Chapter 3: Methodology

3.1 Introduction

As discussed earlier, the research objective is to determine the optimum performance for Malay children vowel recognition in terms of hidden neurons number and signal length. The implemented NN architectures are FFBP and Elman NNs. These two NN architectures were chosen in the current research as they consider among the best supervised learning NNs for classification; since many studies in the literature have proven their efficiency for speech recognition of different languages (Dede & Sazli, 2010).

This chapter outlines the data collection and the preprocessing analysis. Then, introduces the procedures used in constructing, training and testing the NN models. Finally, a brief description on how the performance of NN models was evaluated.

3.2 Data Collection

A supplementary speech database was provided by my supervisor Dr. *Ting Hua Nong*, and used in this study. The speech database is consisted of 360 samples were collected from healthy Malay children with their ages between 7 and 12 years old. This six Malay vowels were obtained from the six Malay words: “Gajah”, “Leher”, “Selipar” (/Səlipar/), “Filem”(/Filəm/), “Yoyo” and “Sudu”. The speech signals were recorded at 20 kHz sampling rate with 16-bit resolution.

3.3 Signal Preprocessing

For speech feature extraction, two types of frame analysis were applied: single-frame and multi-frame. Various signal lengths of the vowels were tested for both techniques.

3.3.1 Single-frame Analysis

For single-frame analysis, the examined lengths of vowel signals are 10ms, 15ms, 20ms, 25ms, 30ms, 35ms, 40ms, 45ms, 50ms, 55ms, 60ms, 65ms and 70ms. As a result, 13 single-frame data were examined. LPC was used to obtain the cepstral coefficients of each frame. The order of LPC is 24 as suggested by Nong & Yunus (2004). Hence, the 24 cepstral coefficients represent the different attributes of the vowel signal.

3.3.2 Multi-frame Analysis

For multi-frame analysis, the examined lengths of vowel signals are 30ms, 40ms, 50ms, 60ms, 70ms, 80ms, 90ms and 100ms. Hence, 8 multi-frame data were checked. LPC was used as well for getting the cepstral coefficients of each frame; however, the order of LPC is not fixed. The LPC orders of the mentioned frames are 48, 72, 96, 120, 144, 168, 192 and 216, respectively. As a result, each vowel is represented by higher number of cepstral coefficients than single-frame method.

The acquired data were divided into 3 sets according to 3-fold cross validation. In each set, there are 13 single-frame and 8 multi-frame data. Each frame contains 360 samples which were divided, according to *3*-fold cross validation, into 240 samples (2/3) were used in training, and 120 samples (1/3) were used in testing. Figure 3.1 depicts the process of data categorizing.

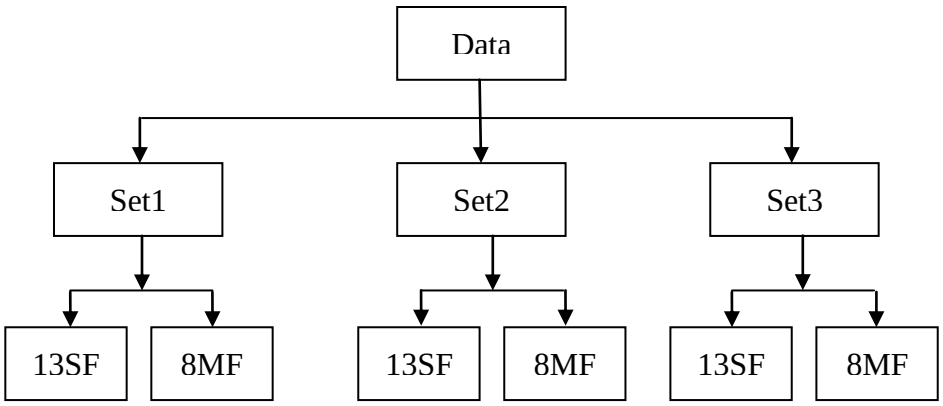

Figure 3.1: Block diagram illustrates data grouping. (SF: Single-Frame, MF: Multi-Frame)

