Qihao HE

Abstract

Urban Morphology involves various continuous features such as building height and area that correlate closely with Urban Heat Island (UHI) phenomenon and Land Surface Temperature (LST). This study applies the Feedforward Neural Network (FNN) machine learning approach in an attempt to quantify the relationship between 9 urban variables and LST in the Pearl River Delta region. Various sets of hyperparameters have been tested out. A 9-32-16-1 network structure is proved to outperform the traditional linear regression model by comparing the correlation coefficient between test set actual LST values and predicted values produced by the model. Further improvements in the current NN model are possible with improvements on data and model hyperparameters.

1. Introduction

Land Surface Temperature (LST) has always been one important variable in urban environment study, where scholars have established the idea of Local Climate Zone (LCZ) to evaluate the connection between urban morphology and Urban Heat Island (UHI) effect by looking at the relationship between LST data and corresponding LCZ area [1].

However, LCZ is a categorical feature rather than a continuous descriptor, which means it can only be associated reasonably with categorical LST [1]. There has been an effort to associate variables like continuous Impervious Surface Cover (ISC) with continuous LST [2], with traditional regression methods limited in complexity, where only a small number of parameters can be tried in the prediction function. Also, only a small group of different variables related to urban morphology could be included in such a model, which means they can only test out a limited number of not very complicated regression models.

For a possibly better regression model that can associate continuous LST data with complex urban data, this study adopts the machine learning method using a simple Neural Network (NN) that is capable of combining multiple quantitative urban morphologic features make more comprehensive predictions. And it will be compared with the linear regression model using the traditional approach.

2. Methodology and Data

2.1 Region of Interest

This study selects a domain in Pearl River Delta in southeast China, covering several major cities, including Guangzhou, Foshan, Dongguan, Shenzhen, and Hong Kong. Cities in this area started urbanization in the 1980s at the beginning of the Economic Reform. Shenzhen acted as a leading figure [3]; hence, the cities are mostly developed, and the city morphology in this region is representative.

The precise coordinate for the upper left corner of this region is [602504.163, 2635258.89] (m) in World Geodetic System (WGS) / Universal Transverse Mercator (UTM) zone 49N projection system that converts longitude/latitude to *Y/X* respectively. In this coordinate system, the location of every pixel in the study area can be easily accessed by fixing a corner (Upper left corner in this case) and the lengths for the sides of the desired data image along the *X* and *Y* axes.

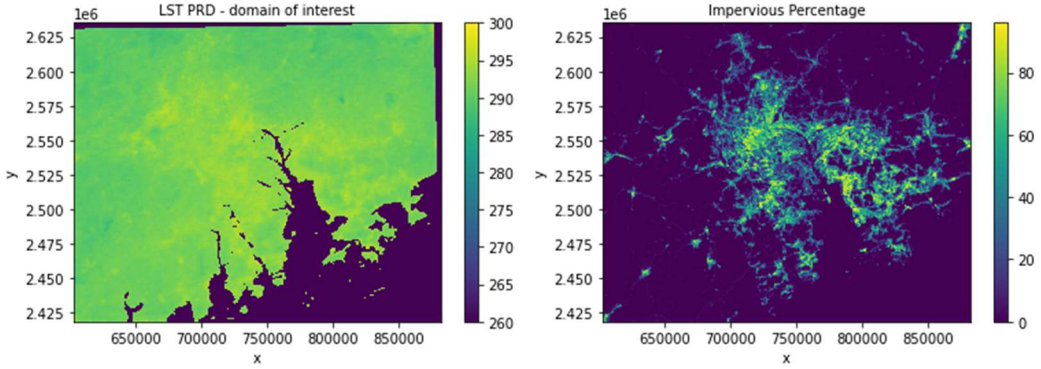


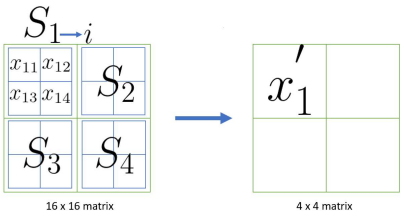
Figure 1. Selected area of interest, LST and Impervious Percentage

2.2. Data

The data used for Feedforward Neural Network (FNN) training is composed of two parts, the LST for the region as the target continuous value and eight kinds of variables that will be used as features to produce predicted LST value. And all features (including LST) are in 1000m spatial resolution in the entire process of training and prediction.

