International Journal on Advanced Science Engineering Information Technology

Vol.10 (2020) No. 6
ISSN: 2088-5334

Calibrating Trip Distribution Neural Network Models with Different Scenarios of Transfer Functions Used in Hidden and Output Layers

Gusri Yaldi[a,1], Imelda M. Nur[a], Apwiddhal[a]

[a]*Politeknik Negeri Padang, Limau Manis, Padang-Sumatera Barat, 25168, Indonesia*
E-mail: [1]gusri.yaldi@gmail.com

***Abstract*— The transfer function is used to process the summation outputs in the hidden and output nodes. It can generally be categorized as either a non-linear or linear function. Examples are Sigmoid and Purelin functions representing non-linear and linear transfer functions. It is often mentioned that there is no standard guideline in the transfer function selection, and the Sigmoid or Logsig is widely used. However, the transfer function and training algorithm have a procedural relationship in training Multilayer Feedforward Neural Network (MLFFNN), a famous Artificial Neural Network model structure. In the feedforward stage, this function transforms the linear summation output to either linear (Purelin) or non-linear form (Sigmoid). In the backpropagation stage, this function is used in calculating the magnitude of change in the connection weights involving its derivative. Nine scenarios of MLFFNN were developed based on different transfer functions used in both hidden and output layers. In order to make fair comparisons, each scenario has the same initial connection weight. The modelling is conducted at the calibration level only; however, it involves different levels of complexity. It was calibrated by using the Levenberg-Marquard training algorithm. The results suggest that some calibrations failed and negative estimations occurred once non-linear transfer functions were used in hidden and output layers. It was found that Purelin was superior to other transfer functions. However, it has a weakness which is its negative estimations.**

***Keywords*— neural network model; transfer function; model calibration; estimated OD matrices.**

I. INTRODUCTION

The concept of Artificial Neural Network (ANN) began with the discovery of neuron in 1836 [1]. The human neural network consists of approximately 10^{11} neurons. Each neuron is connected to up to ten thousand other neurons and hence forms about 10^{14} to 10^{15} interconnections in the human brain neural network system. Figure 1 depicts a single neuron and its components. Those components are (1) Cell body or soma, (2) Axons, (3) Dendrites, and (4) Synapses. These four components have different functions. The ANN is constructed based on these four components and its function.

The cell body or soma contains the nucleus where information received from the dendrites is stored and processed. The received information is summed up and compared with a specific value of threshold to decide whether it should transfer or transmit signals to the other neuron through the axon. Hence, dendrites and axons function to bring the information/electrical signals in and out, respectively. Artificial intelligent experts try to mimic this information processing mechanism in the construction of the ANN, and the most famous one is called Multilayer Feedforward Neural Network (MLFFNN). It was widely used for forecasting and also known as Multiple-layered perceptron [2, 3] or Backpropagation neural network model [4]. The typical structure of MLFFNN is shown in Figure 2.

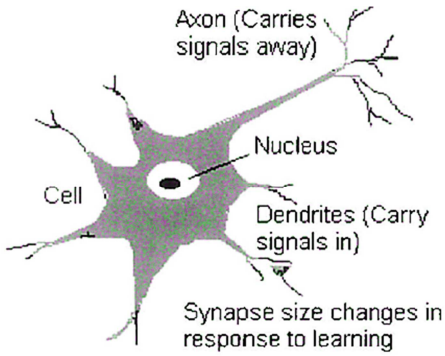

Fig. 1 Biological neuron diagram [1]

MLFFNN comprises input and output layers and a hidden layer between them. There could be more than one hidden layer as reported by Wang & Kim in predicting crash number on urban road networks by using ANN [5]; however, MLFFNN with a hidden layer is often used. Each layer has

several parallel nodes and is connected to other nodes in the adjacent layer. Every node in the same layer connects to other nodes in the previous and subsequent layers depending on its location. However, there is no connection between nodes within the same layer.

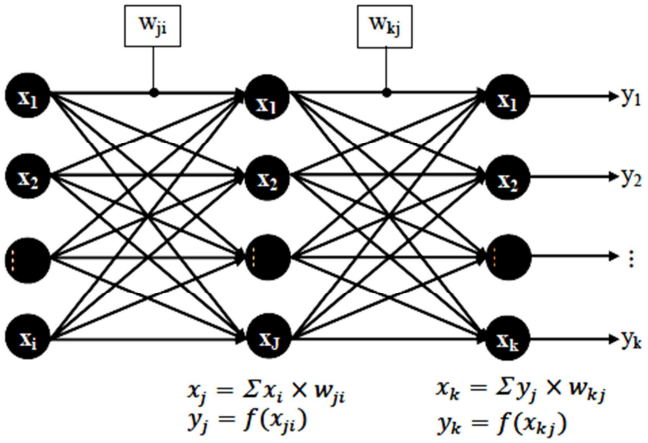

Fig. 2 Multilayer Feedforward Neural Network (MLFFNN)

In the MLFFNN, the nodes in the first layer (i) take the data into the network. The data flow, along with the connections, are scaled by the connection weights (w_{ji}). The output (y_j) is computed by transferring the scaled input by using an internal transfer function ($f(x_{ji})$). The estimation is obtained through the accumulation from all nodes' summation in the hidden layer (x_k). It is then sent to the output node and is commonly transformed by using a logistic transfer function in that node (y_k) [3, 6, 7].

Due to its potential capabilities, ANN has been adopted as a transportation modelling tool since the 1990s, mainly for driver behavior/autonomous vehicle modeling [7]. Its latest usage can be found not only for behavioral modelling but also for transport mode choice [8], real-time transport mode identification [9], traffic impact analysis [10] as well as for trip distribution forecasting [11, 12]. Furthermore, Moretti et al. reported that combining other modelling tools with ANN could increase its capability [13]. The ANN and simple statistical method were combined and were used in modelling urban traffic flow. It resulted in a promising performance. This combined model was called a hybrid modelling approach.

