# Performance of Neural Networks vs. Radial Basis Functions When Forming a Metamodel for Residential Buildings

Philip Symonds, Jon Taylor, Zaid Chalabi, Michael Davies

*Abstract*—Average temperatures worldwide are expected to continue to rise. At the same time, major cities in developing countries are becoming increasingly populated and polluted. Governments are tasked with the problem of overheating and air quality in residential buildings. This paper presents the development of a model, which is able to estimate the occupant exposure to extreme temperatures and high air pollution within domestic buildings. Building physics simulations were performed using the EnergyPlus building physics software. An accurate metamodel is then formed by randomly sampling building input parameters and training on the outputs of EnergyPlus simulations. Metamodels are used to vastly reduce the amount of computation time required when performing optimisation and sensitivity analyses. Neural Networks (NNs) have been compared to a Radial Basis Function (RBF) algorithm when forming a metamodel. These techniques were implemented using the PyBrain and scikit-learn python libraries, respectively. NNs are shown to perform around 15% better than RBFs when estimating overheating and air pollution metrics modelled by EnergyPlus.

*Keywords*—Neural Networks, Radial Basis Functions, Metamodelling, Python machine learning libraries.

## I. INTRODUCTION

IN the coming decades, global temperatures are on average expected to rise by up to 1.4-3°C [1]. This will result in increased mortality rates due to extreme weather and heatwave events [2]. The populations of major cities in developing countries are rising rapidly and governments are faced with housing shortages. Air pollution within the major cities of developing countries is also a major problem. Particulate matter ($PM_{2.5}$) levels in Delhi, for example, are among the highest in the world with an average concentration of 153 $\mu$g/m$^3$ [3]. Housing can play an important role in modifying population exposure to extreme temperatures and high air pollution levels. There is a growing need for a model, which can indicate the ideal building design specification, for low income housing, under various climate and air quality scenarios.

There are a large number of parameters, which may influence indoor temperature and air pollution risks. Indoor temperatures and pollution levels may be moderated by; building geometry and orientation; fabric characteristics such as the thermal mass and conductivity of walls, windows,

P. Symonds is with the Institute of Environmental Design and Engineering, University College London, Central House, 14 Woburn Place, London, WC1H 0NN, UK (e-mail: p.symonds@ucl.ac.uk).

J. Taylor, Z. Chalabi and M. Davies are with the Institute of Environmental Design and Engineering, University College London, Central House, 14 Woburn Place, London, WC1H 0NN, UK.

floors, and roofs [4]; natural or purpose-provided ventilation [5]; occupant behaviour [6]; adaptation measures [7], [8]; and climate, including region [9] or locations within an Urban Heat Island (UHI) [10]. In order to capture this wide range of variants, many simulations are required. The United States Department of Energy building simulation software, EnergyPlus [11], is used for this purpose. This enables the thermal and contaminant transport physics to be modelled accurately. EnergyPlus simulations are CPU time intensive and it is advisable to replicate them with a metamodel. This allows multi-criteria optimisation to be performed, which can help to identify desirable building characteristics for a specific location or climate.

Supercomputing facilities are used to run many thousands of EnergyPlus simulations in parallel. There are various machine learning techniques, which can be used when forming a metamodel. In this paper, the performance of neural networks (NNs) is compared to that of a Radial Basis Function (RBF) algorithm.
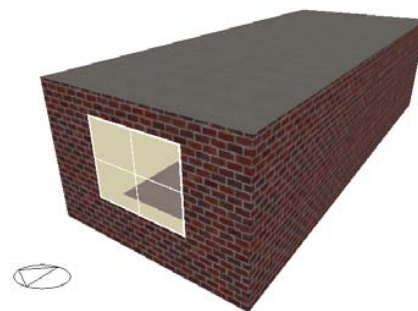
## II. METHODS



Fig. 1 Basic single room building: Image produced using the DesignBuilder software