LST Data. There are two sources of temperature images we can choose from. For the LST data from the Earth Explorer (EE) Landsat 8. The satellite images are in 30m spatial resolution. To make use of the data, firstly, each pixel's value (x being the original value of the pixel in the following equation) needs to be converted to temperature in Kelvin, which is done by a linear function (Landsat 8 Collection 2 Level 2 Science Product Guide [4]): $T = 0.00341802x + 149.0$.

A change on the resolution should also be applied; since other data (shown in following sections) are in 1000m spatial resolution, a downscaling operation is applied on the LST yearly averaged image, the idea is to take average within smaller matrix blocks of the whole temperature image matrix.



$$x'_i = \frac{\sum_{j=1}^{4} x_{ij}}{4}$$

Figure 2. Attempted Resolution Conversion Method

However, this approach does not seem to work well when dealing with a large resolution difference (e.g., 30m to 1000m upscaling in this case). From a previous study, there is a significant correlation between impervious surface area (ISA) and LST [2]. Still, the resolution adjusted LST data has a very low correlation coefficient with ISA, compared to a much higher correlation coefficient between an LST originally in 1000m resolution and ISA originally in 1000m too.

Hence this study turns to adopt the 1000-spatial-resolution yearly averaged nighttime LST image retrieved from the second source, MOD11_L2 product [5]. This product was the result of a split-windows LST algorithm [6]. And we exclude data points in the target region with an LST value smaller than 250K, which are invalid.

ISA Data. This data has been found in the previous study to have a correlation with LST, as mentioned in the LST data part [2], as an attribute of artificial surfaces like pavements, roads, and buildings (City of Derham Public Works Department, n.d.). To find out the ISA distribution over the region of interest, Global Human Built-up and Settlement Extent (HBASE) Dataset from Landsat, v1 [7] was adopted in our study, the valid data range is 0 to 100, representing the percentage of imperviousness, hence certain data points outside this range were excluded in the target region.

Building plan area fraction (λ_P, *or lp2*), Building surface area to plan area ratio (λ_B, *or lb2*) and Area-weighted building height (AH, or ah2) Data [8]. This set of data in 900m-spatial-resolution depicts the building parameters related to urban morphology. They are defined as follows, with A_P being the plan area of buildings at the ground level, A_R being the plan area of rooftops, A_W being the total area of nonhorizontal roughness element surfaces A_T being the total plan area of the region, A_i being the area at the ground level, h_i being the height of each building, these values are measured in pixel (900m resolution) scale in the map, derived from datasets from Baidu:

$$\lambda_P = \mathrm{A_p},\ \lambda_B = A_R + A_W, AH = \frac{\sum_{i=1}^{N} A_i h_i}{\sum_{i=1}^{N} A_i}$$

This set of data is transformed from 900m resolution to 1000m according to the algorithm proposed in the LST data section. For the block sub-matrix size (approximately 100/81 points in one block) is small compared to the earlier 30m to 1000m conversion attempt. The changes made on the data are insignificant that we can confidently do the conversion.