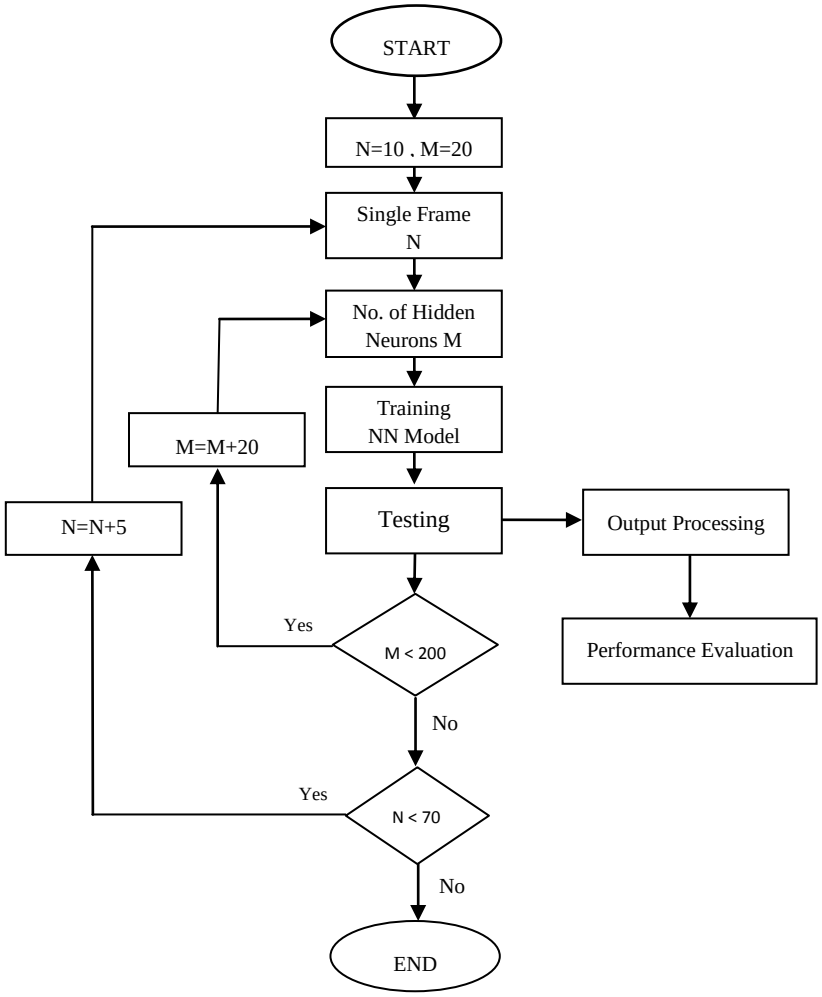

Figure 3.2: Flowchart summarizes the training and testing processes of single-frame vowels' samples.

Flowchart shown in figure 3.2 summarizes the training and testing steps of single-frame vowels' samples. The same procedure was followed for multi-frame vowels' samples with a difference in the frames' numbering (N). 21 frames (13 single-frame + 8 multi-frame) were trained and tested in each set of data. Ten models were created for each frame by changing the number of hidden neurons from 20 to 200 with a step of 20. The total number of NN models that were trained and tested in this study can be calculated as follows:

No. of NN models = 10 (models for each frame) × 21 (Frames) × 3 (Sets) × 2 (NN architectures)

= 1260 Models.

3.4 Constructing NN models

To construct, train and test the NN models, Matlab® Neural Network Toolbox™ was used. Matlab deals with matrices, so three main matrices were created for each model: input, target and test.

