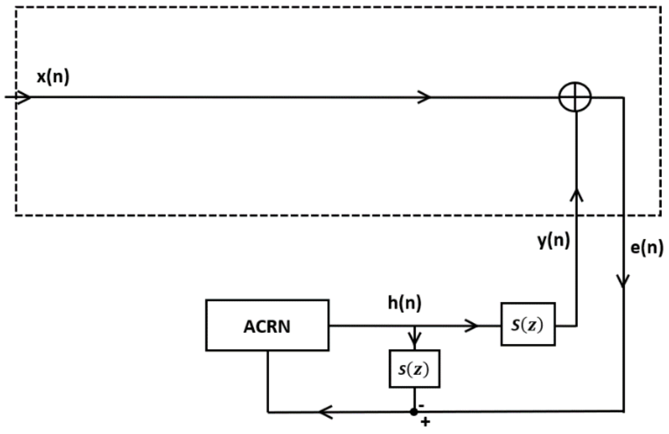


Figure 3-1: Block diagram of proposed feedback ANC system

3.2 Proposed architecture

The proposed ACRN architecture is inspired by the Wavenet architecture (Oord and Dieleman, 2016) for generating speech purposes and DaNSe (Mishra and Basu, 2019) for electricity load forecasting purposes. The details of the ACRN are illustrated in Figure 3-2. It is carefully designed by integration of traditional convolutions, atrous scaled convolution (ASC) modules [composed of 1D atrous convolution, a scaled exponential linear unit (SeLU), and pointwise convolution], a recurrent neural network (RNN), and dense layers to generate anti-noise signals. The ASC modules use 1D atrous convolution to expand the receptive field without increasing computational cost and to extract multiple levels of invariant features from the input data. The RNN is used to capture spatiotemporal information of the noises. To avoid the problem of a vanishing gradient, skip/residual connections are provided from one block to another. He and Zhang (2016) suggest that residual connections can make network optimization easier and can boost accuracy.

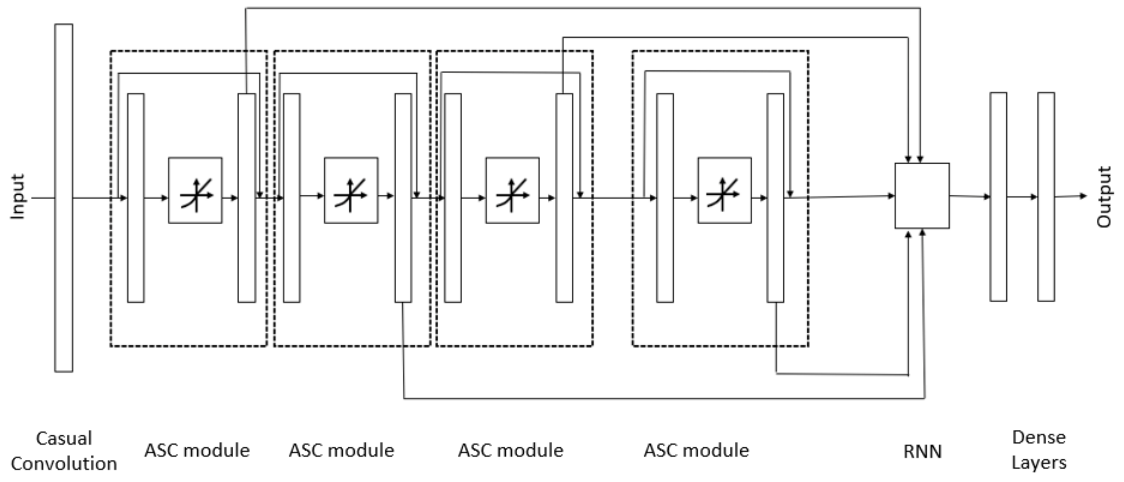


Figure 3-2: Proposed deep ACRN model

The ASC module is concatenated, as shown in Figure 3-3. The extracted multi-level features are then fed into recurrent neural units along with the skip connections from the traditional convolution to avoid loss of low-level features. The recurrent cells in the RNN performs the recurrence and captures temporal information of the input data. Output from each residual unit is then feed-forward into the fully connected layers to predict the anti-noise needed to cancel noises. The details of each operator are explained in the following sections.

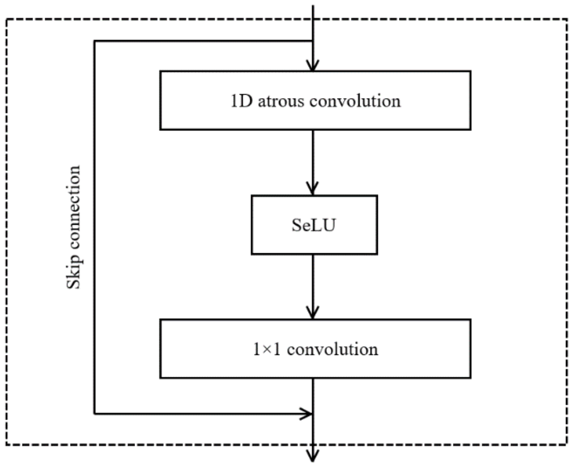


Figure 3-3: Developed ASC module

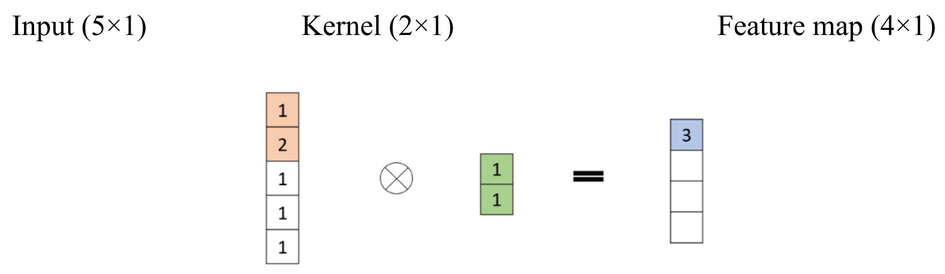
3.2.1 Convolution operation:

The convolution layer is the main building block of traditional convolution neural networks (CNNs). The hyperparameters required for a typical convolution layer are the number of filters and the filter (i.e., kernel) sizes. The mathematical meaning of the convolution is the dot product using the filter, which has learnable parameters through the training process of the proposed network. These convolution layers mimic the response of the visual cortex when subject to different patterns (Chalkiadakis et al., 2016; Jogin and Madhulika, 2018). The main advantage of the convolution layer is spatial and local connectivity. The spatial and local connectivity is maintained by convolving smaller filters over the input channels and then passing the output to the next neuron or filter.

The filters possess the ability to extract the features from the input data. For example, a filter of size 32×32 has 1,024 trainable parameters. The initial values of these filters are randomly assigned. If the size of the filter is 1×1, it is called a pointwise convolution. The number of features extracted from a given input data series is dependent on the number of filters. The process of convolving the filters over the input refers to superimposing the filter on top of the input and calculating the dot product of the overlapping area, as shown in Figure 3-4. After one convolution, the filter moves one pixel to the right or down depending on the array of the input data and then performs the dot product again. The number of pixels that a filter moves is expressed as stride value. The size of the output feature map can be mathematically given in Equation (3.1).

$$O = \frac{I - K}{S} + 1 \tag{3.1}$$

where O is the size of the feature map, I is the size of the input, K is size of the filter, and S is the stride size. Figure 3-4 shows the 2×1 filter convolving along the input data of dimension 5×1 with stride 1.



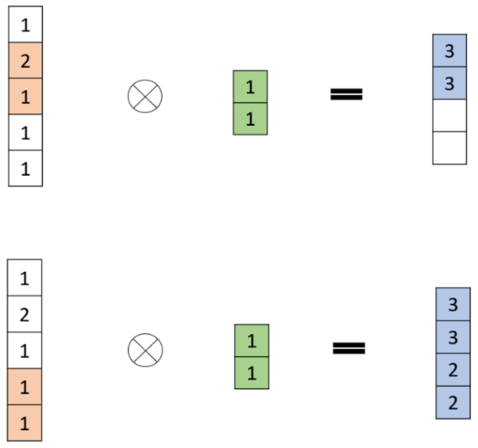


Figure 3-4: Casual convolutional operation

3.2.2 Atrous convolution:

In order to cancel existing noises, an inverse noise pulse should be generated. The inverse noise signals can be predicted by the proposed deep ACRN controller. Noise prediction is dependent on the previous noise data points. To predict the likelihood of the data point x(t) at time t, a 1D atrous convolution is used. Figure 3-5 shows the schematic view of atrous convolution with rates of one, two, and four. These rates of filter convolution reduce the computational cost and increase the output speed.

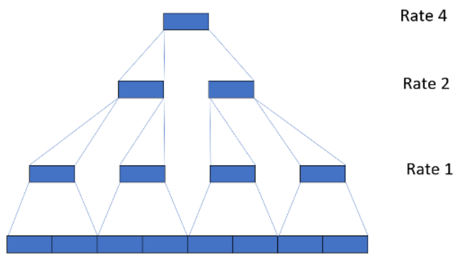


Figure 3-5: 1D atrous convolution

The exponential growth of the dilation rate gives the exponential growth of the receptive field without increasing the number of computations. In the proposed method, the dilation rate for each dilated layer is given in Equation (3.2). Each dilated layer is then followed by a non-linearity.

$$d \in \{2^0, 2^1, 2^2, \ldots.., 2^{L-1}\}, \tag{3.2}$$

where d is the dilation rate and L is the number of stacked ASC modules. The output feature maps from the 1D atrous convolution can be mathematically described in Equation (3.3) (Borovykh and Bohte, 2018).

$$z^l(i,h) = \sum_{j=-\infty}^{j=\infty} \sum_{m=1}^{M_{l-1}} w_h^l(j,m)\, f^{l-1}(i - d.j, m) \tag{3.3}$$

where $z^l(i,h)$ represents feature maps from dilation, d is dilation rate, f^{l-1} is the input feature map from the previous convolution, $w_h^l(j,m)$ is the filter weights, and M_l is the number of channels.

3.2.3 Recurrent neural networks:

Recurrent neural networks (RNNs) are mostly used for their faster convergent capabilities to map non-linear prediction (Wang and Fang, 2016). A RNN captures the temporal information of the data by making recurrent connections. A simple recurrent unit has a loop to preserve the information over time. Figure 3-6 shows a traditional recurrent unit. In Figure 3-6, x_0 is the input at time step t=0 and h_t is the output of the unit at the same time. Apart from computing the output at each time step, the hidden state of the unit (F) is also calculated. Then this hidden state at time t=0 is passed to next time step at time t=1. The hidden state from the previous hidden state is connected consecutively to next hidden state by set of weights. Equation 3.4 shows the mathematical form of recurrence relation related to each time step.

$$F_t = \mathrm{F}(F_{t-1}, x_n) \tag{3.4}$$

where F_t = hidden state of the unit, F_{n-1} = hidden state of the previous unit, x_n= input at time step t, F = function set by weighted connections.

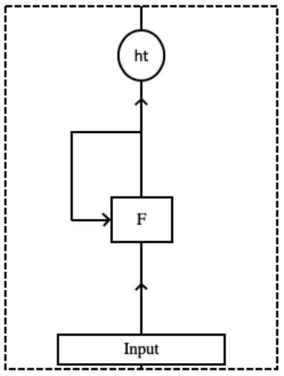


Figure 3-6: Traditional recurrent unit.

Equation 3.5 represents the calculation of hidden state and Equation (3.6) output from a recurrent unit.

$$F_t = \tanh\,(x_n . U + W . F_{t-1}), \tag{3.5}$$

$$h_t = \mathrm{softmax}\,(V . F_t), \tag{3.6}$$

where x_n = input at time ‘t’, U = the weights of input to hidden cell, W = the weighted hidden state, F_{t-1} = the previous hidden state, and V = the weight of hidden cell to output.

3.2.4 Fully connected layer:

Fully connected layers shown in Figure 3-7 are sets of neurons connected to each other, similar to in an MLP. The connections are composed of a set of weights that are updated through back-propagation in the training process. The features from the previous layers are flattened down to a single vector.

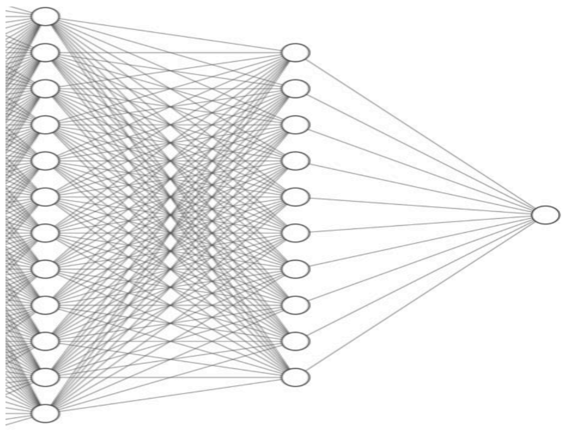


Figure 3-7: Example of fully connected layers

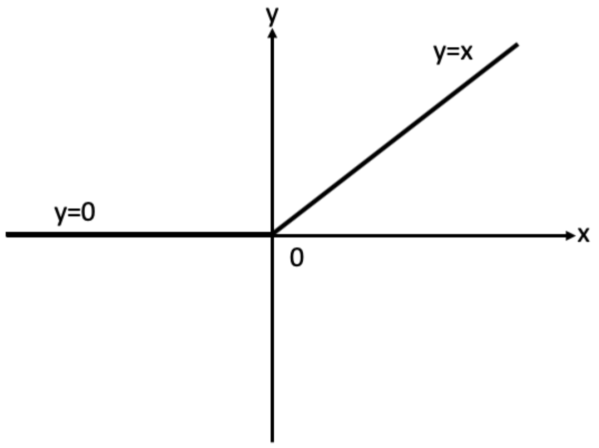
3.2.5 Activation function:

In the proposed architecture, two non-linear functions, a rectified linear unit (ReLU) and a scaled exponential linear unit (SeLU) are used. The following is the detailed explanation of these functions.

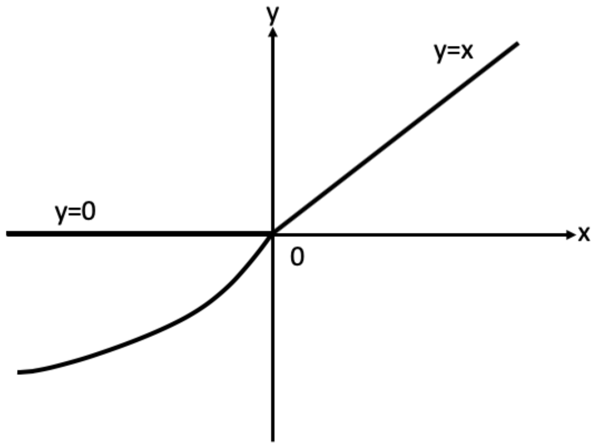
ReLU: is comparatively easy to train, mathematically defined in Equation (3.7), and visualized as shown in Figure 3-8(a).

$$output = \begin{cases} 0\,; & input \leq 0 \\ input\,; & input > 0 \end{cases} \tag{3.7}$$

Considering the close resemblance of ReLU to linear functions, they are easy to train using gradient-based optimizations. Unlike other nonlinear functions, such as sigmoid and tanh, it does not have the problem of gradient saturation. Most convolution neural networks use this activation function. Another main advantage of the ReLU is its sparsity (Nair and Hinton, 2010). The output will be zero when the input is less than or equal to zero (Figure 3-8(a)), which makes some neurons inactive. These types of dead neurons make the model more concise and less overfitted to the data.