Low income housing is likely to be basic and so a simple, single room building has been modelled, as shown in Fig. 1. A metamodelling approach has been used as this is proven to give a reliable approximation of complex models [12]–[14]. All steps of metamodel generation are performed using open source software. This includes several python packages and the EnergyPlus building simulation software. The workflow of metamodel creation takes the following steps:

TABLE I
BUILDING INPUT PARAMETERS

| Parameter | Distribution(Range) | Units |
|---|---|---|
| Length | U(3-10) | m |
| Width | U(3-10) | m |
| Height | U(1.6-3.1) | m |
| Number of exposed facades | U(1-4) | NA |
| Orientation | U(0-360) | ° |
| Permeability | N($\mu$ =20,$\sigma$ =10) | $m^3/h/m^2$ @ 50 Pa |
| Wall U-value | U(0.2-1.8) | $W/m^2/K$ |
| Window U-value | U(1.8-3.8) | $W/m^2/K$ |
| Roof U-value | U(0.13-2.3) | $W/m^2/K$ |

Input variables and the corresponding value ranges in the model relating to the built form. The continuous variables are sampled from; Truncated Normal distributions (N) with mean ($\mu$) and standard deviation ($\sigma$), and Uniform (U) distributions. The calculation of U-values is performed using the thermal resistance of construction materials as described in (1).

(A) Generation of EnergyPlus input building files
(B) Building simulation using EnergyPlus version 8.1
(C) Read metrics of interest from EnergyPlus output files
(D) Metamodel training
(E) Metamodel validation

All of the above steps need to be performed for each location or climate. To begin with, we aim to select several locations that cover a range of climatic regions including hot and humid, hot and dry, and wet and cold. The focus will be on locations in developing countries where the population is rising rapidly.

### A. Generation of EnergyPlus Input Definition Files

EnergyPlus input files are generated using an in house python tool, EnergyPlus Generator 2 (EPG2). This software is supplemented with the pyDOE python package, which has the Latin Hypercube random sampling method [15]. This experimental design is chosen such that similar runs are never repeated, thus maximising the available parameter space. The building input parameters are selected according to the ranges and distributions shown in Table I. The only occupancy variable that is randomly sampled is the temperature threshold at which occupants open the windows. Values for this variable are taken from a normal distribution with a mean of 22°C and with a standard deviation of 5°C. Normal distributions are truncated such that un-physical values are not produced. Simulated buildings have no heating or air conditioning such that only the construction properties are varied. In this way, we can reduce temperature and air quality risks without the requirement of energy consumption from heating or cooling systems. U-values are a measure of the thermal transmittance of building construction elements. They are calculated by summing over the inverse of the thermal resistances, $R_k$, for each construction layer, $k$:

$$\text{U-value} = \sum_k \frac{1}{R_k} \qquad (1)$$

### B. Building Simulation

EnergyPlus is a commonly used open source building simulation software developed by the United States Department of Energy [11]. It is able to model the thermal

TABLE II
MODEL OUTPUTS

| Parameter | Symbol | Unit |
|---|---|---|
| Mean Maximum Daytime Temperature | $T_{\text{day}}^{\max \alpha}$ | °C |
| Mean Minimum Night Temperature | $T_{\text{night}}^{\min \beta}$ | °C |
| Time Above Temperature Mortality Threshold | $t > T_{\text{hot}}{}^{\kappa}$ | hours |
| Time Below Temperature Mortality Threshold | $t < T_{\text{cold}}{}^{\kappa}$ | hours |
| Relative Humidity | RH | % |
| $PM_{2.5}$ indoor/outdoor ratio | I/O* | NA |