3.4.1 Input Matrix

The input matrix contains the input data used in training the NN model. The dimension of the train matrix is N×M, where N is the number of cepstral coefficients in the frames representing the different attributes of the vowel signal, and M is the number of the given examples 1440 (240 samples × 6 vowels). The samples' vectors were ordered as follows:

Input = [/a/ /∂/ /e/ /i/ /o/ /u/],

where each vowel denotes a vector of cepstral coefficients. This matrix is repeated 240 times in order to get the final train matrix with a dimension of N×1440.

3.4.2 Target Matrix

The target matrix contains the desired output which was assumed. This target was used for classifying the six vowels by comparing it with the actual output derived upon simulating the NN model. The dimension of the target matrix is K×M, where K is 6 (representing the 6 vowels), and M is the same number of input examples (number of columns in the input matrix). The building block of the assumed target matrix was designed as follows:

$$\text{Target} = \begin{bmatrix} 0.9 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.9 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.9 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.9 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.9 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.9 \end{bmatrix}$$

where, /a/ $= \begin{bmatrix} 0.9 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \end{bmatrix}$, /ə/ $= \begin{bmatrix} 0.1 \\ 0.9 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \end{bmatrix}$, /e/ $= \begin{bmatrix} 0.1 \\ 0.1 \\ 0.9 \\ 0.1 \\ 0.1 \\ 0.1 \end{bmatrix}$, /i/ $= \begin{bmatrix} 0.1 \\ 0.1 \\ 0.1 \\ 0.9 \\ 0.1 \\ 0.1 \end{bmatrix}$, /o/ $= \begin{bmatrix} 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.9 \\ 0.1 \end{bmatrix}$

and /u/ $= \begin{bmatrix} 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.9 \end{bmatrix}$.

This 6×6 matrix was replicated 240 times to form the final 6×1440 target matrix. The values “0.9” and “0.1” were chosen in the target matrix instead of “0” and “1” to avoid the requirement of extremely large weight values when the sigmoidal function is employed in the hidden layer (Sorsa, Koivo, & Koivisto, 1991).

3.4.3 Test Matrix

After training the NN model using the input and target matrices, the test matrix was applied to simulate the model in order to evaluate its generalizability. The test matrix belongs to an independent set of data (120 examples). The dimension of the test matrix is N×L, where N is the number of cepstral coefficients in the frames representing the different attributes of the vowel signal, and L is the number of the given examples 720 (120 samples × 6 vowels). The order of the vowels’ vectors is the same as the input matrix. It is done in the same order only to simplify the explanation of the confusion matrix later on.

3.4.4 Networks’ Initialization

After creating the three matrices of each NN model, networks were initialized with specifications shown in table 3.1. Weights and biases were initialized randomly. Trial and error approach was implemented to investigate the optimum number of neurons in the hidden layer. In the performed experiments, the number of hidden neurons ranges from 20 to 200 with a step of 20.

Table 3.1: Networks' initialization parameters.

Neural Network Architecture	FFBP	Elman
Training Function	Levenberg-Marquardt (for Single-frame Data) Resilient Backpropagation (for Multi-frame Data)	Variable Learning Rate Gradient Descent (for Single-frame and Multi-frame Data)
Performance Function	Mean Squared Error	Mean Squared Error
Adaption Learning Function	Gradient descent with momentum weight/bias learning function	Gradient descent with momentum weight/bias learning function
Data Division	Random	Random
Number of Layers	2 (One hidden layer and one output layer)	2 (One hidden layer and one output layer)
Transfer Function in the Hidden Layer	Tan-sigmoid	Tan-sigmoid
Transfer Function in the Output Layer	Linear	Linear