(a) ReLU function



(b) SeLU function

Figure 3-8: Activation functions

SeLU: can be mathematically defined in Equation (3.8).

$$output = \lambda \begin{cases} \alpha e^{input} - \alpha\,; & input \leq 0 \\ input\,; & input > 0 \end{cases} \tag{3.8}$$

Figure 3-8(b) shows the graphical representation of SeLU. This function mimics ReLU when the input is greater than zero. However, when the input is less than equal to zero than the output is defined by two parameters 'α' and 'λ'. The values of λ and α are given as α = 1.6733, λ = 1.0507 (Klambauer and Unterthiner, 2017). This function converges the weights towards zero mean and unit variance. Because of scaled output SeLU is very robust and can be trained on the many deep layers.

3.2.6 Skip connections

In order to address the problem of the accuracy saturation and degradation, skip connections were introduced (He and Zhang, 2016). The model needs to learn a mapping function $\aleph$ given input x as shown in Equation (3.9). Figure 3-10 shows the ASC module from the proposed architecture of ACRN.

$$\aleph(x) = y \tag{3.9}$$

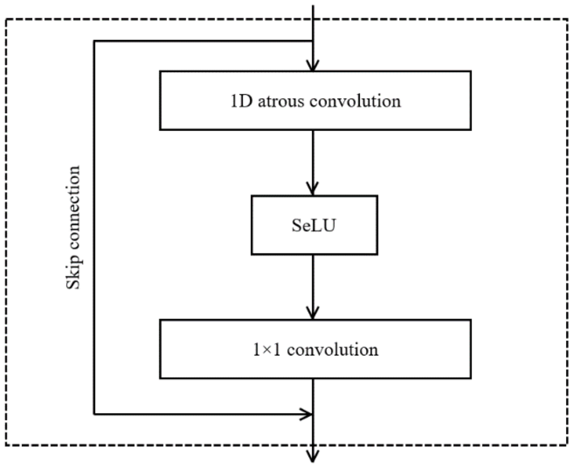


Figure 3-9: Skip connection

These skip connections help the model to take advantage of difference between the $\aleph(x)$ and x given in Equation (3.10).

$$\aleph(x) = \mathcal{F}(x) + x. \tag{3.10}$$

He and Zhang (2016) state the optimization of $\mathcal{F}(x)$, which is the function estimated by nonlinear layers, is easier than that of $\aleph(x)$.

Four ASC modules are selected after casual convolution with one filter and kernel size of 2. Tanh activation is used for casual convolution along with a bias. Table 3-1 shows the parameters used for the ASC modules. Zero padding is used to maintain the data size from convolution layers.

A dropout of 10% was used for the fourth atrous convolution layer to avoid overfitting. Two recurrent units are used, followed by two dense layers with 18 units plus ReLU activation and 1 unit plus linear activation. The total number of trainable parameters is 444.

Table 3-1: Parameters for ASC module

1D atrous convolution layers	**# of filters**	**Size of filter**	**Rate**	**Activation**
Layer 1	10	2	1	SeLU
Layer 2	20	2	2	SeLU
Layer 3	30	2	4	SeLU
Layer 4	40	2	8	SeLU

3.2.7 Filter weight initialization

The weights of filters are initialized from a Gaussian distribution having standard deviation between specified values and mean of zero. In order to make the results reproduceable, a random number generator was fed with the constant seed of 2019. In order to reduce the overfitting and better generalization, $l2_{regularization}$ in Equation (3.11) was applied on the kernels (Cortes and Mohri, 2012).

$$l2_{regularization} = \lambda \sum_{i=1}^{n} {w_i}^2 \tag{3.11}$$

where λ is a regularization parameter and w_i is i^{th} weight.

3.2.8 Loss function:

A mean square error (MSE) was used as a Loss function in the suggested model to train the proposed ACRN. It is mean squared absolute difference of target and predicted values. Equation (3.12) shows the formula for MSE.

$$MSE = \frac{1}{N}\sum_{K=0}^{N} |predicted_K - true_K|^2 \tag{3.12}$$

where N is number of data points.

Because we are using the mean squared difference of absolute values, the bigger weight changes were applied to the bigger errors. MSE is considered more suitable than mean absolute error for a dataset that is closer to zero (Hyndman and Koehler, 2006).

3.2.9 Network optimizer

To optimize the tunable parameters of the network, an ADAM optimizer (Kingma and Ba, 2014) is used. ADAM is a combination of the RMS prop (Tieleman and Hinton, 2012) and stochastic gradient descent with momentum (SGDM) (Qian et al., 1999). The optimization of the network is based on calculating momentum by moving average and correcting learning rates using the gradient squares.

The loss function as $E_{w,b}$ and gradient descent of mini batch for weights 'w' and bias 'b' is given in Equations (3.13-15), respectively.

$$E_{w,b} = \tfrac{1}{N}\textstyle\sum_{K=0}^{N} |predicted - true|^2$$

$$E_{w,b} = \tfrac{1}{N}\textstyle\sum_{K=0}^{N} |error_K|^2 \tag{3.13}$$

For individual data set,

$$\frac{\partial E}{\partial w} = 2. error. \frac{\partial}{\partial w} |error| \tag{3.14}$$

$$\frac{\partial E}{\partial b} = 2. error. \frac{\partial}{\partial b} |error|. \tag{3.15}$$

The ADAM optimizer uses these mini batch gradients to calculate the moving average and squared gradients. The mathematical representation of the moving average gradient and squared gradient for mini batch 'j' is given in the Equation (3.16) and Equation (3.17). β_1 and β_2 are hyper-parameters having values of 0.9 and 0.999 (Kingma and Ba, 2014).

$$m_j = \beta_1 m_{j-1} + (1 - \beta_1) \frac{\partial E_j}{\partial w_j} \tag{3.16}$$

$$v_j = \beta_2 v_{j-1} + (1 - \beta_2) \frac{\partial E_j}{\partial w_j}^2 \tag{3.17}$$

Bias corrections on the estimators m'_j and v'_j are performed using the Equation (3.18) (Kingma and Ba, 2014).

$$\begin{gathered} m'_j = \frac{m_j}{1 - \beta_1^j} \\ v'_j = \frac{v_j}{1 - \beta_2^j} \end{gathered} \tag{3.18}$$

Finally, the weight updating with learning rate η and $\varepsilon = 10^{-8}$ is given as follows:

$$w_1 = w_0 - \eta \frac{m'_1}{\sqrt{v'_1} + \varepsilon}$$

$$w_2 = w_0 - \eta \frac{m'_1}{\sqrt{v'_1} + \varepsilon} - \eta \frac{m'_2}{\sqrt{v'_2} + \varepsilon}$$

$$w_3 = w_0 - \eta \frac{m'_1}{\sqrt{v'_1} + \varepsilon} - \eta \frac{m'_2}{\sqrt{v'_2} + \varepsilon} - \eta \frac{m'_3}{\sqrt{v'_3} + \varepsilon}$$

$$w_j = w_{j-1} - \eta \frac{m'_j}{\sqrt{v'_j} + \varepsilon}$$

Chapter 4. Network Training and Testing

To train and test the proposed deep learning-based ANC, various noise data were established. Using these data sets, various parametric studies of the proposed method, including training and testing, are reported in this chapter.

4.1 Established datasets

To measure the performance of the model, four different types of noisy environments were considered: airplane cockpits, a vehicle interior, a military vehicle interior and construction sites. Noise data from the last three environments were collected from the online data repository Signal processing information base (SPIB, 2020). Details of the noise data sets from these environments are as follows:

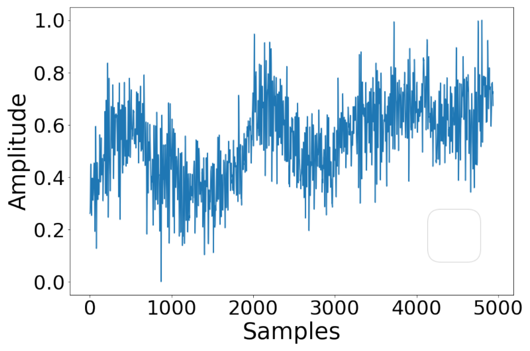
- Airplane cockpit noise data from various flights. The sounds were analyzed with the help of a NTiXL2 sound level meter with a type-1 NTi microphone. The sounds were recorded on a number of flights using an M-Audio data recorder, and the data extracted from these flights were divided into a number of small data sets named Flight1, Flight2, Aero1, Aero2, Aero3 and Aero4. Figure 4-1 shows the data used for the training.
- Vehicle interior noise (Volvo 340): This data set was gathered while the vehicle was being driven at 140 km/h on an asphalt road in the rain. This noise data was recorded with a ½″ B&K condenser microphone and stored on a digital audio tape (DAT).
- Military vehicle noise (Leopard): The noise samples were recorded using a ½″ B&K condenser and stored on DAT. The microphone was mounted on a Leopard-1 traveling at 70 km/h.

- Construction/cutting and welding noises: The noise sample included noises coming from plate cutting machinery and welding equipment. The samples were collected using a ½″ B&K condenser microphone.

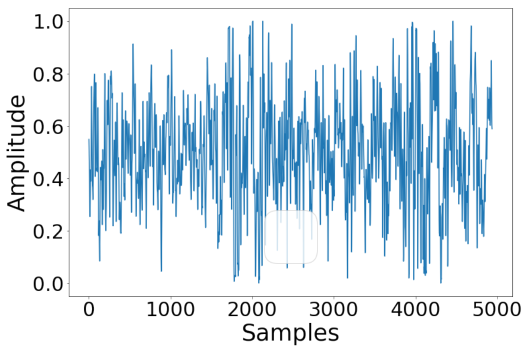
Table 4-1 presents the nine training and nine testing noise datasets generated from the original four noise data sets, and Figure 4-1 shows the time history for each data set. For the training data sets, Flight1 (10,000), Flight2 (10,000), Aero1 (5,000), Aero2 (5,000), Aero3 (5,000), Aero4 (5,000), Volvo (2,000), Construction/Cutting (2,000) and Leopard (2,000) samples were selected. The data sets were normalized within the range of 0 and 1 during the training. Flight1, Flight2, Aero1, Aero2, Aero3 and Aero4 were sampled at a sampling rate of 44.1 kHz, whereas Volvo, Construction/Cutting and Leopard were sampled at a rate of 19.98 kHz with 16-bit precision.

Table 4-1: Established data sets

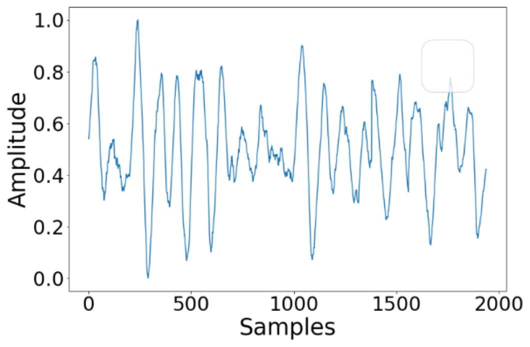
Data	Training samples	Testing samples	Samples per input
Flight1	10,000	4,937	20
Flight2	10,000	4,937	20
Leopard	2,000	1,937	20
Volvo	2,000	1,937	20
Aero1	5,000	4,957	20
Aero2	5,000	4,957	20
Aero3	5,000	4,957	20
Aero4	5,000	4,957	20
Construction	2,000	1,937	20



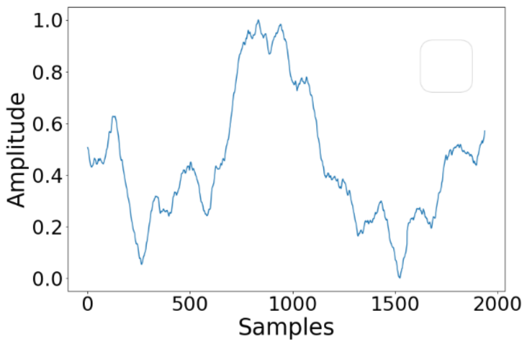
(a) Flight1



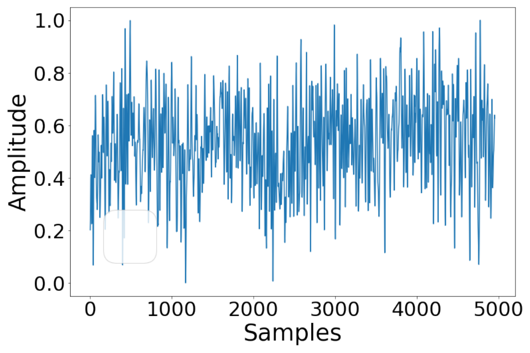
(b) Flight2



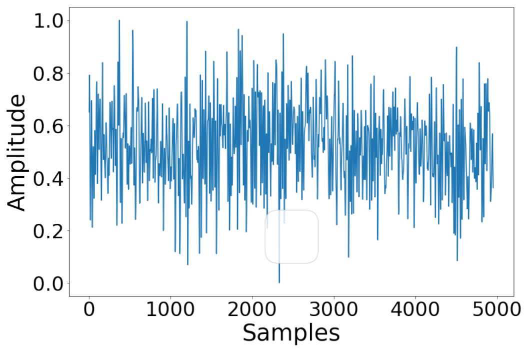
(c) Leopard (SPIB, 2020)



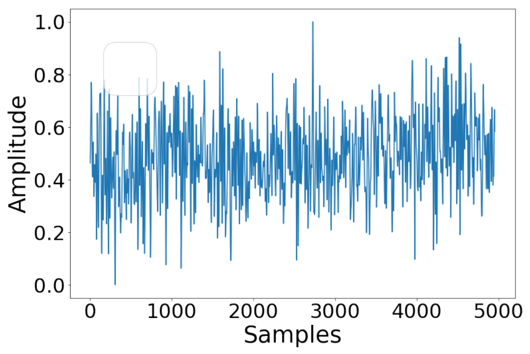
(d) Volvo (SPIB, 2020)

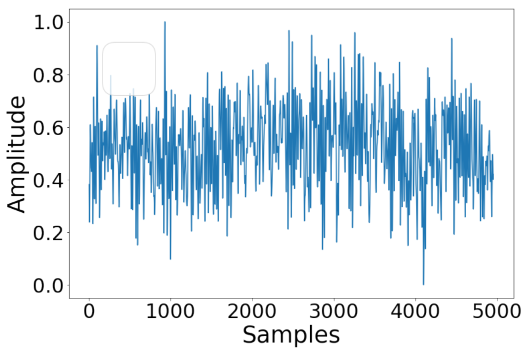


(e) Aero1

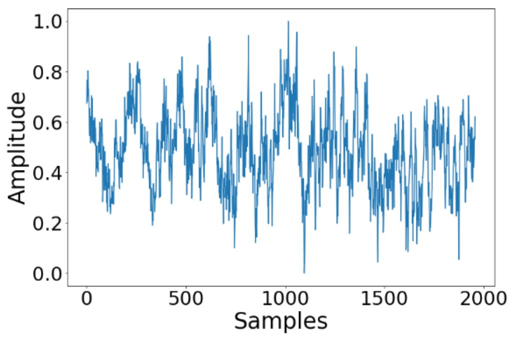


(f) Aero2





(g) Aero3 (h) Aero4



(i) Construction/ Cutting (SPIB, 2020)

Figure 4-1: Noise training data sets

4.2 Tuning model hyperparameters

To configure the optimum hyperparameters for the ACRN model, a number of possible models were tested. The effects of the learning rate, dropout, optimizers and learning rate decay on the model losses were studied, and the hyperparameters were selected for the final network based on the studies described below.

4.2.1 Effect of learning rate

A learning rate highly affects the convergence of the algorithms based on back propagation (Wilson and Martinez, 2001). To investigate the effect of learning rate on the model, four learning rates, 0.001, 0.01, 0.05 and 0.1, were selected. Because of the complex nature of noise, a learning rate of 0.001 was initially selected, and the effects of the other parameters were studied using this learning rate. Figure 4-2 shows the model losses for the different learning rates. Model loss 1 corresponds to a learning rate of 0.001, and model loss 2, model loss 3 and model loss 4 correspond to 0.01, 0.1 and 0.05, respectively.

