Final Project: Artificial Neural Network for predictive CFD

Victor Edske, Jose Moussa, Ramyar Tajik 12/09/2022

Professor: Dr. J. Lo

INTRODUCTION: BACKGROUND

An artificial neural network is a type of artificial intelligence modeled after the human brain's structure (almost!) (White 1963). It comprises many interconnected processing nodes, or "neurons," which work together to process information and make decisions.

The significance of artificial neural networks lies in their ability to perform complex tasks that are difficult or impossible for traditional computers to solve (Yegnanarayana 2009). For example, they are capable of recognizing patterns in data, such as images or sounds, and making predictions based on those patterns. This makes them particularly useful in fields such as image and speech recognition, natural language processing, and predictive analytics.

In the context of computational fluid dynamics (CFD) (Lira, Riella et al. 2022), artificial neural networks can be used to make predictions about the behavior of fluids in different situations, such as how they will flow around a particular object or how they will respond to changes in temperature or pressure. This can be particularly useful in the field of built environment, where the behavior of fluids can have a significant impact on the design and performance of buildings and other structures.

For example, artificial neural networks can be used to predict the airflow around a building and the resulting patterns of wind and temperature, which can help architects and engineers to design buildings that are more energy efficient and comfortable for occupants. They can also be used to predict the behavior of fluids in HVAC systems, which can help to optimize the design of these systems and improve their performance. Additionally, artificial neural networks can be used to simulate the flow of water in plumbing systems and predict the behavior of fluids in other applications, such as fire suppression systems and wastewater treatment facilities.

All in all, the use of artificial neural networks in computational fluid dynamics can improve the performance of built environment systems and make them more efficient, sustainable, and effective.

In addition to their ability to perform complex tasks, artificial neural networks have several other advantages over traditional computing methods. For instance, they are able to learn and adapt to new information over time (Hoskins and Himmelblau 1988), allowing them to improve their performance without the need for explicit programming. They are also highly parallel in nature, meaning that they can process large amounts of data simultaneously, making them well-suited to tasks that require rapid decision-making.

Despite these advantages, there are also some challenges associated with artificial neural networks. One of the main challenges is that they can be difficult to design and train, requiring a significant amount of time and expertise. Additionally, the inner workings of neural networks are often difficult to understand and interpret, making it difficult to know why a particular decision was made.

Overall, artificial neural networks are a powerful tool for solving complex problems in fields such as machine learning and artificial intelligence. Despite their challenges, they have the potential to revolutionize many areas of science and technology and are likely to play an increasingly important role in our daily lives in the years to come.

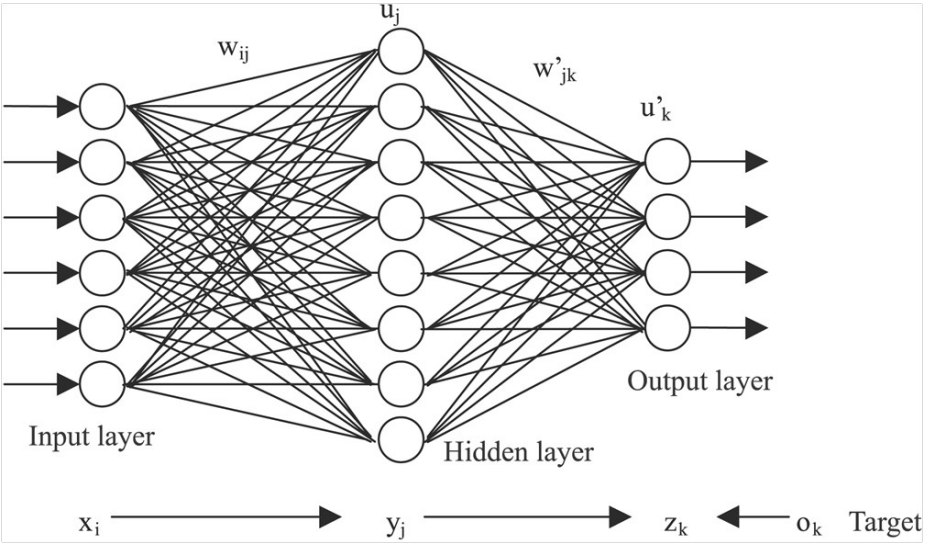


Figure 1 generic Artificial Neural Network architecture with one hidden layer

METHODOLOGY

An artificial neural network (ANN) is a type of machine learning algorithm that is well-suited for predicting the interior ambient temperature that the occupant experiences. The input layer of the ANN can receive data about the conditions inside and outside the building, such as the temperature and humidity levels, the amount of sunlight, and the presence of occupants. The hidden layers of the ANN process this data using nonlinear activation functions, such as the Sigmoid function, to learn the complex relationships between the input variables and the output variable (the interior ambient temperature). The output layer of the ANN produces a predicted value for the interior ambient temperature, which can be used to control the heating and cooling systems in the building.

