# Performance of Artificial Neural Network Using Heterogeneous Transfer Functions

Tayo P. Ogundunmade [a,b,1,*], Adedayo A. Adepoju [c,2]

[a] Research Exchange Scholar, Academy Mobility Project, University of Tlemcen, Algeria
[b] Laboratory for Interdisciplinary Statistical Analysis (UI-LISA), Department of Statistics, University of Ibadan, Ibadan, Nigeria
[c] Department of Statistics, University of Ibadan, Ibadan, Nigeria
[1] ogundunmadetayo@yahoo.com; [2]pojuday@yahoo.com
* corresponding author

ARTICLE INFO

**Article history**
Received August 21, 2021
Revised October 7, 2021
Accepted November 15, 2021

**Keywords**
activation functions
artificial neural network
multi-layer perceptron
mean square error
mean absolute error

ABSTRACT

Neural networks have been very important models across computer vision, natural language processing, speech and image recognition, aircraft safety and many more. It uses a variety of architectures that centres on the Multi-Layer Perceptron (MLP) which is the most commonly used type of Artificial Neural Network. MLP has been found to be good in terms of model precision in the usage of Homogenous Transfer/activation Functions (HTFs), especially with large data set. Based on the preliminary investigations of ranking of transfer functions by error variance (Udomboso, 2014), three HTFs are considered to perform better than other HTFs in prediction. These HTFs are the Hyperbolic Tangent Transfer functions (TANH), Hyperbolic Tangent Sigmoid Transfer function (TANSIG) and the Symmetric Saturating Linear Transfer Function (SSLTF). In this work, the performance of two Heterogeneous Transfer Functions (HETFs), which came as a result of the convolution of the three best HTFs, were compared with the performance of the three above listed HTFs. The hidden neurons used are 2, 5 and 10, while the sample sizes include 50, 100, 200, 500 and 1000. The data were divided into training sets of 90, 80 and 70 respectively. The results showed that the HETFs performed better in terms of the forecast using Mean Square Error (MSE), Mean Absolute Error (MAE) and Test Error as the forecast prediction criteria.



## 1. Introduction

Deep neural networks have bested notable benchmarks across computer vision, reinforcement learning, speech recognition, and natural language processing. However, neural networks still have deficiencies. For instance, they have a penchant to over-fit, and large data sets and careful regularization are needed to combat this tendency. Artificial Neural Networks use a variety of architectures. This study centers on the Multi-Layer Perceptron (MLP) which is the most commonly used type of ANN. The MLP is also known as the Feed-Forward Network (FFN). MLP has been found to be powerful in terms of model precision in the usage of homogeneous transfer functions (TFs), especially with complex or large data set. The choice of MLP is because it is the only ANN type that allows for statistical inference.

Blundell, C. et al (2015) introduced a new, efficient, principled and back propagation-compatible algorithm for learning distribution, a probability on the weights of a neural network, called Bayes by Back prop. Lee et al.(2017) derived the exact equivalence between infinitely wide deep networks and GPs. Matthews et al. , (2018) studied the relationship between random, wide, fully

 ISSN 2722-2039


connected, feed forward networks with more than one hidden layer and Gaussian processes with a recursive kernel definition.

Garriga-Alonso et al. (2019) showed that the output of a (residual) CNN with an appropriate prior over the weights and biases is a GP in the limit of infinitely many convolutional filters for dense networks. Laurence Aitchison (2020) They argued that getting Bayesian neural networks to perform comparably to artificially reduce uncertainty using a ”tempered” or ”cold” posterior. This is extremely concerning if the prior is accurate.

Wenzel et al.(2020) demonstrated through careful MCMC sampling that the posterior predictive induced by the Bayes posterior yields systematically worse predictions compared to simpler methods including point estimates obtained from SGD. Dusenberry et al. (2020) proposed a rank-1 parameterization of BNNs, where each weight matrix involves only a distribution on a rank-1 subspace. Garriga-Alonso et al. (2021) Correlating the weights maintains the correlations in the activations. Varying the amount of correlation interpolates between independent-weight limits and mean-pooling. Hafner, D. etal (2020) proposed noise contrastive priors (NCPs) to obtain reliable uncertainty estimates. The key idea is to train the model to output high uncertainty for data points outside of the training distribution. Tim Pearce et al.(2020) designed a simple, flexible approach to creating expressive priors in Gaussian process (GP) models makes new kernels from a combination of basic kernels, e.g. summing a periodic and linear kernel can capture seasonal variation with a long term trend.

Cui et al. (2021) proposed a new joint prior over the local (i.e., feature-specific) scale parameters that encodes knowledge about feature sparsity, and a Stein gradient optimization to tune the hyper parameters in such a way that the distribution induced on the model’s PVE matches the prior distribution. Vincent Fortuin et al. (2021) studied summary statistics of neural network weights in different networks trained using SGD. They found that fully connected networks (FCNNs) display heavy tailed weight distributions, while convolutional neural network (CNN) weights display strong spatial correlations.

The aim of this study is to compare the performance of ANN using heterogeneous transfer functions and homogenous transfer Functions. The rest of this paper are the methodology, data simulation for the study, presentation of result and conclusion.

