Constructing Large Scale Surrogate Models from Big Data and Artificial Intelligence

Richard E. Edwards[a], Joshua New[b,*], Lynne E. Parker[a], Borui Cui[b], Jin Dong[b]

[a] *University of Tennessee, Knoxville, TN USA 37996*
[b] *Oak Ridge National Laboratory, Oak Ridge, TN USA 37831*

Abstract

EnergyPlus is the U.S. Department of Energy's flagship whole-building energy simulation engine and provides extensive simulation capabilities. However, the computational cost of these capabilities has resulted in annual building simulations that typically requires 2–3 minutes of wall-clock time to complete. While EnergyPlus's overall speed is improving (EnergyPlus 7.0 is 25–40% faster than EnergyPlus 6.0), the overall computational burden still remains and is the top user complaint. In other engineering domains, researchers substitute surrogate or approximate models for the computationally expensive simulations to improve simulation and reduce calibration time. Previous work has successfully demonstrated small-scale EnergyPlus surrogate models that use 10–16 input variables to estimate a single output variable. This work leverages feed forward neural networks and Lasso regression to construct robust large-scale EnergyPlus surrogate models based on 3 benchmark datasets that have 7–156 inputs. These models were able to predict 15-minute values for most of the 80–90 simulation outputs deemed most important by domain experts within 5% (whole building energy within 0.07%) and calculate those results within 3 seconds, greatly reducing the required simulation runtime for relatively close results. The techniques shown here allow any software to be approximated by machine learning in a way that allows one to quantify the trade-off of accuracy for execution time.

*Corresponding author
Email address: newjr@ornl.gov (Joshua New)

Keywords: Machine Learning, EnergyPlus, Building simulation, Energy modeling, Surrogate model

1. Introduction

1.1. Background of research

It is estimated that there were 4.7 million commercial buildings and 114 million residential buildings in the U.S. in 2008, which consumed 39% of the U.S. primary energy (72% of the electrical energy), more than the industrial or transportation sectors [1]. Building energy efficiency constitutes the low-hanging fruit for slight to moderate reductions in energy and corresponding greenhouse gas emissions.

A central challenge in building energy efficiency is to realistically model the energy-related physics of an individual building. This capability is necessary to reliably project how specific policy decisions or retrofit packages would help meet national energy targets or maximize return-on-investment. This challenge is complicated by the fact that individual buildings, unlike cars or airplanes, are manufactured in the field and vary greatly from what may be considered a prototypical building. Since most whole-building simulation engines, such as EnergyPlus, have thousands of very specific required inputs, most of these engines suffer greatly from the user expertise, time, and associated costs required to create an accurate virtual model of a real-world building. Moreover, this manual process of tuning a model to measured data is neither repeatable nor transferable.

EnergyPlus is currently DOE's flagship whole-building energy simulation engine developed with active involvement by many participating individuals and organizations since 1995, and is posted open-source on GitHub [2]. EnergyPlus consists of 1.2 million lines of code with the core consisting of 748,731 lines of C++ code. It uses a more extensible, modular architecture than DOE-2, the previous and still widely used simulation program, to perform the energy analysis and thermal load analysis for a building. The computational costs of

these capabilities has resulted in annual building simulations that, depending on the complexity of the building information, often requires 5+minutes (10x-100x slower than DOE-2 [3]) of wall-clock time to complete. Simulation runtime of this program is practically important as it is used internationally to help create new buildings that are energy efficient, define optimal retrofit of existing buildings, helps define building codes, and is increasingly used by utilities in energy efficiency and demand side management programs.

Reducing the runtime of EnergyPlus is the top priority of the development team with EnergyPlus 7.0 being 25%-40% faster than EnergyPlus 6.0 [4]. But even with a 40% reduction in runtime, manually tuning EnergyPlus building models to align with utility data so that one creates a legally-useful software model of a building is still a slow and tedious process. For example, an engineer manually tuning a simulation is not likely to wait the 3-7 minutes required to run an EnergyPlus simulation before proceeding to the next tuning step; likewise, the Autotune methodology [5] runs 1024 simulations, which at only 3 minutes per simulation would require over 2 days. One solution is to construct surrogates to reduce the overall computational burden. Surrogates, which are generally statistically generated models, are built to provide rapid approximations of the original model, and require less computational resources [6].