Temperature and air pollution metrics used with corresponding symbols and units.
$^{\alpha}T_{\text{day}}^{\max}$ is the mean of the maximum temperatures recorded in the building during day time hours (08:00-22:00) over the course of the year.
$^{\beta}T_{\text{night}}^{\min}$ is the mean of the minimum temperatures recorded in the building during night time hours (22:00-08:00) over the course of the year.
$^{\kappa}T_{\text{hot}}$ and $T_{\text{cold}}$ vary depending on the region. In this study a critical temperature of 29°C is used for overheating risks and 19°C for risk due to cold. These values were taken from the ISOTHERM study [17], which looked at how temperatures coincided with an increase in mortality rates.
*The deposition of a generic contaminant within the building is modelled at a fractional rate of $1.0 \times 10^{-5}$ s$^{-1}$ [11].

and air infiltration physics over the course of a specified time period. In our case, simulations are run over the course of a year to capture the risks of both hot and cold weather. Each simulation requires a weather file containing hourly weather data and a building input file. Weather files are available from various sources. In this study, a Delhi weather file was used provided by ISHRAE [16]. Individual simulations take around half a minute for a simple box model. UCL's high performance computing facilities, Legion, have been used in order to run simulations in parallel. Legion has 7,500 cores, which makes running many thousands of simulations within an hour possible. In this study, 1,000 simulations have been run for training with an independent sample of 200 used for validation (testing) of the machine learning algorithms.

### C. Reading Metrics of Interest

Each EnergyPlus simulation produces an output file containing hourly information, such as indoor temperatures, relative humidity and air contamination ($PM_{2.5}$) concentrations. Output files are about 1 MB in size and are post processed in order to calculate the output variables shown in Table II.

### D. Metamodelling Techniques

Many different machine learning algorithms can be used when forming a metamodel. Implementations are available in many different statistical packages such as MATLAB, Stata and R. In this work, we have used algorithms from python libraries. The two machine learning algorithms studied are neural networks and a radial basis function. These methods are examples of supervised machine learning and are able to reproduce non-linear and non-monotonic relations between input and output variables.

*1) Neural Networks:* comprise of a set of neurons that connect input and output model variants [18]. The inputs, outputs and hidden neurons are connected to one another by synapses that carry weights and biases, which determine the strength of the connections. Each neuron is associated with a

transfer function. Linear transfer functions are often used at the input and output layers, whilst connections to hidden neurons are usually charactersed by a sigmoid transfer function. With the inputs to a hidden neuron denoted as $\{x_1, x_2, \ldots, x_n\}$, and the regression weights, $w_i$, the output of a neuron, $y$, is given by:

$$y = \frac{1}{1 + e^{-\eta}} \qquad (2)$$

where $\eta = \sum_{i=1\ldots n} w_i x_i + \beta$, with $\beta$ the bias value. The weights and the bias values are updated in the fitting of the output metrics over a pre-defined number of training epochs.

The Python-Based Reinforcement Learning, Artificial Intelligence and Neural Network (PyBrain) Library [19] is used to implement the NN algorithm. This software is flexible and allows the user to construct the network architecture in terms of the number of layers and neurons, the connections and the transfer functions. The PyBrain implementation of NNs is able to fit all output metrics simultaneously. The number of training epochs can be specified or there is also the option to allow the training to continue until convergence. Training until convergence can be fairly time consuming and can result in over training, therefore, 100 training epochs are used. Networks were set up with various architectures having either one or two hidden layers with each layer containing 4-22 neurons. When two hidden layers are used, the same number of neurons are present in both layers.

Two training algorithms are implemented, the backwards propagation (Backprop) and the reverse propagation minus (RProp-) trainers. These algorithms work by propagating the errors backwards through the network with respect to the training weights and biases [20]. The main difference between the two algorithms is that the RProp- trainer doesn't have weight backtracking [21], meaning all training samples have the same weight.

*2) Radial Basis Function:* is a another method used for parameter estimation and is a type of support vector machine (SVM) [22]. RBFs are real valued functions, which depend on the radial distance, $r = \sqrt{\sum(x_0 - x_i)^2}$, to a point of origin, $x_0$. An output metric is then given by:

$$y = \sum_{i=1\ldots n} v_i \phi(r) \qquad (3)$$

A Gaussian function is commonly used to represent $\phi(r)$, e.g. $\phi(r) = e^{-\epsilon r^2}$. Training is performed in order to determine the parameters $v_i$, $\epsilon$ and $x_0$. A least squares method is used to do this such that metamodel output values are close to those of the true original model.