## 2. Methodology

### 2.1. Artificial Neural Networks (ANNs)

As simple statistical models, ANNs have been useful to many functions, such as forecasting, curve-fitting, and regression in the fields of engineering, earth sciences, medicine, hydrology, etc. ANN models study data and carry out jobs such as classification or forecasting. The nature of the data is used to assess the network model in the building procedure, unlike other models that use before postulations. ANN arrangements are structured in levels positioned as input, hidden, and output levels. Within every level, there are interconnected elements known as neurons. Weights are the essential variables of the ANN models used to resolve a hitch. The total of the weighted inputs and the bias terms are entered into an activation function that is executed to avert the output from getting bigger. Frequently executed sets of activation functions include the sigmoid, hyperbolic tangent, and the rectified linear unit (ReLU) functions.

The statistics neural network model is given as

$$y = f(x) + ei = \sum x_i w_i + ei \quad (1)$$

$$\text{Output} = \text{sum (weights*inputs)} + \text{bias} \quad (2)$$

ISSN 2722-2039


Where y is the dependent variable, $x_i = (x_0 = 1, x_1, \ldots, x_n)$ is a vector of independent variables, where w = is the network weight and ei = is the stochastic term that is normally distributed (that is, e ~ N(0,$\sigma^2 I$)).

## 2.2. ANN Model Development

This study utilized a multilayer perceptron (MLP) feed-forward network. The multilayer perceptron reduced the error between the ANN model outputs and observed values by renewing the weights between each node. The choice of the hidden nodes in the complicated area in ANN modelling. To date, there are no precise strategies for matters such as how many hidden layers and hidden nodes should be integrated into an ANN model. Thus, a trial-and-error method was utilized to find the best number of nodes for the hidden layer. In this study, the data was split into training and test sets, training set and testing set (70%, 80%, 90%), hidden layers (2,5,10) and activation functions (sigmoid, hyperbolic tangent and rectified linear unit). Thus, the results obtained in the results section are the estimations of the performance of the ANN on the test data. All input data are normalized using the following equations:

$$Y_i = \frac{X_{oi}}{X_{o\,max}}, X_{oi} \geq 0 \quad (3)$$

$$Y_i = \frac{X_{oi}}{|X_{o\,min}|}, X_{oi} < 0 \quad (4)$$

Where $X_{oi}$ is the observed value, $X_{o\,min}$ and $X_{o\,max}$ are respectively the minimum and maximum data in the input time series.

## 2.3. Activation functions

The perception of the handling of neural networks is mainly attained through the activation functions. An activation function is a mathematical function that changes the input variable to an output variable. In default of activation functions, the operation of neural networks will be similar to linear functions. A linear function is a function where the output variable is exactly related to the input variable.

Nevertheless, most of the limitations the neural networks try to unravel are nonlinear and complicated. The activation functions are utilized to attain the nonlinearity. Nonlinear functions are high-level polynomial functions. The graph of a nonlinear function is curved and combines the complication element. Activation functions provide the nonlinearity element to neural networks and render them accurate universal function approximations.

### 2.3.1. Sigmoid

The sigmoid function is a mathematical function that gives a sigmoidal curve; a characteristic curve for its S shape. This is the oldest and frequently used activation function. This compresses the input to any value between 0 and 1 and makes the model logistic. This function is known as a special case of logistic function defined by the following formula:

$$f(x) = 1/1(1 + e^{-x}) \quad (5)$$

### 2.3.2. Hyperbolic tangent

Another common and mostly utilized activation function is the tanh function. This is a nonlinear function, characterized in the scale of values (-1, 1). One thing to make clear is that the gradient is

better for tanh than sigmoid (the derivatives are steeper). Settling between sigmoid and tanh will be based on the gradient strength prerequisite. Like the sigmoid, tanh also has the missing slope constraint. The function is specified by the formula:

$$f(x) = \tanh(x) \tag{6}$$

This looks like sigmoid; it is a scaled sigmoid function.

### 2.3.3. Rectified Linear Unit

Rectified Linear Unit (ReLU) is a predominantly utilized activation function. It is a simple specification and has merits over the other functions. The function is defined by the following formula:

$$f(x) = 0 \text{ when } x < 0$$

$$x \text{ when } x >= 0 \tag{7}$$

The scale of the result is between 0 and infinity. RELU finds usage in computer vision and speech identification using deep neural networks.

## 2.4. Artificial Neural Network with Heterogeneous Transfer Function

The hardware was provided for the training process depicted in table 1. GPU was necessary in the training process to speed up the time process for eye modelling. Tensorflow-GPU version 1.12 was selected to handle the training process with Keras support. Beside the Tensorflow-GPU, OpenCV was also installed to run the model in real time application using webcam.

The model below gives a neural network model with a homogenous transfer function

$$F(x, w) = \alpha X + \sum_{h=1}^{H} \beta_h [\, g(\sum_{i=0}^{I} \gamma_{hi} X_i)] \tag{8}$$

Where g (.) is the transfer functions, which makes the equation (2.8) above is called an Homogenous SNN (HSNN) model. Given a convoluted form of the artificial neural network model given above, using the product convolution, we have

$$F(x, w) = \alpha X + \sum_{h=1}^{H} \beta_h [\, g_1(\sum_{i=0}^{I} \gamma_{hi} X_i) g_2(\sum_{i=0}^{I} \gamma_{hi} X_i)] \tag{9}$$

Where $g_1(.)$ and $g_2(.)$ are transfer functions, which are HTFs but combined in equation 3 above to make a heterogeneous Transfer function (HETFs). Equation (2.9) above is called the Heterogeneous SNN (HETSNN) model.

