Predicting solar radiation with Artificial Neural Network based on urban geometrical classification

Anas M. Hosney Lila[1,2], Wassim Jabi[2], Simon Lannon[2]
[1]Architecture and Built Environment Department, Faculty of Environment and Technology, University of West England, Bristol, United Kingdom
[2]Welsh School of Architecture, Cardiff University, Cardiff, United Kingdom

Abstract

This research introduces the adaptation and development of an open-source Artificial Neural Network (ANN) with the aim of predicting solar radiation for newly generated neighbourhoods in Aswan, Egypt as an example of a hot arid zone. The outcomes are the result of training the ANN on a database of classified urban geometries and their solar radiation simulation results for local weather conditions. The classification of this database was first introduced and discussed in (Lila and Lannon 2019). This paper discusses the different stages of developing the ANN code and its final version capabilities.

The ANN code was developed to differentiate the training process from the prediction code to allow for the reuse of the trained ANN in multiple tests. The ANN code was tested for different database sizes to predict individual buildings' solar radiation and was also used to predict solar radiation for urban configurations that were not part of the training process. The results of these ANN predictions were compared to conventional solar radiation simulation results to establish the accuracy and time saved.

Key Innovations

The focus on solar radiation is a part of a multi-stage proof-of-concept framework that produces a novel method to optimize performance-based neighbourhood geometry. This research shows the building of an ANN that is based on novel geometrical classification of urban solar radiation as a new method to save simulation time and achieve significant accuracy. This ANN, which is based on a pre-existing open-source code, is built to be reused for the same weather data, can be used in the Grasshopper environment(Scott Davidson, 2017), and written in the Python Language. This research saves computational time to allow for performance-based design decisions to be included in the early stages of urban design. The time savings will also pave the way for generating optimized neighbourhood geometrical options with solar radiation as a fitness function for this optimization process.

Practical Implications

This study shows the potential of applying ANN methods to predict solar radiation at the urban scale. The paper also provides a proof-of-concept ANN node that can reuse trained networks to predict simulation results in the Grasshopper environment.

Introduction

Urban performance computational modelling and simulation has gained attention because of increasing awareness of urban growth outcomes and the need to mitigate urban microclimate environmental impacts. This attention focused on the early stages of urban design to develop evidence-based approaches to provide better performing urban environments. Some critical issues have been addressed during research to develop and produce computationally modelled urban environments and environmentally optimized design decisions on an urban scale for the early stages of design. The complexity and level of details of urban modelling are among the issues which have been widely investigated by different studies (Biljecki et al., 2014; Martin & March, 1972; Picco & Marengo, 2015; Schwarz, 2010; Stewart & Oke, 2012). Other challenges were the performance simulation methods and techniques. Research is still investigating and developing different approaches to the urban performance simulation process and its inputs, outputs and visualizations (Greenberg & Erdine, 2014; Naboni, 2014; Nault et al., 2016; Trigaux et al., 2014). Moreover, another interesting aspect involves dealing with the resultant data from these processes and the way to feed it into the urban design and decision-making process, either for optimization goals or for evidence-based computationally generated urban designs (Beirão, 2012; Hillier, 2007; Koenig, 2011; Pinto Duarte et al., 2005).

Most of these studies have utilized the principles of parametric design within the urban modelling context. Parametric urban design can be represented simply as a group of arranged buildings and urban geometrical variables shaped by scripted algorithms. This interpretation provides a different vision and capability for investigating urban design, geometry and performance (Schumacher, 2009a, 2009b). This idea of parametric modelling and its tools have provided urban design and modelling with new computational applications. Computational optimization solvers form an important aspect of these new capabilities of urban computational design in its early stages. (Tamke et al., 2018) have discussed the role of merging machine learning methods and state of the art simulation tools in parametric environments and discussed the emergence of this approach in the current architectural practices.

Moreover, there has been a continuous development to include machine learning methods and principles under the notion of built environment optimization. Generally, the major difference in machine learning is where the problem, or parts of it, are predefined and pre-introduced to the solver. This means the algorithm is mainly predicting results for the problem based on existing "training data" (Dounis et al., 2014; Krarti, 2003; Zhao & Magoulès, 2012). Machine learning methods have different mathematical models, each of which follow a certain way of predicting the results of the proposed problem. One of the models, Artificial Neural Networks (ANN), aims to imitate the way the human neural system works (Cui & Cai, 2013). An overview and comparisons of the different mathematical models is presented in (Ali et al., 2018). As an emerging technology, accessibility to these principles and its application in user friendly platforms like Grasshopper has been developed too (Cichocka et al., 2017; Poving Ground, 2018; Wortmann, 2017). This allowed more visualization of the design optimization and performance predictions process in different stages which led to allow for user control over the process in order to overcome a challenge known as "Black box" which about these solvers and applications, being uncontrolled and ambiguous tools (Wortmann, 2017; Wortmann & Nannicini, 2016).