Based on the definition and the concept of ANN previously described, certain elements or factors configure ANN. According to Dougherty [7], those elements are namely (1) A set of processing elements (nodes), (2) Connectivity of those elements (connection weights), (3) Transfer functions, and (4) Learning algorithms (training algorithms). The transfer function and training algorithm have a procedural relationship in training MLFFNN. In the feedforward stage, this function will transform the linear summation output to either linear (Purelin) or non-linear form (Sigmoid). In the backpropagation stage, this function is used in calculating the magnitude of change in the connection weights involving its derivative.

The nature of ANN was considered as a Black-box modelling approach [9]. Therefore, the ANN approach's adoption must be supported by adequate empirical works since the inappropriate usage of transfer functions will lead to unrealistic model outputs. There is a lack of reported studies on the behavior of ANN concerning these properties. Once ANN was used in the transport modelling purposes, the selection of transfer functions was rarely discussed such as in a study by Chen et al. in forecasting tourist arrival [2] and Jabbar and Dia in predicting and monitoring traffic condition on freeways [4]. The same case was found in the modeling air pollution due to traffic impact on street and urban level [10] and modeling crash prediction in urban road networks [5]. Therefore, a comprehensive work on defining those properties, especially in the trip distribution modeling, can help this approach's users. Thus, this study investigates the performance of ANN in trip distribution modelling, especially in using different transfer functions for both hidden and output layers.

II. Materials and Method

Teodorovic and Vukadinovic [14] claimed Sigmoid and Linear functions as the most commonly used transfer functions. However, the lack of study reported the impact of different transfer functions toward neural network model performance even though transfer function is one of the ANN's main properties.

Black [6] reported the utilization of neural network spatial distribution model for strategic forecasting. The study compared the gravity and neural models. The neural models were developed based on the gravity model's traditional form and used Sigmoid as the transfer function. Figure 3 shows the neural network model structure developed by Black called Gravity Artificial Neural Network/GANN). Three case studies were reported, (1) A three-region flow problem; (2) A commodity flow problem, and (3) A migration flow problem. Although both gravity and neural models produced acceptable results, the neural models were found as a better calibration tool and were recommended for future flow forecasting.

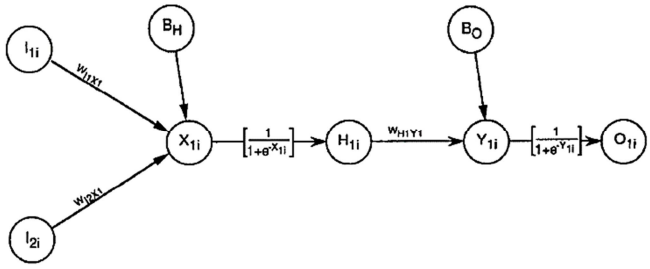

Fig. 3 Gravity Artificial Neural Network (GANN) used by Black [6]

Mozolin et al. [15] reported another strategic forecasting by ANN (see Figure 4 for the neural model structure). The difference between the studies by Black [6] and Mozolin et al. [15] is that Black [6] used Sigmoid as transfer functions in all layers resulted in values typically ranging within (0) and (+1). Meanwhile, Mozolin et al. [15] used Tansig as the transfer functions, which produced numbers ranging within (-1) and (+1). Consequently, the output of the model representing the estimated trip numbers may comprise both negative and positive values, while the trip values are always a positive value. Furthermore, Mozolin et al. [15] extended the analysis to the testing level. Although both studies

developed neural models based on the structure of the traditional constrained gravity model, Mozolin et al. [15] reported that the neural model had a poor generalization performance despite their superior calibration ability claimed by Black [6].

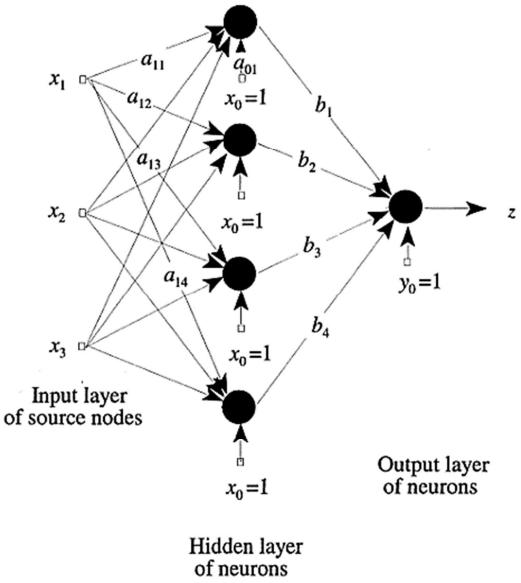

Fig. 4 Neural Network Architecture used by Mozolin et al. [15]

Dantas et al. [16] used neural models to forecast travel demand where the data were mainly obtained from remote sensing images processed in the geographical information system. Hardlim transfer function was used in the output node resulted in negative forecasted trip interchange volumes contradicting to the reality where there are no negative trip numbers. Meanwhile, Yaldi et al. [11] simulated some neural network models where the input data were normalized with the same transfer function used in the output layer with the neural network model structure depicted in Figure 5. A significant improvement was found in the model outputs compared to those normalized by using simple normalization method where the entire input pattern vector, except the distance, was divided by the total number of trips.

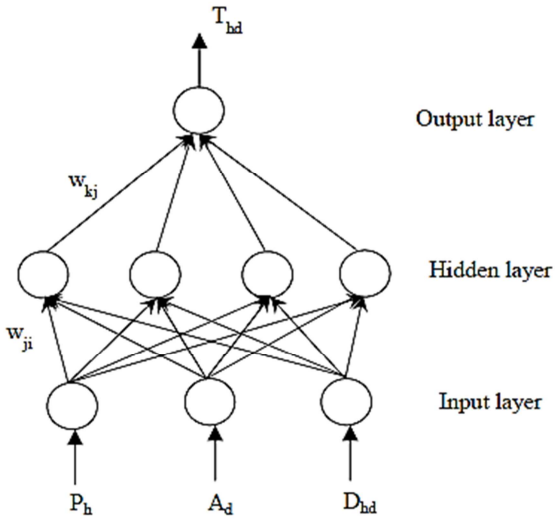

Fig. 5 Neural Network Architecture used by Yaldi et al. [11]

Its maximum value normalized the distance. The advantage of this normalization method is that there is only one unique factor used in normalizing the data, and hence it becomes simpler than dividing them by their maximum values. This method was used by Black [6]. Thus, it confirmed that the transfer function plays essential roles in optimizing neural network model performance.