## 2.5. Heterogeneous Transfer Functions (HETFs)

Based on the above listed best HTFs, two convoluted HETFs were derived using the principle of convolution i.e. $g_1(.)$ x $g_2(.)$ such that the newly derived transfer functions are also a probability density function. These two HETFs below are derived using the convolution of Symmetric Saturating Linear Transfer Function and the Hyperbolic Tangent Transfer Function (SSLHT) and the convolution of the Symmetric Saturating Linear Transfer Function and the Hyperbolic Tangent Sigmoid Transfer Function (SSLHTS) (Udomboso, 2014).

The summary of the derived function is given as:
(1) Symmetric Saturating Linear and Hyperbolic Tangent (SSLHT)

$$g(x) = (x+1)\log(e^{x} + e^{-1})^{-1} \quad \text{For} \quad -1 < x < 1$$

ISSN 2722-2039


(2) Symmetric Saturating Linear and Hyperbolic Tangent Sigmoid (SSLHTS)

$$g(x) = (2x^2 + 3x + \frac{1}{2}) - x(\sum \frac{e^{-2ipx}}{p} - \sum \frac{e^{2ipx}}{p})^1 \quad \text{For} \quad -1 < x < 1$$

Where p is the number of parameters.

### 2.6. Data Simulation for the Study

The data to be used for this study was generated using the model below:

$$y = x + 0.3\sin(2\pi(x + ei)) + ei \tag{10}$$

Where ei ~ N(0,0.02) and x ~N(0.1).

The results are based on the prediction and model selection criterion given at different levels of hidden neurons at different sample sizes. The hidden neurons used are 2, 5 and 10, while the sample sizes include 50, 100, 200, 500 and 1000. The data was also divided into training and testing sets of 90 and 10, 80 and 20 and 70 and 30 respectively.

### 2.7. Prediction Selection Criteria

The prediction selection criteria used in this work are the Mean Square Error (MSE), Mean Absolute Error (MAE) and the Test Error. Mean square error measures the average of the squares of the errors or the average squared between the estimated values and the actual value. Mathematically, it can be represented as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Yi - \hat{Y}i)^2$$

Mean Absolute Error (MAE) is a measure of prediction accuracy. It is the measure of absolute error between the forecast value and the true value. Mathematically, it can be written as:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|Yi - \hat{Y}i|^2$$

Test Error is used when a model is to be validated. When we calculate the error on data which was unknown in the training phase, we are calculating the test error.

## 3. Results and Discussion

This present the analyses of the performance of ANN using homogenous transfer functions and heterogeneous transfer functions. Tables 1 to 3 below shows the forecast performance measures results for the simulated data using the mean square error, mean absolute error and the test error respectively. The tables show the performance of the transfer functions (HTFs and HETFs) under different training set numbers (70%, 80% and 90%) and under different hidden neurons (2, 5, and 10) under different activation functions and at different sample sizes. The results obtained from Tables 1 to 3 reflects the performance of the activation functions, for three the homogenous transfer functions and two heterogeneous transfer function. Considering the activation function, RELU produced majority of the lowest mean square errors (MSEs), mean absolute error (MAEs) and the test error among the homogenous transfer functions while the two heterogeneous transfer functions produced lowest mean square error, mean absolute error and the test error compared to the homogenous transfer functions considered which make them better in prediction. It can also be seen from the results that, as the sample size increases, the value of the mean square error decreases.