This paper discusses a part of ongoing research aiming to produce an urban design optimization framework that can break down urban complexity by geometry classification and reach for optimal neighbourhood's geometry based on its solar radiation performance prediction using ANNs. This paper focuses on the use of ANN's principles to significantly reduce computational cost and time in such scale of performance optimization by depending on acceptable prediction accuracy while avoiding the creation of another black box. Furthermore, the tool is intended to be available in commonly used parametric environments like Grasshopper and to allow for the separation between the training phase and the prediction phase of the process. This ANN code allows for the reuse of trained networks to save the training time for multiple sites when it shares the same weather data. The paper also discusses the testing of this ANN and its outcomes regarding time saving and achieved prediction accuracy when compared to traditional simulation results.

Methods

ANN node development

The literature has illustrated the use of machine learning techniques in architectural and urban design problems. An Artificial Neural Network (ANN) is one of these techniques. It is clear from its name that it is a mathematical way of imitating how the human brain neurons work. A concise guide to ANN concepts and core models can be found in Jian's "Artificial Neural Networks: A Tutorial" (Jain et al., 1996). This paper discusses the implementation of ANN to enable the recognition of urban geometries that are not typical to the saved database of classified geometry with its attribute tags and solar radiation performance without sacrificing too much of the accuracy achieved from conventional simulation methods.

ANN is tested to assess its accuracy in predicting the solar radiation performance of urban configurations based on the classified geometry of buildings. The first trial in this investigation was to use a ready-made available tool, LunchBox ML, for ANN principles within the platform of Grasshopper (Proving Ground, 2018). However, the time consumption in these tests proved to be one of the limitations using this LunchBox ANN tool. The fact that the number of requested predictions does not have a significant effect on the time for training and prediction has indicated that this can be enhanced by separating the training from the prediction process. In this way the time needed for training the ANN once can be balanced by using the trained network to predict a larger number of entries. Although the LunchBox node has an open-source code available, the code uses unclear classes, and it does not have enough documentation to reveal the source code core principles (Proving Ground, 2017). This led the research to look for another resource to apply ANN principles within this research scope. The idea is to find an open-source code for an ANN that does not rely on libraries and can be edited to enable the separation of the training and the prediction codes.

The ANN node used for this research is an open-source Python code originally inherited from a blog website that teaches how to build a neural network in a simple method with no loaded libraries (The Codacus, 2017) and went through different stages of enhancements until it reached the final version used. In this section, the parts added during development to the original code will be discussed, along with the initial and final stages of testing the utilization of ANN principles on the database at hand to get the optimal timesaving and accuracy needed.

The main idea of editing this code was to separate the training process from the prediction process. This took place by creating two Grasshopper Python components. The first component was used for the training part of the ANN application. It consisted of the original code classes for Connection, Neuron and Network. Then we added one more class instead of the original training class in which the training took place and then extracted the trained network as an output to be fed into the second separate component. This added class of code identifies different parts in the ANN. It starts with calling static values adopted from the original code, the neuron eta and alpha, and set by an integer input to the component. Then it identifies "maxiterations" which is the maximum number of iterations allowed for learning over the whole ANN before it starts to output predictions. This number is set as an input to the Grasshopper component to be set by the user. Then it calls the feed-forward and back propagation process over the input data and its targeted results. These are also fed to the Grasshopper component as a list of target data. This input name is "training P". Another input is the tree of lists containing the data used to train the network. Each list contains one column of the training database parameters and is named "Training Data" when input into the component. Another input is the "Error

Threshold" which is used to break the training process by comparing it with the ANN calculated error. Then the network is defined from these inputs and built based upon the "topology" input which is a list with the neuron number of each layer. The last lines of the code export the trained ANN as an output with the name "nn". Figure 1 is showing the two-component interface. Figure 1 (a) shows the training component. In addition to the mentioned inputs, there are the inputs of setting the eta which means the learning rate and sigmoid's Alpha for this component and those are integer inputs. The last input for this component is where to find the saved python code for "pickling" the neural network which allows for saving the trained ANN for reuse. The outputs include the neural network trained for live prediction followed by the pickled neural network to be reused in another file if needed. The third output provides the number of iterations done before stopping the training. The next output is a list of mean square errors for each iteration and the last output is the reason of stopping the training. The prediction component interface has less inputs and outputs as shown in Figure 1 (b). It basically receives the trained ANN either from a pickled version or directly from the training component in case both are used in the same instance. The other input is the data which needs to be predicted.