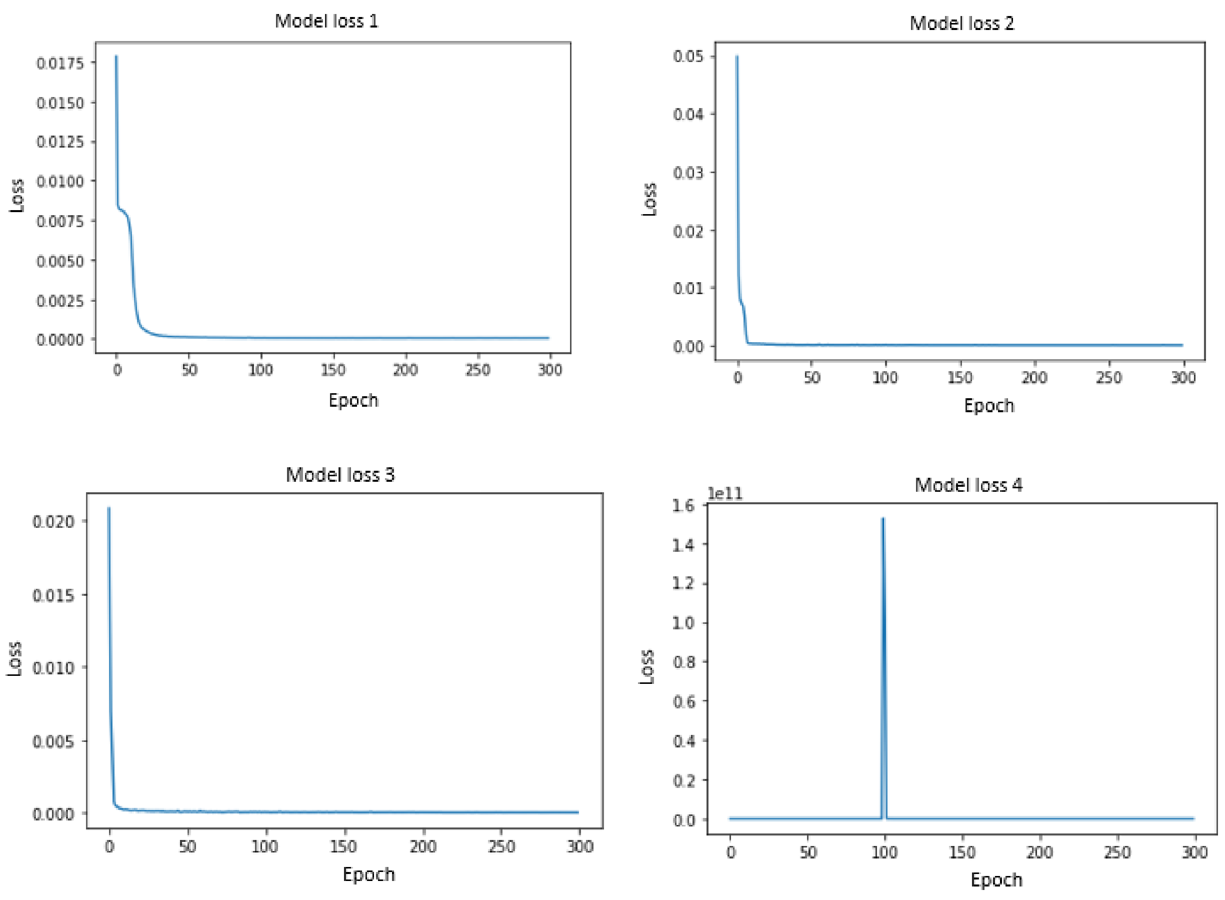


Figure 4-2: Model training losses with different learning rate

Figure 4-3 shows the effect of learning rate on the training loss. The training data sets were used to calculate the training loss. The highest training loss was observed when a learning rate of 0.05 was selected. The training loss corresponding to this learning rate was 0.026200. However, the other learning rates provided remarkably less losses, as shown in Table 4-2.

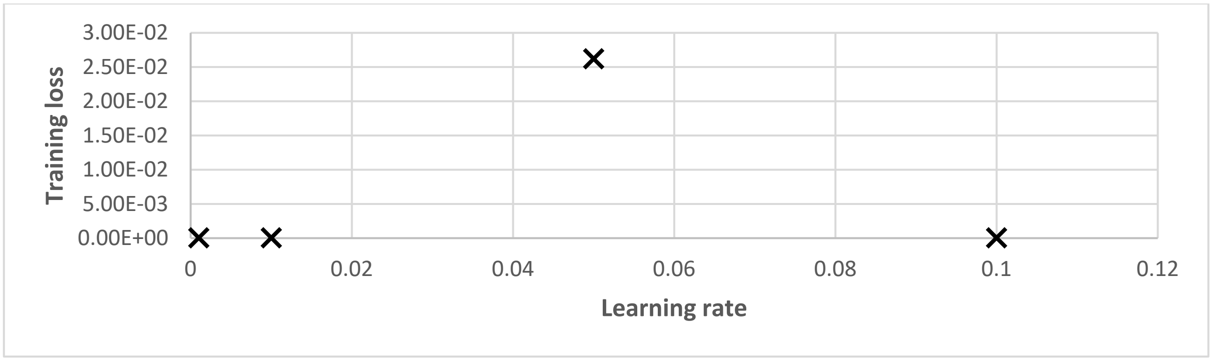


Figure 4-3: Effect of learning rate on training loss

Table 4-2 : Learning rates with training loss

Learning rate	**Training loss**
0.001	0.000053
0.010	0.000058
0.100	0.000052
0.050	0.026200

4.2.2 Effect of dropout

Dropouts were introduced in the model to avoid overfitting (Srivastava and Hinton, 2014). Three different dropout rates were selected: 0.1, 0.5 and 0.9. This meant that there were 1 in 10, 1 in 2 and 1 in 1.11 chances that a node would be excluded from each weight update cycle. Figure 4-4 shows the effect of dropout rate on training loss.

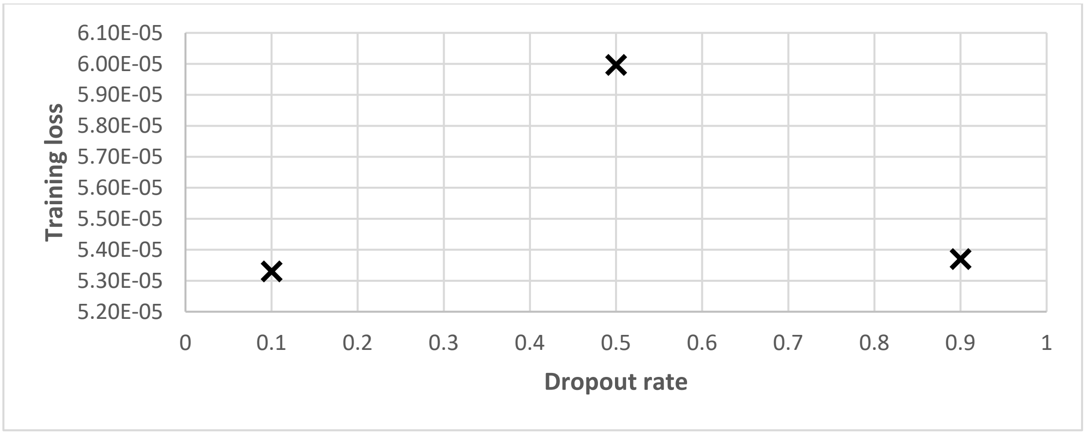


Figure 4-4: Effect of dropout on training loss

As seen in Table 4-3, a dropout rate of 0.1 corresponded to less training loss compared to dropout rates of 0.5 and 0.9.

Table 4-3: Dropout with training loss

Dropout	Training loss
0.10	0.000053
0.50	0.000060
0.90	0.000054

4.2.3 Effect of learning rate decay

To investigate the effect of learning rate on the model, four learning rates of 0.00001, 0.0005, 0.001 and 0.1 were selected. You and Long (2019) suggested that learning rate decay can help a network learn more complex data. Figure 4-5 shows the plot of training loss versus learning rate decay.

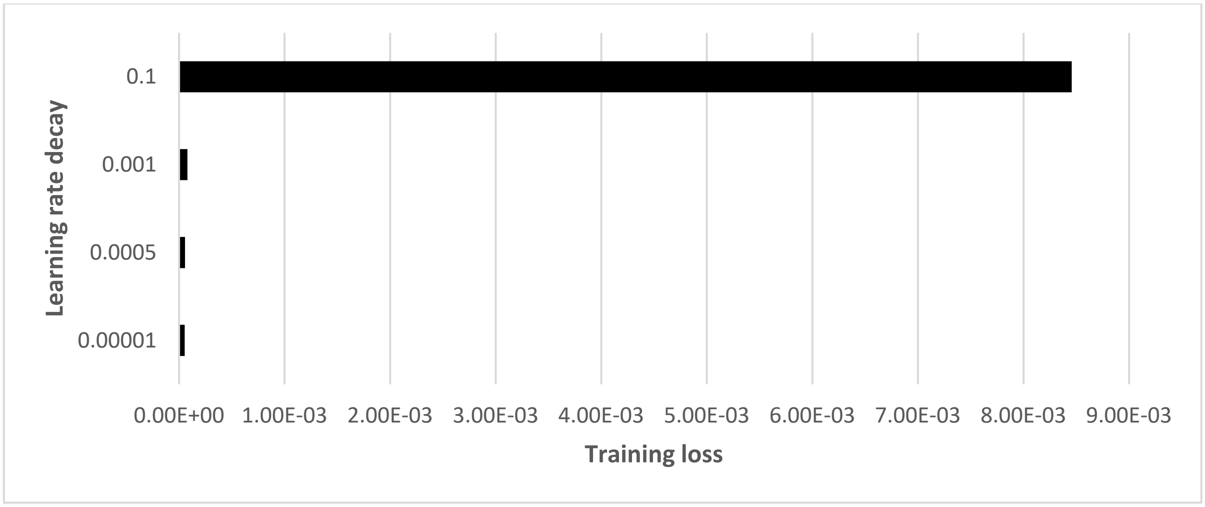


Figure 4-5: Effect of learning rate decay

Table 4-4 shows that the lower the decay rate, the lower the lower the training loss. The best result was yielded by a learning rate decay of 0.00001, corresponding to a training loss of 0.000053.

Table 4-4: Learning rate decay with training loss

Learning rate decay	**Training loss**
0.000010	0.000053
0.000500	0.000056
0.001000	0.000080
0.100000	0.008457

4.2.4 Effect of optimizer

To select the best optimizer for the network, four network optimization methods were studied. Figure 4-6 shows the results of Adagrad (Duchi and Hazan, 2011), RMSprop (Tieleman and

Hinton, 2012), SGD (Qian et al., 1999) and ADAM optimizers in terms of training loss. ADAM outperformed the other three optimizers; it was the fastest and required much less fine tuning.

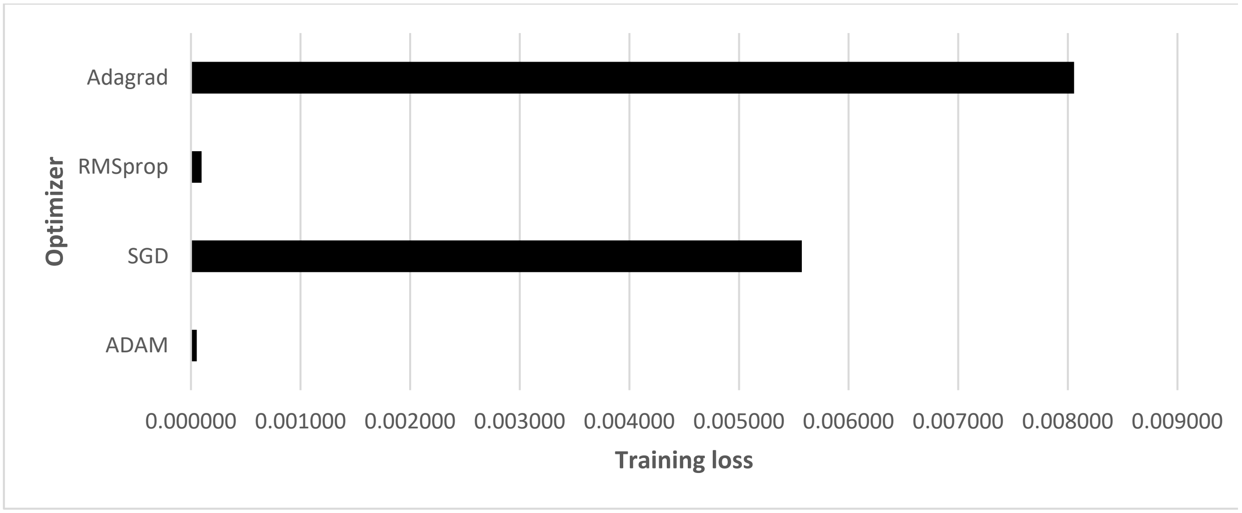


Figure 4-6: Performances of optimizer on training

Table 4-5 shows the training loss corresponding to the optimizer type. The best result was yielded by ADAM, with a training loss of 0.000053.

Table 4-5: Optimizers with training loss

Optimizer	**Training loss**
ADAM	0.000053
SGD	0.005573
RMSprop	0.000096
Adagrad	0.008057

4.3 Training of network and model loss for flight1 dataset

The training was performed using 10,000 samples and 100 epochs. The training data set was divided into 9,781 training samples and 219 validation samples. A batch size of 100 was selected to speed up the process. The training was performed on a GPU (TESLA K80, NVIDIA) with 12 GB available RAM. The time taken for each epoch was 3 seconds. The training and validation losses both flattened after 95 epochs. However, a slightly declining loss was still observed at 100 epochs. The training mean square error at epoch 100 was 7.4460e-05, with an accuracy of 0.0019, as shown in Figure 4-7. Moreover, the mean square error for the validation data set at the same epoch was 5.4184e-05. The other data sets were similarly used for training and prediction.

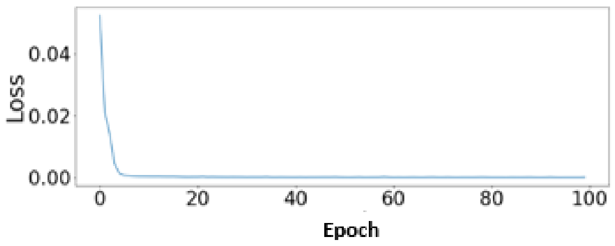
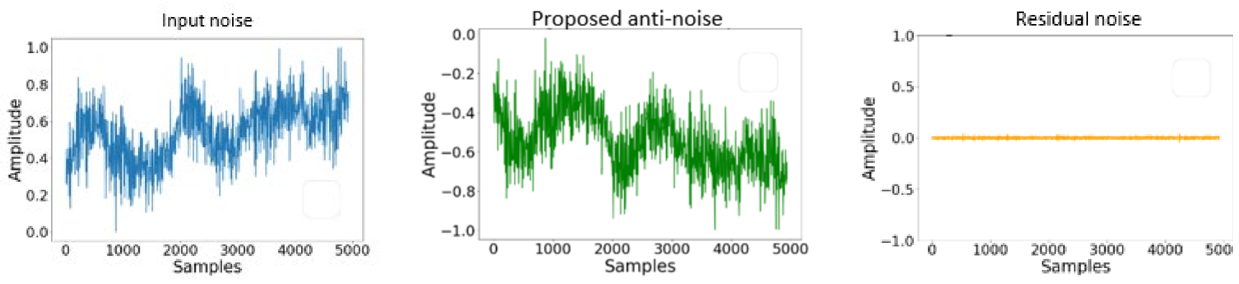


Figure 4-7: Training loss curve

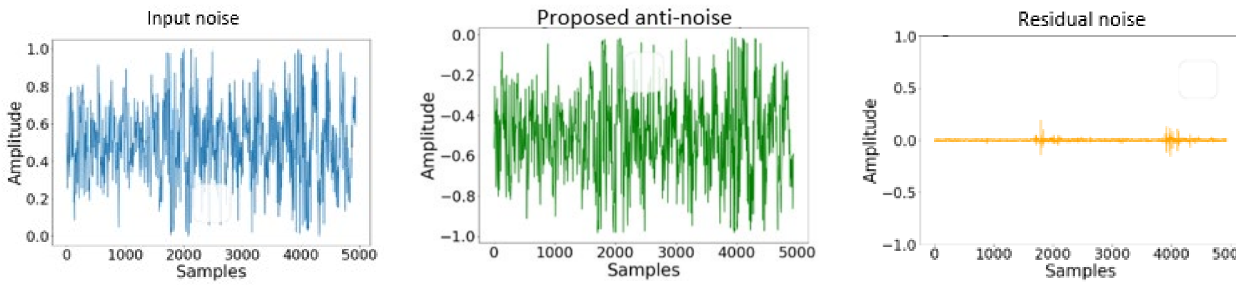
4.4 Testing results

Testing samples Flight1(4,937), Flight2(4,937), Leopard(1,937), Volvo(1,937), Aero1 (4,957), Aero2 (4,957), Aero3 (4,957), Aero4 (4,957), and Construction/Cutting (1,937) were selected to test the model. This testing was also performed on a GPU (K80) with 12 GB available RAM. These samples were never used by the model during training. Figure 4-8 shows the outputs of the proposed method with the inputs and residual noise.