In addition to the significant computational load issue, another main concern is the accuracy of the simulation engines for realistically modeling a virtual building such that matches a real-world building. A 2008 study [7] found 190 Home Energy Saver, REM/Rate, and SIMPLE residential simulation models had 25.1%-96.6% error compared to actual monthly electrical energy usage. Another 2012 study [8] found that 859 residential models across Home Energy Saver, REM/Rate, and SIMPLE simulation engines had a mean absolute percent difference of 24% from actual monthly electrical energy usage and 24%-37% from actual natural gas use for a sample of 500 houses. It should be noted that all of these studies use comparisons to monthly utility bill data; the challenge of accurately matching hourly or 15-minute data for dozens of submetered data channels is significantly more difficult.

The challenge for simulation accuracy can be reduced to two primary issues: 1) a gap between the as-modeled and as-built structure, and 2) limitations of the modeling engine's capabilities. Gaps between as-modeled and as-built structures have many sources, but ultimately the fault lies in inaccurate input files rather than simulation engine itself. For example, infiltration, the rate at which air and the energy in it flows through the building envelope is not currently able to be cheaply tested despite its importance for energy efficiency. Blower-door tests can determine infiltration rate at a given pressure (usu. 50 Pascals) but is a 1-time measurement that, in reality, experiences significant variances as a function of temperature, wind speed, and wind direction. A second issue is the schedule for building usage, which includes number of occupants, times of occupancy, heating, ventilation and air-conditioning (HVAC) set-points, operations schedule, and other factors. For many of these, cost-effective sensors simply do not exist or are not typically deployed in a building. In many cases, occupancy schedules and relatively static set-point temperatures are estimated and then used later to "tune-up" a simulation to match whole-building data without regard to the accuracy of the actual HVAC thermostat set-points.

1.2. Literature review

Statistical energy models have been widely used for energy prediction [9, 10], and energy optimization [11, 12]. Building energy models calibration is critical in bringing simulated energy use closer to the actual consumption [13]. Researchers have shown an increasing interest in using various statistical tools for building energy models calibration [14, 15, 16, 17, 18, 19, 20, 21, 22]. Though many statistical energy models have been proposed for building energy analysis, they can be divided into two categories: data-driven models when detailed engineering energy models are available, and surrogate model-driven when only computationally cheap models are provided. There have also been attempts to combine data from both field measurement and computer simulations for calibration of building energy simulation models [16]. In contrast to simple linear regression, Gaussian process (GP) models [15] are used to capture the features

of complex nonlinear and multivariable interactions of building energy behavior. Correlation analysis and hierarchical clustering has been utilized [19] to determine and choose informative energy data. The Bayesian technique becomes popular in this area since it is capable of parameter estimation even when there are missing energy data which are considered as uninformative output data. Bayesian technique based model can be used for multiple purposes, e.g. retrofit analysis, model-based optimal controls and energy diagnostics [23]. Provided a case without complete or a sufficiently large dataset, bootstrap is a powerful statistical tool to assess the accuracy of an estimator by random sampling with replacement from an original dataset [18].

Uncertainties and sensitivity analysis in building energy simulation has been investigated [24, 25, 26, 27, 28, 29]. Uncertainty analysis (UA) takes into account uncertainties due to inherent simplifications of any model and lack of information with regard to input data. Understanding how uncertainties in energy use predictions from simulation software is important to achieve more effective energy efficiency upgrade packages and operational strategies for buildings [30]. On the other hand, sensitivity analysis (SA) consists of modifying model inputs in order to explore the relationship between input parameter variations and overall energy performance of the building [31]. The sensitivity analysis can also identify the most influential parameters to determine which should be tuned at high priority [32]. Both UA and SA should be integrated within calibration methodologies since they play an important role in building model accuracy [33]. To overcome the difficulties of getting information from SA using detailed models, macroparameters that characterize the building are utilized to define and propagate uncertainties of input parameters of building models [25].

Machine learning is a popular technology for improving the accuracy of building models from data obtained from simulations or experiments. To the best of our knowledge, machine learning work within the domain of building energy modeling generally focuses on predicting whole-building utility consumption as a function of environmental measurements. The Building Energy Predictor Shootout, hosted by ASHRAE, had participants predict hourly whole building

electrical consumption for an unknown building using environmental data and user defined domain knowledge. The competition included 150 competitors [34].