Levenberg-**M**arquardt (LM) algorithm was chosen for training the single-frame data in FFBP network because it has proved in the literature high performance with high convergence rate (Kermani, Schiffman, & Nagle, 2005). However, LM algorithm demands huge memory especially in training large networks (Lera & Pinzolas, 2002). This makes training multi-frame data in FFBP network not practical; since the number of LPC coefficient reaches up to 216 for each vowel. As a result, resilient backpropagation training function was used for training multi-frame data in FFBP network. Resilient learning function is fast technique since it requires less computation ; and hence less memory

requirement (Riedmiller & Braun, 1993). Moreover, resilient backpropagation algorithm is the fastest among other algorithms in pattern recognition problems (Kandaswamy, Kumar, Ramanathan, Jayaraman, & Malmurugan, 2004). One the other hand, variable learning rate gradient descent algorithm was chosen for training Elman networks as recommended by Beale, Hagan, & Demuth (2010). Therefore, choosing the training function for each network was justified.

Graphical **U**ser **I**nterface (GUI) Matlab® Neural Network Toolbox™ was used in networks initializing, training and testing. GUI was used because it is practical and easy to handle; however the full programming code, which can be run in Matlab® workplace, is provided in the appendix A. Figure 3.3 depicts network initialization window. Through this window, network architecture type, input and target data, training function, performance function, number of layers, number of hidden neurons and transfer function were assigned in order to initialized each NN model.

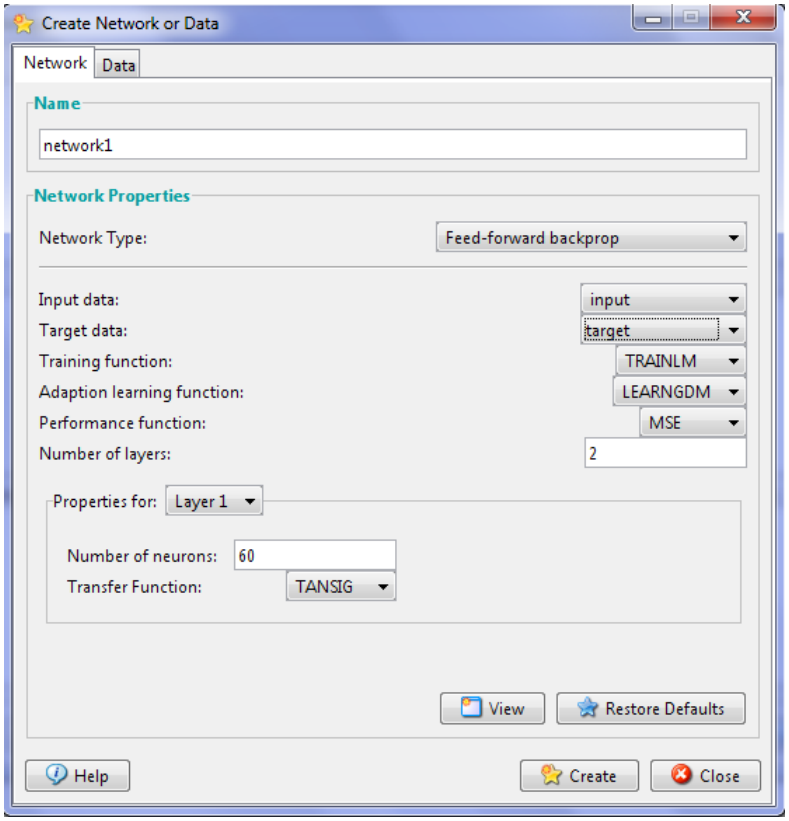

Figure 3.3: Network initialization window in Matlab® Neural Network Toolbox™.

3.5 Training NN Models

Before training the NN models, each training data set was divided into three groups: training, validation and testing. The training set is employed in computing the gradient and updating the network weights and biases. The validation set is used to check network generalization, and to stop training when generalization stops improving. On the other hand, the testing set has no influence on training; hence, it provides an independent measure of network performance during and after training (Demuth & Beale, 1992). In the current study, the percentage of training, validation and testing subsets are 60, 20 and 20%, respectively.