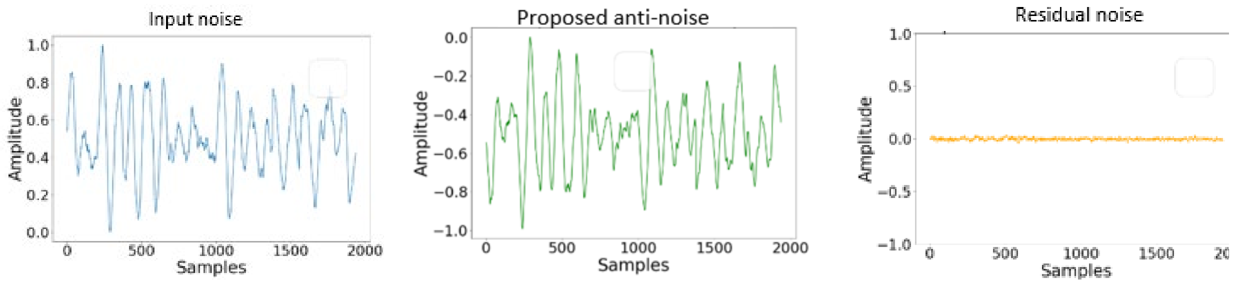
Input noise **Anti-noise generated** **Residual noise**



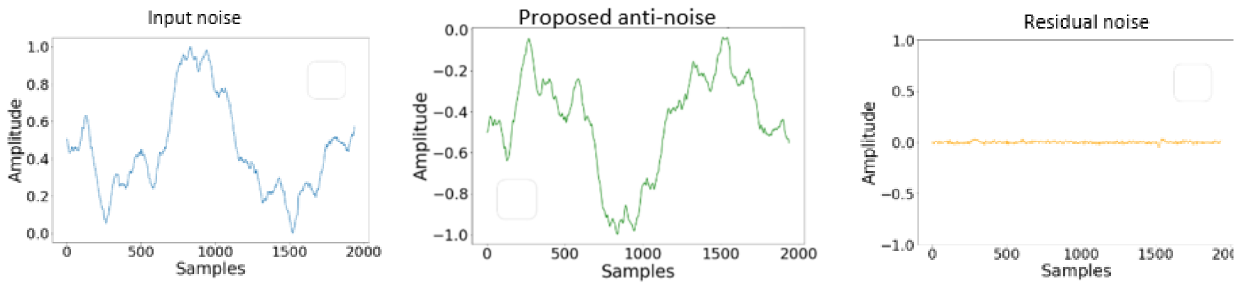
(a) Flight1



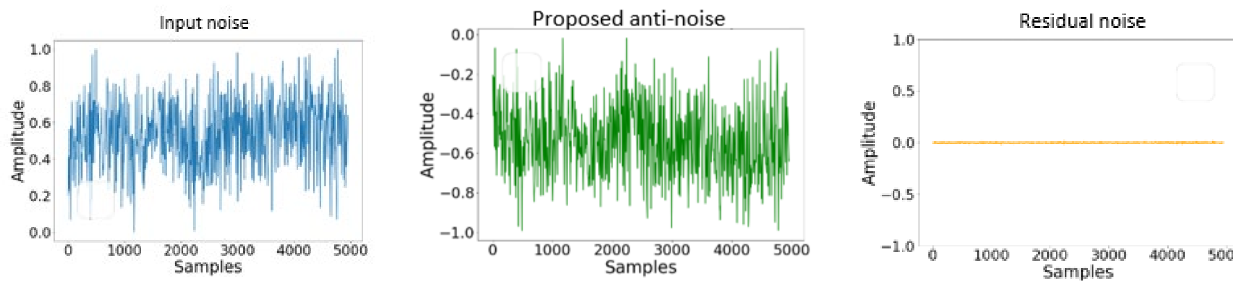
(b) Flight2



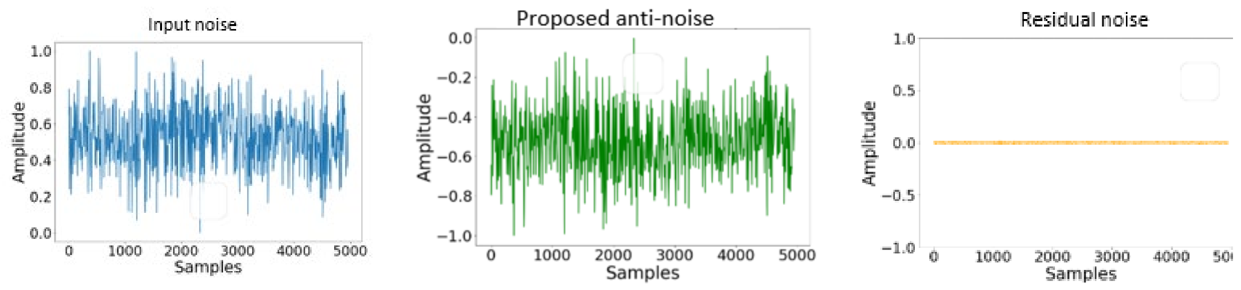
(c) Leopard



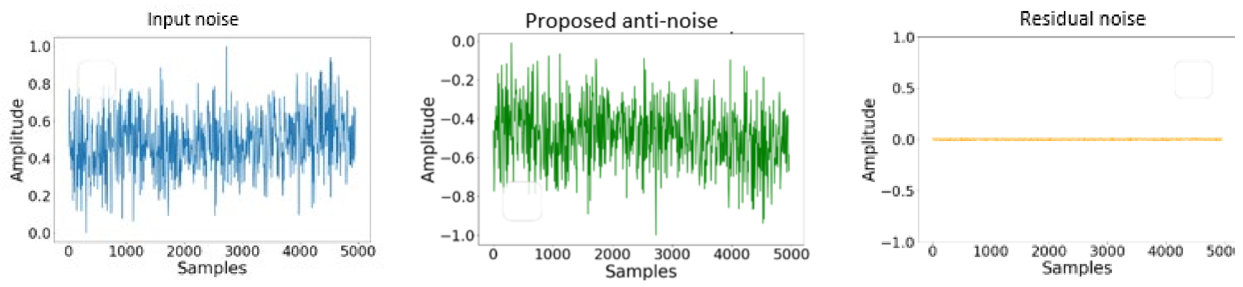
(d) Volvo



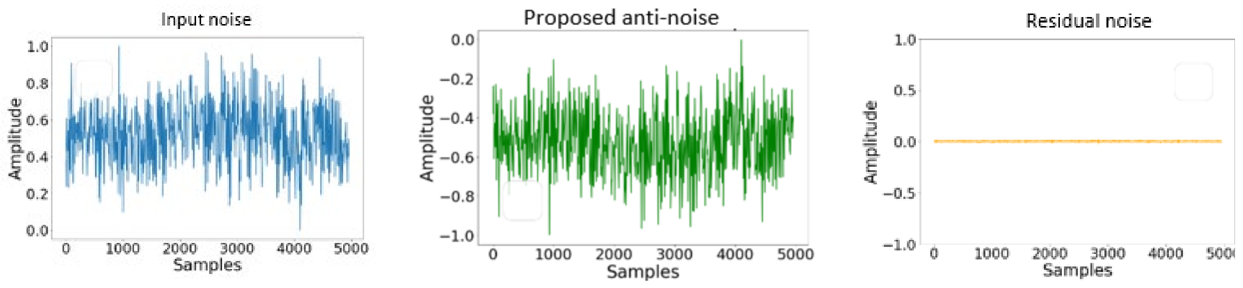
(e) Aero1



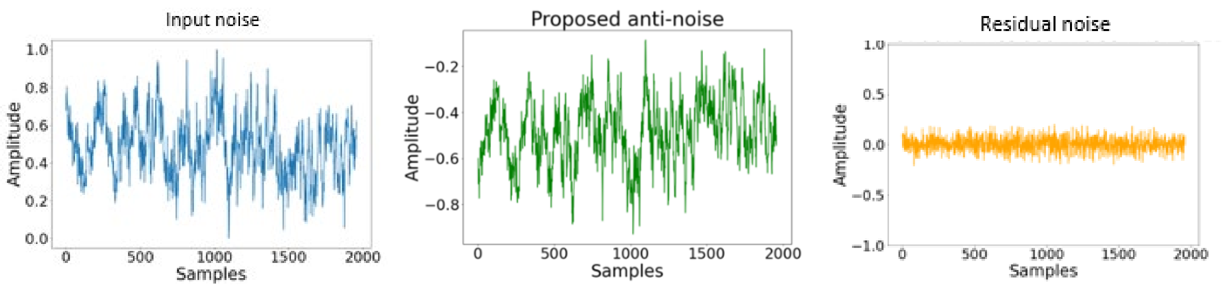
(f) Aero2



(g) Aero3



(h) Aero4



(i) Construction

Figure 4-8: Proposed model outputs and residual noises

Chapter 5. Comparative Studies

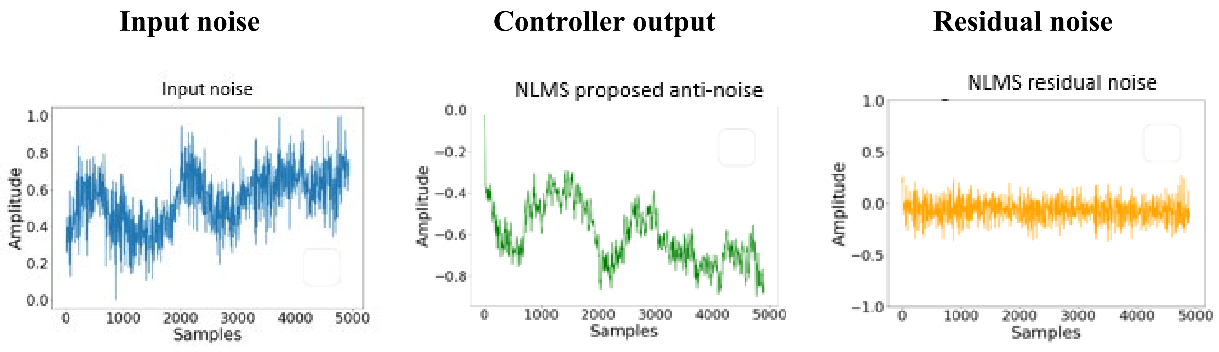
In order to conduct comparative studies to investigate the performances of the proposed ACRN model, four different algorithms were selected: normalized least mean square (NLMS), multi-layered perceptron (MLP), long short-term memory (LSTM) and convolution neural network (CNN). These algorithms, proposed by different researchers for ANCs, were used as benchmarks for the comparisons carried out in this study.

5.1 Normalized least mean square algorithm

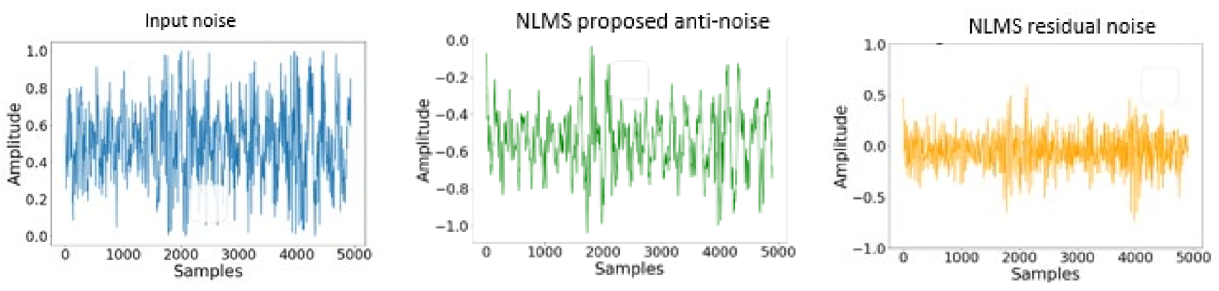
To measure the performance of the NLMS algorithm (Dixit and Nagaria, 2017), a convergence controlling factor of 9.094947017729282e-13 was selected, and the number of taps selected for the Finite impulse response (FIR) was 20. Weight updating was achieved using Equation (5.1) as follows:

$$w(n+1) = w(n) + \mu(n).\, x_f(n).\, e(n) \qquad (5.1)$$

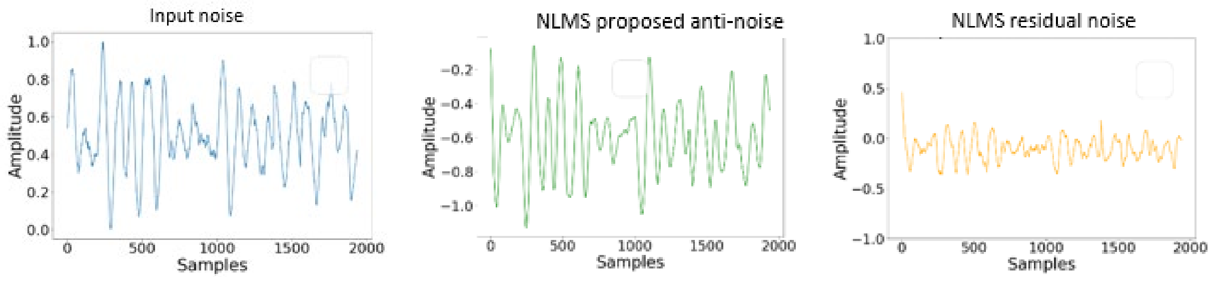
where $\mu(n)$ is the learning rate; $w(n)$ is the filter weight; $x_f(n)$ is the input sample and $e(n)$ is the error, which differs between the prediction and target value. Figure 5-1 shows the output of the NLMS algorithm.