Three popular machine learning techniques are *Bayesian Feed Forward Neural Networks (FFNNs)*, *FFNNs ensembles* and *piecewise linear regression*. These techniques helped establish the direction for machine learning-based energy modeling within the building science domain [35, 36]. The only deviations from these classical techniques use Support Vector Regression [37] and Least Squares Support Vector Machines [36], which were not mature techniques when the competition was originally held.

Surrogate generation or meta-modeling often leverages a few classical statistical techniques - Ordinary Least Squares (OLS) linear regression [38, 39], Multivariate Adaptive Regression Splines (MARS) [40], Kriging method [41], and Radial Basis Functions (RBF) [9]. Each technique has its own strengths and weaknesses [42]. There has been a comparison [43] of predictive performance between two linear approaches (full linear and Lasso) and four non-parametric methods (MARS multivariate adaptive regression spline, SVM support vector machine, bagging MARS, and boosting) where SVM models achieve the best performance for both gas and electricity, followed by bagging MARS. Lasso also provides similar prediction accuracy to the full linear model. Overall calibration quality depends on the surrogate model's estimation accuracy. Surrogate model work in the buildings domain often involves relatively small scale (16 inputs and 1 output) EnergyPlus surrogates [14], in which case they are able to produce accurate distribution estimates over parameter settings for buildings based on actual measured data. In addition, linear regression and MARS have been used [14] to generate surrogate models and highlight the need to explore other surrogate model options. The work focuses on macro-scale building stock parameter estimation, which reduces the overall surrogate model's size and complexity. Recently, variable importance analysis and meta-model construction with correlated variables has been studied in [44, 45], where statistical energy meta-models are obtained through linear and non-parametric regression models.

1.3. Motivation and research objective

Previous research has demonstrated that surrogate models have the ability to provide computational advantages and its calibration utility completely depends on the model's accuracy. The few available studies in the building science domain explore a limited number of envelope parameters, operation parameters, and outputs (10 envelope parameters [46] or 16 envelope and operational parameters [14]). These studies estimate a single output. A vast majority of surrogate studies in other engineering disciplines frequently use a limited number of inputs and outputs. Therefore, it is difficult to ascertain how well the surrogate model created in this study can approximate EnergyPlus on a scale relative to other studies. Large scale simulation calibration produces significant scalability issues with fitting MARS, Kriging, or RBF surrogates. In particular, the computational time and memory requirements quickly become intractable as model training data increases. Therefore, machine learning methods leveraging world-class high performance computing resources and state-of-the-art data mining methodologies for big data are needed to mitigate scalability limitations. FFNN and Lasso regression using Alternating Direction of Method of Multipliers [47] are the methods adopted for this study. These methods can produce large scale surrogate models and quantify their overall effectiveness at quickly producing accurate EnergyPlus simulation outputs.

Two surrogate models were constructed using FFNN and Lasso regression respectively. In leveraging the previously demonstrated inaccuracies of current simulation engines, we explore the possibility of using machine learning techniques to quantify the trade-off between this innate inaccuracy to more quickly run approximated EnergyPlus simulations. The surrogate models were generated using three very large EnergyPlus simulation datasets for a residential building. The datasets cover fine grain parameter sweeps over seven envelope parameters with 80 simulation outputs and coarse broad parameter sampling over 156 envelope parameters with 90 simulation outputs. Data generation and sampling is covered in more detail in Section 2. Using these datasets, we evaluate the two generated surrogate models' abilities.

This research is part of a project named Autotune [5]. The Autotune project's [5] goal is to create an automated process for tuning simulation inputs so that simulation output can match measured data. This work facilitates that aim by constructing EnergyPlus surrogates that can be used to improve the overall calibration execution time. The project relies on 300+ channels of 15-minute sensor data from an automated-occupancy research home (real-world data), supercomputer simulations of millions of EnergyPlus simulations resulting in a 267TB database (simulation data), a mathematical mapping between real-world data and simulation output data, and sensitivity analysis/data mining to determine intelligent ways to quickly find the proper set of building inputs to match measured data.

The remainder of the paper is organized as follows: Section 2 discusses the simulation parameter sampling process (data generation); Section 3 discusses the developed approximation methods; Section 4 presents the evaluation criteria; Section 5 presents the approximation results; Section 6 discusses our prediction results and interesting observed phenomena found through the experimentation process; Section 7 summarizes the findings as well as possible future directions.