The parameters, which were used for training the NN models of FFBP and Elman, are shown in table 3.2. Performance goal was set to zero because of the fact that learning until error on the training set falls under specific threshold (performance goal) can lead to overfitting (Demuth & Beale, 1992). These training parameters were assigned using the windows shown in figure 3.4.

Table 3.2: Training parameters for FFBP and Elman NNs.

Neural Network Architecture	FFBP	Elman
Performance Goal	0	0
Minimum Gradient performance	$1e^{-10}$	$1e^{-10}$
Epoch	1000	1000
Time	Infinite	Infinite
Max. Validation Checks	6	6

Elman NN models were trained according to gradient descent with momentum and adaptive learning rate backpropagation. The initial learning rate was 0.01, while the ratios to increase and decrease the learning rate were 1.05 and 0.7, respectively. Moreover, the momentum constant was fixed at 0.9.

On the other hand, FFBP models were trained according to LM algorithm optimization. The initial adaptive value μ was set to 0.001, while the increasing and decreasing factors of μ were 10 and 0.1, respectively.

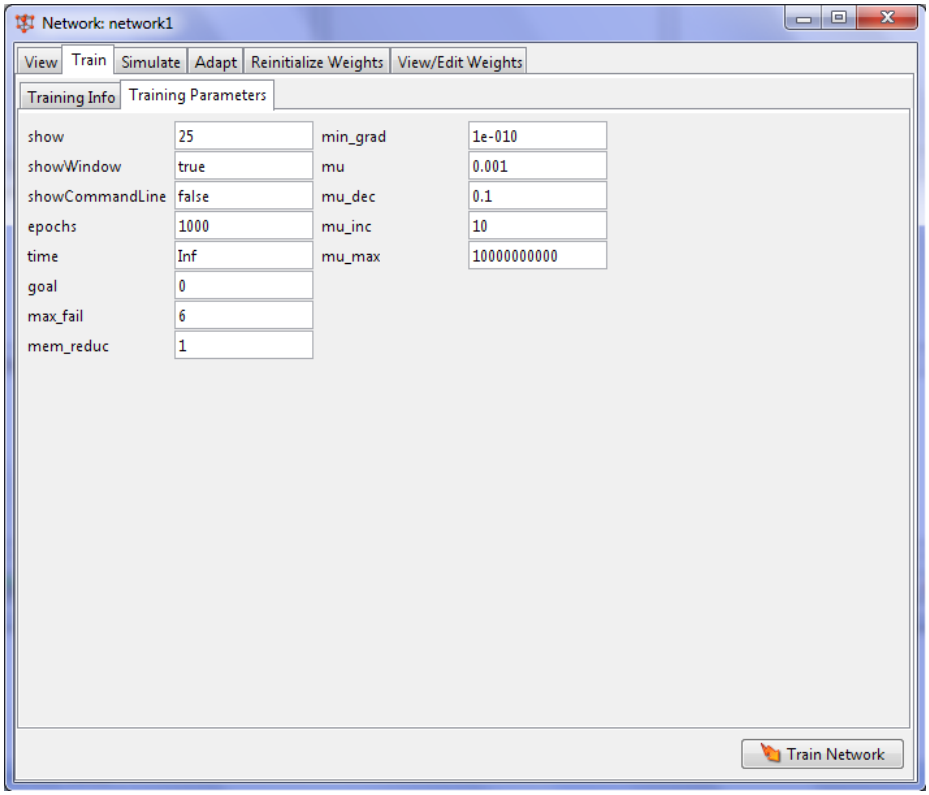

Figure 3.4: Window for assigning training parameters in Matlab® Neural Network Toolbox™.

Figure 3.5 shows the training window in Matlab® Neural Network Toolbox™. Training process stops when any of these conditions occurs:

- The number of successive iterations that the validation performance fails to decrease reaches 6.
- The magnitude of the gradient performance falls below $1e^{-10}$.
- The number of epochs reaches 1000.