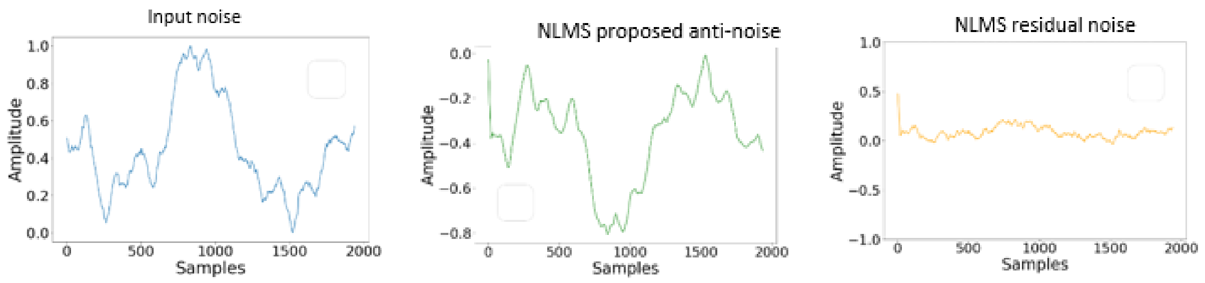
(a) Flight1



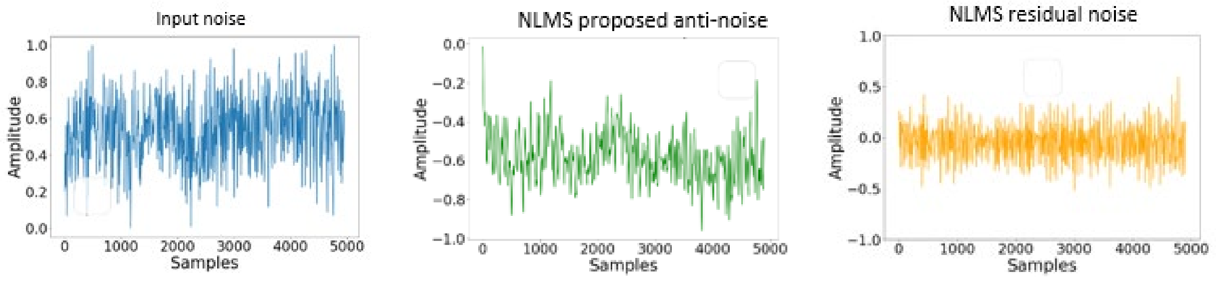
(b) Flight2



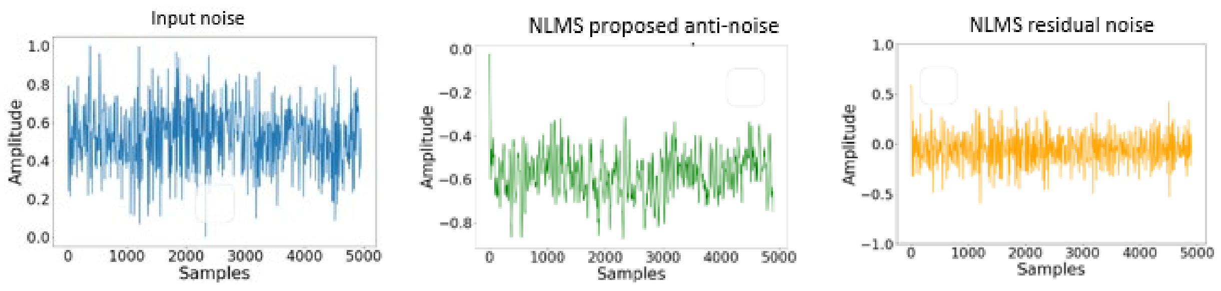
(c) Leopard



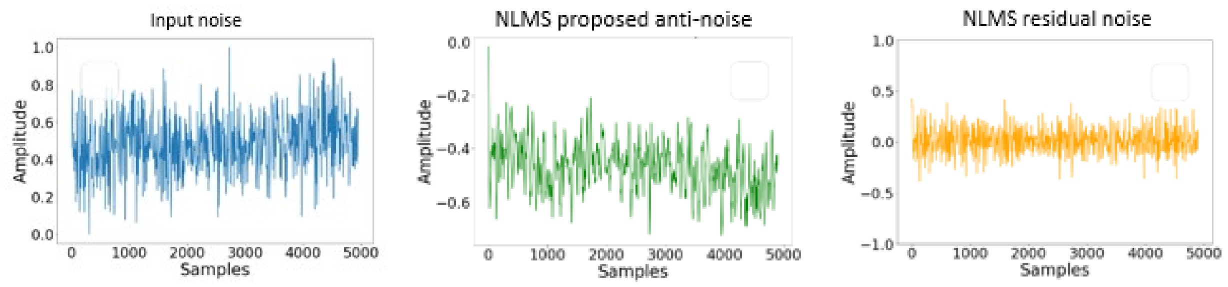
(d) Volvo



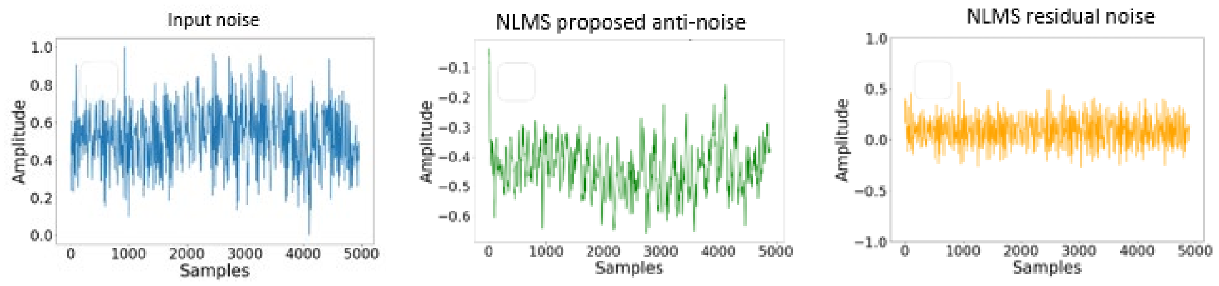
(e) Aero1



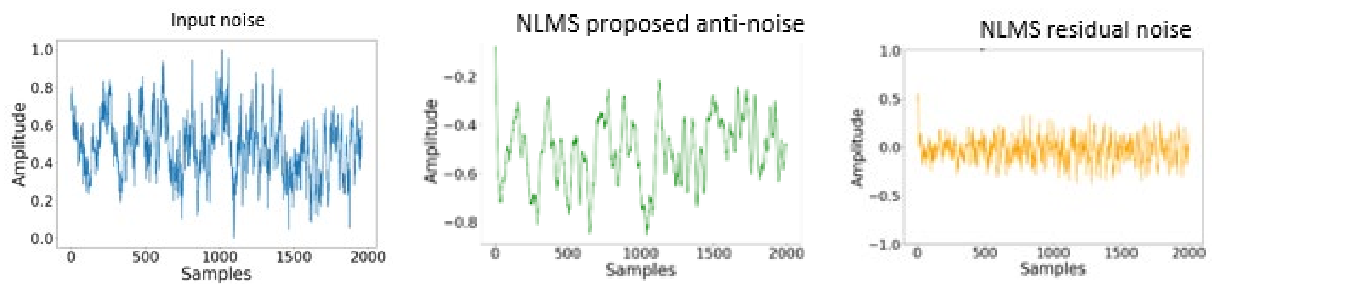
(f) Aero2



(g) Aero3



(h) Aero4



(i) Construction

Figure 5-1: Normalized least mean square (NLMS) algorithm outputs.

5.2 Multi-layered perceptron

The MLP was used to predict the nine noise datasets. The MLP architecture (Park and Patterson, 2019) is composed of 20 input unit layers followed by 20 hidden units and one output layer, as shown in Figure 5-2. Linear activation and tanh activation functions were used as the activation functions for neurons. The incoming training noise was normalized within the range of 0 and 1.

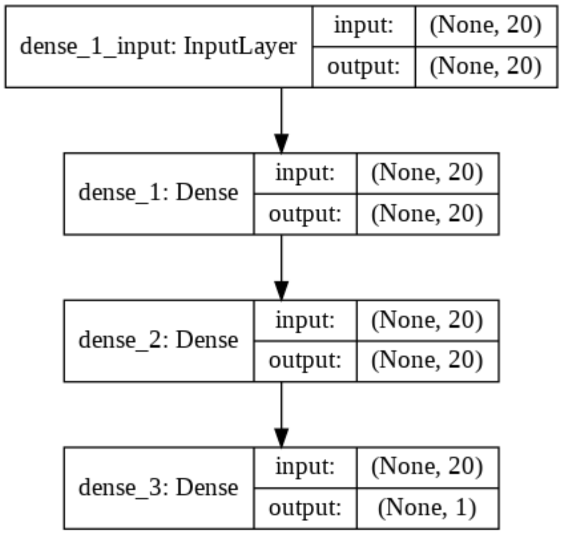
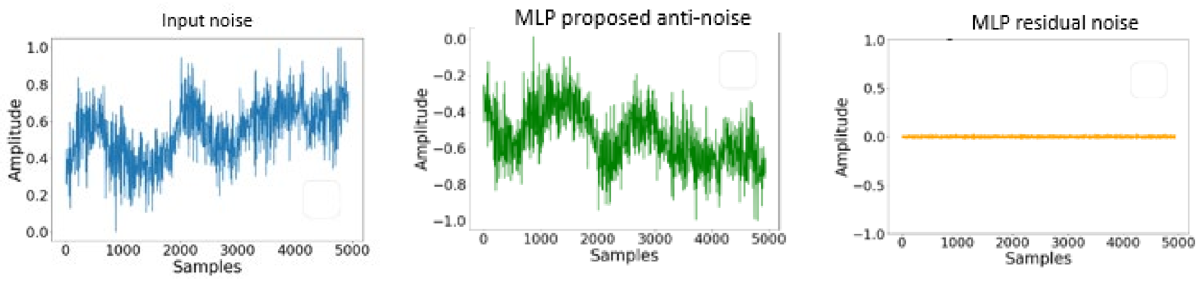


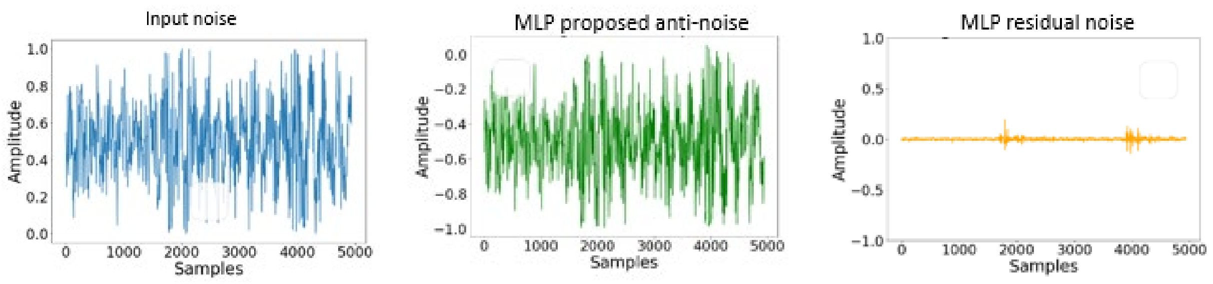
Figure 5-2: Multi-layered perceptron (MLP) network.

The training was performed using 200 epochs and a batch size of 100. The training and testing were performed on a GPU (TESLA K80, NVIDIA) with 12 GB available RAM. The input, output and residual noise are presented in Figure 5-3.

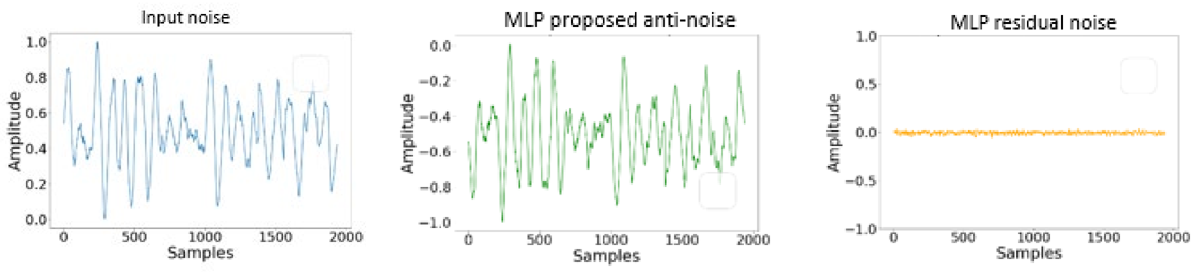
Input noise **Controller output** **Residual noise**