2. Simulation Sampling

Oak Ridge National Laboratory (ORNL) operates four 2,800 ft^2 residential buildings that robotically emulate occupancy according to Building America benchmarks as part of the ZEBRAlliance project [48]. One of these houses—which has 269 channels of 15-minute data including on-site weather data—is leveraged to allow high-resolution comparison between existing EnergyPlus models of this building and real-world sensor data. For this reason, and because U.S. homes consume 22% of the nation's primary energy [1], the residential building model was selected as a primary building type that needs to be included in Autotune's large-scale sensitivity analysis and calibration studies.

There are only 4.7 million commercial buildings in the United States, but

they consume 19% of US primary energy. Commercial buildings have more clearly-defined building types associated with designs that typically align with their use. DOE has previously released a detailed report on the major US commercial reference buildings [49]. We selected 3 of the 16 commercial building types for this study—stand-alone retail and warehouse buildings, due to the size of total floor area, and the medium office building since it is the most prevalent building type as far as number of buildings.

To conduct thorough sensitivity analyses and calibration experiments, 5 million simulations for the residential building and 1 million simulations for each commercial building type were computed and stored to sample the input parameter space. Subject matter experts prioritized hundreds of the approximately three thousand variables for each of these buildings, as well as the most important output; some of these variables are meta-parameters that may be a material instantiated in several places throughout the building (e.g. gypsum board) or a grouping of inputs that vary together as a single parameter (e.g. infiltration at multiple zones treated as whole-building infiltration). The most important variables consisted of 156 input variables and 90 output variables being collected for the total of 8 million simulations at 15-minute resolution. With the input file corresponding to approximately 300 KB and output of 35 MB, total simulation data amounts to 267 TB detailing 26.9 trillion data points that has been shared with the research community.

The total search space for 156 variables, with defined distributions and discretized ranges, is computationally infeasible as it amounts to $5\text{x}10^{52}$ simulations. This research was done on desktop systems, the 1024-core Nautilus supercomputer from the University of Tennessee, the 2048-core Frost supercomputer, and the 299,000-core Titan supercomputer at Oak Ridge National Laboratory. Titan, at the time the fastest supercomputer in the world, would require $4.5\text{x}10^{31}$ lifetimes of the known universe to brute-force every combination of 156 variables for just one building. Even to run the relatively small subset of 8 million simulations used in this study, it required approximately 110 compute years. To intelligently search the space, knowledge must be gleaned

from simulation differences for a small subset of the potential simulations. The experimental design is such that machine learning agents can quickly learn on available data but then leverage more complex data as additional simulations are completed.

The methodology employed for running simulations is according to Markov order in which increasing orders of complexity take into account the combinatorial effect from a correspondingly increasing number of variables. Each simulation contains 35,040 simulation output vectors (15-minute data for an annual simulation, for each output variable). The simulations in all experiments use a constant set point for the entire year. In a simplified example with only a min, average, and max value for each of N input variables, Markov order 1 (MO1) simulations consist of all simulations that hold all variables at the average/baseline value, but they change variable 1 to the min value for one simulation and to the max value for a second simulation, and then proceed to variable 2 and ultimately all other variables. This results in a total of 2N+1 simulations. For Markov order 2 (MO2), all simulations not previously run in MO1 are computed for each pair of inputs. In MO2, if you consider each simulation a variable pair,

$$MO2 = [(V1_{min}, V2_{min}), (V1_{min}, V2_{max}), (V1_{max}, V2_{min}), (V1_{max}, V2_{max}), \\ (V1_{min}, V3_{min}), \dots] ;$$

the result is a total of 4×C(N,2) simulations. There is a combinatorial increase for every step up this Markov order; we proceed until the planned number of simulations is complete for every building type. This work leverages the MO1 and MO2 residential simulations.

In addition to the MO1 and MO2 simulation datasets, we sampled simulations using a brute force sampling we store and refer to as Fine Grained (FG). The simulations in the FG dataset use the same inputs as the MO1 and MO2 simulations, but we varied only seven envelope input parameters. In addition, we captured only 80 output variables. In total, the small FG data set contains 12,000 simulations and is approximately 143 GB. This dataset was constructed

to estimate how well the surrogate models are able to approximate EnergyPlus when presented with densely sampled points within the design space of critical building envelope parameters.