A. Common transfer functions

Figure 6 shows the common transfer functions used in ANN [14] and also in this research, namely:

1) Tansig:

$$f(x) = \left(\frac{exp(x)-exp(-x))}{(exp(x)+exp(-x))}\right. \quad (1)$$

2) Sigmoid/Logsig:

$$f(x) = (1 + exp(-(x)))^{-1} \quad (2)$$

3) Linear:

$$f(x) = x \quad (3)$$

Among the three transfer functions, the Sigmoid or logistic function (Logsig) is the most widely used [5-7]. Each node in the network may use different transfer functions; yet, the same transfer function is commonly used within a layer. Both hidden and output layer nodes usually use the Sigmoid function. There is no standard rule in selecting the transfer function. The authors rarely explain the selection process of transfer function; however, it can be assumed that the Sigmoid function is used to capture the non-linear relationship among the model variables. It has a range of value typically within (0) and (+1) for Logsig, and (-1) and (+1) for double logistic/Tansig.

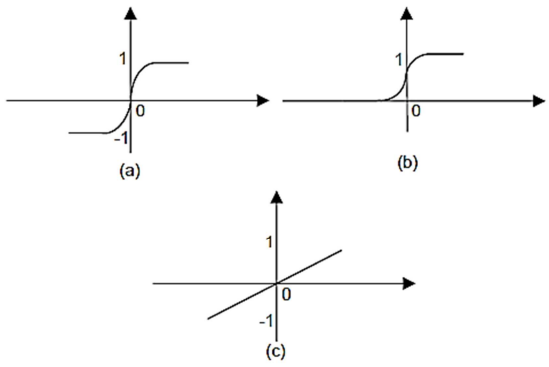

Fig. 6 Commonly used transfer functions

B. Model Data

This study uses the US 1965-1970 migration data in addition to Black's three region flow problem [6]. The data represents nine major census regions in the USA namely (1) New England, (2) Middle Atlantic, (3) East North Central, (4) West North Central, (5) South Atlantic, (6) East South Central, (7) West South Central, (8) Mountain, and (9) Pacific. Different spatial interaction modelling has also been used these data, as reported by Black [6]. This data is a nine-square matrix, which is nine times bigger than the Black three region flow problem.

C. Model Scenario

The ANN performance towards different transfer functions for the output layer in calibrating trip distribution model was reported by Yaldi et al. [17] and a linear transfer function for the output layer was proposed. Thus, this study expands it by varying the transfer function in the output layer and the hidden layer. Consequently, it results in nine different scenarios, where each scenario has the same initial connection weight in order to make fair comparisons.

Table 1 presents the scenarios used in the training process. These are categorized based on the types of transfer functions used in both hidden and output layers. The performance of the neural model for those scenarios can then be compared.

TABLE I
NEURAL MODEL SCENARIO BASED ON TRANSFER FUNCTIONS

Scenario #	Transfer function		Experiment #	Remark
	Hidden node	Output node		
1	Tansig	Tansig	30	Called Tan scenario
2	Tansig	Logsig	30	
3	Tansig	Purelin	30	
4	Logsig	Tansig	30	Called Log scenario
5	Logsig	Logsig	30	
6	Logsig	Purelin	30	
7	Purelin	Tansig	30	Called Pur scenario
8	Purelin	Logsig	30	
9	Purelin	Purelin	30	
Total	9 scenarios		270 trials	

D. Model Level

The modelling is conducted at the calibration level only; however, it involves different levels of complexity. Firstly, the neural models are developed for Black's three region flow [6], where the problem's size is a 3x3 matrix. The results are then ranked based on the model performance for each scenario.

The best scenarios are then further assessed by using a higher degree of data complexity, in this case, a 9x9 matrix of US 1965-1970 migration flow data. The evaluation considers the accuracy and precision of the estimated trip interchange volumes by using the Root Mean Square Error (RMSE), R^2, and absolute error distribution for different error range categories. MLFFNN with a hidden layer and ten hidden nodes is used in training. The training is conducted using the Levenberg-Marquardt algorithm as this was the most effective and efficient training algorithm [18].

III. RESULTS AND DISCUSSIONS

A. Black's three region flow model

Table 2 shows the performance of neural models for nine combinations of sub-scenarios. The first column shows the transfer function in the hidden nodes, while the second row shows it for the output node. The best scenario in terms of the values of RMSE and R^2 belongs to the Log scenario (transfer function in all the hidden nodes is a Logsig function). Although not all the RMSE for each Log sub-scenario (logsig_tansig, logsig_logsig, and logsig_purelin) are lower than other sub-scenarios, two of them have lower RMSE than others. The same trend is also indicated for the R^2 coefficients. Its value for Log scenario is higher than other scenarios, except for log_tan sub-scenario. Then, the sub-scenarios involving Purelin transfer function performs better than other sub-scenarios; however, this only happens when Purelin is used in output nodes (see also Figures 7-9).

TABLE II
AVERAGE RMSE AND R2 FOR DIFFERENT SCENARIOS, BLACK'S SAMPLE

HL transfer function	OL transfer function					
	RMSE			R^2		
	Tan	Log	Pur	Tan	Log	Pur
Tan	8.968	0.546	0.004	0.999	0.998	1.000
Log	5.967	0.001	0.000	0.999	1.000	1.000
Pur	1.564	1.988	1.558	0.958	0.936	0.959

When Purelin is used in hidden layer nodes, log_log sub-scenarios tend to have a higher performance grade. This finding is related to the non-linearity and linearity issues. A neural model with a hidden layer tends to perform better than the one without it [12]. The hidden layer can capture the non-linear relationship in the data. Thus, the logistic function is more suitable for this purpose than the linear transfer function as indicated by the results in Table 2.

Meanwhile, the sub-scenarios where Purelin is used in the output node tend to perform better than other kinds of transfer functions. This result can be explained by looking at the relationship between the method used to normalize the target vector and the type of transfer function (linear or non-linear) used in the output node. Simple normalization method was used to normalize the target vector. Thus, Purelin is more suitable as a transfer function in the output layer. Zhang et al. [19] suggested that the linear transfer function is suitable for continuous data forecasting.