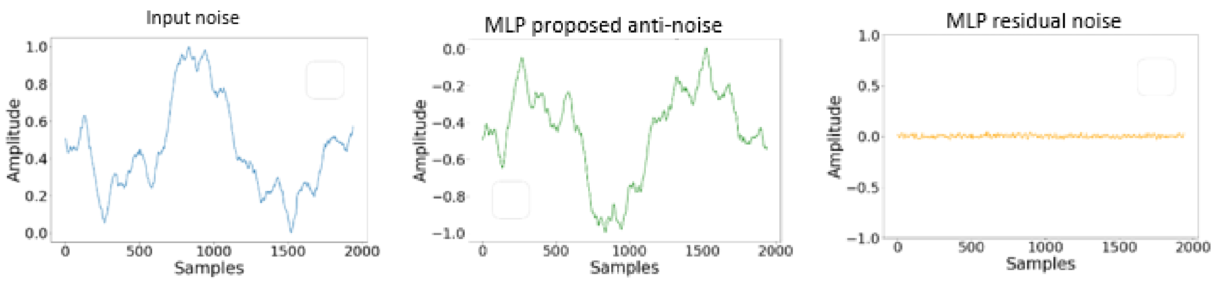
(a) Flight1



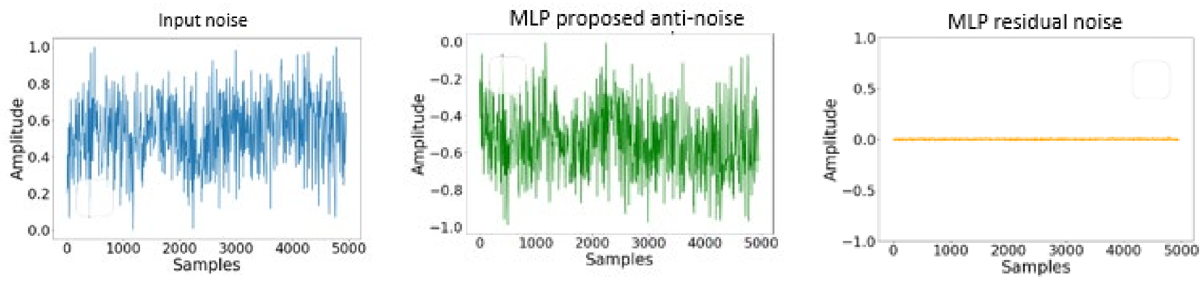
(b) Flight2



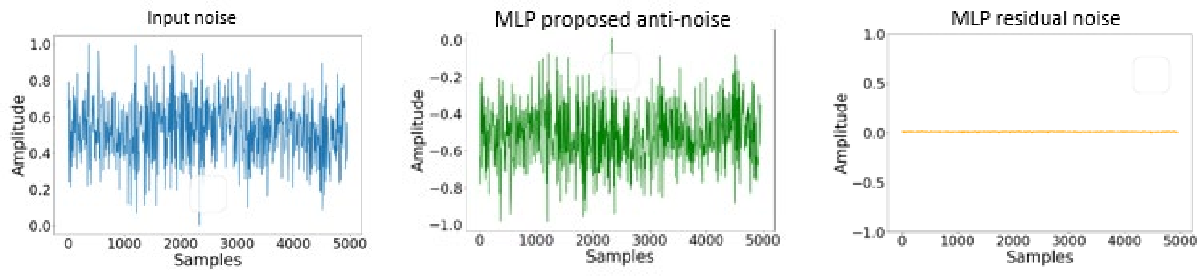
(c) Leopard



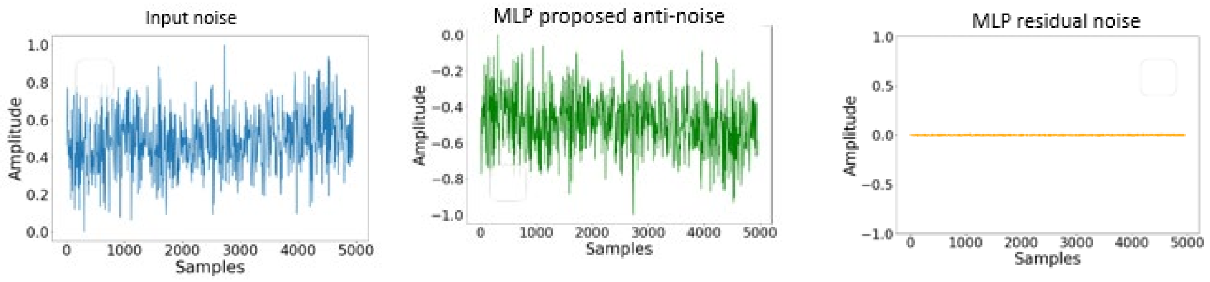
(d) Volvo



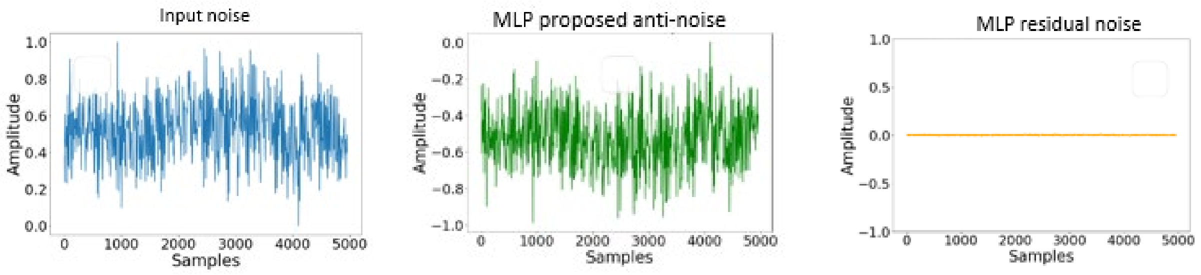
(e) Aero1



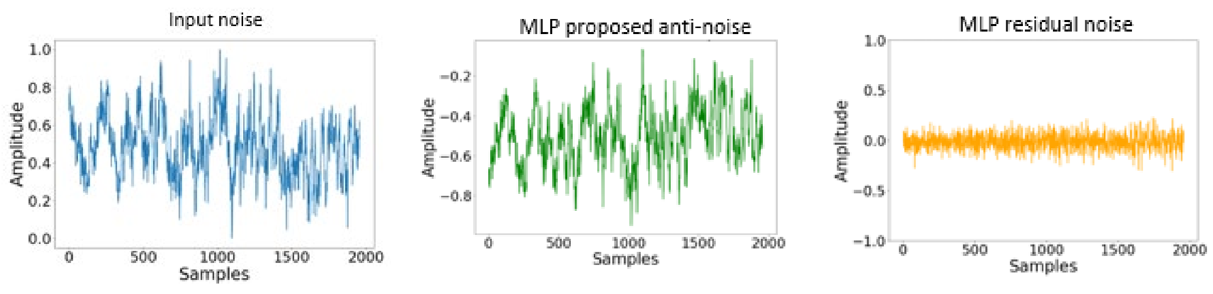
(f) Aero2



(g) Aero3



(h) Aero4



(i) Construction

Figure 5-3: Outputs of MLP network.

5.3 Long short-term memory

An LSTM network with 20 inputs, 20 hidden states and one output (Park and Patterson, 2019) was selected, as shown in Figure 5-4.

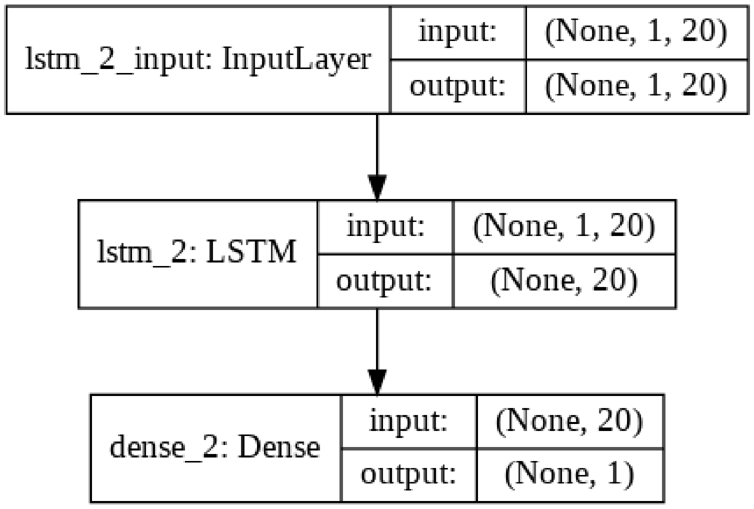
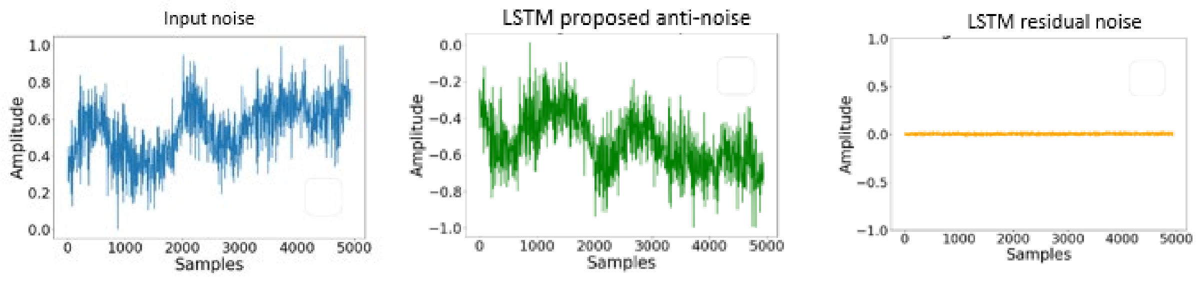


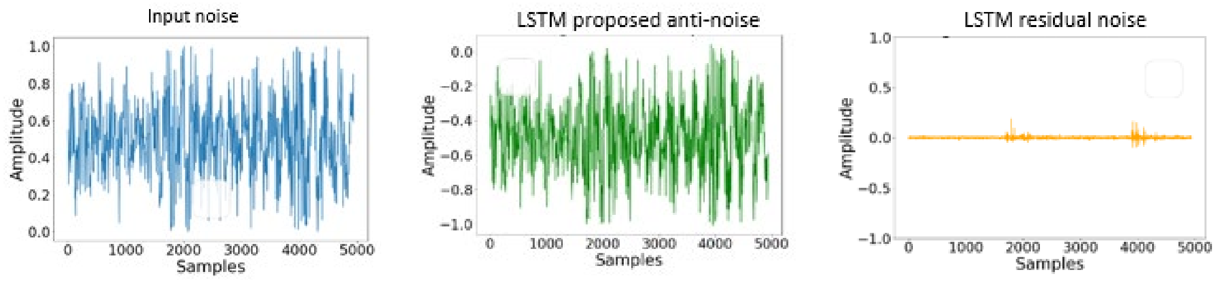
Figure 5-4: LSTM network.

Tanh activation and linear activation functions were used for the hidden layer and output layer, respectively. A similar batch size of 100 and 200 epochs were selected for the data set, as explained in chapter 4. Data normalization was performed on the data set. A GPU (TESLA K80, NVIDIA) with 12 GB available RAM was used to train and test the network. Figure 5-5 presents the input noise, controller output and residual noise.

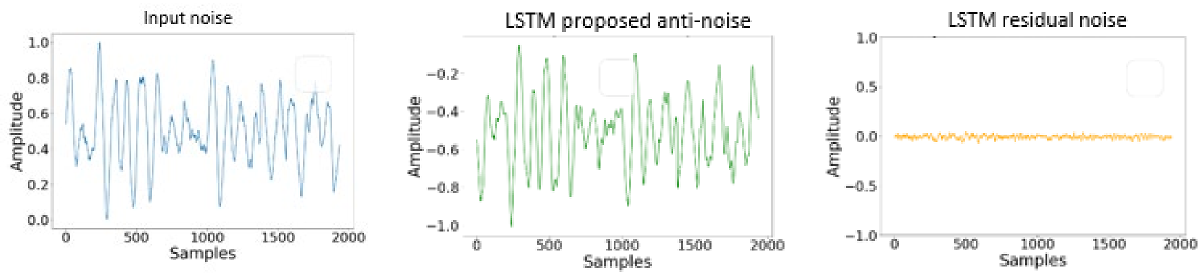
Input noise Controller output Residual noise



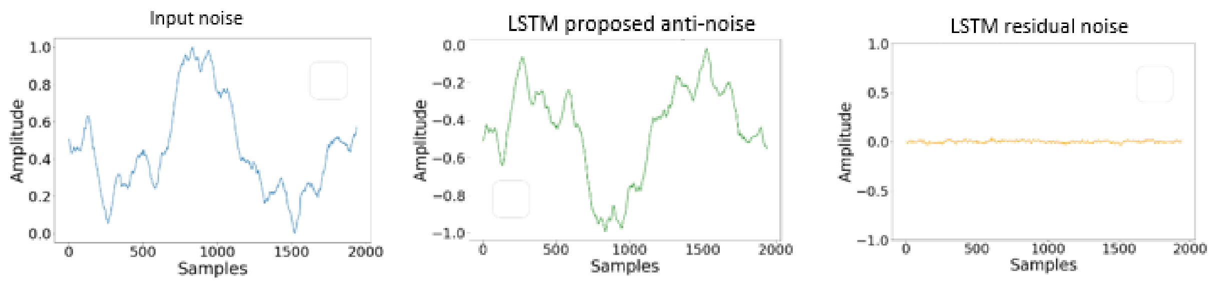
(a) Flight1



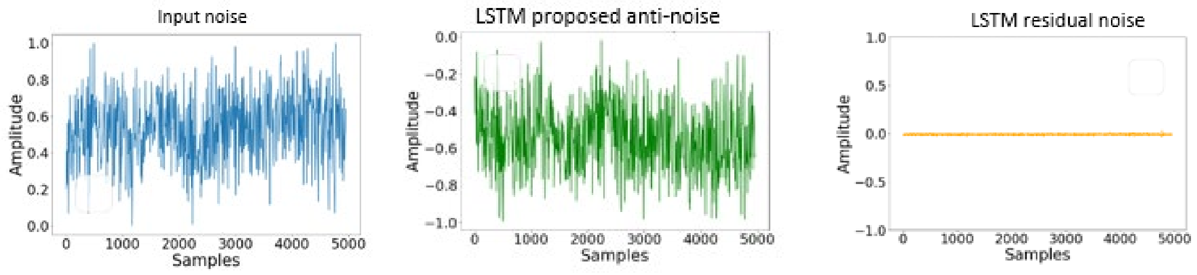
(b) Flight2



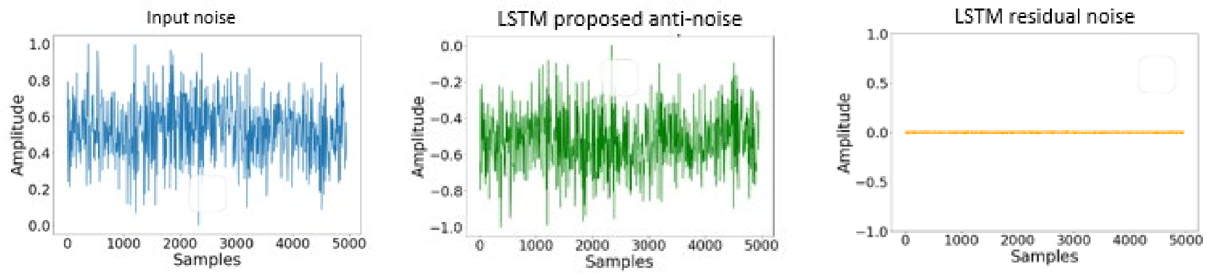
(c) Leopard



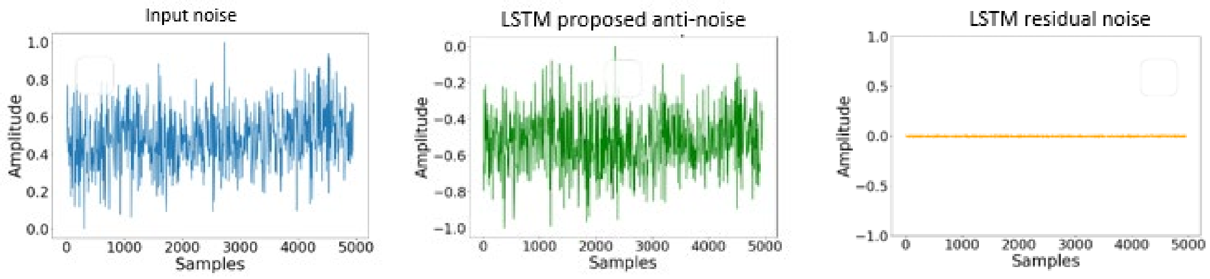
(d) Volvo



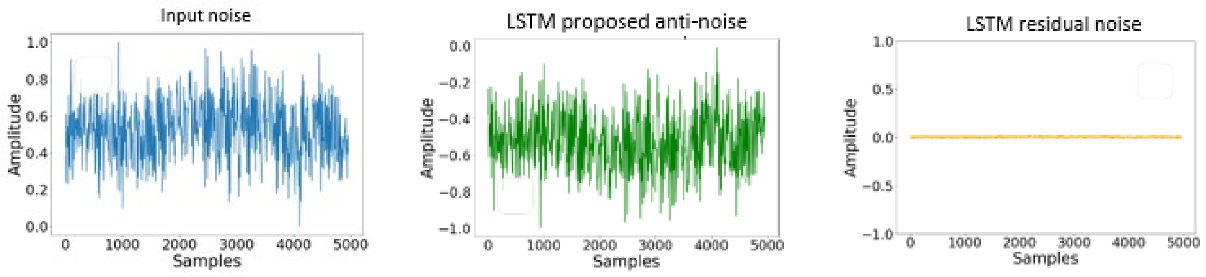
(e) Aero1



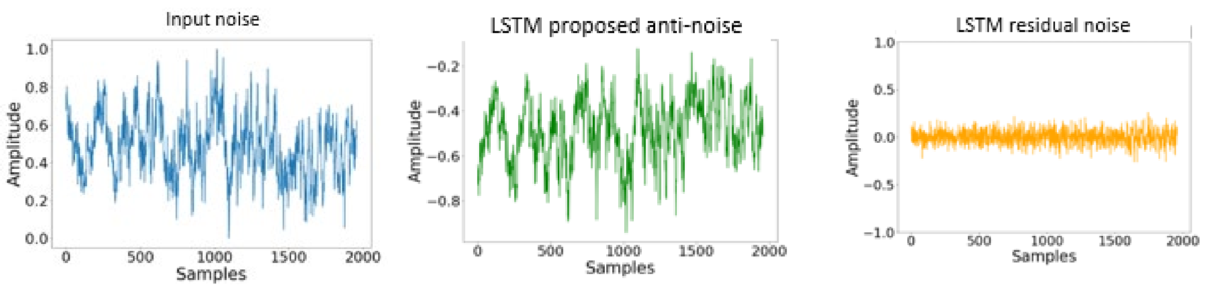
(f) Aero2



(g) Aero3



(h) Aero4



(i) Construction

Figure 5-5: Outputs of LSTM algorithm.

5.4 Convolution neural network

A CNN with one convolution layer, four filters and a kernel size of 1 × 5 was used. The tanh activation function was used for the convolution layer, and zero padding was used to keep the output size the same as the input size. A max pooling layer was applied after 1-D convolution, and the selected pooling size for this layer was two. The output from the max pooling layer was flattened and passed to a dense layer consisting of forty neurons, and this layer output one prediction as anti-noise sample. Figure 5-6 shows the CNN architecture proposed by Park and Patterson (2019).