3. Approach

Two different methods were explored for approximating EnergyPlus. The first approach uses standard FFNN with a modified training process. The training process was adjusted to accommodate large datasets, which ultimately allows computationally tractable large-scale FFNN learning. FFNN background information is presented in Section 3.1 and the training procedure is presented in Section 3.2.

The second approach uses Lasso regression, a linear model, which has the ability to automatically select relevant input variables. This allows users to determine whether there is sufficient information within the datasets to produce predictions good enough for a particular use case, or determine if a more complex model (FFNN) is required. However, the standard Lasso regression learning algorithms are not designed to support large-scale learning. To overcome this difficulty, we use a recently developed decentralized optimization framework, Alternating Direction Method of Multipliers (ADMM) [47]. This method supports arbitrary large-scale learning by dividing the original problem into smaller, local optimization problems. These problems are either distributed across multiple computers or solved locally on a single memory-constrained computer that uses the hard drive as additional storage. Section 3.3 discusses Lasso regression, Section 3.4 the ADMM framework, Section 3.5 Lasso regression's ADMM formulation, and Section 3.6 the best parameter settings found for Lasso and FFNN models for this problem.

3.1. Feed Forward Neural Network

Previous research has shown that FFNN can be used to approximate nonlinear functions for predicting electrical consumption, and much more [50, 51,

52, 53]. Essentially, FFNNs can learn to approximate continuous functions that map $\Re^m \rightarrow \Re$ without prior assumptions about the relationships between the inputs and the outputs. While the FFNN model is general, it requires the user to create the model structure defined by parameters such as number of hidden layers, hidden units, and activation function.

An FFNN with a single hidden layer was used to approximate EnergyPlus. Other work has shown that a single-hidden-layer FFNN performs well on prediction tasks within the building spaces domain [50, 51, 52, 53]. An FFNN with a single hidden layer for function approximation has the following mathematical representation:

$$f(x) = \sum_{j=1}^{N} w_j \Psi_j \left[\sum_{i=1}^{M} w_{ij} x_i + w_{io} \right] + w_{jo}$$

where N represents the total number of hidden units, M represents the total number of inputs, and Ψ represents the activation function for each hidden unit. In this work, $\tanh(x)$ was selected as the activation function because it allows hidden layer output values to range from $[-1, 1]$, which allows for a wide variety of possible functions.

A FFNN's weights are learned using gradient descent-based methods, such as Newton-Raphson, by minimizing a user-specified error function. There are many possible error functions, such as mean squared error, sum squared error (SSE), and root mean squared error (RMSE). In this research, the SSE function was used.

A gradient descent learning approach poses two problems. The first problem is over-fitting. The FFNN can adjust its weights so that it performs well on the training examples, but it will be unable to produce accurate responses for novel input examples. This problem is addressed by splitting the training set into two parts – a set for training and a set for validation. When the error increases on the validation set, the learning algorithm should halt, because any further weight updates will only result in over-fitting the training examples.

The second problem involves finding a globally optimal solution in the presence of many local minima. A local minimum is a point at which it is impossible

to further minimize the objective function by following the gradient, even though the global minimum is not reached. However, it is not possible to determine if any particular set of weights is a globally optimal solution or a local minimum. It is not possible to completely address this problem, but it is possible to avoid shallow local minima by using momentum and an adaptive learning rate. Momentum incorporates a small portion of the previous weight changes into the current weight updates. This can allow the FFNN to converge faster and to possibly step over shallow local minima. An adaptive learning rate dynamically changes the gradient descent step size so that the step size is larger when the gradient is steep and smaller when the gradient is flat. This mechanism will allow the learning algorithm to escape local minima if they are sufficiently shallow.

3.2. Large-Scale Feed Forward Neural Network Training

There are two gradient-based methods for training FFNN – stochastic and batch. The stochastic method uses a single observation to compute the gradient and update the network. It is extremely scalable to large datasets, because it makes updates per training example. However, stochastic gradient descent only approximates the gradient using local information, which means it lacks the global information required to follow the objective function's true gradient. This allows the stochastic gradient descent method to scale well, but it may produce less accurate models because an approximate gradient is substituted for the exact gradient.

The batch gradient descent method uses all training examples to compute the gradient and update the network. This method is much less scalable than the stochastic method, because it has to process all examples for every update. Computing the gradient using the entire dataset allows this method to produce better gradient estimates, which may lead to more accurate networks. However, this method is not typically practical since it requires hundreds of passes over a gigabyte dataset.