The activation function is a crucial component of an artificial neural network (ANN) model (Sharma, Sharma et al. 2017). It is a nonlinear function that is applied to the output of each neuron in the network, and its purpose is to introduce nonlinearity into the model.

In the early days of neural networks, activation functions were mostly linear, which meant that the output of a neuron was simply a weighted sum of its inputs. However, it was quickly realized that using only linear activation functions limited the expressiveness of the network, making it difficult to model complex real-world data.

Nonlinear activation functions, such as the Sigmoid function and the hyperbolic tangent (tanh) function, were introduced to address this limitation. These functions take in a real-valued input and squashes it to a value between 0 and 1, or -1 and 1, respectively. This nonlinear transformation allows the neurons in the hidden layers of the network to learn complex nonlinear relationships in the data, which is essential for making accurate predictions.

The choice of activation function can have a significant impact on the performance of an ANN model. Different activation functions have different characteristics, such as the range of their output and the smoothness of their gradient. These characteristics can affect the speed and stability of the training process, as well as the final accuracy of the model.

For example, the Sigmoid activation function has the advantage of being easy to differentiate, which makes it suitable for use with optimization algorithms that rely on gradient descent. However, it can suffer from vanishing gradients, where the gradient becomes very small as the input to the function becomes large, which can slow down the training process. The tanh activation function, on the other hand, has a wider range of output values, which can make the training process faster, but its gradient is not as smooth, which can make it more difficult to optimize.

The Sigmoid activation function is particularly useful for predicting the interior ambient temperature because it allows the ANN to learn nonlinear relationships in the data. The interior ambient temperature is influenced by a wide range of factors, including the temperature and humidity levels, the amount of sunlight, and the presence of occupants. These factors can interact in complex ways, and a nonlinear activation function like the Sigmoid function is better equipped to capture these interactions than a linear activation function. This can improve the accuracy of the ANN's predictions.

The optimizer function is an essential component of an artificial neural network (ANN) model. It is an algorithm that is used to adjust the weights of the network in order to minimize the loss function, which measures the difference between the predicted output of the model and the true output.

The most common type of optimizer function is the stochastic gradient descent (SGD) algorithm (Bottou 2010) (Figure 2). SGD calculates the gradient of the loss function with respect to the weights of the network and updates the weights in the opposite direction of the gradient. This process is repeated for multiple iterations, or epochs, until the loss function reaches a minimum value or the model reaches a satisfactory level of accuracy.