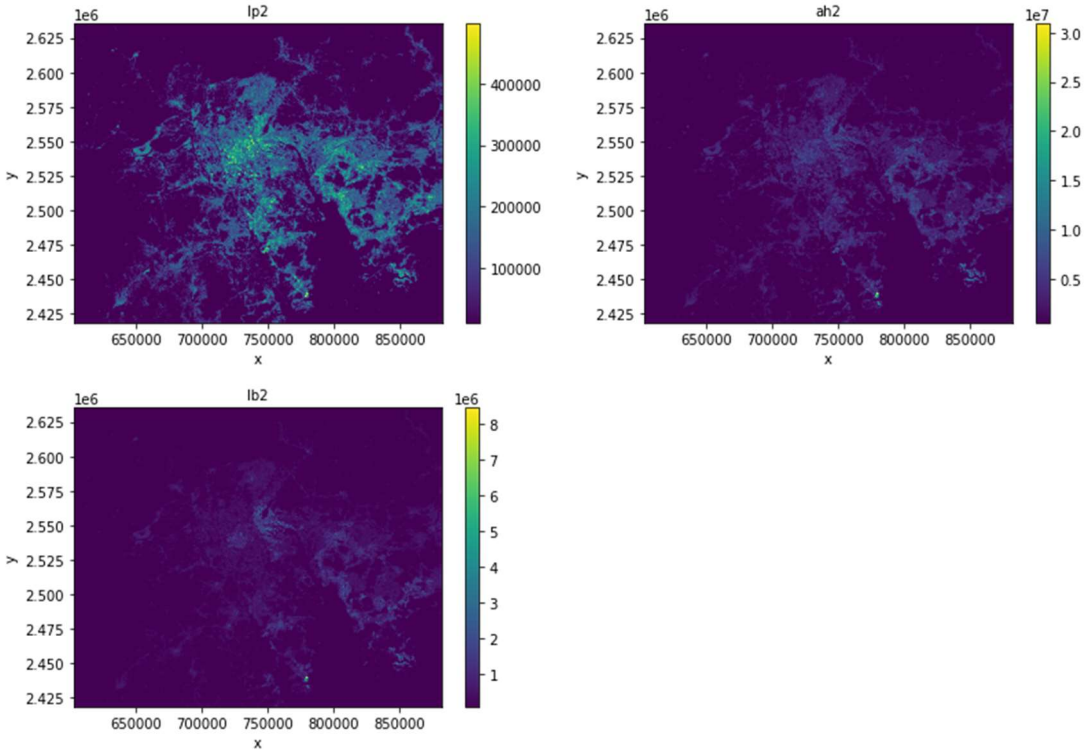


Figure 3. Spatial maps of lp2, ah2 and lb2

***X* (approximately longitude), *Y* (approximately latitude), UN WPP-Adjusted Population Density [9], altitude and 2019 Global Forest Canopy Height [10] Data.** This set of data in 1000m-spatial-resolution describes less the urban morphology but is essential to the estimation of LST. In a previous study, the LCZ concept, which includes this kind of data, is developed to be compared with LST to find their relationships [11]. Corresponding *X* and *Y* are embedded in the .tiff files of LST data. The altitude data came from the Enhanced Shuttle Land Elevation Data, generated from NASA's Shuttle Radar Topography Mission (SRTM) [12].