ISSN 2722-2039


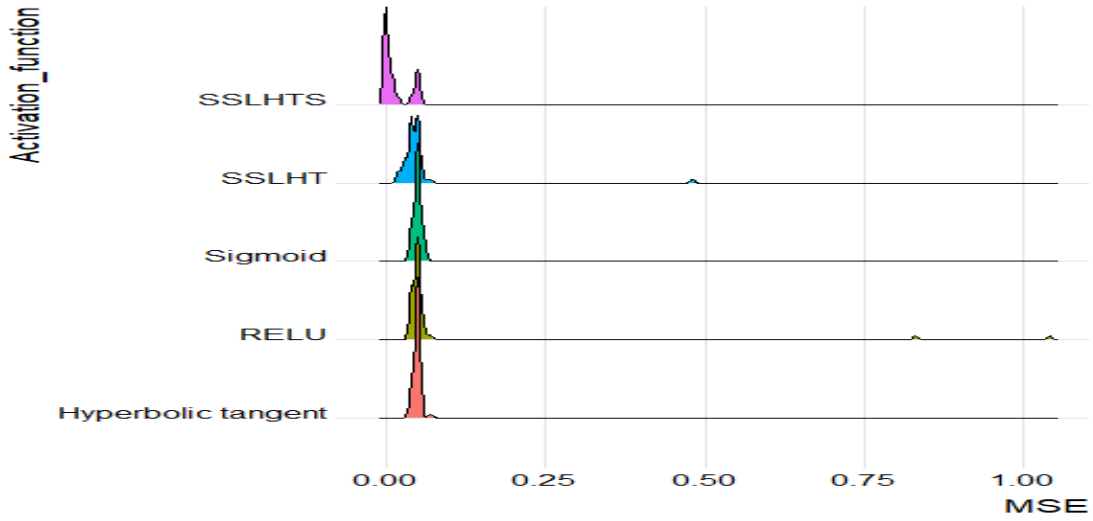


**Fig. 1** Distribution of performance of the activation functions using MSE

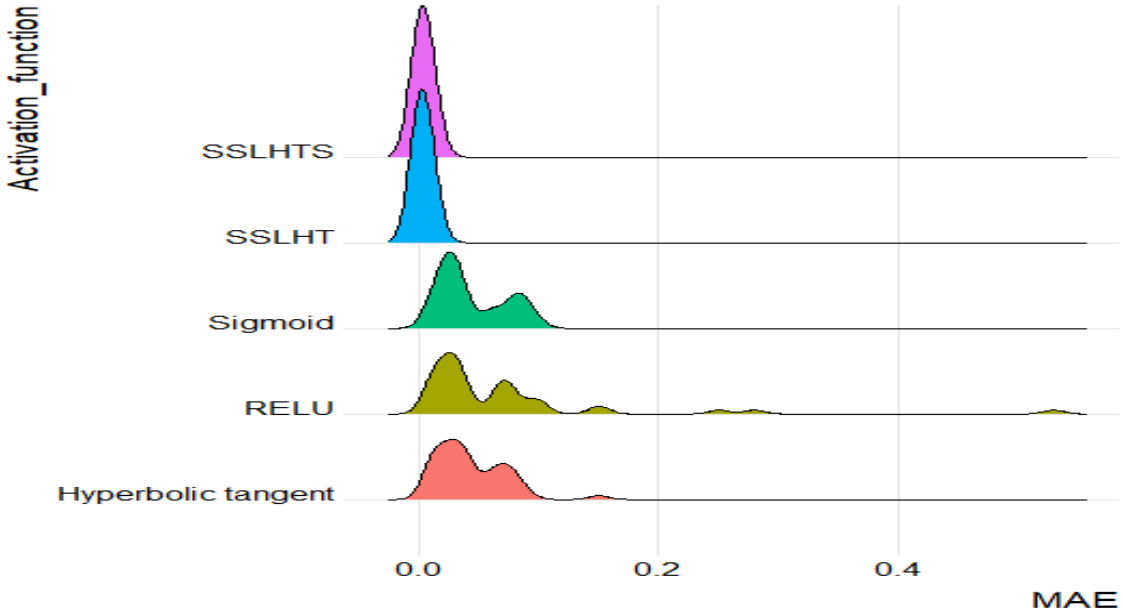


**Fig. 2** Distribution of performance of the activation functions using MAE

ISSN 2722-2039


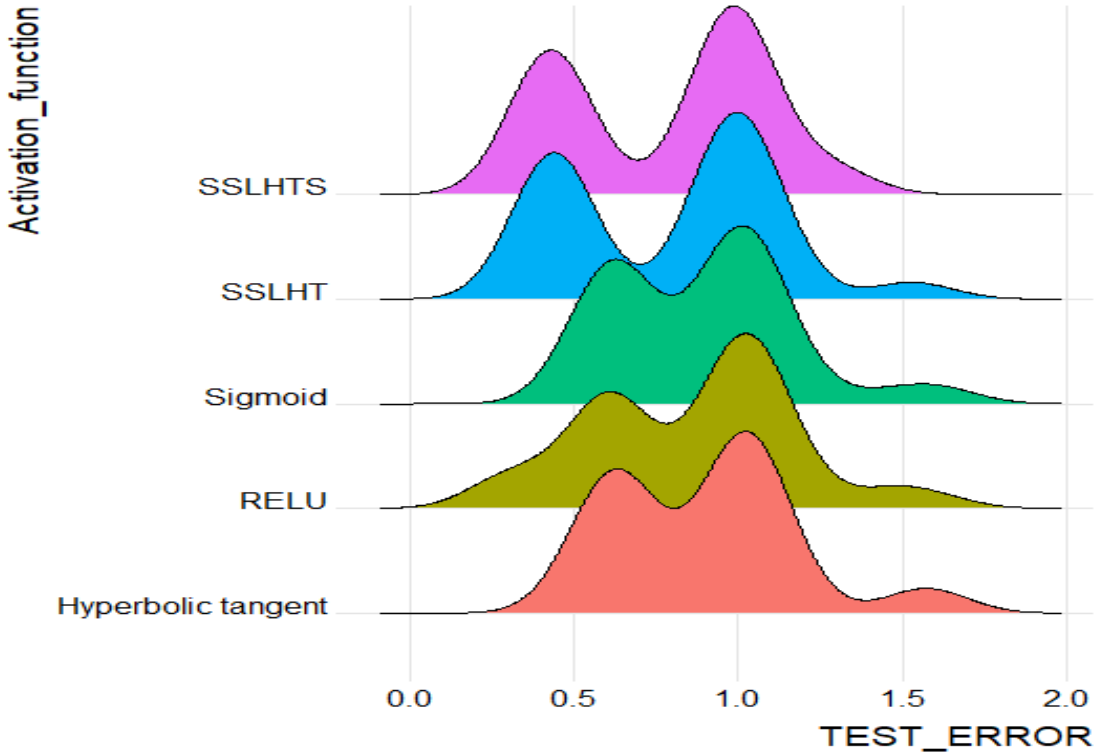


**Fig. 3** Distribution of performance of the activation functions using Test Error

**Table 1.** Forecast Performance using MSE

| | Activation functions | Training:70%; Testing:30% | | | Training:80%; Testing:20% | | | Training:90%; Testing:10% | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) |
| When n=50 | RELU | 0.04536065 | 0.04564356 | 0.0492185 | 0.04089795 | 0.04824148 | 0.04872315 | 0.04601659 | 0.04480303 | 0.04618498 |
| | Sigmoid | 0.05656073 | 0.05022231 | 0.04483313 | 0.05572514 | 0.0575734 | 0.05603065 | 0.06303656 | 0.05302699 | 0.05153979 |
| | Hyperbolic tangent | 0.0484365 | 0.0480776 | 0.0451661 | 0.04988979 | 0.05074671 | 0.05414198 | 0.04942139 | 0.05035907 | 0.04780267 |
| | SSLHT | 0.03796254 | 0.03120936 | 0.03477873 | 0.03650677 | 0.03219296 | 0.02039492 | 0.0331542 | 0.03995348 | 0.03773181 |
| | SSLHTS | 0.04369801 | 0.04369801 | 0.0140745 | 0.00898904 | 0.00898904 | 0.00898904 | 0.00701129 | 0.00701129 | 0.00701129 |
| | | Training:70%; Testing:30% | | | Training:80%; Testing:20% | | | Training:90%; Testing:10% | | |
| | Activation functions | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) |
| When n=100 | RELU | 0.0405356 | 0.04843695 | 0.04852811 | 0.0550651 | 0.05122153 | 0.0512218 | 1.037169 | 0.04466014 | 0.04486319 |
| | sigmoid | 0.04143232 | 0.04887886 | 0.0481793 | 0.06226006 | 0.04868746 | 0.05041625 | 0.04517857 | 0.04503382 | 0.0424108 |
| | Hyperbolic tangent | 0.04801389 | 0.04907249 | 0.04781495 | 0.05205736 | 0.04831357 | 0.05047595 | 0.04772446 | 0.0495061 | 0.04437785 |
| | SSLHT | 0.4825532 | 0.04549503 | 0.04182575 | 0.04366129 | 0.04229004 | 0.04124163 | 0.04100606 | 0.04520576 | 0.0237550 |
| | SSLHTS | 0.00137821 | 0.00137821 | 0.00137821 | 0.00139706 | 0.00139706 | 0.00139706 | 0.001655153 | 0.00165515 | 0.00165515 |
| | | Training:70%; Testing:30% | | | Training:80%; Testing:20% | | | Training:90%; Testing:10% | | |
| | Activation functions | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) |