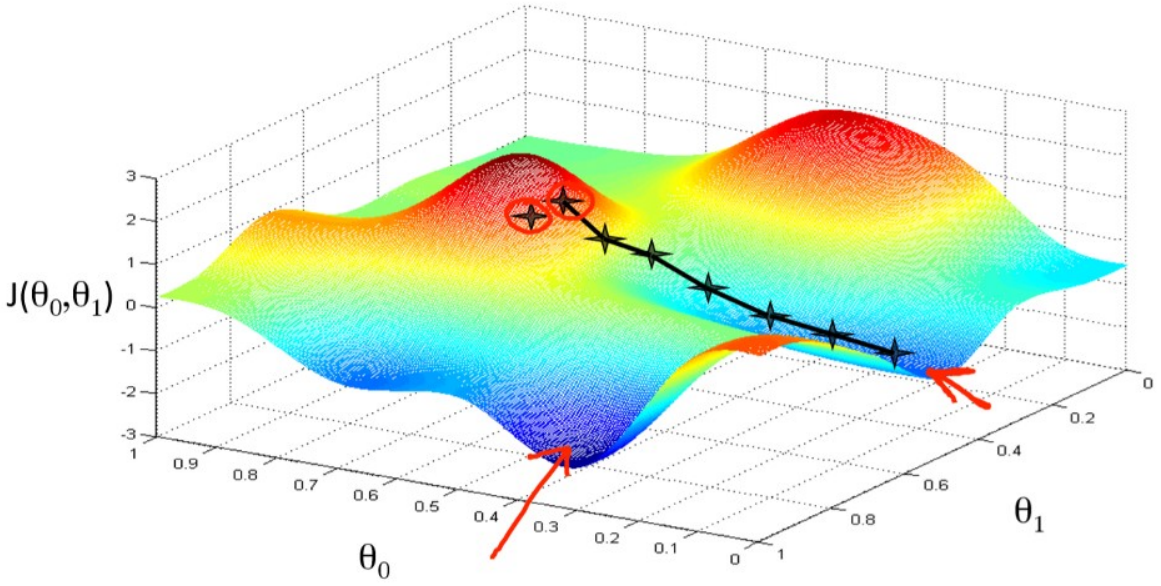


Figure 2 Gradient descent is the algorithm used to find the optimal parameters (weights and biases) for our network

One limitation of SGD is that it uses a fixed learning rate, which determines the size of the updates to the weights. A learning rate that is too small can cause the training process to be slow, while a learning rate that is too large can cause the model to overshoot the minimum value of the loss function and become unstable.

To address this limitation, more advanced optimizer functions, such as ADAM and RMSprop, have been developed. These algorithms incorporate estimates of the first and second moments of the gradient, which are used to adapt the learning rate for each weight in the network. This allows the optimizer to use a larger learning rate, which can speed up convergence and improve the performance of the model. Additionally, these algorithms incorporate moving averages of the gradient and the squared gradient, which can help to reduce the variability of the gradients and improve the stability of the training process.

The ADAM optimizer (Kingma and Ba 2014) is a valuable tool for training an ANN to predict the interior ambient temperature because it allows the model to adapt the learning rate for each weight in the network. This can improve the performance of the model by speeding up convergence and reducing the risk of overfitting. Additionally, the moving averages of the gradient and the squared gradient that are incorporated into the ADAM algorithm can help to stabilize the training process and improve the quality of the model's predictions.

The loss function is a crucial component of an artificial neural network (ANN) model. It is a measure of the difference between the predicted output of the model and the true output, and its purpose is to provide a basis for adjusting the weights of the network during the training process.

The loss function is used in conjunction with an optimizer function, which is an algorithm that adjusts the weights of the network in order to minimize the value of the loss function. The goal of the training process is to find a set of weights that minimizes the loss function and produces the most accurate predictions.

There are many different loss functions that can be used in ANNs, and the choice of loss function depends on the type of problem that the model is trying to solve. For example, in a regression problem, where the goal is to predict a continuous-valued output, the mean squared error (MSE) loss function is often used. This function calculates the difference between the predicted and true output for each sample in the dataset, squares the difference, and averages over all the samples.

In a classification problem, where the goal is to predict a discrete-valued output, the cross-entropy loss function is often used. This function calculates the difference between the predicted and true output for each sample in the dataset and applies a logarithmic transformation to the difference. This transformation has the effect of penalizing large errors more heavily than small errors, which can improve the performance of the model.

The choice of loss function can have a significant impact on the performance of an ANN model. Different loss functions have different characteristics, such as their sensitivity to large errors and their sensitivity to the scale of the true and predicted output. These characteristics can affect the speed and stability of the training process, as well as the final accuracy of the model.

The NRMSE loss function (Sewdien, Preece et al. 2020) is a useful performance metric for evaluating the accuracy of an ANN's predictions of the interior ambient temperature. It is calculated by dividing the root mean square error (RMSE) by the standard deviation of the true interior ambient temperature and subtracting the mean of the true interior ambient temperature. This normalization step helps to account for differences in the scale of the true and predicted temperatures, which can make the loss function more stable and easier to interpret. A low NRMSE value indicates that the ANN is making accurate predictions of the interior ambient temperature.

In summary, using an ANN with a Sigmoid activation function, ADAM optimizer, and NRMSE loss function is a powerful methodology for predicting the interior ambient temperature that the occupant experiences. The ANN learns complex nonlinear relationships in the data, the ADAM optimizer adapts the learning rate for each weight in the network, and the NRMSE loss function provides a stable and interpretable performance metric. These features can improve the accuracy and performance of the model, making it a valuable tool for controlling the heating and cooling systems in a building.