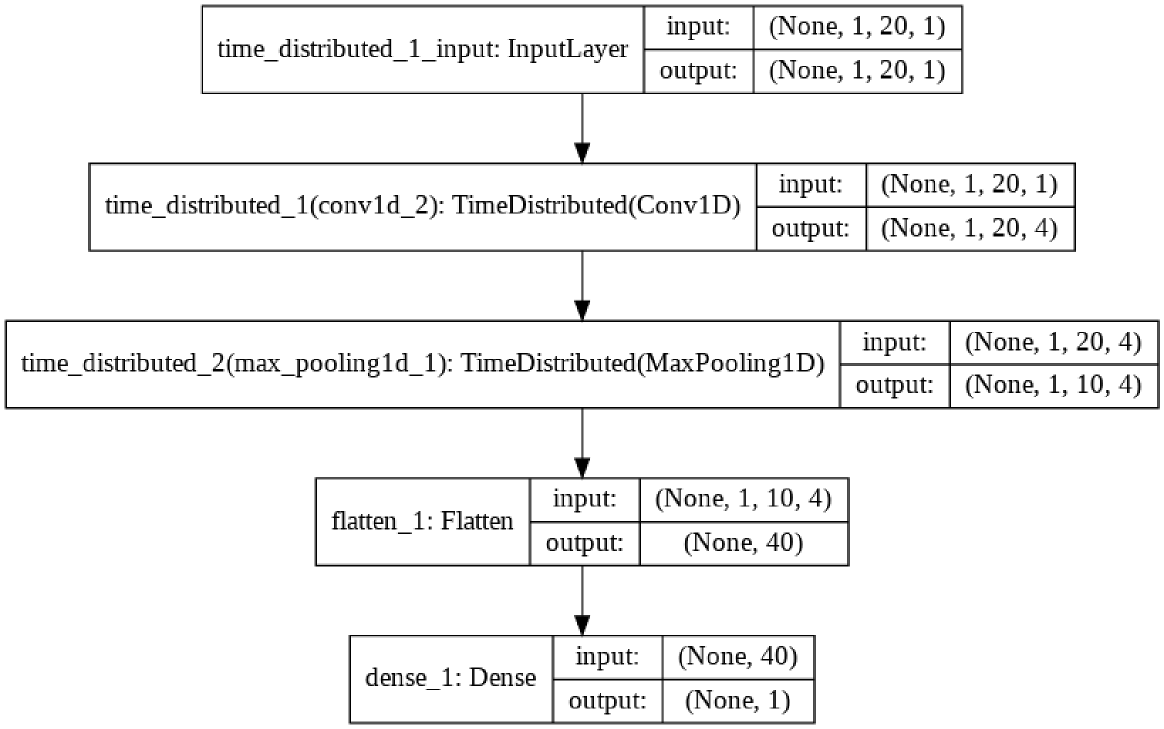
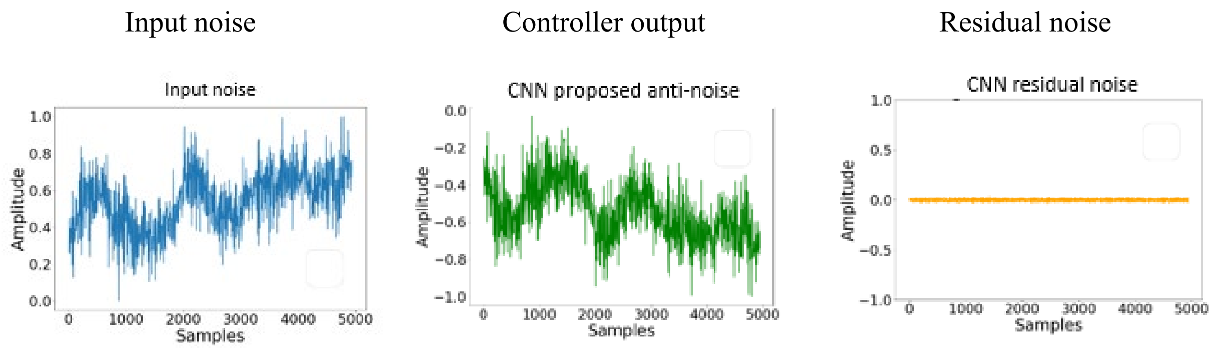
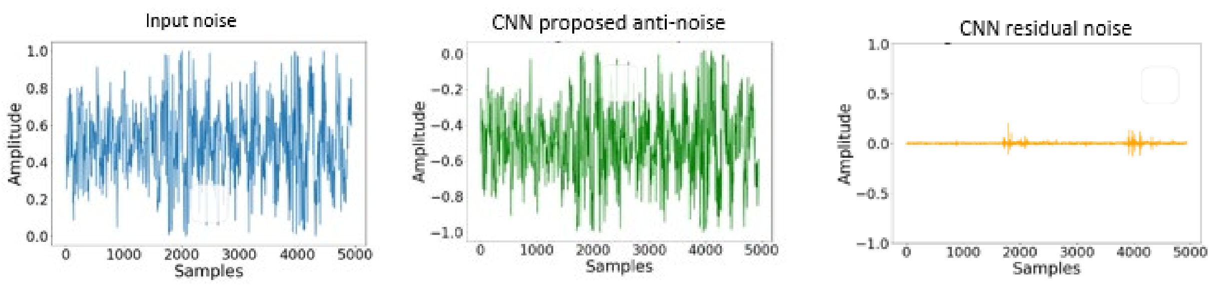


Figure 5-6: Architecture of the CNN

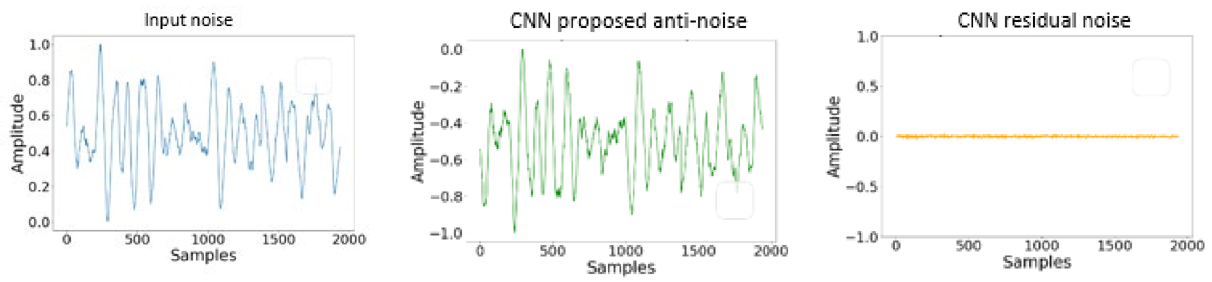
To train the network, a GPU (TESLA K80, NVIDIA) with 12 GB available RAM was used. A batch size of 100 was selected for training the network with the ADAM optimizer. The output, input noise and residual noise are presented in Figure 5-7.



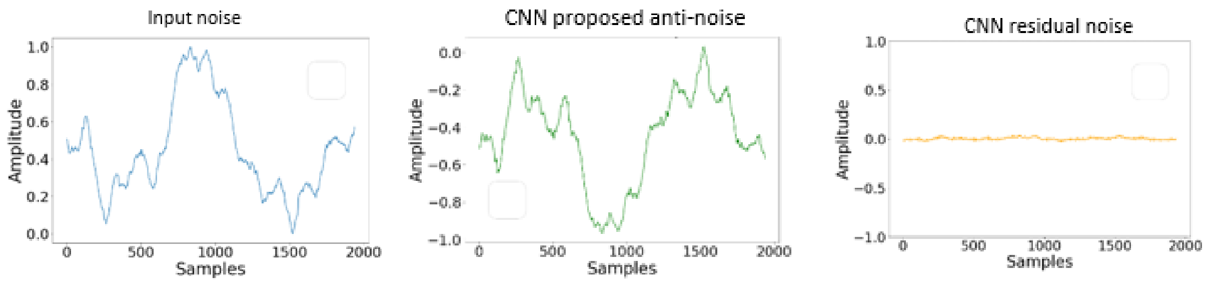
(a) Flight1



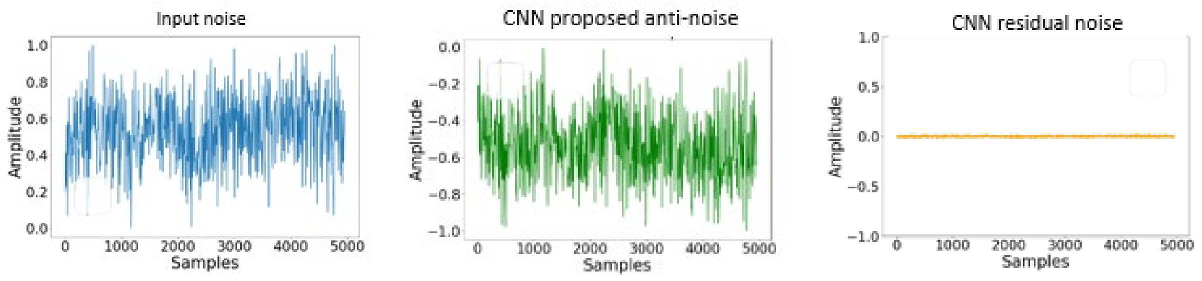
(b) Flight2



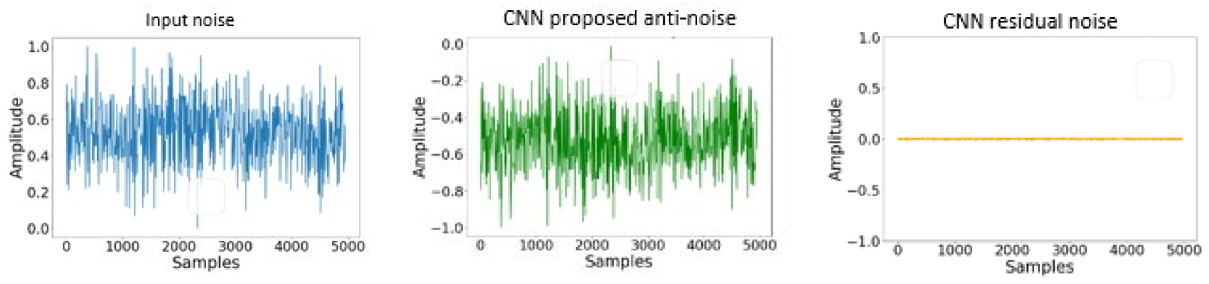
(c) Leopard



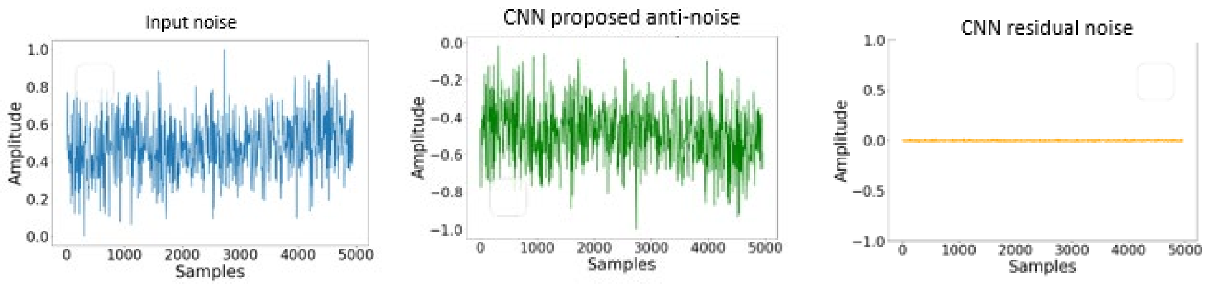
(d) Volvo



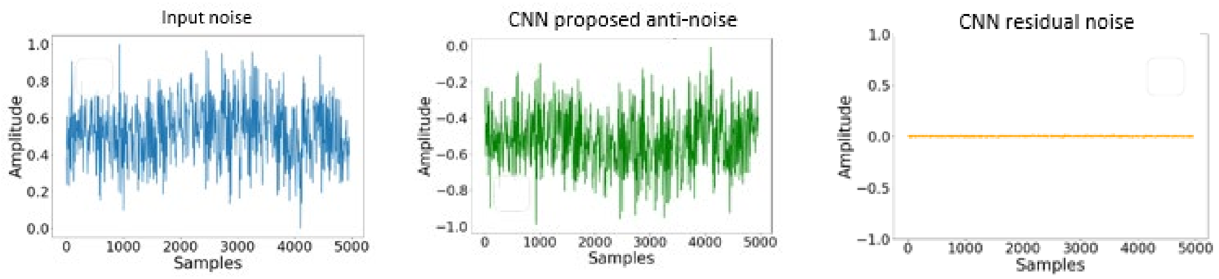
(e) Aero1



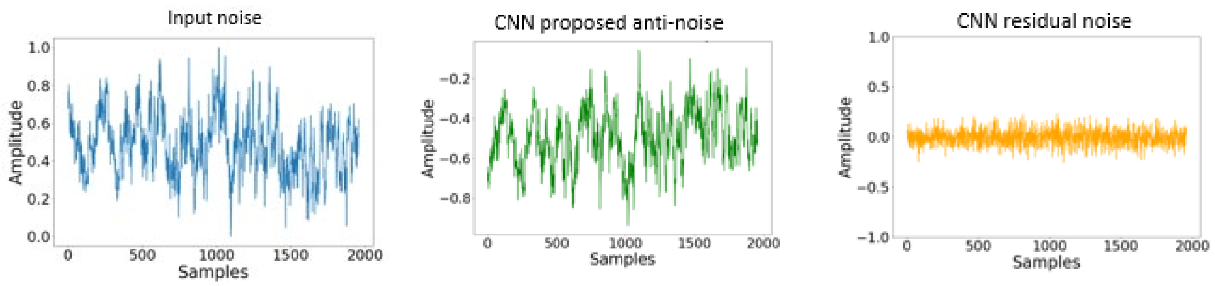
(f) Aero2



(g) Aero3



(h) Aero4



(i) Construction

Figure 5-7: Outputs of CNN algorithm.

5.5 Evaluation metrics

To compare the performances of the four different ANCs, three evaluation metrics were used: noise attenuation, root mean square error (RMSE) and number of trainable parameters.

5.5.1. Noise attenuation

Noise attenuation is a representation of the amount of noise cancellation (i.e., the higher the noise attenuation value, the higher the amount of noise cancellation). The noise attenuation was calculated using Equation (5.2), as follows:

$$Noise\ attenuation\ (dB) = 10. log_{10}. \frac{E_N}{E_R} \qquad (5.2)$$

where E_N is the energy of noise signal and E_R is the energy of the residual noise signal. The calculations for the proposed methods, LSTM, MLP and NLMS are given in Table 5-1. As shown in the Table 5-1, the proposed method resulted in the best performance.

Table 5-1: Noise attenuation levels (dB)

Data/Method	**ACRN**	**LSTM**	**MLP**	**NLMS**	**CNN**
Flight1	*37.83*	34.81	36.63	13.69	35.31
Flight2	*32.65*	32.29	31.63	10.24	32.54
Leopard	*36.35*	28.86	31.15	10.83	35.99
Volvo	*33.76*	32.76	31.11	13.89	31.68
Construction/ Cutting	*17.89*	16.90	16.28	12.38	16.58
Aero1	*40.70*	35.18	40.67	10.62	37.60
Aero2	*43.76*	39.98	33.73	11.03	40.86
Aero3	*42.08*	38.50	40.28	12.44	39.83

Aero4	*43.96*	39.24	39.81	11.07	40.38
Average	*38.86*	35.20	35.62	11.72	36.77

5.5.2 Root mean square error

Neill and Hashemi (2018) suggested the usefulness of the RMSE to compare the accuracies of different models. The performance of the proposed method was compared to the other methods based on RMSE metrics calculated using Equation (5.3). The RMSE is the square root of the mean of the square of the difference between the target value (t_i) and the observed value (o_i), as follows.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(t_i - o_i)^2} \qquad (5.3)$$

where n is number of samples. Table 5-2 shows the RMSEs for the different methods when applied to the different noise data sets. The proposed ACRN model resulted in the best performance in terms of RMSE.