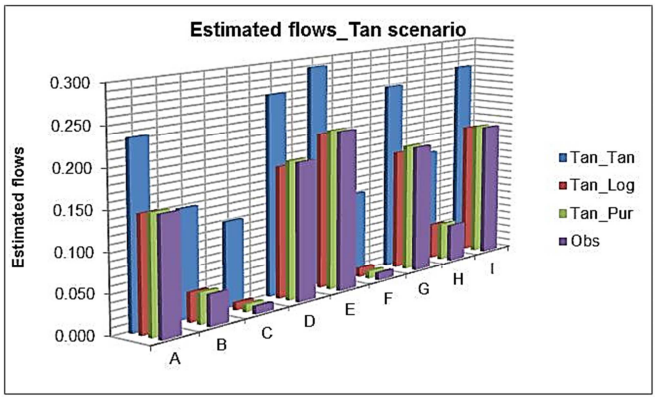

Fig. 7 Estimated flow numbers for Black's sample, Tan scenarios

Tan scenario, which uses Tansig in the output node, generates overestimated outputs, as seen in Figure 7. However, the output is closer to the real one when the transfer function is switched to either Logsig or Purelin. Although the usage of Tansig function also generates negative outputs, fortunately training with this simple and small size matrix does not result in any negative outputs for this scenario (see Figure 7). However, the Tansig usage in

the output node leads to the occurrence of failed training up to ten per cent of the total 30 trials. Thus, there is the possibility of training being unsuccessful when Tansig is used in the output node.

The next one is Log scenario. Figure 8 depicts the estimated flows for each of Log sub-scenario and illustrates the average flows estimated for 30 trials. Among the Log sub-scenarios, log_pur tends to superior to the other sub-scenarios. Log_tan sub-scenario has overestimated flows number for all samples from A to I, the same as the one in Tan scenario. The overestimations occur due to the failed training experienced by log_tan scenario.

Both log_log and log_pur sub-scenarios have almost the same performance level, except that the RMSE for log_pur is lower. This performance is related to the output of Purelin (linear) function and target vector, which is also linearly normalized. The target vector in this model is linearly normalized, matching with the output of neural models for log_pur sub-scenario which generates linear outputs due to the usage of the linear transfer function in the output node (Purelin). It transfers the output linearly according to Equation 3. It is related to the summation of the inputs multiplied with the relevant connection weights from the hidden nodes and received by the output nodes.

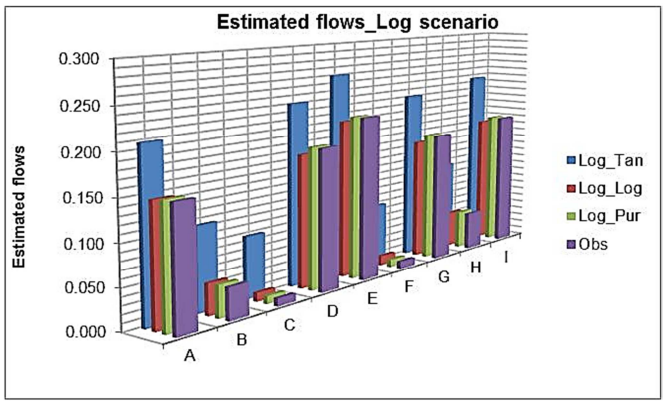

Fig. 8 Estimated flow numbers for Black's sample, Log scenarios

Unlike Logsig and Tansig, Purelin just receives the summation output according to Equation 3, and the results are used in the computation of the difference between the neural model output ($O'_k(T_{hd})$) and the target vector (t_{hd}) counted by the following equation:

$$diff = t_{hd} - O'_k(T_{hd}) \quad (4)$$

This equation tries to calculate the difference based on the neural model outputs and the target vectors, which are both linearly transformed or normalized.

A different situation occurs when either Logsig or Tansig function is used in the output node. The neural model output is non-linearly transformed, and the error or the difference is obtained by subtracting the non-linearly transformed neural model outputs from the linearly transformed target vector. A systematic error occurs due to the mismatch between the nature of the neural model output and the target vector. This error affects the training results, as indicated by the neural model performance reported in Table 2.

The utilization of a linear normalization method is a disadvantage for log_log sub-scenario. It is due to the neural model that calculates the difference by using Equation 2 and uses it in adjusting the connection weight. In this equation, the estimated trip (T_{hd}) is transformed non-linearly by Logsig function while the observed trip (t_{hd}) is transformed linearly. It is expected that the performance of log_log sub-scenario will improve when the target vector is also non-linearly transformed according to Logsig function. It should also not be forgotten that the sample size is relatively small and simple. A more complex sample will have different trends. Like Tan scenario, no negative estimation belongs to any Log sub-scenario, although Tansig and Purelin functions are used in the output node.

It can be seen from Table 2 that using linear transfer function for both hidden and output layers is also not recommended. It is better to use a combination of logsig_purelin than purelin_purelin sub-scenario. Figure 9 shows that some estimations are negative in contradiction to the real ones which are positive (see sample C). The negative estimation occurs when Purelin is used in both hidden and output nodes, for 30 trials. Using Purelin in both hidden and output nodes means that the linear function is used to capture the relationship. Moreover, the neural model output is also linearly estimated.

Meanwhile, the performance of Tan scenario is also seen to be worse than the other scenarios, and hence it is also not recommended. There are two reasons for this trend, namely:

1) Up to ten percent of Tan scenarios: Up to ten percent of Tan scenarios experienced failed training, especially tan_tan sub-scenario: The results of failed training negatively contribute to Tan scenario's average performance. The estimations become greatly overestimated (see Figure 7). The failed training may result from the utilization of double logistic function in the output node. It generates non-linear transformed estimation, while the target vector is linearly transformed. Thus, the usage of linear data normalization is also a disadvantage for tan_tan sub-scenario. The training fails when the minimum gradient (1E-10) is reached, which means that there is no change in the weight adjustment process.