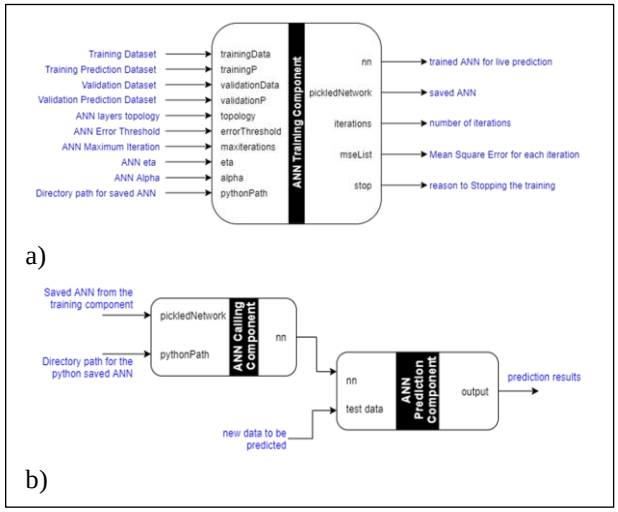

Figure 1 showing a) the training component in Grasshopper environment with its inputs and b) the prediction component calling a save trained ANN

The following component is responsible for receiving the trained ANN and the list of new data to be predicted. The code basically sets the input data under the name "test data" and creates a list of it to be predicted. Then it activates the trained ANN. The rest of the code is about activating the "feed-forward" function and obtaining the prediction results of the input data, either as one entry or a list of entries to be predicted at the same time. Then, the component exports the output predictions.

Through this two-component sequence, the time for iterating the prediction was reduced and the training time is accounted as just one time and the trained ANN is reused for different inputs following the training time. This saved the overall time and allowed for predicting different urban configurations in less time. As discussed earlier, the size of the input training data has a significant impact on the training time. Yet, this method of separating the training enabled the loading of larger datasets along with enhancing the possibility of testing it against different urban iterations.

One of the additions to the original code was to add the feature of "Cross-validation". This is a method to hold part of the training dataset and use it for testing the predictions to enhance the predictions for new data entries. So, the preserved subset of the data is used for testing the predictions during the training process to ensure the ANN is not trained only for the input data set. Thus, it allows the predictions to be more accurate when it predicts data that it did not see before during the training phase. There are different ways of conducting this method within the training phase (Arlot & Celisse, 2010; Pedregosa et al., 2010; Varoquaux et al., 2015). The simplest application of cross-validation was chosen to be implemented in this research. These cross-validation techniques are about holding a part of the training data and testing the ANN prediction on it to make sure the prediction error from the validation data set is less than the one calculated from the training dataset (Arlot & Celisse, 2010).

The code starts with calling the inputs and defining them. Then it starts the ANN training for the training data and collecting the error ratios for the training. Following this, the prediction starts on the cross-validation dataset followed by the collection of the validation error by comparing it with the validation target inputs. The last section of this class is to determine when to stop the training. Although the training still consumes some time, training 100,000 entries for 1,000 iterations using an ANN with two hidden layers typically consumed 7 to 8 hours. Although this may seem time consuming, it is still reasonable when factored into the ability to obtain performance predictions in a maximum of 20 seconds. This capability of benefiting from the trained network was further developed by importing a Python library, named "Pickle", that enables the saving of the trained ANN to be recalled and used for different iterations and on different PCs (Python Software Foundation, 1990).. It has the function to call for a version of the ANN code saved outside Grasshopper in the form of plain text and use it to predict the dataset inputs. It generates a version of the trained ANN that can be saved in text format too, to be reused in predicting different newly generated entries. It is important to note that in case the trained data was remapped the prediction results must follow the same remapping process.

The testing of this ANN took place during the development phase on both data that were introduced to the code as training and brand-new data that were unseen by the ANN during the training phase. The testing was only trying to detect the time saved for using ANN predictions instead of running solar radiation simulations for the same number of iterations. Also, it was looking for the accuracy achieved by ANN predictions against simulation results.

Database build-up

The database used for this research consisted of solar radiation performance of buildings attributed to classification tags for them. The generation of urban configurations and initial creation of these classification tags was discussed in earlier publication (Lila & Lannon, 2019). The final stage of the classification tags consisted of 11 fragments to act as an indicator for the classified building by building height, area, typology, location and orientation within the neighbourhood, and the surrounding buildings heights. Figure 2 is showing an example of one building geometrical status and its classification tag.

The segments are replaced by integer indicators to make it easier for the training and prediction process. The 8 azimuth directions are replaced by numbers from 0 to 7 starting with east and ends with north east respectively. Other fragments that indicate the exposure to main street, exposure to urban voids, number of edges for the building typology and courts existence was replaced by a 0 or 1 indicator. It is important to note that these tags are classifying individual buildings and its solar radiation results within the urban context. Thus, urban configurations simulations results were calculated by summing the results of its individual buildings. Consequently, ANN code was trained on individual buildings results and the predictions of urban configurations was also a sum of buildings' prediction results.