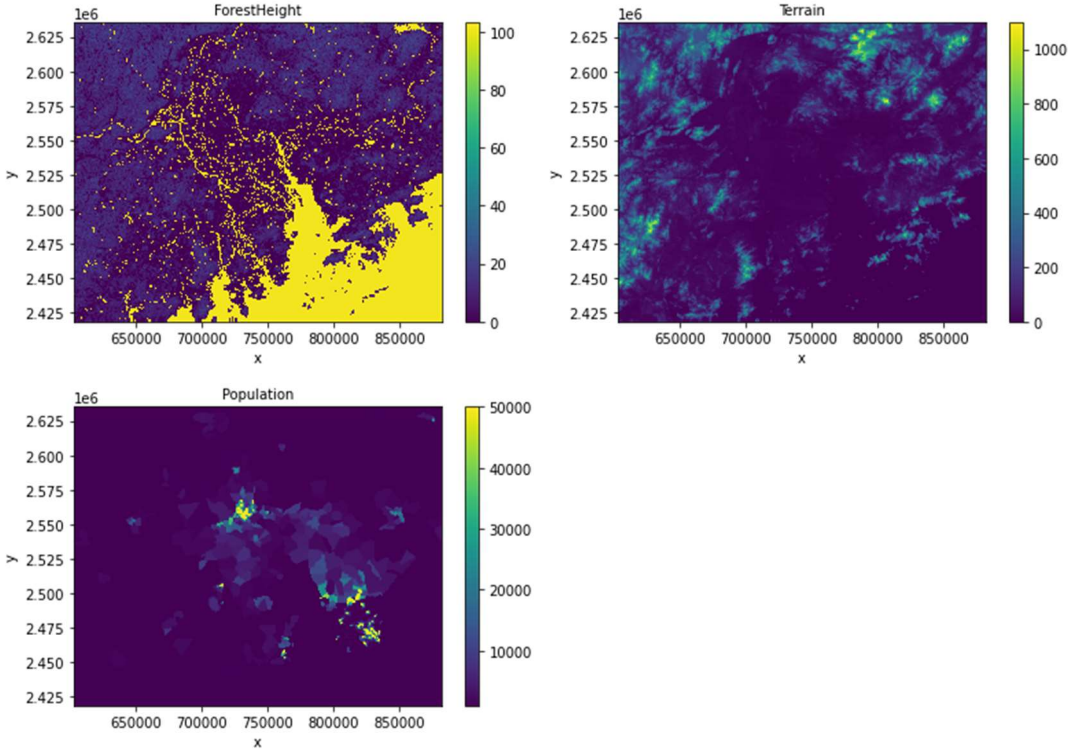


Figure 4. Spatial maps of Forest Height, Terrain and Population

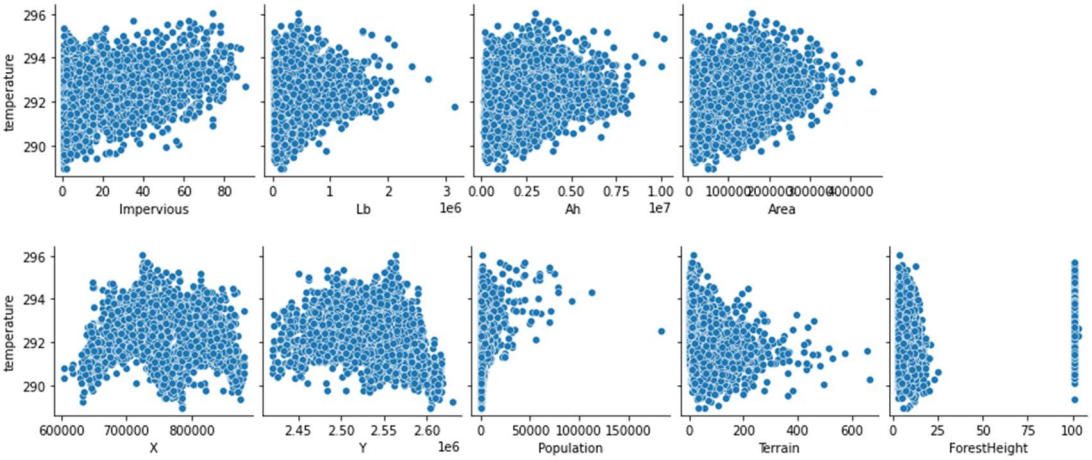


Figure 5. Scatter Plot of temperature relationships with regard to 9 variables

2.3. The Neural Network (NN) Model

To produce estimated LST values from 9 variables with the help of machine learning, we make use of the traditional Feedforward Neural Network (FNN) with Back Propagation (BP) Gradient Descent Based Algorithm [13] to associate these 9 input variables with LST through the network structure. There are two important components, Optimizer, and loss function, and an important method called backpropagation that makes an FNN work properly. This FNN is based on the basic regression model from the TensorFlow GitHub repository [14].

Basic Gradient Descent (GD). This optimization algorithm is a classic method in the core of most traditional machine learning models [13]. In the case where multiple variables contribute to a cost function, the optimization process that minimizes the cost function can be expressed as follows:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

Where $min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ is the optimization goal, and θ_j is each parameter in the model (called weight in Neural Network) that is updated simultaneously with a learning rate or step size α that indicates how fast in each iteration the change is made towards the minimum point. Usually, a larger learning rate results in a shorter learning time. Still, poorer result compared to a lower learning rate which can find the optimized set of parameters, reaching the global minimum point that is more time-consuming.