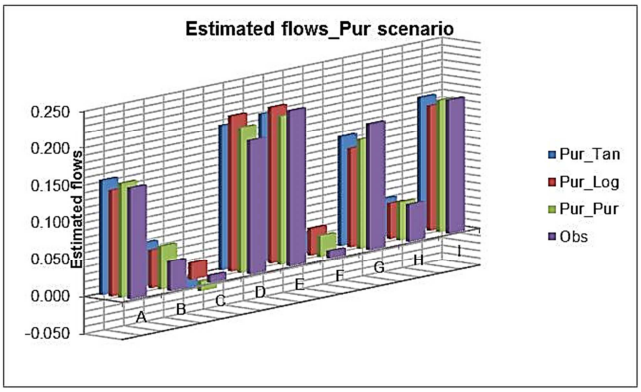

Fig. 9 Estimated flow numbers for Black's sample, Pur scenarios

2) The performance of the Tan scenario worsens: The performance of the Tan scenario worsen since the negative results are generated by the neural model resulted from the usage of double logistic function (see Equation 1 and Figure 6(a)). This function transforms the results non-linearly between negative and positive values. Negative estimations contradict with the reality as negative trip interchange

volumes do not exist. It was found that the number of negative outputs was ten per cent of the total samples.

Thus, the scenarios can be ranked from the best one, namely Log scenario, followed by Tan and Pur scenarios, respectively. Based on this result, the Log scenario will be used again in the training of the same neural models. However, the training is conducted for a different and more complex dataset, namely the US 1965-1970 migration flow data.

B. US 1965-1970 migration flow model

Given Log scenario, as the best scenario resulted from 30 trials by Black three region flow [6], the same neural models have trained again. At this time, the nine-square matrix based on the US 1965-1970 migration flow data was used as the dataset. The training again comprises three different sub-scenarios, namely (1) Log_tan, (2) Log_log, and (3) Log_pur.

It is expected that the impacts of double logistic and linear transfer functions could be seen more clearly in a larger dataset. According to Figures 6 (a) and (c), the usage of both Tansig and Purelin generates positive and negative results. However, this was unseen in the previous experiment with the Black sample. Moreover, the usage of Logsig was successful without experiencing failed training like the Tansig. These situations are expected to occur when a more complex dataset is used.

The training was conducted for two different maximum epoch numbers, namely 10 and 100 iterations. It sought to explore the advantage of training the neural model for real data with more epochs. However, it is limited to 100 iterations. Training the model with a higher number would potentially cause over-fitting.

In order to communicate the results better, the distribution of the error is illustrated based on specific error range categories, as seen in Tables 3 and 4, and Figures 10-12. There are 11 categories of error ranges. These tables and figures show the error distribution for the first trial of log_tan, log_log, and log_pur sub-scenarios for both 10 and 100 epochs. Showing the results for all trials will be good; however, it will also be crowded and hence considered possibly ineffective. The average errors based on 30 trials are also reported.

TABLE III
DISTRIBUTION OF ERROR FOR THE FIRST TRIAL (10 EPOCHS)

Error range	10 epochs		
	Log_Tan	Log_Log	Log_Pur
<=10%	20	15	19
10<X<=20	10	12	9
20<X<=30	15	12	21
30<X<=40	19	12	14
40<X<=50	4	9	9
50<X<=60	6	6	5
60<X<=70	1	1	4
70<X<=80	2	4	1
80<X<=90	1	5	4
90<X<=100	5	0	1
>100	17	23	15

The typical trend shown by these tables and figures is that the error decreases when the neural model is trained with more epochs. The numbers inside the brackets show the gap of error between the training of 10 and 100 epochs. The most significant gap belongs to log_log sub-scenario, for the first error range category. As expected, more samples have errors, not more than 10 per cent after the epoch is increased to 100 iterations. Other error ranges decrease.

TABLE IV
DISTRIBUTION OF ERROR FOR THE FIRST TRIAL (100 EPOCHS)

Error range	100 epochs		
	Log_Tan	Log_Log	Log_Pur
<=10%	30(10)	47(32)	30(11)
10<X<=20	22(12)	11(-1)	15(6)
20<X<=30	10(-5)	5(-7)	14(-7)
30<X<=40	7(-12)	5(-7)	12(-2)
40<X<=50	4(0)	7(-2)	2(-7)
50<X<=60	1(-5)	5(-1)	2(-3)
60<X<=70	1(0)	1(0)	0(-4)
70<X<=80	1(-1)	4(0)	2(1)
80<X<=90	5(4)	1(-4)	1(-3)
90<X<=100	2(-3)	0(0)	0(-1)
>100	16(-1)	14(-9)	21(6)

TABLE V
DISTRIBUTION OF AVERAGE ERROR FOR 30 TRIALS (10 EPOCHS)

Error range	10 epochs		
	Log_Tan	Log_Log	Log_Pur
<=10%	14	12	16
10<X<=20	14	11	13
20<X<=30	11	9	15
30<X<=40	11	9	13
40<X<=50	7	7	9
50<X<=60	6	5	6
60<X<=70	3	3	4
70<X<=80	2	2	3
80<X<=90	2	3	3
90<X<=100	2	17	2
>100	28	20	18

TABLE VI
DISTRIBUTION OF AVERAGE ERROR FOR 30 TRIALS (100 EPOCHS)

Error range	100 epochs		
	Log_Tan	Log_Log	Log_Pur
<=10%	25(11)	32(20)	26(10)
10<X<=20	16(2)	13(2)	16(3)
20<X<=30	12(1)	8(-1)	13(-2)
30<X<=40	10(-1)	5(-4)	12(-1)
40<X<=50	5(-2)	5(-2)	5(-4)
50<X<=60	3(-3)	4(-1)	3(-3)
60<X<=70	2(-1)	2(-1)	2(-2)
70<X<=80	2(0)	1(-1)	2(-1)
80<X<=90	2(0)	1(-2)	2(-1)
90<X<=100	2(0)	15(-2)	2(0)
>100	22(-6)	14(-6)	16(-2)

In the second place is log_pur sub-scenario. The gap is lower for the first error range, compared to log_log sub-

scenario. More errors are concentrated within 10-40 percent error range categories, counted for 41 percent compared to the first category's error, which is only 30 percent. Then, some error ranges experience increasing percentages, although the epoch is increased. The examples are error range 70<X≤80, and >100.