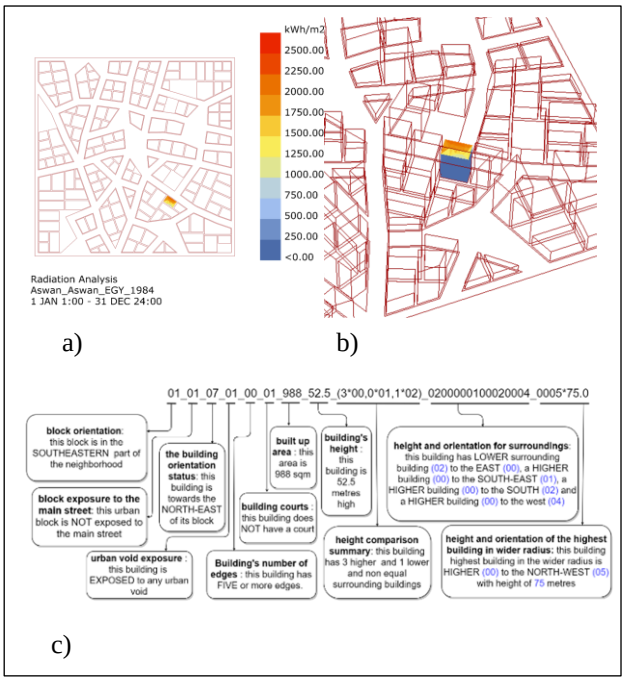

Figure 2 a) show case top view, b) show case perspective c) show case classification tag.

Solar radiation simulations were conducted using Ladybug Tools simulation package in Grasshopper (Sadeghipour & Pak, 2013). The simulation used the Egyptian Typical Meteorological Year (ETMY) weather file for Aswan city in southern Egypt (24.0889° N, 32.8998° E). Tested geometries surfaces were divided into an average of 15 by 15 metre grid cells for testing the direct solar radiation falling on these surfaces. The sky matrix was set to the Tregenza sky matrix (Lee et al., 2018) which is the default setting of the tool.

Testing phase

The testing conducted was for 3 different groups. The first group was 10,000 individual classification tags. The two other groups were focused on testing urban summed results for 100 configuration each with an average of 150 buildings. One group was selected with similar settings to the configurations used in the training phase. The other group was consisted 100 configurations randomly generated within the same context and changing the same parameters used to generate the training database. Another phase of the testing was to select an existing neighbourhood in the same weather data and test the accuracy of the prediction for some generated configuration withing its existing context.

The tests with unseen data were conducted with different sample sizes. The one shown in this paper is trained on a 200,000 tags database. To have a better understanding of the performance of the ANN on this sample, multiple settings were tested against the same three groups of predictions. The first ANN setting had the same settings as the previously tested ANNs. To minimize the time consumption there was only one hidden layer with seven nodes in this ANN. The training parameters were set to 250 iterations with an error threshold of .001. Alpha was set .015. Another ANN was set to have two hidden layers of seven and four neurons respectively in the order from input to output direction. The maximum number of iterations was set to 1,000 iterations. The rest of the settings have not been changed. Both ANNs were set to have 25% of input data as cross validation data.

The location of the case study was made based on the weather file used to build the training database which was the weather file of the city of Aswan in Upper Egypt with hot arid zone climate. The location of that existing neighbourhood boundary was selected from the city of New Aswan in southern Egypt (**Error! Reference source not found.**). This is one of the new cities commissioned by Egyptian government to accommodate the Egyptian population growth. Being a twin city and an extension to the original city it falls within the same climate and weather conditions. The total planned area of the city is 91.532112 km^2. It is targeting to accommodate 850,000

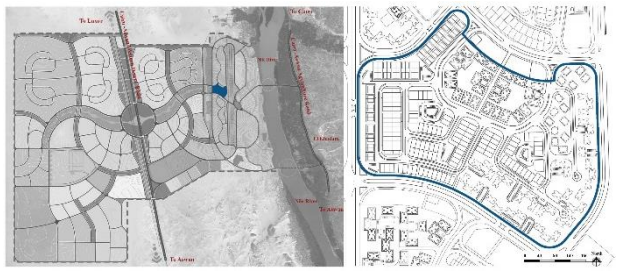

Figure 3 (To right) case study neighbourhood location in the city land use map. (To left) detailed urban design of the selected neighbourhood and boundary highlighted in colour edited by the researcher.

inhabitants by the year 2023 (New Urban Communities Authority at The Ministry of Housing. Utilities & Urban Communities [no date]).

Results

The second trained ANN came up with better mean square error for its last iterations and the reason it stopped was the reaching of the maximum iteration number while the first ANN was stopped based on achieving a mean square error closer to the error threshold. This was the reason it was selected to be tested with the existing neighbourhood's case study. Yet both were tested for the 3 groups of testing in its first phase. The time consumed for the first ANN was five hours of training and the ANN with two hidden layers took six hours and 24 minutes.