The SVM python module from the scikit-learn package [23] was used to implement the RBF method. This implementation only allows one output to be trained at a time, which can make training multiple outputs more time consuming than using a NN. A comparison of time effectiveness of the NN and RBF methods is made in sections III and IV. The SVM training procedure requires two input parameters to be provided; $\epsilon$ and $x_0$ (denoted $\gamma$ and $C$ in scikit-learn, respectively). The $\gamma$ parameter defines the influence of a single training sample and can be seen as the inverse of the radius of influence of the samples selected by the model as support vectors.

TABLE III
OPTIMAL $\gamma$ AND $C$ VALUES FOR RBF TRAINING

| Symbol | $\gamma$ | $C$ | $R^2$ |
|---|---|---|---|
| $T_{\text{day}}^{\text{max}}$ | 0.01 | 100.0 | 0.78 |
| $T_{\text{night}}^{\text{min}}$ | 0.001 | 100000.0 | 0.93 |
| $t > T_{\text{hot}}$ | 0.001 | 100000.0 | 0.64 |
| $t < T_{\text{cold}}$ | 0.01 | 1000.0 | 0.79 |
| RH | 0.01 | 100.0 | 0.72 |
| I/O | 0.001 | 100000.0 | 0.94 |

Optimal $\gamma$ and $C$ values determined using the scikit-learn GridSearch feature. $R^2$ is a score of the goodness of fit, described in (5).

The $C$ parameter trades off the mis-calculation of training samples against the simplicity of the decision surface. A low $C$ makes the decision surface smooth, whilst a high $C$ gives the model freedom to select more samples as support vectors. In order to determine optimal values for $\gamma$ and $C$, scikit-learn has a GridSearch feature. A logarithmic range of $\gamma$ $\{1 \times 10^{-1}, 1 \times 10^{-2}, 1 \times 10^{-3}\}$ and $C$ $\{1, \ldots, 1 \times 10^{-5}\}$ values are provided and the values which yield the lowest training error are selected. Table III presents the optimal $\gamma$ and $C$ values for each output variable.

*E. Metamodel Validation*

Validation of the metamodelling techniques involves passing an independent set of inputs through each trained metamodel. The outputs produced by each metamodel are then compared to the outputs produced by the true EnergyPlus model.

Several goodness of fit metrics, as used in previous studies [12], allow the accuracy of the algorithms to be compared. These metrics include the Root Mean Square Error ($RMSE$), the coefficient of determination ($R^2$), and the Maximal Absolute Error ($MAE$) and are defined as follows:

$$RMSE = \sqrt{\frac{1}{m} \sum_{j=1\ldots m} (\hat{y}_j - y_j)^2} \qquad (4)$$

$$R^2 = 1 - \frac{\sum_{j=1\ldots m}(\hat{y}_j - y_j)^2}{\sum_{j=1\ldots m}(\bar{y}_j - y_j)^2} \qquad (5)$$

$$MAE = \max(|\hat{y}_1 - y_1|, \ldots, |\hat{y}_m - y_m|) \qquad (6)$$

where $y_j$ represents the original model outputs (EnergyPlus), $\bar{y}$ the mean original model output value, $\hat{y}_j$ the metamodel output, and $m$ the number of testing samples. These performance metrics are calculated for each output variable. Since the metamodels are fit to several inputs and outputs, all variables are normalised such that they have a mean of zero, and a standard deviation of one prior to training. This ensures that the algorithms are not biased towards a particular variable and allows the performance of individual outputs to be compared. The sum of the normalized error of each output metric gives the overall performance for a candidate metamodel.