DATA AND MODEL

1. DATA

For our project, we have more than 20000 rows of hourly data collected on indoor air temperature. First thing to do to prepare this data for our ANN model is to divide into two sets of features and output(s). To do so, we need to clearly distinguish which columns are the output(s) that we want to predict, and which columns are features that have effects on the output. In our case, we have 13 features that impact our only output which is the indoor ambient temperature the occupant experiences (Figure 3).

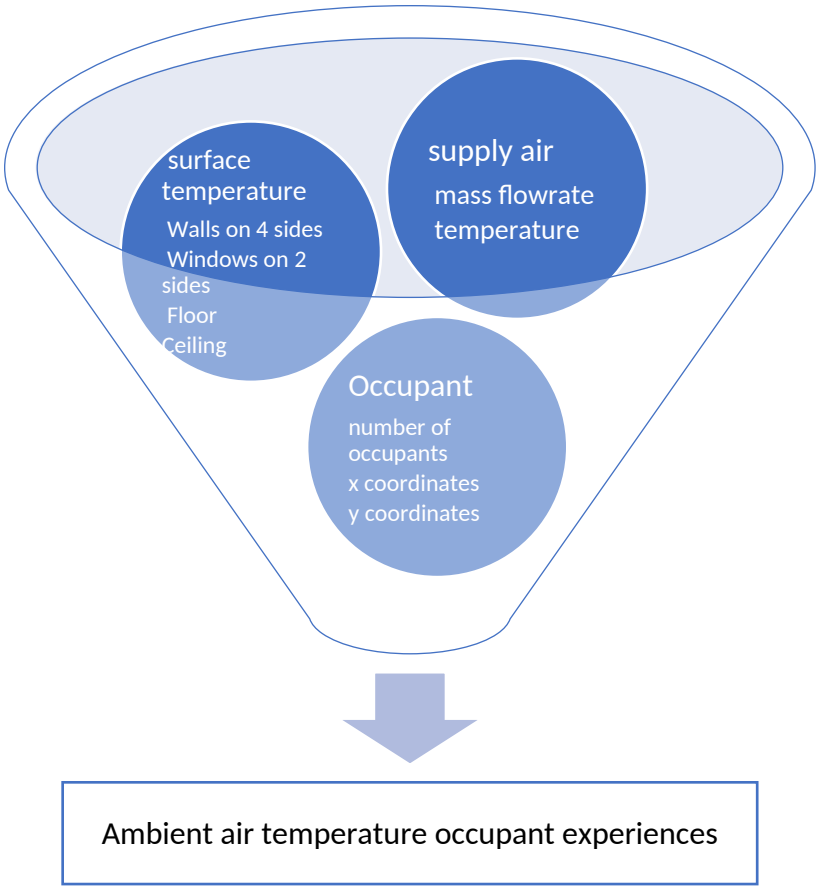


Figure 3 13 features are provided as input data to the model to predict the ambient air temperature occupant experiences

Next step is to preprocess the dataset for modeling. Preprocessing the data before training an artificial neural network (ANN) model is an important step that helps to ensure that the model is able to learn effectively from the data. Preprocessing the data typically involves several steps, including cleaning the data to remove any missing or invalid values, scaling the data to ensure that all features are on the same scale, and splitting the data into training, validation, and test sets.

One common preprocessing step is to remove any missing or invalid values from the data. This is important because ANNs are sensitive to missing data and having missing values in the data can cause the model to make inaccurate predictions. To remove missing values, you can use techniques such as imputation, where missing values are replaced with estimates based on the other values in the dataset. Fortunately, in our case, the data was already cleaned up and it did not have any missing points.

Another important preprocessing step is to scale the data. This is necessary because ANNs typically work best when the input features are on the same scale. For example, if some features are in the range 0-1 and others are in the range 0-100,000, the model may not be able to learn effectively from the data. To scale the data, you can use techniques such as min-max scaling, where each feature is scaled to a specific range (e.g., 0-1), or standardization, where each feature is scaled to have a mean of 0 and a standard deviation of 1. We used sklearn.preprocessing._data.StandardScaler which standardizes features by removing the mean and scaling to unit variance.

The standard score of a sample x is calculated as:

$$z = (x - u) / s$$

where u is the mean of the training samples or zero if with_mean=False, and s is the standard deviation of the training samples or one if with_std=False.