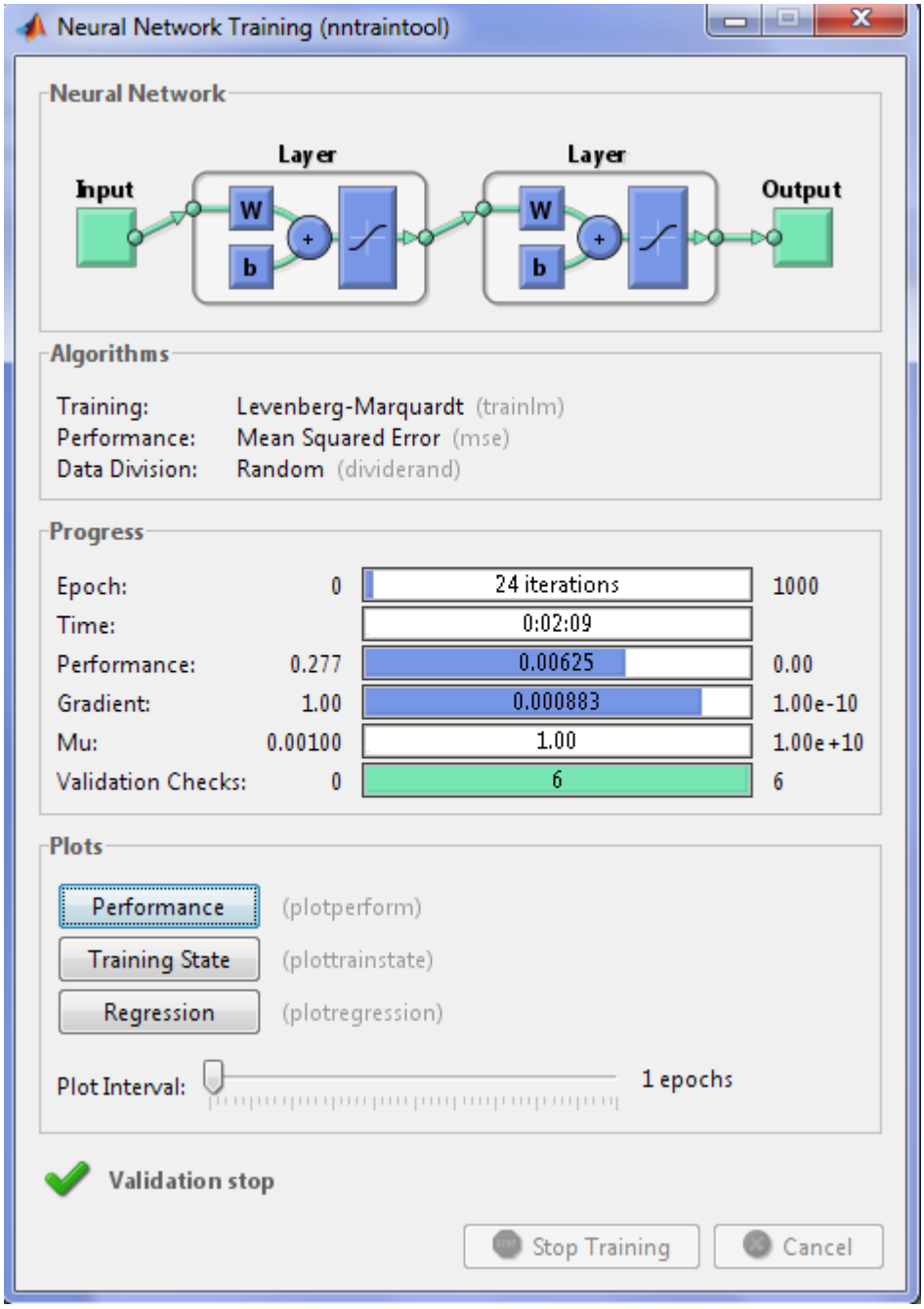

Figure 3.5: Training window in Matlab® Neural Network Toolbox™

3.6 Simulating NN Models

The NN models are simulated to evaluate their performances after the training process is done. To simulate the model, the independent test set is fed to the NN model. The output of the simulation is a K×L matrix, where K is the number of rows in the target matrix representing the 6 vowels and L is the number of the examples in the test matrix (720). This output was subjected to an analysis process; before being compared to the target in the confusion matrix.