Adam Optimizer. This method is based on Stochastic Gradient Descent (SGD), and its name is based on adaptive moment estimation [15]. It is one of the most frequently used and efficient optimizers that work well with many machine learning problems.

Model Structure. A Neural Network (NN) is essentially a function with numerous parameters represented by the neurons' connections in the network. In a NN, every line connecting two nodes (neuron) denotes a parameter θ_j That is also called weight described earlier in the optimization algorithm part, they work as multipliers on input neurons, and the multiplication results from all neurons are summed to produce the value of neuron(s) in the next layer. Apart from these multipliers on variables, there is another group of parameters present in the NN that works as bias, representing a shift in output values for the function. This bias parameter is represented by an additional neuron in every layer that does not connect with any neurons in the previous layer but connects to all neurons in the next layer.

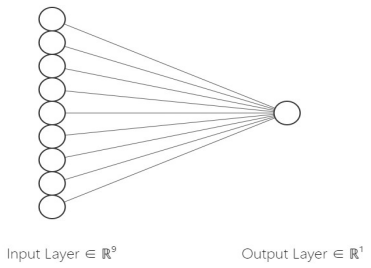


Figure 6. Simplest Linear NN

NN can represent an either a linear or non-linear relationship. In a linear NN, there is no hidden layer between the input and output layers, and the normalized input variables are directly connected to the output layer. The total number of parameters trainable is the number of input and output neurons. In our linear NN model, it is 9 weights of 9 lines that connect inputs to an output, and one bias that connects to the output neuron, in total ten parameters trainable.

In a non-linear NN, the weights and biases are similar to that of a linear NN. In an L-layer NN, the number of parameters can be represented by a number of neurons in layers, suppose in layer l there are S_l neurons:

$$\prod_{l=1}^{L-1} S_l S_{l+1} + \sum_{l=2}^{L} S_l$$

With the first term being the number of weights and the second term the number of biases. However, it is the activation function that makes a multi-layer NN non-linear. Otherwise, it is essentially the same as a one-layer NN with an equal number of neurons.

An activation function is applied on the summed result of neurons' values multiplied with weights in each layer before the output layer to transform linear inputs to non-linear ones that feed into the next layer's neurons. With the help of activation function/layer, a NN can produce a prediction for non-trivial problems. In our model, Rectified Linear Unit (ReLU) is chosen for it is selected by many state-of-the-art NNs [16]:

$$f(x) = \max(0, x)$$

Loss function and Back Propagation. Loss function in NN is equivalent to the cost function described in the earlier GD optimization part, which is the core of the training process of a NN model. Our FNN incorporates Mean Absolute Error (MAE) as the training target, implemented with the Backpropagation (BP) algorithm where the input data are initially fed to the network and go through all neurons in the hidden layers, results from all neurons and layers are computed with weights and biases in their randomly initialized values. And we obtain a value from the output neuron, which is the predicted value generated for the first time. This process is called Feedforward, corresponding to the name FNN. We obtain an initial value for the cost function, which is the error between the predicted result and the real LST value. According to the previously discussed GD algorithm, every weight in the previous layers contributes to this cost function (error).

Now we could compute the partial derivative of the cost function with regard to each weight. This can be solved by the chain rule in multivariable calculus, for each layer is a function of transformations of multiplication, summations, and activation (from linear to nonlinear) on input values using these weights, biases and an activation function, and the entire network is one big composite function. To compute these partial derivatives efficiently, the backpropagation algorithm is developed where the error propagates from the output layer to all previous layers [17]. It essentially avoids doing repetitive multiplications of partial derivatives from the forward direction (input to output) and calculating the error for each neuron in the backward direction from the output layer to the input layer.

Hyper Parameter tuning. Hyper Parameters are a set of parameters determined by designers of a NN before training, including the type of optimizer, learning rate, number of neurons, number of hidden layers, etc. These parameters, unlike weights and biases that change to improve the performance of the model during the training automatically, can only be configured by NN designers themselves to improve the training process, for they do not define the weights and biases that map values from inputs to outputs in the function the model is learning but regulate the way NN itself is structured and how it learns the function. For example, a linear NN cannot learn non-linear behaviors, and a Deep NN (DNN) can learn even more complex behaviors than a shallow NN. In our NN, several sets of a number of neurons and layers were tested. And the current number of hidden layers is set to be two, with the number of neurons for the two layers 32 and 16 as described by Figure 7.