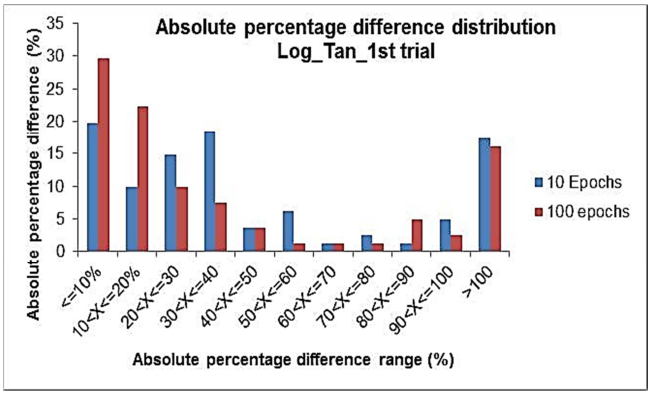

Fig. 10 Absolute percentage different distributions, log_tan, 1st trial

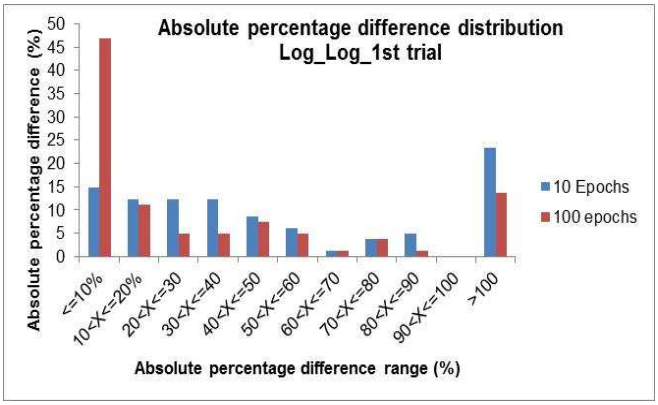

Fig. 11 Absolute percentage different distributions, log_log, 1st trial

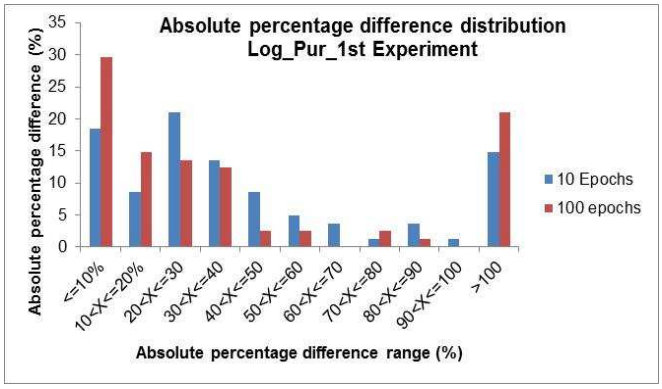

Fig. 12 Absolute percentage different distributions, log_pur, 1st trial

However, this is for the first trial only. The average results for 30 trials show that all of error range categories decrease except for the first category, as expected (see Table 6). The third-place belongs to log_tan sub-scenario. The percentage of error distributed for the first category is slightly lower than log_pur sub-scenario, for both the first trial and the average results. Log_tan sub-scenario has initially 20 per cent of error distributed in the first error category. When the epoch is increased to 100 iterations, it was found that about 30 per cent of the sample has error not more than ten per cent.

The training for 30 trials results in about seven per cent unsuccessful training. It also can be seen from Tables 3-6 that the percentage of error distributed for >100 per cent error range category is high, especially for log_log sub-scenario. This occurrence is due to:

*1) Negative estimations:*The error is calculated by using the following formula:

$$Absolute\left(\frac{Observation-estimation}{Observation}\times 100\%\right) \qquad (5)$$

Thus, the usage of this formula results in a more significant percentage as a result of negative estimation. The negative estimation is due to Tansig and Purelin's utilization, which allow and generate both positive and negative outputs. The negative estimation is counted for an average of six per cent of the total sample numbers. It remains after the epoch is increased to 100 iterations. This problem can be solved by using Logsig function, which only generates positive continuous numbers.As the consequence of Tansig function usage, up to 12 per cent (from 81 samples) of the neural outputs are negative. This value drops to eight per cent when the training is conducted for 100 epochs. However, the average estimation is positive for all samples because of the failed training, which generates +1 output for the Tansig function.

2) Overestimations: In general, overestimations dominate more than negative estimations. It is in the average of (5-10) per cent of the total sample number for ten epoch-training, dropping to (2-4) per cent when the epoch is increased to 100 iterations. Therefore, increasing the epoch number can minimize overestimations.

3) Zero trips were estimated as a non-zero trip: Neural models are unable to map zero value observations correctly. The models estimate the zero trips as numbers very close to zero. When the zero-value observation is estimated as either positive or negative non-zero estimation (although it is so small that it is close to zero), the difference is considered above 100 per cent. This result occurs for all sub-scenarios. This problem remains even when the epoch is increased to 100 iterations.

Table 3-6 and Figures 13-15 shows the average absolute error distribution for eleven error range categories. They also display the results for neural models trained with 10 and 100 epochs. Figures 13 and 14 show almost identical pictures as those in Figures 10 and 11, except that the error distribution for the last two error range categories is much larger. It is due to the first two figures displaying the average results, including the failed training. Log_tan and log_log sub-scenarios have 7 and 17 per cent of unsuccessful training respectively, in contrast to log_pur sub-scenario which has all its training successful. Therefore, the first trial of log_pur sub-scenario and its average results show relatively typical trends. Increasing epoch number cannot solve the unsuccessful training as the training stops at the first iteration.

It was previously reported in the training of Black's three region flow that all trials were successful. The usage of Tansig for Log scenario was reported to have up to ten per

cent unsuccessful training. It is related to the method used in normalizing the target vector and the output generated by the logistic function. The experiments with real data and with higher complexity resulted in 7 and 17 per cent of failed training for log_tan and log_log sub-scenarios, respectively. The impact of the mismatch between the linear normalization method applied for the target vector, and the result of the non-linear logistic function in the output node now becomes more evident.