Given that both approaches provide different benefits, a hybrid method for

training the FFNN was implemented. The method can be considered a stochastic gradient descent that performs updates using mini-batches, rather than updates per single training example. This allows us to balance training time performance and gradient estimation quality. In the developed approach, we select a random simulation and divide the simulation into mini-batches. Before the mini-batches are constructed, each sampled simulation is randomly shuffled. Randomly sampling the simulations and shuffling the internal simulation data provides the stochastic gradient characteristics. In addition, making network updates per randomized mini-batch provides a pseudo batch gradient descent characteristic. In summary, we sample a simulation, randomize the simulation data vectors, divide the data into mini-batches, and update the network using each mini-batch.

3.3. Lasso Regression

Standard Lasso regression fits a linear model by modifying a multiplicative weighting factor for each input and adding the weighted inputs to create the outputs. The final model has the same functional form as linear regression and ridge regression, but the learning criteria inserts a term to penalize the absolute size of the regression coefficients. This allows automatic feature selection and overcomes standard regression problems with overweighting highly correlated predictors. The following equation shows the Lasso regression optimization criteria:

$$\sum_{i=1}^{n}(y_i - b - w^{\mathsf{T}}(\vec{x_i})) + \lambda||w||$$

where $\vec{x_i}$ is an input vector, y_i is the response, w is the model weights, b is the y intercept, and λ produces a trade-off between fitting and sparsity. Increasing λ leads to a model with more zero value weights. This means, under an appropriate λ value, irrelevant inputs in $\vec{x_i}$ are ignored, resulting in a sparser, more robust model. Note that robustness is defined based on the idea that a simplistic model is most likely to generalize to new scenarios, which is based on model complexity studies [54, 55, 56].

Lasso regression can easily be extended to nonlinear functions using kernels, but this was not explored in this study for two reasons. First, linear Lasso regression is computationally fast and its performance indicates whether the more computationally-expensive nonlinear FFNNs is necessary. If a linear model is sufficient, then it can substantially reduce the overall training time for larger datasets. Second, Lasso regression's variable selection capabilities make it more interpretable based on the learned values for w. This allows an expert to analyze which information is important for making predictions and can help a user ascertain if required information is missing within the dataset.

3.4. Alternating Direction Method of Multipliers

To maximize resource utilization, ADMM [47] was selected instead of other equally capable distributed optimization methods because it does not use a master-slave paradigm. While the following detailed ADMM description illustrates solving a redundant secondary optimization problem per computer, the optimization problem in practice is extremely lightweight. This makes it more efficient to redundantly solve the problem locally rather than communicate the solution to slave computers.

ADMM is a fully-decentralized, distributed, optimization method that can scale to very large machine learning problems. It solves the optimization problem directly without using approximations during any phase of the optimization process. The optimization process works by splitting the criteria function into separate subproblems and optimizing over those individual problems with limited communication. The following is a formal explanation from [47]:

$$\text{minimize} f(x) + g(z)$$

with the following constraints $Ax + Bz = c$ where c is a targeted response or agreed target value that correlates the two functions. In addition, f and g are convex, closed, and proper functions. The functions $f(x)$ and $g(z)$ are independent, meaning both can be minimized in parallel. The equality constraint provides consensus across the two functions. More importantly, the following

Lagrangian is introduced [47] for this particular optimization problem:

$$L_p(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + (\rho/2)||Ax + Bz - c||_2^2$$

where ρ defines a tunable parameter that determines the trade off between violating the equality constraint and fitting the target function. After some additional algebraic simplifications of the above Lagrangian, the final ADMM optimization process is as follows:

$$\begin{aligned} x^{k+1} &= \underset{x}{\operatorname{argmin}}(f(x) + \frac{\rho}{2}||Ax + Bz^k - c + u^k||_2^2) \\ z^{k+1} &= \underset{z}{\operatorname{argmin}}(g(z) + \frac{\rho}{2}||Ax^{k+1} + Bz^k - c + u^k||_2^2) \\ u^{k+1} &= u^k + x^{k+1} - z^{k+1} \end{aligned}$$

Iterating over these optimization equations provides guaranteed convergence, and establishes a method to minimize x and z independently with limited communication between the two optimization problems.