| When n=200 | RELU | 0.04283649 | 0.04282593 | 0.0428457 | 0.0461478 | 0.04095318 | 0.04098674 | 0.8261478 | 0.04095318 | 0.04098674 |
| | Sigmoid | 0.04592385 | 0.04522716 | 0.04638254 | 0.04211514 | 0.04236094 | 0.04076459 | 0.03885812 | 0.03815169 | 0.03722988 |
| | Hyperbolic tangent | 0.04251323 | 0.04545386 | 0.04758545 | 0.04185362 | 0.03908446 | 0.04039074 | 0.03795205 | 0.03717498 | 0.03710731 |
| | SSLHT | 0.05345796 | 0.04138555 | 0.04157519 | 0.04254912 | 0.03990308 | 0.0430479 | 0.0308365 | 0.0310234 | 0.0374535 |
| | SSLHTS | 0.00192205 | 0.00192205 | 0.00192205 | 0.00163919 | 0.00163919 | 0.00163919 | 0.00205974 | 0.02059741 | 0.02059741 |
| | | Training:70%; Testing:30% | | | Training:80%; Testing:20% | | | Training:90%; Testing:10% | | |
| | Activation functions | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) |
| When n=500 | RELU | 0.04226684 | 0.04706437 | 0.04697518 | 0.05910555 | 0.04706283 | 0.05580163 | 0.05096383 | 0.0682076 | 0.0509657 |
| | sigmoid | 0.04756158 | 0.04742588 | 0.04681848 | 0.05169109 | 0.04871218 | 0.05151577 | 0.05112607 | 0.05182675 | 0.05082465 |
| | Hyperbolic tangent | 0.04882288 | 0.04756547 | 0.04841315 | 0.05337090 | 0.05033169 | 0.0527485 | 0.05185673 | 0.0688195 | 0.05189598 |
| | SSLHT | 0.05317327 | 0.04940695 | 0.06562853 | 0.05453931 | 0.0423414 | 0.04676842 | 0.0483453 | 0.06449127 | 0.05288543 |
| | SSLHTS | 0.003122762 | 0.003122762 | 0.003122762 | 0.003649035 | 0.003649035 | 0.003649035 | 0.003565195 | 0.003565195 | 0.003565195 |
| | | Training:70%; Testing:30% | | | Training:80%; Testing:20% | | | Training:90%; Testing:10% | | |
| | Activation functions | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) |
| When n=1000 | RELU | 0.04737156 | 0.0473454 | 0.04733953 | 0.04664931 | 0.04664866 | 0.0466491 | 0.05072219 | 0.05069983 | 0.05072245 |
| | sigmoid | 0.04970722 | 0.04864827 | 0.04884184 | 0.04895137 | 0.04704777 | 0.04681032 | 0.05151714 | 0.05199655 | 0.05220437 |
| | Hyperbolic tangent | 0.0479398 | 0.04760941 | 0.04896131 | 0.04791326 | 0.04817565 | 0.04765268 | 0.05220662 | 0.05257291 | 0.05031869 |
| | SSLHT | 0.0479398 | 0.04733635 | 0.04733035 | 0.04736291 | 0.04633459 | 0.04850382 | 0.02210718 | 0.05062815 | 0.05180198 |
| | SSLHTS | 0.04789064 | 0.04789064 | 0.04789064 | 0.04738028 | 0.04738028 | 0.04738028 | 0.05180198 | 0.05180198 | 0.05180198 |