Finally, it's important to split the data into training, validation, and test sets. The training set is used to train the model, the validation set is used to evaluate the model's performance during training, and the test set is used to evaluate the model's performance on unseen data after training. This is important because it allows you to evaluate the model's performance on different subsets of the data, which can help you to avoid overfitting (where the model performs well on the training data but poorly on new data) and underfitting (where the model does not perform well on either the training data or new data).

For our case, we divided the data into two sets of 30% for testing and 70% for training with 42 shuffles before the split to keep the data's random state.

2. ANN MODEL

Like any other ANN model, we have an input layer comprising of all of our input parameters or features. We then have two hidden layers. They are called hidden layers because we do not observe their behavior throughout the process. Finally, we have an output layer, which houses our only output parameter. Each node in each of our layers is connected to all the nodes of the layers adjacent to them. This is also called a densely connected neural network architecture.

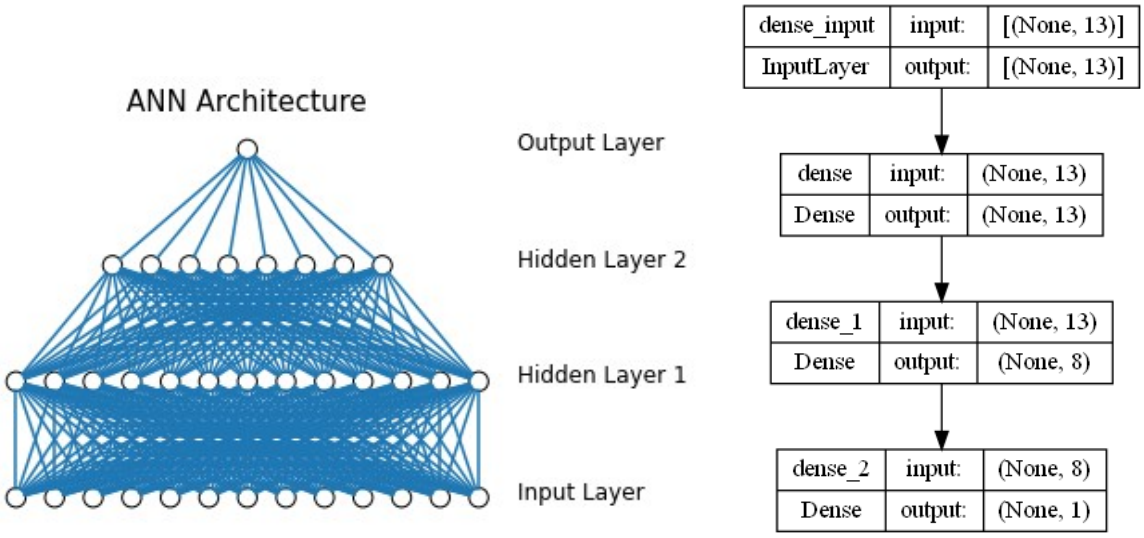


Figure 4 the architecture of our ANN model persists of 4 layers, 13 input parameter nodes, 13 nodes in the first hidden layer, 8 nodes in the second hidden layer and 1 node in the output layer.

Since we have 13 input parameters and one output parameter in our raw data, our input layer has 13 nodes, and our output layer has one node. For our hidden layer 1, keras automatically chooses same number of nodes as the number of nodes in the input layer and for hidden layer 2, we chose 8 nodes (Figure 4). All layers are activated using Sigmoid activation which squishes the values of inputs between 0 and 1. The significance of using an activation method in an ANN model is that it allows us to perform a better prediction of the output by introducing a higher dimensionality by a higher dimension function

such as Sigmoid. This process increases the features to observe which allows us to make a better prediction of the output.

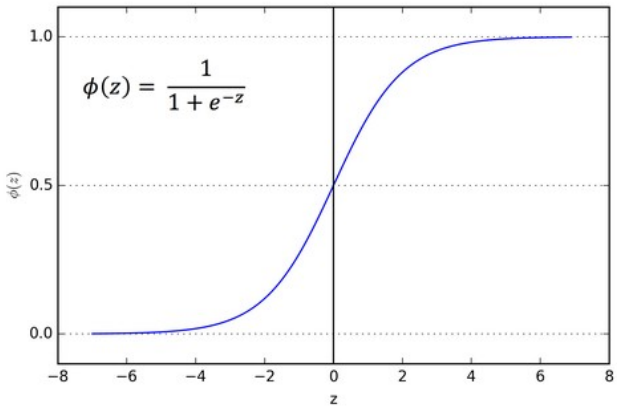


Figure 5 All layers are activated using Sigmoid activation which squishes the values of inputs between 0 and 1.