The above form can be deconstructed further into multiple sub-problems across $f(x)$ by sub-dividing the function across the independent components within x. This creates independent sub-problems that are solved locally via the first minimization step, which allows multiple computers to optimize $f(x)$ locally, and pass information to other computers or processes about x^{k+1}, resulting in a global optimization over z^{k+1} at each individual process. This means all processes can work to optimize and compute their individual updates by only communicating their local beliefs for x^{k+1}.

3.5. Large-Scale Lasso Regression

There exist several common substructures for constrained convex optimization problems [47]. In particular, the general minimization problem is defined as follows:

$$\text{minimize} f(x)$$

with the following constraints $x \in C$, where C defines a constrained solution space. This general minimization problem is formulated as the following under

ADMM:

$$\text{minimize} f(x) + g(z)$$

with the constraint $x - z = 0$, where g is an indicator function. Using an indicator function for g allows ADMM to represent the original convex optimization constraints, and the $x - z = 0$ constraint guarantees that the x that minimizes $f(x)$ obeys the original constraints.

While [47] used this general solution format to solve many different convex optimization problems, we are only interested in the version used to solve Lasso regression. The distributed optimization steps for solving large scale linear Lasso regression problems are the following[1]:

$$x_i^{k+1} = \underset{x_i}{\text{argmin}}(\frac{1}{2}||A_i x_i - b_i||_2^2 + \frac{\rho}{2}||x_i - z^k + u_i^k||_2^2)$$
$$z^{k+1} = S_{\frac{\lambda}{\rho N}}(\bar{x}^{k+1} + \bar{u}^k)$$
$$u_i^{k+1} = u_i^k + x_i^{k+1} - z^{k+1}$$

The individual local subproblems are solved using ridge regression, and the global z values are computed by evaluating a soft thresholding function S. This function is defined as follows:

$$S_{\frac{\lambda}{\rho N}}(v) = max(0, v - \frac{\lambda}{\rho N}) - max(0, -v - \frac{\lambda}{\rho N})$$

The soft thresholding function applies the Lasso regression sparsity constraints over z, which are incorporated into the local subproblem solutions on the next optimization iteration.

The key advantage of this particular Lasso regression formulation is that the main step is solved exactly once. The direct method for computing x_i^{k+1} requires computing $(A^\mathsf{T} A + \rho I)^{-1}$. The resulting matrix never changes throughout the entire optimization process. Storing this result allows the distributed optimization method to perform a very computationally intensive task once and

[1]This version assumes we are only splitting the optimization problem across the training samples, and not the features. It is possible to split across both [47].

reduce all future x_i^{k+1} computational steps. Caching the values used to compute x_i^{k+1} to disk allows a 2.2Ghz Intel Core i7 laptop to solve a univariate 3.9GB Lasso regression problem in approximately 17 minutes. By way of comparison, the best FFNN model with 15 hidden units and 10 outputs completed training in 24 hours on the same 3.9GB dataset.

3.6. Model selection

The final Lasso regression model's performance is greatly dependent upon the λ value used during the training process. Similarly, a FFNN's performance is greatly dependent upon the total number of hidden units selected. Selecting a λ value that is too small can result in overfitting, while selecting a value that is too large can lead to underfitting. The same possibilities apply to FFNN, but selecting too few hidden units can lead to underfitting, and selecting too many can lead to overfitting.

Selecting the best parameter setting is achieved by evaluating a model selection criteria, which measures how well the learned model will generalize to unseen examples. There are several different model selection techniques. For example, cross-validation methods estimate how well a model generalizes to unseen data by periodically testing the current model on a validation set. An online validation process is one that terminates the learning process when the model begins to perform poorly on the validation set. K-Folds cross-validation is another approach that divides the data into K partitions, and builds a model using $K-1$ partitions as training data. The omitted partition is used to evaluate the model as testing data. This training and testing process is repeated such that each partition is used as the testing set at least once. K-Folds primary advantage over other methods is that it can provide an almost unbiased error estimate for any particular model as K approaches the dataset's sample size [57].

Ideally, a combination of cross-validation and K-Folds would be used to select the best parameter values. Cross-validation enables online FFNN learning termination, and K-Folds facilitates selecting the correct number of hidden

units. On the other hand, Lasso regression uses the validation set to select λ and K-Folds to approximate the model's overall error. However, the large dataset makes K-Folds cross-validation computationally expensive. Therefore, a pure cross-validation method for parameter selection was selected. Each model has a training set, a single validation set, and a testing set. For the FFNN models, the validation set was used to prevent overfitting, and hidden unit settings using the unseen testing data were compared. The Lasso regression models use the validation set to select the best λ value by picking the one that maximizes prediction accuracy for the validation set. This parameter selection method allows us to estimate the Lasso regression model's true prediction capabilities by using the unseen testing data. In addition, the testing data results can be directly compared with the FFNN results.