For the first phase of testing (Figure 5), the first group of randomly generated urban configurations achieved an R^2 value, coefficient of determination, of 0.793 and this was enhanced with the added hidden layer to be 0.832 for the correlation of predictions and simulation results.

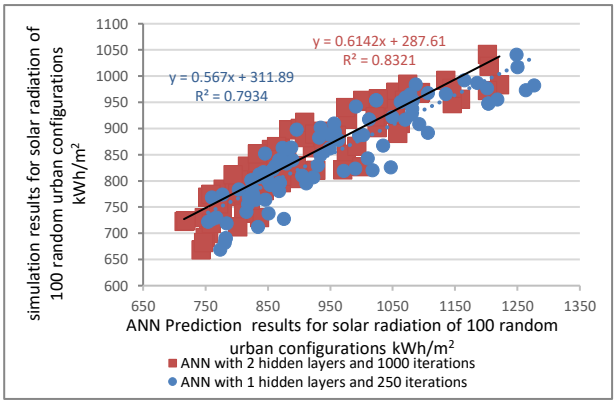

Figure 5 Correlation of simulation and prediction for 100 random urban configurations for 2 ANNs with 200,000 training dataset.

The other 100 configurations that had similar features of the training database resulted in a high value of correlation reaching to 0.99 for the ANN with two hidden layers and 0.96 R^2 value for the ANN with one hidden layer (Figure 7).

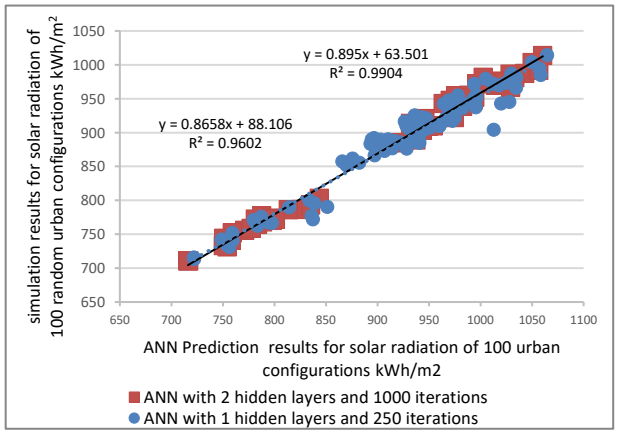

Figure 7 Correlation of simulation and prediction for 100 urban configurations for 2 ANNs with 200,000 training dataset

Finally, the 1,000 individual Tag results, shown in Figure 4, had a closer variation between the results of the two tested ANNs. The R^2 for the ANN with one hidden layer was 0.937. This was slightly enhanced for the ANN with two hidden layers by getting a value of 0.939 for the R^2 of the tested correlation.

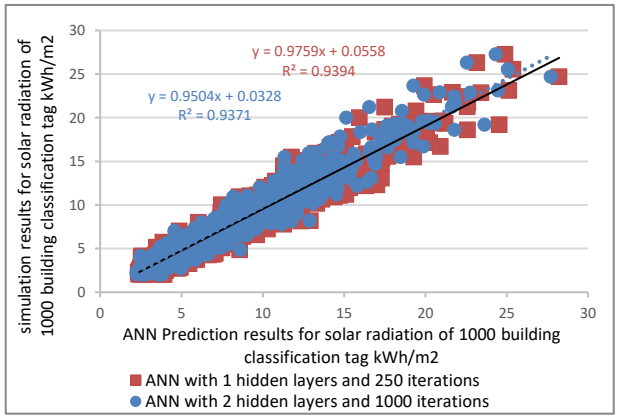

Figure 4 Correlation of simulation and prediction for 1,000 building classification tag for ANN 200,000 training dataset.

The second phase was testing the trained ANN on a newly generated classified geometry in an existing context to investigate the applicability of this feature. The tested urban configurations were generated through the same classification framework meaning each generated configuration has its individual buildings tagged and classified. This is how the ANN is predicting solar radiation for these tested configurations based on the same tag features used to train it. The test was about running a solar radiation simulation for 1,000 random urban configurations and comparing its results with our ANN prediction results for the same configurations

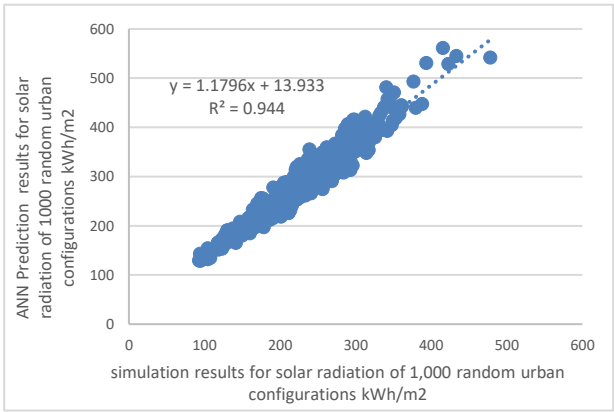

Figure 6 Correlation of simulation and prediction for 1,000 urban configurations for the existing case study iterations with the same trained network.