Table 5-2 : RMSE for all methods

	Leopard	**Volvo**	**Flight1**	**Flight2**	**Aero1**	**Aero2**	**Aero3**	**Aero4**	**Const./ Cutting**	**Average**
ACRN	0.00810	0.01050	0.00839	0.01073	0.00517	0.00352	0.00398	0.00345	0.06518	0.01322
LSTM	0.01919	0.01178	0.01126	0.01118	0.00977	0.00545	0.00601	0.00594	0.07304	0.01707
MLP	0.01475	0.01425	0.00938	0.01228	0.00519	0.01120	0.00490	0.00556	0.07841	0.01732
NLMS	0.15304	0.10354	0.11586	0.16524	0.16530	0.16530	0.12064	0.15220	0.12260	0.14041
CNN	0.00844	0.0133	0.01119	0.01078	0.00739	0.00493	0.00516	0.00521	0.07579	0.01580

5.5.3 Comparison of trainable parameters

Table 5-3 shows the number of trainable parameters for each active noise cancellation model. The greater the number of parameters, the greater the time delay and the increased consumption of the system's inbuilt memory. Active noise cancellation systems require minimum delay, and it is desirable to have a small number of trainable parameters to reduce the processing time. Table 5-3 shows that the proposed architecture had just 444 parameters, while the LSTM and MLP algorithms had almost seven to two times more trainable parameters. However, the CNN algorithm had just 65 trainable parameters.

Table 5-3 : Numbers of trainable parameters

Model	**Number of trainable parameters**
ACRN	444
LSTM	3301
MLP	861
CNN	65

Chapter 6. Conclusion and Future Work

Section 6.1 summarizes the testing of the proposed method as an ANC, and the limitations of the study and proposed work are listed in sections 6.2 and 6.3.

6.1 Conclusions

A deep learning model comprising an ASP module based on combining a 1-D atrous convolution and a recurrent unit was proposed in this study. The 1-D atrous convolution layer with an exponential dilation rate was selected to extract the high-level features from the input. Nine noise data sets were used to train and test the proposed model. To train the model, 10,000, 10,000, 2,000, 2,000, 2,000, 5,000, 5,000, 5,000 and 5,000 samples were selected from the Flight1, Flight2, Leopard, Construction/Cutting, Volvo, Aero1, Aero2, Aero3 and Aero4 noise data sets, respectively. The high-level features were fed into a recurrent neural network followed by a dense layer to predict the anti-noise. The model parameters were tuned using the first four data sets (Flight1, Flight2, Leopard and Volvo), and the last five data sets (Aero1, Aero2, Aero3, Aero4 and Construction/Cutting) were used to test the robustness of the proposed method.

The input size for the proposed method was selected as 20 samples by considering previous studies in the same field (Park and Patterson, 2019). Each data set was normalized between the range of 0 and 1, and to regulate the input and output of the convolution layers, padding was used. The proposed model was optimized using the ADAM optimizer. To test the model, nine testing data sets were used, which were never seen by the model. The test data sets Leopard, Volvo, Construction/Cutting, Aero1, Aero2, Aero3, Aero4, Flight1 and Flight2 contained 1,937, 1,937, 1,937, 4,957, 4,957, 4,957, 4,957, 4,937 and 4,937 samples, respectively. To measure the performance of the proposed model, two metrics were used: RMSE and noise attenuation. It was shown that the proposed network outperformed the other methods based on these evaluation metrics. This thesis concluded the following:

- The network optimizer, along with the learning weight decay, were the most sensitive hyperparameters of the proposed model based on model tuning.
- The proposed method outperformed the traditional NLMS algorithm and other deep learning methods, including CNN, LSTM and MLP.
- Based on the performance metrics, the model achieved higher noise attenuation from 1 dB to 10 dB compared to deep learning methods and about three times better noise attenuation than the NLMS algorithm.
- The number of trainable parameters for each model was noted. The proposed model performed better with fewer trainable parameters, which is important for controlling the computational cost.

6.2 Limitations

The limitations of the proposed model are as follows:

- The proposed model was studied by only considering pure delay between the samples.
- The effect of a secondary path on the proposed ANC controller was not studied.
- The noise data set used for the network came from a single channel. The network might have behaved differently under noise coming from more than one channel.

6.3 Future works

To eliminate these limitations, the following future works are recommended:

- Study the effect of a secondary path on the model.
- Explore the behavior of the proposed model as a multichannel ANC system.
- Explore the hardware implementation of the proposed method.

References

[1] Aloysius, N. and Geetha, M., 2017, April. A review on deep convolutional neural networks. In *2017 International Conference on Communication and Signal Processing (ICCSP)* (pp. 0588-0592). IEEE.

[2] Borovykh, A., Bohte, S. and Oosterlee, C.W., 2017. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*.

[3] Britz, D., 2015. Understanding convolutional neural networks for NLP. *URL: http://www. wildml. com/2015/11/understanding-convolutional-neuralnetworks-for-nlp/(visited on 11/07/2015)*.

[4] Cai, M., Pipattanasomporn, M. and Rahman, S., 2019. Day-ahead building-level load forecasts using deep learning vs. traditional time-series techniques. *Applied Energy*, *236*, pp.1078-1088.

[5] Cartes, D.A., Ray, L.R. and Collier, R.D., 2002. Experimental evaluation of leaky least-mean-square algorithms for active noise reduction in communication headsets. *The Journal of the Acoustical Society of America*, *111*(4), pp.1758-1771.

[6] Cha, Y.J., Choi, W. and Büyüköztürk, O., 2017. Deep learning-based crack damage detection using convolutional neural networks. *Computer-Aided Civil and Infrastructure Engineering*, *32*(5), pp.361-378.

[7] Chalkiadakis, I., 2016. Convolutional Neural Networks for Object Detection: A visual cortex perspective.

[8] Chang, C.Y. and Chen, D.R., 2010. Active noise cancellation without secondary path identification by using an adaptive genetic algorithm. *IEEE Transactions on Instrumentation and Measurement*, *59*(9), pp.2315-2327.

[9] Chen, C.K. and Chiueh, T.D., 1996, May. Multilayer perceptron neural networks for active noise cancellation. In *1996 IEEE International Symposium on Circuits and Systems. Circuits and Systems Connecting the World. ISCAS 96* (Vol. 3, pp. 523-526). IEEE.

[10] Cortes, C., Mohri, M. and Rostamizadeh, A., 2012. L2 regularization for learning kernels. *arXiv preprint arXiv:1205.2653*.

[11] Coulibaly, P. and Baldwin, C.K., 2005. Nonstationary hydrological time series forecasting using nonlinear dynamic methods. *Journal of Hydrology*, *307*(1-4), pp.164-174.

[12] Denenberg, J.N., 1992. Anti-noise. *IEEE potentials*, *11*(2), pp.36-40.

[13] Deng, L. and Yu, D., 2014. Deep learning: methods and applications. *Foundations and trends in signal processing*, *7*(3–4), pp.197-387.

[14] Dixit, S. and Nagaria, D., 2017. LMS Adaptive Filters for Noise Cancellation: A Review. *International Journal of Electrical & Computer Engineering (2088-8708)*, *7*(5).

[15] Douglas, S.C., 1999. Fast implementations of the filtered-X LMS and LMS algorithms for multichannel active noise control. *IEEE Transactions on speech and audio processing*, *7*(4), pp.454-465.

[16] Duan, Y., Yisheng, L.V. and Wang, F.Y., 2016, November. Travel time prediction with LSTM neural network. In *2016 IEEE 19th international conference on intelligent transportation systems (ITSC)* (pp. 1053-1058). IEEE.

[17] Duchi, J., Hazan, E. and Singer, Y., 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, *12*(7).

[18] Elliott, S.J. and Nelson, P.A., 1993. Active noise control. *IEEE signal processing magazine*, *10*(4), pp.12-35.

[19] Elman, J.L., 1990. Finding structure in time. *Cognitive science*, *14*(2), pp.179-211.

[20] Fawaz, H.I., Forestier, G., Weber, J., Idoumghar, L. and Muller, P.A., 2019. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, *33*(4), pp.917-963.

[21] Fu, R., Zhang, Z. and Li, L., 2016, November. Using LSTM and GRU neural network methods for traffic flow prediction. In *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)* (pp. 324-328). IEEE.

[22] Gonzalez, A., Ferrer, M., De Diego, M., Pinero, G. and Garcia-Bonito, J.J., 2003. Sound quality of low-frequency and car engine noises after active noise control. *Journal of Sound and Vibration*, *265*(3), pp.663-679.

[23] Gupta, S. and Ghatak, C., 2011. Environmental noise assessment and its effect on human health in an urban area. *International Journal of Environmental Sciences*, *1*(7), pp.1954-1964.

[24] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[25] Hesselbach, J., Hoffmeister, H.W. and Loeis, K., 2009, December. Multiple channel filtered-X LMS-RLS vibration control in wood machining. In *2009 IEEE International Conference on Control and Automation* (pp. 2060-2065). IEEE.

[26] Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. *Neural computation*, *9*(8), pp.1735-1780.

[27] Hyndman, R.J. and Koehler, A.B., 2006. Another look at measures of forecast accuracy. *International journal of forecasting*, *22*(4), pp.679-688.

[28] Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J. and Keutzer, K., 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. *arXiv preprint arXiv:1602.07360*.

[29] Jaeger, H., 2003. Adaptive nonlinear system identification with echo state networks. In *Advances in neural information processing systems* (pp. 609-616).

[30] Jogin, M., Madhulika, M.S., Divya, G.D., Meghana, R.K. and Apoorva, S., 2018, May. Feature extraction using Convolution Neural Networks (CNN) and Deep Learning. In *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)* (pp. 2319-2323). IEEE.

[31] Kingma, D.P. and Ba, J., 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[32] Klambauer, G., Unterthiner, T., Mayr, A. and Hochreiter, S., 2017. Self-normalizing neural networks. In *Advances in neural information processing systems* (pp. 971-980).

[33] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

[34] Kuo, S.M. and Morgan, D.R., 1996. *Active noise control systems* (pp. 17-199). New York: Wiley.

[35] Kuo, S.M. and Morgan, D.R., 1999. Active noise control: a tutorial review. *Proceedings of the IEEE*, *87*(6), pp.943-973.

[36] Kuo, S.M., 2000. Active noise control systems with the TMS320 family. In *DSPSFEST 2000, DSP Technology and Education Conference, August 2000. NIU001 in conference proceeding CD-ROM SPRC045*.

[37] Kwon, N., Park, M., Lee, H.S., Ahn, J. and Shin, M., 2016. Construction noise management using active noise control techniques. *Journal of Construction Engineering and Management*, *142*(7), p.04016014.

[38] Larsson, M., 2011. *Active control of noise in ventilation systems: Analysis and experiments* (Doctoral dissertation, Blekinge Institute of Technology).

[39] Li, X.D., Ho, J.K. and Chow, T.W., 2005. Approximation of dynamical time-variant systems by continuous-time recurrent neural networks. *IEEE Transactions on Circuits and Systems II: Express Briefs*, *52*(10), pp.656-660.

[40] Liu, Z.S., Lee, H.P. and Lu, C., 2006. Passive and active interior noise control of box structures using the structural intensity method. *Applied Acoustics*, *67*(2), pp.112-134.

[41] Marburg, S., 2002. Developments in structural-acoustic optimization for passive noise control. *Archives of computational methods in engineering*, *9*(4), pp.291-370.

[42] Mishra, K., Basu, S. and Maulik, U., 2019, December. DaNSe: A Dilated Causal Convolutional Network Based Model for Load Forecasting. In *International Conference on Pattern Recognition and Machine Intelligence* (pp. 234-241). Springer, Cham.

[43] Na, K. and Chae, S.I., 1997, June. Single-sensor active noise cancellation using recurrent neural network predictors. In *Proceedings of International Conference on Neural Networks (ICNN'97)* (Vol. 4, pp. 2153-2156). IEEE.

[44] NagaJyothi, G. and SriDevi, S., 2017, March. Distributed arithmetic architectures for fir filters-a comparative review. In *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)* (pp. 2684-2690). IEEE.

[45] Nair, V. and Hinton, G.E., 2010, January. Rectified linear units improve restricted boltzmann machines. In *ICML*.

[46] Nassiri, P., Monazam, M., Dehaghi, B.F., Abadi, L.I.G., Zakerian, S.A. and Azam, K., 2013. The effect of noise on human performance: A clinical trial. *Int J Occup Environ Med (The IJOEM)*, *4*(2 April), pp.212-87.

[47] Neill, S.P. and Hashemi, M.R., 2018. Chapter 8-ocean Modelling for resource characterization. *Fundamentals of Ocean Renewable Energy*, pp.193-235.

[48] Oord, A.V.D., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. and Kavukcuoglu, K., 2016. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

[49] Park, S., Patterson, E. and Baum, C., 2019, September. Long Short-Term Memory and Convolutional Neural Networks for Active Noise Control. In *2019 5th International Conference on Frontiers of Signal Processing (ICFSP)* (pp. 121-125). IEEE.

[50] Qian, N., 1999. On the momentum term in gradient descent learning algorithms. *Neural networks*, *12*(1), pp.145-151.

[51] Rey, G.J., Bitmead, R.R. and Johnson, C.R., 1991. The dynamics of bursting in simple adaptive feedback systems with leakage. *IEEE Transactions on circuits and Systems*, *38*(5), pp.476-488.

[52] Ross, B.W. and Burdisso, R.A., 1999. Low frequency passive noise control of a piston structure with a weak radiating cell. *The Journal of the Acoustical Society of America*, *106*(1), pp.226-232.

[53] Ruder, S., 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

[54] Shynk, J.J., 1989. Adaptive IIR filtering. *IEEE Assp Magazine*, *6*(2), pp.4-21.

[55] Signal Processing information base (SPIB). http://spib.linse.ufsc.br/noise.html. (Accessed on 02/12/2019)

[56] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), pp.1929-1958.

[57] Stansfeld, S.A. and Matheson, M.P., 2003. Noise pollution: non-auditory effects on health. *British medical bulletin*, *68*(1), pp.243-257.

[58] Suter, A.H., 2002. Construction noise: exposure, effects, and the potential for remediation; a review and analysis. *Aiha Journal*, *63*(6), pp.768-789.

[59] Tichy, J., Warnaka, G.E. and Poole, L.A., 1983. Active noise cancellation in ducts. *The Journal of the Acoustical Society of America*, *74*(S1), pp.S25-S25.

[60] Tieleman, T. and Hinton, G., 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, *4*(2), pp.26-31.

[61] Tobias, O.J. and Seara, R., 2002, September. Performance comparison of the FXLMS, nonlinear FXLMS and leaky FXLMS algorithms in nonlinear active control applications. In *2002 11th European Signal Processing Conference* (pp. 1-4). IEEE.

[62] Toha, S.F. and Tokhi, M.O., 2008, September. MLP and Elman recurrent neural network modelling for the TRMS. In *2008 7th IEEE International Conference on Cybernetic Intelligent Systems* (pp. 1-6). IEEE.

[63] Wang, J., Fang, W. and Niu, H., 2016. Financial time series prediction using elman recurrent random neural networks. *Computational intelligence and neuroscience*, *2016*.

[64] Wilson, D.R. and Martinez, T.R., 2001, July. The need for small learning rates on large problems. In *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222)* (Vol. 1, pp. 115-119). IEEE.

[65] You, K., Long, M., Wang, J. and Jordan, M.I., 2019. How Does Learning Rate Decay Help Modern Neural Networks?.

[66] Zhao, B., Lu, H., Chen, S., Liu, J. and Wu, D., 2017. Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, *28*(1), pp.162-169.

[67] Zhao, T., Liang, J., Liang, Y., Wang, L., Pei, X. and Li, P., 2016. Secondary Path Modeling Method for Active Noise Control of Power Transformer. In *Proceedings of the 2015 International Conference on Applied Mechanics, Mechatronics and Intelligent Systems (AMMIS2015)* (pp. 427-433).