Using the ADAM optimizer, and NRMSE as our loss function, we started training the model on the training dataset. The training dataset was fed into the model in different batch sizes from 5 to 50 and different epoch values from 20 to 200. Evaluating the model based on the NRMSE loss function and RUNTIME for each step, we fine-tuned the model's hyperparameters (such as the learning rate and the number of hidden units) to improve its performance. Finally, a batch size equal to 36, meaning 36 rows of data at a time, and 100 epochs, meaning 100 different variations of the data sequence and the same number of nodes for the two hidden layers chosen initially, was chosen as the optimal combination of these inputs.

Table 1 Final values of loss function and runtime after hyperparameters were fine-tuned

Final	
NRMSE:	3.99
Runtime:	0s 861us/step

OUTCOME

After training the model, we fed it the testing dataset that had been set aside at the outset. Comparing the projected values, the model generated from these never-before-seen data points to the actual result values reveals that the model does an excellent job forecasting the interior ambient temperature experienced by the tenant. With a Normalized Root Mean Square Error (NRMSE) less than 4, the graph of anticipated Temperature against actual Temperature of ambient air is relatively linear (Figure 6). It can be seen from the graph that temperatures beyond 24 degrees Celsius deviate from the actual value somewhat more. This variation may be attributable to model error at higher temperatures or to the fact that a larger proportion of the data points fall within this temperature range.

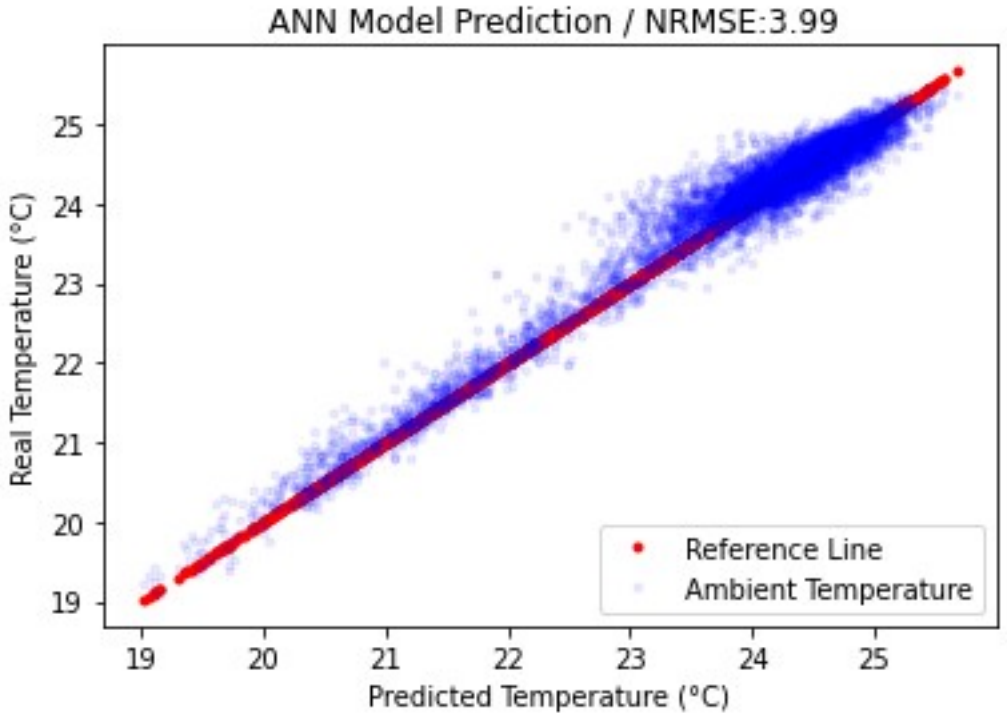


Figure 6 Real verses Predicted ambient temperature that occupant experiences regarding the data points in testing dataset.