(Figure 6). The results have shown a high correlation between the two results. It had a 0.94 R^2 value for the correlation which is a significant positive correlation noting that the training data of this ANN node is generated from different settings, neighbourhood boundary and context conditions.

Discussion

The assessment of time-saving performance is calculated by comparing the time consumed by the ANN node with the time consumed by the conventional simulation methods. On a building level, ANN predictions consume 3% of the simulation time. This is due to the capability of predicting the whole set of building tags for a configuration in one run instead of simulating the buildings with their different context and getting the results for it.

It is important to note that the training time is not factored into these comparisons due to the difference in the number of buildings and configurations that can be done using these trained ANNs. Moreover, solar radiation simulation is one of the least time-consuming simulation aspects when compared to other simulation aspects like energy demand or daylighting performance. This finding result shows the potential of utilizing this classification method accompanied with ANN predictions to get even better time-performing frameworks that will reach even better time-saving results for other more time-consuming aspects of environmental performance.

The two tests with the 200,000-database had similar results for its correlation R^2. These high results were also accompanied with a similar correlation in the actual values between the prediction and simulation at the individual building level. This means that the error in prediction values follows almost the same percentage as shown in Figure 4. The high value of correlation with this simple ANN has shown the capability of utilizing this method of classification with ANN. This finding provides some support for the conceptual premise that applying ANN principles can achieve such a high accuracy of predictions with the benefit of saving time for the desired simulation and the reuse of the trained ANN nodes to be utilized with different case studies using the same weather file and location.

The urban scale testing had two groups, each of which had 100 urban configurations in it. The first group comprised urban configurations selected with shared urban inputs as the ones included in the database for training. The other group was randomly generated within the pool of iterations for this case study as the framework generation discussed in previous publications from the research team. The accuracy for predicting the solar radiation on the urban configuration was investigated by comparing the urban configuration simulation results against the outcome of the framework process and the ANN prediction. The result of the tag predictions is summed to get the final prediction for the solar radiation on the urban configuration. This number is compared with the simulation outcome.

The R^2 results for the similarly selected 100 configurations were 0.99 for the tested training sample as shown in Figure 7. This value was lower for the randomly generated 100 configurations predictions' correlations. Figure 5 shows that R^2 value result for the 200,000-training dataset was 0.83. This difference in correlation coefficient values for the two groups of urban configuration predictions can show the impact of the similarities between the tested configurations to the ones used in training. The choice of R^2 to assess the accuracy of prediction was caused by the aim of the analysis as this accuracy is merely looking for the prediction accuracy as a total without paying attention to the individual parameters impact on the prediction (Zikmund & Carr, 2000). The accuracy achieved in these tests have passed the significance level suggested by (Henseler et al., 2009) where it is important to mention that the level of a good R^2 value is rarely determined yet, These achieved accuracy levels can be found similar to other investigations of applying ANN methods on different sets of data and for different prediction goals in urban design context (Chan & Chau, 2019; Lin et al., 2020). Although these findings for prediction results on an urban scale seem in line with the findings of the building prediction scale as it shows a linear positive correlation, yet these correlations are not at the same level of accuracy. This highlights the need for further investigation on the relative input contribution on the prediction results. Moreover, this can be noted from the difference in values between prediction and simulation outcomes for the tested urban configurations, especially the random selected ones. A possible explanation of this might be the selection of the training database. The training database was built from similar groups of urban configurations and it was not selected from different feature groups in the available pool of iterations. This was due to the limitation of automating a random selection of the buildings in the database. Another explanation for this is the expected aggregation of the prediction error becoming clearer with the addition of the classification tag prediction results. These findings show that there is still room for improvement when it comes to using the prediction on an urban scale applying this classification method. However, there are some immediately dependable conclusions for the framework aiming for a proof of concept. These are based on the found linear positive correlation to utilize this node of ANN in the next optimization stage applying genetic algorithm principles to highlight the optimal solution in this pool of iterations.

Conclusion

This paper has shown different tests utilizing ANN principles to gauge its potential in predicting solar radiation performance when compared to simulation results along with consideration of the accuracy of predicting the performance at building and urban level and the potential time saved in the process. The Artificial Neural Network principles have been applied on the classified database of building classification tags and its attributed solar radiation results.

The trial of available ANN tools led to the use of a basic open-source code for a neural network to create a sequence that separates the training and prediction time.

The testing discussed in this paper were aimed at calculating the time saved by the ANN prediction in comparison with the time consumed for the simulation and it aimed to compare the results for both methods to

measure the accuracy achieved by the ANN prediction when compared to the simulation results acting as the benchmark for accuracy.