III. RESULTS

Table IV compares the performance of the best performing NN to the RBF. Results are shown for individual output
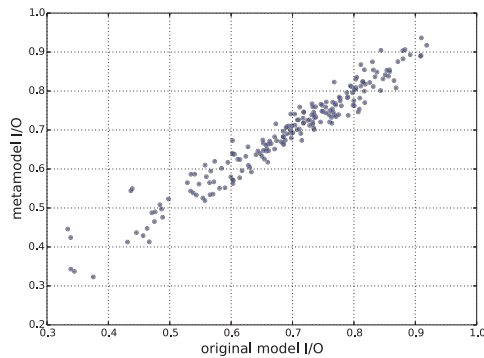
Fig. 2 Original EnergyPlus simulation I/O values plotted against the outputs of the best performing metamodel, this being the RBF
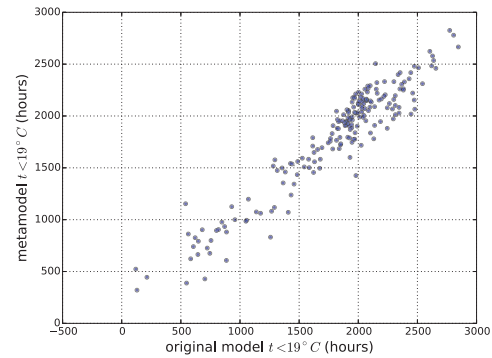


Fig. 4 Original EnergyPlus simulation $t < 19°$C values plotted against the outputs of the best performing metamodel. In this case it was the NN with 2 layers with 15 neurons in each layer
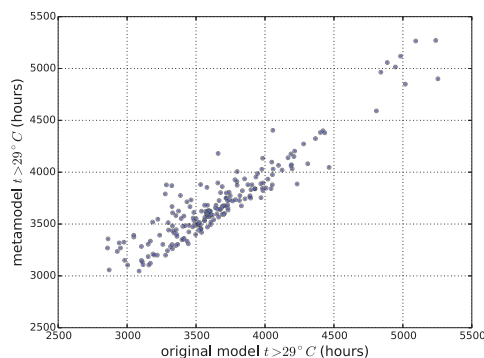


Fig. 3 Original EnergyPlus simulation $t > 29°$C values plotted against the outputs of the best performing metamodel. In this case it was the NN with 2 layers with 11 neurons in each layer

variables for the building as well as the sum over all output variables. Figs. 2-4 show the I/O, $t > 29°$C, and $t < 19°$C values simulated using EnergyPlus plotted against the outputs using the best performing metamodel. Comparisons of the performance of various NN architectures trained with the BackProp and the RProp- algorithms are made in Appendix A.

In general, the overall performance of the best NN is around 15% better than an RBF in terms of $RMSE$, $R^2$ and $MAE$ values. An individual NN can be trained 90% faster than an RBF. NNs trained using the BackProp algorithm were observed to perform better than networks trained using the RProp- algorithm.

## IV. Discussion and Future Work

This work compares the performance of two machine learning methods using two different python packages. PyBrain was used to implement neural networks whilst scikit-learn provided the radial basis function algorithm. The results indicate that an NNs perform around 15% better than an RBF in terms of $RMSE$ and training is quicker, as all output metrics are trained in one go.

There are some advantages and disadvantages to both methods which should be considered when when deciding

which one to use. The PyBrain implementation allows a lot of flexibility when it comes to designing the NNs architecture. This flexibility comes with complexity, and finding the best architecture and training algorithm can be a time consuming procedure. Different networks also give better performance for different output variables, so if the absolute best performance is desired, multiple networks must be used. Scikit-learn's SVM package offers less in terms of flexibility and hence is slightly easier to use. The grid search method allows the best training parameters for the RBF to be determined. This procedure is CPU time intensive as training has to be performed for a range of the parameters, $\gamma$ and $C$. Another issue is that only pre-defined values can be chosen for these training parameters. This means that the absolute best values are unlikely to be selected or will take a long time to find.