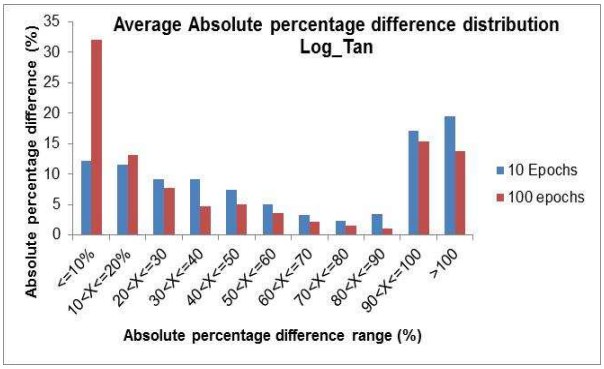

Fig. 13 Average absolute percentage different distributions, log_tan

Because of failed training, Tansig generates outputs as +1 for all samples, while Logsig generates very small positive numbers for all samples. Therefore, the log_tan sub-scenario's average error for 30 trials shows no negative output (see Figure 13). Meanwhile, failed training for log_log sub-scenario has distributed the error for the error range category 90<X≤100. Tables 5 and 6 show the error distribution for this range. The error reaches 17 and 15 per cent for 10 and 100 epochs, respectively. The percentage drops two per cent when the epoch is increased to 100 iterations.

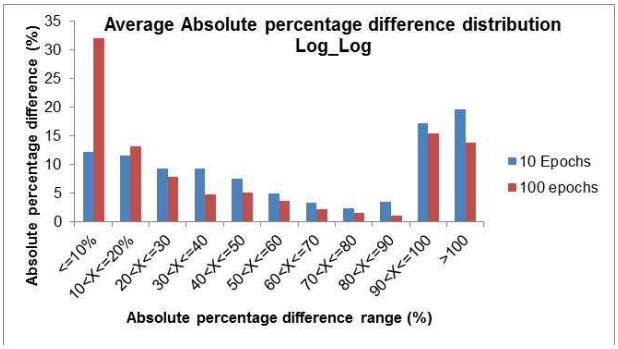

Fig. 14 Average absolute percentage different distributions, log_log

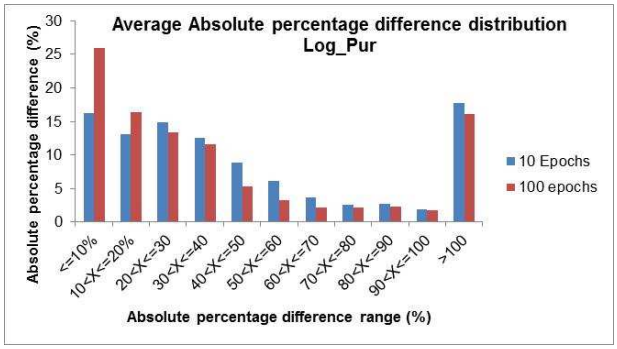

Fig. 15 Average absolute percentage different distributions, log_pur

C. *Estimated O-D matrix*

The estimated O-D matrix for each sub-scenario of Log scenario can be seen in Tables 7-10. The results presented in this section are for 100 epochs only. The US migration flow problem is more complicated than Black's problem and hence requires more epochs to generate good results. The general results of Black three-region flow are (1) The training for log_pur sub-scenario is relatively quicker than log_log sub-scenario. It only requires in average five epochs to converge. At the same time, it is triple for log_log sub-scenario, and (2) Log_tan sub-scenario experiences seven per cent failed training. Hence, it is unable to generate good estimations to construct O-D matrix even though the epoch is increased many times. Its performance only can be solved when failed training can be avoided.

It should be remembered that these results are based on a relatively simple three-square matrix. Some expected results did not occur. Training with more complex and real data like the US 1965-1970 migration flow has resulted in the occurrence of anticipated events. Examples are (1) Failed training happened not only for log_tan sub-scenario, but also for log_log sub-scenario, and (2) Negative outputs were generated by log_tan and log_pur sub-scenarios as the result of the usage of Tansig and Purelin functions. Thus, these findings affect the O-D matrices estimated by each sub-scenario, as reported in Tables 7.8-7.10.

TABLE VII
AVERAGE O-D MATRIX FOR US 1965-1970 MIGRATION FLOW: LOG_TAN (100 EPOCHS)

Origin	Destination									Total	Gap
	1	2	3	4	5	6	7	8	9		
1	67	78	73	68	81	67	69	68	75	646	-1069
2	86	68	97	74	118	75	80	75	88	761	-399
3	73	85	66	90	109	85	83	75	96	763	-339
4	70	75	92	68	81	75	84	77	85	706	-616
5	75	87	90	75	68	87	79	74	87	722	-403
6	69	73	88	73	93	68	80	72	75	692	-763
7	69	75	83	79	83	78	66	74	85	692	-642
8	69	71	76	73	75	69	73	66	92	664	-727
9	72	77	84	76	87	71	81	84	65	696	-460
Total	650	688	751	674	795	675	696	665	749	6341	
Gap	-1081	-632	-416	-780	-293	-913	-592	-666	-330		

TABLE VIII
AVERAGE O-D MATRIX FOR US 1965-1970 MIGRATION FLOW: LOG_ LOG (100 EPOCHS)

Origin	Destination									Total	Gap
	1	2	3	4	5	6	7	8	9		
1	67	78	73	68	81	67	69	68	75	646	-1069
2	86	68	97	74	118	75	80	75	88	761	-399
3	73	85	66	90	109	85	83	75	96	763	-339
4	70	75	92	68	81	75	84	77	85	706	-616
5	75	87	90	75	68	87	79	74	87	722	-403
6	69	73	88	73	93	68	80	72	75	692	-763
7	69	75	83	79	83	78	66	74	85	692	-642
8	69	71	76	73	75	69	73	66	92	664	-727
9	72	77	84	76	87	71	81	84	65	696	-460
Total	650	688	751	674	795	675	696	665	749	6341	
Gap	-1081	-632	-416	-780	-293	-913	-592	-666	-330		

TABLE IX
AVERAGE O-D MATRIX FOR US 1965-1970 MIGRATION FLOW: LOG_ PUR (100 EPOCHS)