This method has already shown acceptable accuracy and it also opens the door for other simulation aspects to be considered utilizing the same method of depending on the individual building classified results to build a database for predicting urban configuration results.

This paper has discussed a stage of creating a proof-of-concept framework which adopt machine learning principles within its data flow to provide the capability of computational prediction of solar radiation performance aiming for this to save the simulation time consumption without sacrificing the accuracy achieved by the conventional simulation engines. To expand the scope, Other weather files will be used for the training and testing of this developed ANN and it will utilize more investigation for variables and its relative contribution on the prediction results. The significantly positive accuracy and time-saving results allowed the framework to go forward with its investigation and use these prediction results to answer the question about the capability of optimization in the early stages of design.

References

Ali, U., Shamsi, M. H., Alshehri, F., Mangina, E., & Donnell, J. O. (2018). *Application of Intelligent Algorithms for Residential Building Energy Performance Rating Prediction UCD Energy Institute , University College Dublin , Belfield , Dublin 4 , Ireland School of Computer Science and Informatics , University College Dublin (U*.

Arlot, S., & Celisse, A. (2010). A survey of cross-validation procedures for model selection. *Statistics Surveys*, *4*, 40–79. https://doi.org/10.1214/09-SS054

Beirão, J. (2012). CItyMaker; Designing Grammars for Urban Design. In *A+BE | Architecture and the Built Environment* (Vol. 2, Issue 5). Delft University of Technology, Faculty of Architecture, Department Architectural Engineering+ Technology, Department of Urbanism]. https://doi.org/10.7480/a+be.vol2.diss5

Biljecki, F., Ledoux, H., Stoter, J., & Zhao, J. Q. (2014). Formalisation of the level of detail in 3D city modelling. *Computers Environment and Urban Systems*, *48*, 1–15. https://doi.org/10.1016/j.compenvurbsys.2014.05.004

Cichocka, J. M., Migalska, A., Browne, W. N., & Rodriguez, E. (2017). *SILVEREYE – The Implementation of Particle Swarm Optimization Algorithm in a Design Optimization Tool* (pp. 151–169). Springer, Singapore. https://doi.org/10.1007/978-981-10-5197-5_9

Cui, Z., & Cai, X. (2013). Artificial Plant Optimization Algorithm. In *Swarm Intelligence and Bio-Inspired Computation* (pp. 351–365). Elsevier Inc. https://doi.org/10.1016/B978-0-12-405163-8.00016-8

Dounis, A. I., Science, A., & Piraeus, T. E. I. (2014). Artificial intelligence for energy conservation in buildings Artificial intelligence for energy conservation in buildings. *Advanes in Building Energy Research*. https://doi.org/10.3763/aber.2009.0408

Greenberg, E., & Erdine, E. (2014). Computing the Urban Block - Local Climate Analysis and Design Strategies. *Thompson, Emine Mine (Ed.), Fusion - Proceedings of the 32nd ECAADe Conference - Volume 1, Department of Architecture and Built Environment, Faculty of Engineering and Environment, Newcastle upon Tyne, England, UK, 10-12 September 2014, Pp. 145-152*.

Henseler, J., Ringle, C. M., & Sinkovics, R. R. (2009). The use of partial least squares path modeling in international marketing. *Advances in International Marketing*, *20*, 277–319. https://doi.org/10.1108/S1474-7979(2009)0000020014

Hillier, B. (2007). *Space is the machine: a configurational theory of architecture*. Space Syntax.

Jain, A. K., Mao, J., & Mohiuddin, K. M. (1996). Artificial neural networks: A tutorial. In *Computer* (Vol. 29, Issue 3, pp. 31–44). https://doi.org/10.1109/2.485891

Koenig, R. (2011). Generating urban structures: a method for urban planning supported by multi-agent systems and cellular automata. *Przestrzeń i Forma*, *nr 16*, 353–376. https://www.infona.pl/resource/bwmeta1.element.baztech-article-BPS1-0047-0065

Krarti, M. (2003). An overview of artificial intelligence-based methods for building energy systems. *Journal of Solar Energy Engineering, Transactions of the ASME*, *125*(3), 331–342. https://doi.org/10.1115/1.1592186

Lee, E. S., Geisler-Moroder, D., & Ward, G. (2018). Modeling the direct sun component in buildings using matrix algebraic approaches: Methods and validation. *Solar Energy*, *160*, 380–395.

Lila, A. M. H., & Lannon, S. (2019). Classifying urban geometry impact on solar radiation. In V. Corrado, E. Fabrizio, A. Gasparella, & F. Patuzz (Eds.), *the International Building Performance Simulation Association (IBPSA) 2019 16th Conference*.

Martin, L., & March, L. (1972). *Urban Space and Structures. Cambridge Urban and Architectural Studies, No. 1*. Cambridge University Press.