**Table 2.** Forecast Performance using MAE

| | Activation functions | Training:70%; Testing:30% | | | Training:80%; Testing:20% | | | Training:90%; Testing:10% | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) |
| When n=50 | RELU | 0.01084481 | 0.01583126 | 0.03362715 | 0.01033206 | 0.01431494 | 0.01230493 | 0.07415554 | 0.05585283 | 0.06902516 |
| | sigmoid | 0.01913261 | 0.05988184 | 0.01294782 | 0.02172133 | 0.01963479 | 0.02395511 | 0.08484791 | 0.05532735 | 0.05746984 |
| | Hyperbolic tangent | 0.02443442 | 0.01557737 | 0.0158182 | 0.08705485 | 0.0137796 | 0.01161389 | 0.06070249 | 0.05248329 | 0.05708698 |
| | SSLHT | 0.00318743 | 0.004393331 | 0.003620382 | 0.003599549 | 0.003123241 | 0.004051393 | 0.001471838 | 0.006308208 | 0.006316871 |
| | SSLHTS | 0.001589913 | 0.001589913 | 0.001782391 | 0.001148536 | 0.001148536 | 0.001148536 | 0.00983228 | 0.00983228 | 0.00983228 |
| | Activation functions | Training:70%; Testing:30% | | | Training:80%; Testing:20% | | | Training:90%; Testing:10% | | |
| | | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) |
| When n=100 | RELU | 0.279912 | 0.02808561 | 0.02877417 | 0.2503624 | 0.07158606 | 0.07178141 | 0.5304296 | 0.07728433 | 0.07624355 |
| | sigmoid | 0.02858596 | 0.03014854 | 0.02917777 | 0.07906109 | 0.07510564 | 0.07556928 | 0.08843629 | 0.09180876 | 0.09133624 |
| | Hyperbolic tangent | 0.02860824 | 0.02876265 | 0.02917051 | 0.07348370 | 0.07439191 | 0.07323994 | 0.07688096 | 0.06795592 | 0.0780563 |
| | SSLHT | 0.002428043 | 0.006019314 | 0.005896373 | 0.007284321 | 0.006863603 | 0.007715116 | 0.005123343 | 0.00779468 | 0.001477266 |
| | SSLHTS | 0.002834303 | 0.002834303 | 0.002834303 | 0.001414445 | 0.001414445 | 0.001414445 | 0.005339173 | 0.005339173 | 0.005339173 |
| | Activation functions | Training:70%; Testing:30% | | | Training:80%; Testing:20% | | | Training:90%; Testing:10% | | |
| | | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) |
| When n=200 | RELU | 0.01555635 | 0.01592215 | 0.01561166 | 0.1504472 | 0.03425668 | 0.03454515 | 0.1504472 | 0.03425668 | 0.03454515 |
| | sigmoid | 0.01506126 | 0.01600192 | 0.01924248 | 0.03271113 | 0.03198777 | 0.04034036 | 0.02779946 | 0.02715333 | 0.02726782 |
| | Hyperbolic tangent | 0.01227057 | 0.01887132 | 0.02146642 | 0.03277947 | 0.03601551 | 0.03803863 | 0.03116556 | 0.02906871 | 0.02816523 |
| | SSLHT | 0.00262705 | 0.00222336 | 0.00252556 | 0.00492054 | 0.00376587 | 0.00301193 | 0.00125838 | 0.00355436 | 0.00288313 |
| | SSLHTS | 0.00341757 | 0.00341757 | 0.00341757 | 0.00352640 | 0.00352640 | 0.00352640 | 0.00355436 | 0.00355436 | 0.00355436 |
| | Activation functions | Training:70%; Testing:30% | | | Training:80%; Testing:20% | | | Training:90%; Testing:10% | | |
| | | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) |
| When n=500 | RELU | 0.07049311 | 0.02869529 | 0.02942641 | 0.03408281 | 0.01640909 | 0.09467976 | 0.07084378 | 0.01809918 | 0.07092672 |
| | sigmoid | 0.07970243 | 0.08534412 | 0.05957753 | 0.03398892 | 0.02442058 | 0.03394872 | 0.02853253 | 0.007577316 | 0.06930484 |
| | Hyperbolic tangent | 0.03722658 | 0.01050819 | 0.01255912 | 0.15214931 | 0.02551606 | 0.04395881 | 0.05523077 | 0.02457272 | 0.05645807 |
| | SSLHT | 0.008363895 | 0.002497509 | 0.001445503 | 0.002883166 | 0.001324457 | 0.004179057 | 0.00213211 | 0.002030934 | 0.00929881 |
| | SSLHTS | 0.008970462 | 0.008970462 | 0.008970462 | 0.008247808 | 0.008247808 | 0.008247808 | 0.00931740 | 0.009317405 | 0.009317405 |
| | Activation functions | Training:70%; Testing:30% | | | Training:80%; Testing:20% | | | Training:90%; Testing:10% | | |
| | | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) |
| When n=1000 | RELU | 0.01391879 | 0.01362747 | 0.01378003 | 0.09982207 | 0.09948592 | 0.09971453 | 0.03724168 | 0.03702656 | 0.03723718 |
| | sigmoid | 0.08474242 | 0.01070048 | 0.01165833 | 0.02439061 | 0.01111752 | 0.09931213 | 0.03453425 | 0.03540605 | 0.03602913 |
| | Hyperbolic tangent | 0.01090115 | 0.01198944 | 0.01398887 | 0.07063271 | 0.07923062 | 0.08239547 | 0.03609895 | 0.03606323 | 0.03867038 |
| | SSLHT | 0.001090115 | 0.001779952 | 0.001684105 | 0.00201341 | 0.00131349 | 0.00526225 | 0.007921446 | 0.003855935 | 0.003636896 |
| | SSLHTS | 0.001346109 | 0.001346109 | 0.001346109 | 0.00109871 | 0.00109871 | 0.00109871 | 0.003636896 | 0.003636896 | 0.003636896 |