Then, we examined ten random data points from Figure 7 inside the testing data set. Due to the smaller amount of data points, our NRMSE score climbs to 7.68, but this is still an excellent figure and indicates that the model is functioning well.

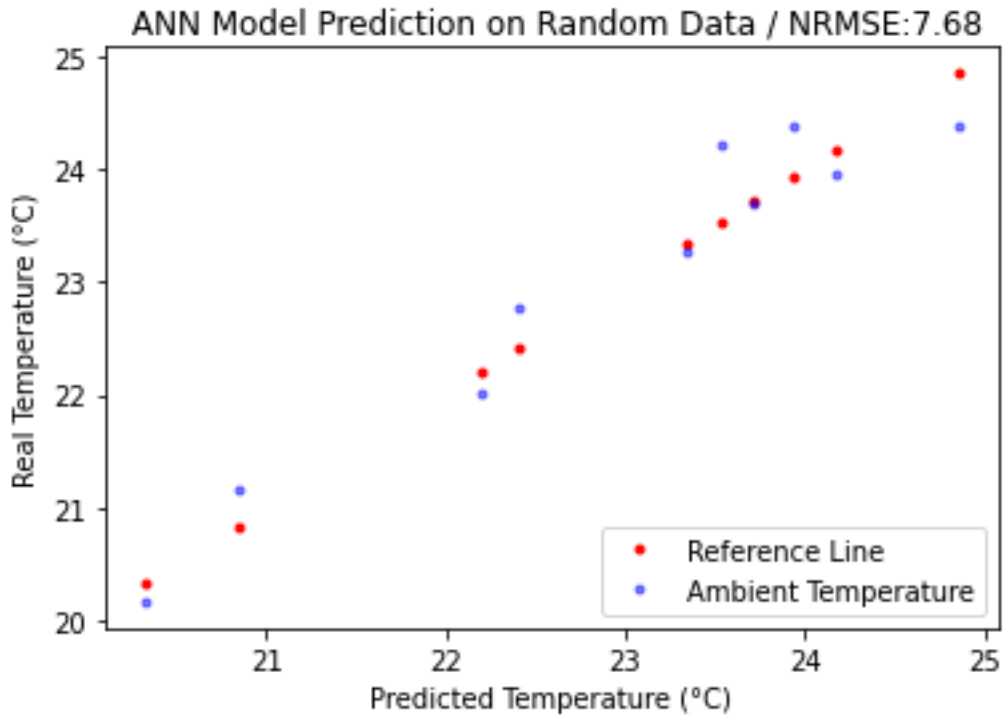


Figure 7 10 random data points in testing dataset is looked at closely to see the performance of our model in specific cases.

CONCLUSION

In conclusion, our model demonstrates that utilizing an ANN with a Sigmoid activation function, ADAM optimizer, and NRMSE loss function is an effective way for forecasting the interior ambient temperature experienced by the occupant. The ADAM optimizer modifies the learning rate for each weight in the network, and the NRMSE loss function offers a stable and interpretable performance indicator. These elements increase the model's accuracy and effectiveness, making it a valuable tool for regulating a building's heating and cooling systems. Due to the extremely large number of epochs utilized in our model, there is a chance that it has been overfit to this dataset. This assertion must be tested further by giving the model additional unseen data points and assessing its performance.

REFERENCES

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. Proceedings of COMPSTAT'2010, Springer**:** 177-186.

Hoskins, J. C. and D. M. Himmelblau (1988). "Artificial neural network models of knowledge representation in chemical engineering." Computers & Chemical Engineering **12**(9-10): 881-890.

Kingma, D. P. and J. Ba (2014). "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980.

Lira, J. O., et al. (2022). "Computational fluid dynamics (CFD), artificial neural network (ANN) and genetic algorithm (GA) as a hybrid method for the analysis and optimization of micro-photocatalytic reactors: NOx abatement as a case study." Chemical Engineering Journal **431**: 133771.

Sewdien, V., et al. (2020). "Assessment of critical parameters for artificial neural networks based short-term wind generation forecasting." Renewable Energy **161**: 878-892.

Sharma, S., et al. (2017). "Activation functions in neural networks." towards data science **6**(12): 310-316.

White, B. (1963). Principles of neurodynamics: Perceptrons and the theory of brain mechanisms, JSTOR.

Yegnanarayana, B. (2009). Artificial neural networks, PHI Learning Pvt. Ltd.