Origin	Destination									Total	Gap
	1	2	3	4	5	6	7	8	9		
1	0	11	7	1	15	1	3	1	9	48	13
2	20	1	32	8	54	9	15	9	22	170	-11
3	6	20	0	25	45	19	18	10	32	175	-1
4	3	9	28	1	15	8	18	10	20	112	-14
5	8	21	24	8	0	22	14	8	22	128	11
6	3	7	24	7	28	1	15	6	8	97	-21
7	2	9	18	13	18	11	0	7	20	98	-5
8	2	5	10	6	9	3	7	0	28	69	14
9	6	11	19	9	22	4	15	18	-1	102	18
Total	51	92	161	78	207	79	103	69	159	998	
Gap	8	2	-11	-2	-2	-19	-3	21	9		

IV. Conclusion

It can be summarized that log_pur sub-scenario appears superior to the other sub-scenarios. Although it has a weakness which is its negative estimations, its performance is slightly better than log_log and much better than log_tan sub-scenarios. The performances of log_log and log_tan sub-scenarios diminish due to some factors, namely (1) Unsuccessful training, and (2) Negative estimations (for log_tan sub-scenario only). Increasing the epoch number cannot solve the failed training; however, it helps in decreasing the incidence of negative estimations.

Another problem faced by all the sub-scenarios and affected their performance is that the results are based on invalidated models, and hence different results may be attained when tested by using new datasets. The gaps between real and estimated row and column totals are still high, from -1081 to +31 trips. Fixing this difference should improve the calibration performance of neural models and its testing performance, which has yet to be considered.

Nomenclature

A	Trip Attraction	Trip
D	Deterrence factor	Km
LM	Levenberg-Marquard	
MLFFNN	Multilayer Feedforward Neural Network	
ANN	Artificial Neural Network	
P	Trip Production	Trip
r	Correlation coefficient	
RMSE	Root Mean Square Error	Trip
T_{hd}	Estimated trip number	Trip
t_{hd}	Observed trip number	Trip
w	Connection weight	
Obs	Observed trip number	Trip

Subscripts

i	input layer
j	hidden layer
k	output layer

References

[1] S. T. Skias, *Methods and procedures for the verification and validation of artificial neural networks*. New York, NY: Springer Science+Business Media, 2006.

[2] W. Chen, *et al.*, "Using a Single Dendritic Neuron to Forecast Tourist Arrivals to Japan," *IEICE Transactions on Information and Systems,* vol. E100.D, pp. 190-202, 2017.

[3] K. Kumar, *et al.*, "Short term traffic flow prediction in heterogeneous condition using artificial neural network," *Transport,* vol. 30, pp. 397-405, 2015/10/02 2015.

[4] R. Abdul Jabbar and H. Dia, "Predictive Intelligence: A Neural Network Learning System for Traffic Condition Prediction and Monitoring on Freeways," *Journal of the Eastern Asia Society for Transportation Studies,* vol. 13, pp. 1785-1800, 2019.

[5] W. Guangxing and K. Jiwon, "A Large Scale Neural Network Model for Crash Prediction in Urban Road Networks " in *Australasian Transport Research Forum,* Auckland, New Zealand, 2017.

[6] W. R. Black, "Spatial interaction modeling using artificial neural networks," *Journal of Transport Geography,* vol. 3, pp. 159-166, 1995.

[7] M. Dougherty, "A review of neural networks applied to transport," *Transportation Research Part C: Emerging Technologies,* vol. 3, pp. 247-260, 1995.

[8] H. Wang, *et al.*, "Detecting Transportation Modes Using Deep Neural Network," *IEICE Transactions on Information and Systems,* vol. E100.D, pp. 1132-1135, 2017.

[9] Y.-J. Byon, *et al.*, "Real-Time Transportation Mode Identification Using Artificial Neural Networks Enhanced with Mode Availability Layers: A Case Study in Dubai," *Appled Sciences,* vol. 7, 2017.

[10] O. Lozhkina, *et al.*, "Motor transport related harmful PM2.5 and PM10: from onroad measurements to the modelling of air pollution by neural network approach on street and urban level," *Journal of Physics: Conference Series,* vol. 772, p. 012031, 2016/11 2016.

[11] G. Yaldi, "Improving the Neural Network Testing Performance for Trip Distribution Modelling by Transforming Normalized Data Non-linearly," *International Journal on Advanced Science, Engineering and Information Technology,* vol. 7, 2017.

[12] G. Yaldi, "Analysing the Behaviour and Performance of Neural Network Trip Distribution Models toward Different Hidden Layer and Node Numbers," presented at the The 11th International Conference of Eastern Asia Society for Transportation Studies, Cebu, Philippines, 2015.

[13] F. Moretti, *et al.*, "Urban traffic flow forecasting through statistical and neural network bagging ensemble hybrid modeling," *Neurocomputing,* vol. 167, pp. 3-7, 2015/11/01/ 2015.

[14] D. Teodorovic and K. Vukadinovic, *Traffic Control and Transport Planning: A Fuzzy Sets and Neural Networks Approach.* Massachusetts, USA: Kluwer Academic Publisher, 1998.

[15] M. Mozolin, *et al.*, "Trip distribution forecasting with multilayer perceptron neural networks: A critical evaluation," *Transportation Research Part B: Methodological,* vol. 34, pp. 53-73, 2000.

[16] A. Dantas, *et al.*, "Neural network for travel demand forecast using GIS and remote sensing," in *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, 2000, pp. 435-440 vol.4.

[17] G. Yaldi, *et al.*, "Improving Artificial Neural Network Performance in Calibrating Doubly-Constrained Work Trip Distribution by Using a Simple Data Normalization and Linear Activation Function," in *Paper of The 32 Australasian Transportation Research Forum,* Auckland, New Zealand. Available at www.patrec.org/atrf.aspx, 2009.

[18] B. M. Wilamowski, *et al.*, "An Algorithm for Fast Convergence in Training Neural Networks," *IEEE,* vol. 3, pp. 1778-1782, 2001.

[19] G. Zhang, *et al.*, "Forecasting with artificial neural networks:: The state of the art," *International Journal of Forecasting,* vol. 14, pp. 35-62, 1998.