**Table 3.** Forecast Performance using Test Error

| | Activation functions | Training:70%; Testing:30% | | | Training:80%; Testing:20% | | | Training:90%; Testing:10% | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) |
| When n=50 | RELU | 1.004461 | 1.017161 | 1.193877 | 1.092693 | 1.067767 | 1.074861 | 1.455757 | 1.597596 | 1.473599 |
| | sigmoid | 0.9899312 | 1.012857 | 1.007863 | 1.1124 | 1.100785 | 1.112796 | 1.406555 | 1.609739 | 1.592005 |
| | Hyperbolic tangent | 1.031811 | 1.014247 | 1.022047 | 1.083076 | 1.08429 | 1.121298 | 1.551501 | 1.603778 | 1.57132 |
| | SSLHT | 0.9706997 | 0.9571166 | 1.070951 | 1.10896 | 1.015166 | 1.012562 | 1.107582 | 1.519653 | 1.540541 |
| | SSLHTS | 0.980448 | 0.931562 | 1.013156 | 1.018861 | 1.018861 | 1.018861 | 1.293583 | 1.293583 | 1.293583 |
| | Activation functions | Training:70%; Testing:30% | | | Training:80%; Testing:20% | | | Training:90%; Testing:10% | | |
| | | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) |
| When n=100 | RELU | 0.5154035 | 0.5654503 | 0.5688913 | 0.2807331 | 0.6337495 | 0.6333759 | 0.3413968 | 0.6446217 | 0.6470433 |
| | sigmoid | 0.5943388 | 0.560711 | 0.5434544 | 0.7121132 | 0.6041872 | 0.6278305 | 0.6355053 | 0.626227 | 0.6092434 |
| | Hyperbolic tangent | 0.5597554 | 0.5750345 | 0.5523297 | 0.6459385 | 0.5994198 | 0.631463 | 0.6690408 | 0.683816 | 0.642954 |
| | SSLHT | 0.4143074 | 0.4591691 | 0.4084604 | 0.4705492 | 0.4233827 | 0.4410327 | 0.4191503 | 0.4141392 | 0.4675494 |
| | SSLHTS | 0.4057916 | 0.4057916 | 0.4057916 | 0.4981263 | 0.4981263 | 0.4981263 | 0.4646266 | 0.4646266 | 0.4646266 |
| | Activation functions | Training:70%; Testing:30% | | | Training:80%; Testing:20% | | | Training:90%; Testing:10% | | |
| | | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) |
| When n=200 | RELU | 0.5873649 | 0.5881179 | 0.5869946 | 0.308537 | 0.634238 | 0.6340868 | 0.308537 | 0.634238 | 0.6340868 |
| | sigmoid | 0.5826101 | 0.5860443 | 0.5720085 | 0.6313711 | 0.6309563 | 0.6386879 | 0.6732878 | 0.6768626 | 0.667942 |
| | Hyperbolic tangent | 0.5970667 | 0.5748272 | 0.5638314 | 0.6322046 | 0.6488517 | 0.6433647 | 0.6707231 | 0.6702858 | 0.6681361 |
| | SSLHT | 0.4391822 | 0.4982035 | 0.4229245 | 0.4590161 | 0.4439311 | 0.4255294 | 0.4890826 | 0.4141392 | 0.4369202 |
| | SSLHTS | 0.4141392 | 0.4141392 | 0.4141392 | 0.4141392 | 0.4141392 | 0.4141392 | 0.4141392 | 0.4141392 | 0.4141392 |
| | Activation functions | Training:70%; Testing:30% | | | Training:80%; Testing:20% | | | Training:90%; Testing:10% | | |
| | | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) |
| When n=500 | RELU | 0.9587568 | 0.9567833 | 0.9596407 | 0.9546522 | 1.027125 | 1.069239 | 0.9498151 | 0.8412903 | 0.9498849 |
| | sigmoid | 0.9395106 | 0.9413349 | 0.9698294 | 0.99048 | 0.9874596 | 0.991759 | 0.9159227 | 0.8983348 | 0.9165466 |
| | Hyperbolic tangent | 0.9909502 | 0.9653519 | 0.9419297 | 0.9811419 | 0.9723048 | 1.071862 | 0.9039508 | 1.0275 | 0.9033646 |
| | SSLHT | 0.9336583 | 0.9750791 | 0.9021643 | 0.9437664 | 0.9324454 | 0.9579638 | 0.9532421 | 0.9018967 | 0.9726117 |
| | SSLHTS | 0.9235326 | 0.9141392 | 0.9324244 | 0.9354447 | 0.9363321 | 0.94321345 | 0.9454432 | 0.9466654 | 0.95443211 |
| | Activation functions | Training:70%; Testing:30% | | | Training:80%; Testing:20% | | | Training:90%; Testing:10% | | |
| | | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) | Hidden Neuron (2) | Hidden Neuron (5) | Hidden Neuron (10) |
| When n=1000 | RELU | 1.045718 | 1.044086 | 1.042721 | 1.097914 | 1.098102 | 1.097986 | 1.014376 | 1.012394 | 1.014541 |
| | sigmoid | 1.054239 | 1.05041 | 1.045192 | 1.110031 | 1.094073 | 1.096389 | 1.023284 | 1.01566 | 1.010466 |
| | Hyperbolic tangent | 1.053457 | 1.049495 | 1.035197 | 1.096626 | 1.090496 | 1.093852 | 1.010068 | 1.008129 | 1.006755 |
| | SSLHT | 1.023457 | 1.025666 | 1.030545 | 1.012341 | 1.089852 | 1.089165 | 1.007187 | 1.004144 | 1.001620 |
| | SSLHTS | 1.039659 | 1.039659 | 1.031659 | 1.044392 | 1.044392 | 1.054392 | 1.001621 | 1.001621 | 1.001621 |