The next step for this work will be to validate the metamodel against monitoring data from at least one location. Occupancy parameters such as window opening temperature thresholds are likely to be an important factor. These occupancy parameters will be estimated during the validation procedure and an uncertainty assigned using a sensitivity analysis. Once metamodel validation has been performed, the model will be extended to cover various locations and climates. Ideally, we will cover all of the world's climates as classified by Koppen [24]. Initially, models for several cities in the developing world will be built. Optimisation of building construction parameters will then be performed. This will help to minimise the risks posed by indoor air quality and extreme temperatures for each climate.

## Appendix A
### Neural Network Performance

Table V compares the overall performance of the various NN architectures using the BackProp and RProp- training algorithms.

TABLE IV
PERFORMANCE OF NNS VS. RBF

| Model | I/O | $T_{\text{day}}^{\max}$ | $T_{\text{night}}^{\min}$ | RH | $t > 29°C$ | $t < 19°C$ | Total |
|---|---|---|---|---|---|---|---|
| $RMSE$ values | | | | | | | |
| NN (2, 13) | 0.29 | 0.40 | 0.23 | 0.37 | 0.40 | 0.31 | 2.00 |
| RBF | 0.24 | 0.43 | 0.23 | 0.46 | 0.53 | 0.41 | 2.29 |
| $R^2$ values | | | | | | | |
| NN (2, 13) | 0.92 | 0.84 | 0.95 | 0.87 | 0.84 | 0.90 | 5.31 |
| RBF | 0.94 | 0.77 | 0.95 | 0.76 | 0.61 | 0.81 | 4.84 |
| $MAE$ values | | | | | | | |
| NN (2, 16) | 0.81 | 1.54 | 0.81 | 1.49 | 1.60 | 0.94 | 7.19 |
| RBF | 0.90 | 1.59 | 0.98 | 1.81 | 2.63 | 1.45 | 9.37 |
| Training CPU time (minutes:seconds) | | | | | | | |
| NN (2, 13) | NA | NA | NA | NA | NA | NA | 01:58.2 |
| RBF | 03:02.5 | 02:00.8 | 02:30.8 | 01:57.7 | 02:12.3 | 02:09.7 | 12:53.8 |

Comparison of the best NN (layers, neurons) to that of a RBF in terms of the performance metrics given in (4)-(6) and in terms of training time in minutes and seconds.

TABLE V
TOTAL PERFORMANCE OF VARIOUS NN ARCHITECTURES USING
BACKWARD AND REVERSE PROPAGATION TRAINING

| Training Algo. | BackProp | | Reverse | |
|---|---|---|---|---|
| Hidden Layers | 1 | 2 | 1 | 2 |
| NN(8) | 2.45 | 2.13 | 3.01 | 3.19 |
| NN(9) | 2.31 | 2.13 | 3.89 | 2.57 |
| NN(10) | 2.30 | 2.09 | 2.87 | 2.88 |
| NN(11) | 2.26 | 2.11 | 2.87 | 3.65 |
| NN(12) | 2.22 | 2.06 | 3.62 | 2.86 |
| NN(13) | 2.20 | 1.99 | 2.88 | 2.51 |
| NN(14) | 2.26 | 2.21 | 2.98 | 2.57 |
| NN(15) | 2.19 | 2.04 | 3.17 | 2.51 |
| NN(16) | 2.17 | 2.02 | 3.03 | 2.52 |

Total $RMSE$ value for several NN(neurons) architectures using the backwards propagation (BackProp) and the reverse propagation minus (RProp-) training algorithm. Results for networks with one or two hidden layers are shown.