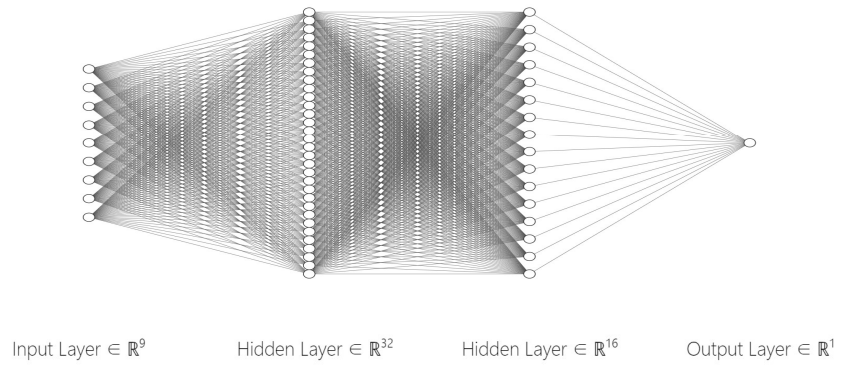


Figure 7. non-linear FNN with two hidden layers

3. Result

The target region valid data are divided into three groups as 64:20:16. With 64 percent of the data defined as a train set, 20 percent of the data as a test set to determine the model performance, and 16 percent of data in the validation set to help spot out possible overfitting in the training process where losses for both training set and validation set are plotted with regard to epoch number.

The Linear Regression Algorithm used for comparison is the *LinearRegression* design from the learn python library implemented with Least Squares Method [14]. And the linear NN model is the simplest NN, treated as a baseline to exclude models with inappropriate hyperparameters if they could not outperform the predictions produced by this simplest model. And the Deeper nonlinear NN is in 9-32-16-1. All NNs are trained for 1000 epochs, as the losses are observed to flatten, and the model stabilized after this length of training time.

The detailed training process and results can be found in our repository: https://github.com/phyqh/UROP_urbanTemperature.git

Prediction Comparison between linear NN and Traditional Least Squares Method. The correlation coefficient for linear NN predicted values and actual LST values is only 0.6400432, which is not compatible with the traditional method where the correlation coefficient between Least Squares linear regression model prediction and actual LST value is 0.67758547.

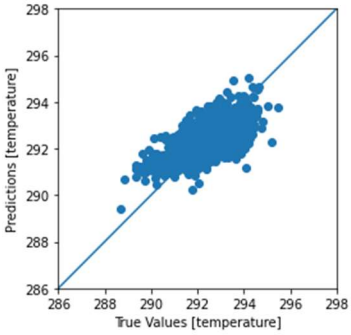


Figure 8. Least Squares Method

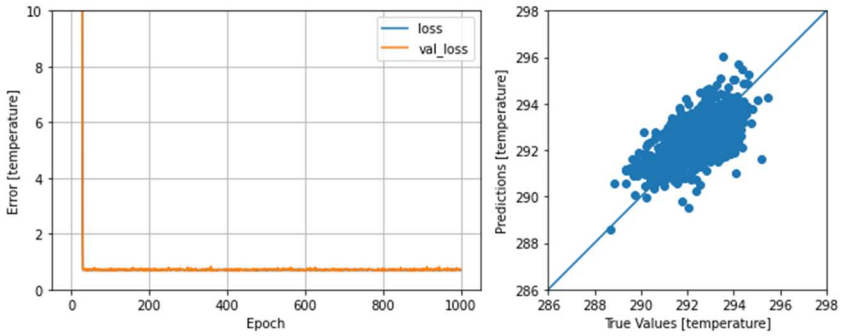


Figure 9. linear NN of 9-1(Number of neurons) structure

Prediction Comparison between nonlinear FNN and Traditional Least Squares Method. The correlation coefficient between nonlinear FNN predictions and actual values is 0.71517678, higher than 0.67758547 in the traditional linear model. The performance of our current FNN seems not good enough to outperform the traditional method by a large margin. Still, it should be a method with the potential, that neither significant underfitting nor overfitting is observed from the result, and it does reasonably better than the baseline linear NN, which shows the promising effect of hyperparameter tuning.