## 4. Conclusion

In this study, Mean Square Error (MSE) was used to assess the performances of all ANN models. Conclusively, RELU produced majority of the lowest mean square errors (MSEs) across the sample sizes. Also, as the training percentage increases, the mean square error increases in most cases. The performance of the heterogeneous transfer functions considered have been good in terms of prediction measures has they produced lowest mean square error in almost all the cases of training sets and at different level of sample sizes considered.

Neural network model is one of the so called important models used is data science for pattern and image recognition, computer vision and so on. Without the use of activation functions, neural network cannot be used because it is the main functional part of the model. In previous studies, homogenous transfer functions have been used in predictions in most of these areas mentioned above. With the result obtained from this study, it is recommended that heterogeneous transfer functions for neural network models should be considered for neural network models. The forecast performance of the heterogeneous transfer functions in this study has shown that, if used in neural network models for the aforementioned areas of research and many more, better results will be attained.

## Acknowledgements

We acknowledgement the active participation of the authors to the success of the research paper.

## References

[1] Aitchison, L. (2020) A statistical theory of cold posteriors in deep neural networks. arXiv preprint arXiv:2008.05912.

[2] Aitchison, L., Yang, A. X., and Ober, S. W. (2020) Deep kernel processes. arXiv preprint arXiv:2010.01590.

[3] Bayes, T. An essay towards solving a problem in the doctrine of chances. Philosophical transactions of the Royal Society of London, 53:370–418, 1763. By the late Rev. Mr. Bayes, FRS communicated by Mr. Price, in a letter to John Canton, AMFRS.

[4] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015) Weight uncertainty in neural networks. arXiv preprint arXiv:1505.05424.

[5] Christopher Godwin Udomboso (2013) On Some Properties of a Heterogeneous Transfer Function Involving Symmetric Saturated Linear (SATLINS) with Hyperbolic Tangent (TANH)Transfer Functions .Journal of Modern Applied Statistical Methods. Volume 12, Issue 2, Article 26.

[6] Christopher Godwin Udomboso (2014) On the level of precision of an heterogeneous statistical neural network model. PhD thesis, Department of Statistics, University of Ibadan, Nigeria.

[7] Heek, J. and Kalchbrenner, N (2019). Bayesian inference for large scale image classification. arXiv preprint arXiv:1908.03491.

[8] Gauss, C. F. Theoria motvs corporvm coelestivm in sectionibvs conicis solem ambientivm. Sumtibus F. Perthes et IH Besser, 1809.

[9] Garriga-Alonso, A. and Fortuin, V. (2021). Exact Langevin dynamics with stochastic gradients. arXiv preprint arXiv:2102.01691

[10] Garriga-Alonso, A. and van der Wilk, M (2021). Correlated weights in infinite limits of deep convolutional neural networks. arXiv preprint arXiv:2101.04097.

[11] Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013) Bayesian data analysis. CRC press.

[12] Nalisnick, E. T.(2018) On priors for Bayesian neural networks. PhD thesis, UC Irvine.

[13] Neal, R. M.(1996) Bayesian learning for neural networks, volume 118. Springer.

[14] Student (1908) The probable error of a mean. Biometrika, pp. 1–25.

[15] Wenzel, F., Roth, K., Veeling, B. S., Swi atkowski, J., Tran, L., Mandt, S., Snoek, J., Salimans, T., Jenatton, R., and Nowozin, S(2020a). How good is the Bayes posterior in deep neural networks really? In International Conference on Machine Learning.

[16] Wilson, A. G. and Izmailov, P. (2020) Bayesian deep learning and a probabilistic perspective of generalization. arXiv preprint arXiv:2002.08791.

[17] Zhang, R., Li, C., Zhang, J., Chen, C., and Wilson, A. G. Cyclical stochastic gradient MCMC for Bayesian deep learning. arXiv preprint arXiv:1902.03932, 2019.