3.7 Output Processing

The output of the simulation process underwent into three stages: output preprocessing, output classification and, finally, creating the confusion matrix.

3.7.1 Output Preprocessing

First, the output of simulating the NN models was transferred from GUI toolbox to Matlab® workplace. The elements' values of the raw output are not well organized. To make it more practical, it was transformed into 0 and 1 form. This transformation was done by changing the values of the maximum elements in each column to 1, while other elements were changed to zeros. An example for this output transformation is as follows:

$$\begin{bmatrix} 0.88 & 0.11 & 0.12 & 0.15 & 0.19 & 0.11 \\ 0.13 & 0.62 & 0.13 & 0.11 & 0.28 & 0.17 \\ 0.10 & 0.13 & 0.84 & 0.21 & 0.53 & 0.33 \\ 0.12 & 0.19 & 0.22 & 0.78 & 0.21 & 0.12 \\ 0.23 & 0.23 & 0.32 & 0.23 & 0.29 & 0.81 \\ 0.18 & 0.33 & 0.12 & 0.17 & 0.71 & 0.32 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

However, there were rare cases; in which two equal maximum elements are existed in one column. Such cases were treated as “others” (will be discussed in the following section). An example of these cases is shown in the first and last columns of the following matrix:

$$\begin{bmatrix} 0.88 & 0.11 & 0.12 & 0.15 & 0.19 & 0.11 \\ 0.88 & 0.62 & 0.13 & 0.11 & 0.28 & 0.81 \\ 0.10 & 0.13 & 0.84 & 0.21 & 0.53 & 0.33 \\ 0.12 & 0.19 & 0.22 & 0.78 & 0.21 & 0.12 \\ 0.23 & 0.23 & 0.32 & 0.23 & 0.29 & 0.81 \\ 0.18 & 0.33 & 0.12 & 0.17 & 0.71 & 0.32 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

3.7.2 Output Classification

In this stage, recognizing each column of the output matrix according to the vowels’ order in the test matrix was done. This was done by comparing the output’s columns to the following columns as follows:

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = /a/, \quad \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = /ə/, \quad \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = /e/, \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = /i/, \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = /o/ \text{ , and } \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = /u/.$$

However, if the output’s column was not matched with any of these columns (the rare case), then it was stored as “others”. (Note: for the code programming, [1 2 3 4 5 6] was used instead of [/a/ /ə/ /e/ /i/ /o/ /u/]).

3.7.3 Confusion Matrix

Confusion matrix was used in order to evaluate the performance of each NN models, and to check whether the NN model is confusing two or more vowels. Each column of the confusion matrix denotes the predicted vowels, while each row denotes the actual vowels. As a result, the output and target patterns of the vowels were fed to the confusion matrix. The recognition rate of each vowel and the total recognition rate of the model were simply calculated from the confusion matrix. An example of the confusion matrix is shown in table 3.3.

Table 3.3: Example of the confusion matrix.

Recognized as / Vowel	/a/	/ə/	/e/	/i/	/o/	/u/	Rec. Rate (%)
/a/	**111**	1	4	0	2	2	**92.50**
/ə/	0	**105**	3	8	2	2	**87.50**
/e/	8	4	**96**	0	2	10	**80.00**
/i/	0	1	0	**112**	3	4	**93.33**
/o/	3	0	4	0	**78**	35	**65.00**
/u/	0	0	1	0	10	**109**	**90.83**
Total Recognition Rate (%)							**84.86**