Naboni, E. (2014). Integration of Outdoor Thermal and Visual Comfort in Parametric Design. *30th International PLEA Conference, December*, 1–10.

Nault, E., Rey, E., & Andersen, M. (2016). A Multi-Criteria Decision-Support Workflow for Early-Stage Neighborhood Design based on Predicted Solar Performance. *PLEA 2016, 36th International*

Conference on Passive and Low Energy Architecture. Cities, Buildings, People: Towards Regenerative Environments. https://infoscience.epfl.ch/record/218840/files/1093-Nault01.pdf?version=1

New Urban Communities Authority at The Ministry of Housing. Utilities & Urban Communities. (n.d.). *Home - New Aswan*. Retrieved March 22, 2020, from http://www.newcities.gov.eg/english/New_Communities/Aswan/default.aspx

Pedregosa, F., Varoquaux, G., Gramfort, A., & Michel, V. (2010). *3.1. Cross-validation: evaluating estimator performance — scikit-learn 0.22.1 documentation*. https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation

Picco, M., & Marengo, M. (2015). On the Impact of Simplificationson Building Energy Simulation for Early Stage Building Design. *Journal of Engineering and Architecture*, *3*, 66–78. https://doi.org/10.15640/jea.v3n1a7

Pinto Duarte, J., Ducla-Soares, G., & Sampaio, A. Z. (2005). Urban Grammars: Towards Flexible Urban Design. *Digital Design: The Quest for New Paradigms [23nd ECAADe Conference Proceedings*, 491–500. http://papers.cumincad.org/cgi-bin/works/Show?2005_491

PROVING GROUND. (2018). *Lunch-Box*. https://provingground.io/tools/lunchbox/

Sadeghipour, M. r, & Pak, M. (2013). Ladybug: a Parametric Environmental Plugin for Grasshopper To Help Designers Create an Environmentally-Conscious Design. *13th Conference of International Building Performance Simulation Association*, 3129–3135. http://www.ibpsa.org/proceedings/bs2013/p_2499.pdf

Schumacher, P. (2009a, July). Parametricism: A New Global Style for Architecture and Urban Design. *Architectural Design*, *79*(4), 14–23. https://doi.org/10.1002/ad.912

Schumacher, P. (2009b, November). Parametric Patterns. *Architectural Design*, *79*(6), 28–41. https://doi.org/10.1002/ad.976

Schwarz, N. (2010). Urban form revisited-Selecting indicators for characterising European cities. *Landscape and Urban Planning*, *96*(1), 29–47. https://doi.org/10.1016/j.landurbplan.2010.01.007

Scott Davidson. (2017). *Grasshopper - algorithmic modeling for Rhino*.

Stewart, I. D., & Oke, T. R. (2012). Local Climate Zones for Urban Temperature Studies. *Bulletin of the American Meteorological Society*, *93*, 1879–1900. https://doi.org/10.1175/bams-d-11-00019.1

Tamke, M., Nicholas, P., & Zwierzycki, M. (2018). Machine learning for architectural design: Practices and infrastructure. *International Journal of Architectural Computing*, *16*(2), 123–143. https://doi.org/10.1177/1478077118778580

The Codacus. (2017). *Build Neural Network From Scratch in Python (no libraries) | The Codacus*. https://thecodacus.com/neural-network-scratch-python-no-libraries/

Trigaux, D., Allacker, K., & Troyer, F. De. (2014). A simplified approach to integrate energy calculation in the Life Cycle Assessment of neighbourhoods. *30th International PLEA Conference,Volume: Sustainable Habitat for Developing SocietiesPLEA*. https://lirias.kuleuven.be/handle/123456789/481910

Varoquaux, G., Buitinck, L., Louppe, G., Grisel, O., Pedregosa, F., & Mueller, A. (2015). Scikit-learn. *GetMobile: Mobile Computing and Communications*, *19*(1), 29–33. https://doi.org/10.1145/2786984.2786995

Wortmann, T. (2017). OPOSSUM Introducing and Evaluating a Model-based Optimization Tool for Grasshopper. In P. Janssen, P. Loh, A. Raonic, & M. A. Schnabel (Eds.), *Protocols, Flows and Glitches, Proceedings of the 22nd International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA)* (pp. 283–293). www.food4rhino.com

Wortmann, T., & Nannicini, G. (2016). Black-box optimisation methods for architectural design. *CAADRIA 2016*, Volume: 177-186. https://www.researchgate.net/publication/299594622_Black-box_optimisation_methods_for_architectural_design

Zhao, H. X., & Magoulès, F. (2012). A review on the prediction of building energy consumption. *Renewable and Sustainable Energy Reviews*, *16*(6), 3586–3592. https://doi.org/10.1016/j.rser.2012.02.049

Zikmund, W. G., & Carr, G. (2000). Business Research Methods. 7th. In *USA, Dryden*.