REFERENCES

[1] D. J. Rowlands et al. Broad range of 2050 warming from an observationally constrained large climate model ensemble. *Nature Geoscience*, 5:256–260, 03/2012 2012.
[2] S. Hajat, S. Vardoulakis, C. Heaviside, and B. Eggen. Climate change effects on human health: projections of temperature-related mortality for the uk during the 2020s, 2050s and 2080s. *Journal of Epidemiology and Community Health*, 2014.
[3] World Health Organisation. Ambient (outdoor) air pollution in cities database 2014.
[4] A. Mavrogianni, P. Wilkinson, M. Davies, P. Biddulph, and E. Oikonomou. Building characteristics as determinants of propensity to high indoor summer temperatures in London dwellings. *Building and Environment*, (55):117–30, 2012.
[5] J. Taylor, A. Mavrogianni, M. Davies, P. Das, C. Shrubsole, and P. Biddulph. Understanding and mitigating overheating and indoor PM2.5 risks using coupled temperature and indoor air quality models. *Building Services Engineering Research and Technology*, (0143624414566474), 2015.
[6] A. Mavrogianni, M. Davies, J. Taylor, Z. Chalabi, P. Biddulph, and E. Oikonomou. The impact of occupancy patterns, occupant-controlled ventilation and shading on indoor overheating risk in domestic environments. *Building and Environment*, (78):183198, 2013.
[7] S. Porritt and P. Cropper. Heat wave adaptations for UK dwellings and introducing a retrofit toolkit. *International Journal of Disaster Resilience in the Built Environment*, (4:3):269–286, 2010.
[8] R. Gupta and M. Gregg. Preventing the overheating of English suburban homes in a warming climate. *Building Research & Information*, (41):281–300, 2013.
[9] J. Taylor, M. Davies, A. Mavrogianni, Z. Chalabi, P. Biddulph, and E. Oikonomou. The relative importance of input weather data for indoor overheating risk assessment in dwellings. *Building and Environment*, (76):81–91, 2014.
[10] E. Oikonomou, M. Davies, A. Mavrogianni, P. Biddulph, P. Wilkinson, and M. Kolokotroni. Modelling the relative importance of the urban heat island and the thermal quality of dwellings for overheating in London. *Building and Environment*, (57):223–38, 2012.
[11] US-DoE. EnergyPlus V8. 2013.
[12] L. Van Gelder, P. Das, H. Janssen, and S. Roels. Comparative study of metamodelling techniques in building energy simulation: Guidelines for practitioners. *Simulation Modelling Practice and Theory*, (49):245–57, 2014.
[13] R. E. Edwards. Predicting future hourly residential electrical consumption: A machine learning case study. *Energy Buildings*, 2012.
[14] B. Eisenhower, Z. ONeill, S. Narayanan, V. A. Fonoberov, and I. Mezi. A methodology for meta-model based optimization in building energy models. *Energy and Buildings*, (47):292–301, 2012.
[15] B. Tang. Orthogonal Array-Based Latin Hypercubes. *Journal of the American Statistical Association*, 2012.
[16] Indian Society of Heating Refrigerating and Air Conditioning Engineers. New delhi weather file.
[17] A. J. McMichael et al. International study of temperature, heat and urban mortality: the 'isothurm' project. *International Journal of Epidemiology*, 37(5):1121–1131, 2008.
[18] W. S. McCulloch and W. Pitts. Neurocomputing: Foundations of research. chapter A Logical Calculus of the Ideas Immanent in Nervous Activity, pages 15–27. MIT Press, Cambridge, MA, USA, 1988.
[19] T. Schaul et al. PyBrain. *Journal of Machine Learning Research*, 2010.
[20] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA, 1988.
[21] C. Igel and M. Hüsken. Empirical evaluation of the improved rprop learning algorithms. *Neurocomputing*, 50:105–123, 2003.
[22] D. S. Broomhead and D. Lowe. Multivariable Functional Interpolation and Adaptive Networks. *Complex Systems 2*, pages 321–355, 1988.
[23] F. Pedregosa et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
[24] M. C. Peel, B. L. Finlayson, and T. A. McMahon. Updated world map of the kppen-geiger climate classification. *Hydrology and Earth System Sciences*, 11(5):1633–1644, 2007.