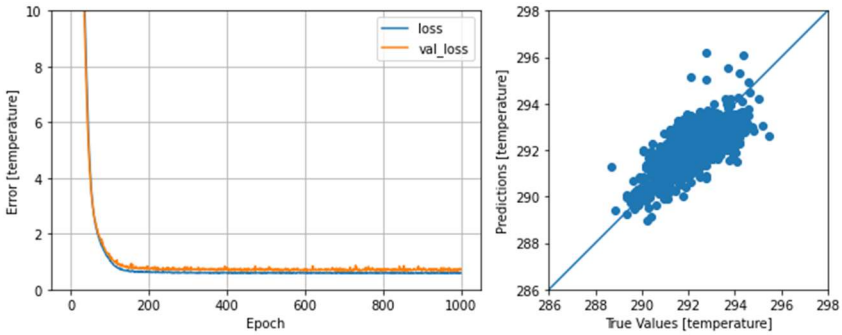


Figure 10. nonlinear FNN of 9-32-16-1(Number of neurons) structure

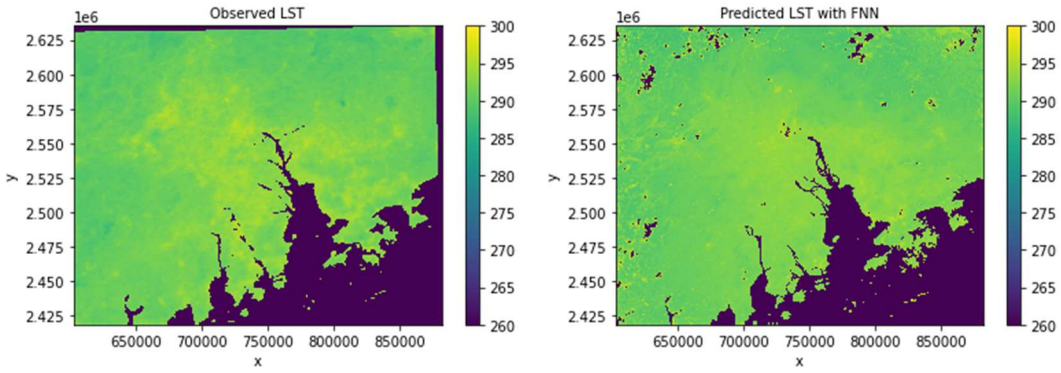


Figure 11. FNN Prediction (Places with value 0, or holes on the map are data points the model cannot produce a reasonable prediction for, since certain invalid data points are pre-excluded before training) compared with Observed LST values in the selected area

4. Conclusion and Future Works

This study attempts to associate 9 different features of the urban environment to LST with FNN model, which turns out to outperform the traditional linear regression model by a small margin. There are two important directions where possible improvements can be made to strengthen the result of this machine learning approach.

Firstly, the data used in this study are all in 1000m spatial resolution, which means a relatively small dataset, where overfitting is more likely to happen, resulting in poorer performance of the model. If more data points can be collected in future studies using a 30m spatial resolution map, the training result from training set data may generalize better to the test set, producing predictions with higher accuracy and correlation with the actual values.

Secondly, hyperparameters are deterministic in NN training, which have been configured manually with limited trials in this study. Research has shown that randomly searching for hyperparameters can be much more efficient [18].

Machine learning is a promising tool that has the potential to reveal many nonlinear relationships, including LST and urban morphologic variables. An advanced NN model should be able to reveal more intricate connections, hence providing more numerical intuitions for theoretical explanations that may better associate urban morphology with the UHI phenomenon.

References

[1] M. Cai, C. Ren, Y. Xu, K. K.-L. Lau, and R. Wang, “Investigating the relationship between local climate zone and land surface temperature using an improved

Wudapt methodology – a case study OF Yangtze River Delta, china," *Urban Climate*, vol. 24, pp. 485–502, 2018.

[2] H. Xu, D. Lin, and F. Tang, "The impact of Impervious surface development on land surface temperature in a SUBTROPICAL CITY: Xiamen, China," *International Journal of Climatology*, vol. 33, no. 8, pp. 1873–1883, 2012.

[3] Weng Q, Yang S, "An approach to evaluation of sustainability for Guangzhou's urban ecosystem.*" International Journal Sustainable Development and World Ecology,* 10: 69-81, 2003.

[4] K. SAYLER, "Landsat 8 Collection 2 level 2 Science PRODUCT GUIDE," *U.S. Geological Survey*, Sep-2020. [Online]. Available: https://www.usgs.gov/media/files/landsat-8-collection-2-level-2-science-product-guide. [Accessed: 15-Aug-2021].

[5] Wan, Z., Hook, S., Hulley, G., MOD11_L2 MODIS/Terra Land Surface Temperature and the Emissivity 5-Min L2 Swath 1km. NASA LP DAAC. 2015. [Online]. Available: http://doi.org/10.5067/MODIS/MOD11_L2.006

[6] Zhengming Wan and J. Dozier, "A generalized split-window algorithm for retrieving land-surface temperature from space," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 34, no. 4, pp. 892–905, 1996.

[7] Wang, P., C. Huang, E. C. Brown de Colstoun, J. C. Tilton, and B. Tan, Documentation for the Global Human Built-up And Settlement Extent (HBASE) Dataset From Landsat. Palisades, NY: NASA Socioeconomic Data and Applications Center (SEDAC). 2017. [Online]. Available: shttps://doi.org/10.7927/H48W3BCM. Accessed DAY MONTH YEAR.

[8] X. He, Y. Li, X. Wang, L. Chen, B. Yu, Y. Zhang, and S. Miao, "High-resolution dataset of URBAN canopy parameters for Beijing and its application to the Integrated WRF/Urban modelling system," *Journal of Cleaner Production*, vol. 208, pp. 373–383, 2019.

[9] "Gridded population of the World (gpw), v4," *SEDAC*. [Online]. Available: https://sedac.ciesin.columbia.edu/data/set/gpw-v4-population-density-adjusted-to-2015-unwpp-country-totals-rev11. [Accessed: 18-Aug-2021].

[10] "Glad," *GLAD*. [Online]. Available: https://glad.umd.edu/. [Accessed: 18-Aug-2021].

[11] R. Wang, M. Cai, C. Ren, B. Bechtel, Y. Xu, and E. Ng, “Detecting Multi-temporal land cover change and land surface temperature in Pearl River delta by adopting local climate zone,” *Urban Climate*, vol. 28, p. 100455, 2019.

[12] “Shuttle radar Topography Mission,” *NASA*. [Online]. Available: https://www2.jpl.nasa.gov/srtm. [Accessed: 18-Aug-2021].

[13] I. A. Basheer and M. Hajmeer, “Artificial neural networks: Fundamentals, computing, design, and application,” *Journal of Microbiological Methods*, vol. 43, no. 1, pp. 3–31, 2000.

[14] François Chollet, “Basic regression: Predict fuel efficiency”, GitHub repository, 2017. [Online] Available: https://github.com/tensorflow/docs/blob/master/site/en/tutorials/keras/regression.ipynb

[15] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv.org*, 23-Apr-2015. [Online]. Available: https://arxiv.org/abs/1412.6980v5. [Accessed: 15-Aug-2021].

[16] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015.

[17] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural Networks*, vol. 61, pp. 85–117, 2015.

[18] Pedregosa *et al.*, “Scikit-learn: Machine Learning in Python,” JMLR 12, pp. 2825-2830, 2011.

[19] Bergstra, J, Bengio, Y, “Random Search for Hyper-Parameter Optimization”, *Journal of Machine Learning Research*, vol. 13, pp. 281-305, 2012.