4. Methods

4.1. Experimental Design

Given the need for scalability and performance, we optimized the FFNN network structure and application performance by determining the maximum number of outputs per network should be 10. This means that eight FFNNs were used to model the FG dataset's 80 outputs, and nine FFNNs to model the MO1 dataset's 90 outputs[2]. In addition, the outputs for each network were selected by grouping the variables based on the order they were stored with groups that represent similar components (e.g. building descriptors) within the simulation.

The Lasso regression method used is only able to approximate univariate response variables. This means a linear model was created for each simulation output. This restriction results in using 80 linear functions to model the FG dataset, and 90 linear functions to model the MO1 dataset. While the overall

[2]The MO1 and MO2 datasets originally contained 96 outputs, but six output variables for all simulations never changed and were removed.

model count is high, the overall training time scales very well using the ADMM optimization technique previously discussed. This allowed computation time to scale better than the FFNN models on average.

Two experiments were defined using the FFG, MO1, and MO2 datasets in combination with the FFNN and Lasso models previously described. The first tests how accurately a model approximates EnergyPlus when only seven simulation variables are varied (FG dataset). This allows us to estimate how sensitive the learned models are to fluctuations in the building parameter inputs by using a very densely sampled simulation input set. In this particular experiment, we selected the best FFNNs by testing models with 5, 10, and 15 hidden neurons, and we selected the best Lasso regression model by testing λ values between 0 and 1 using 0.15 increments. The training set contains 250 simulations and the testing set contains 750 simulations; we selected the models that performed best overall on the testing set. The second experiment measures how well the models approximate EnergyPlus when presented with MO1, defined in Section 2, a very coarse sampling of the simulation input parameters. FFNN models with 5, 10, and 15 hidden nodes were trained. For the Lasso regression models, the best λ value between 0 and 1 was searched using 0.15 increments. Three hundred randomly sampled simulations from the slightly denser MO2 dataset was also used for testing and comparing both methods.

4.2. Performance Metrics

Within the building community, there are four commonly used metrics for comparing prediction accuracy — root mean squared error (RMSE), coefficient of variance (CV), mean absolute percentage of error (MAPE), and mean bias error (MBE). These metrics are defined as follows:

$$\mathrm{RMSE} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

$$\mathrm{CV} = \frac{\mathrm{RMSE}}{\bar{y}} \times 100$$

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - \hat{y}_i|}{y_i}$$

$$\text{MBE} = \frac{\frac{1}{N-1}\sum_{i=1}^{N}(y_i - \hat{y}_i)}{\bar{y}} \times 100$$

where y_i is the actual energy consumption, $\hat{y}_i$ is the predicted energy consumption, and $\bar{y}$ is the average actual energy consumption. RMSE is a measure of absolute error. When y_i and $\hat{y}_i$ are close to zero, RMSE may be absolutely small but relatively large compared to the signal. CV determines the error relative to the target's mean. Predictions with low RMSE and low CV are preferred. Most figures in this article plot RMSE and the mean target response (i.e. average of actual energy consumption) so that the reader can visibly ascertain the absolute prediction error relative to the average actual energy consumption. MAPE measures the percentage of error per prediction, which avoids issues with outliers that may skew the average and average-based metrics (e.g. CV). MBE establishes how likely a particular model is to over-estimate or under-estimate the actual energy consumption where a negative percentage means the predictor generally under predicts the real value. The MBE and MAPE metrics are well described and presented in [34, 36].

5. Results

The datasets generated contain 80+ output variables which makes traditional table presentations difficult. Therefore, figures were presented to provide broader comparisons across the models. The figures of results have been split into non-load variables (not associated with the heating and cooling needs of a building which must be met by an HVAC unit) and load variables. All variables are referenced only by numbers rather than by name to show goodness-of-fit for different models across simulation outputs, although we do later provide explicit names to facilitate discussion in Section 6. The detailed variable list of inputs and outputs is provided in the supplementary material.

In the figures below, the two values needed to compute the CV metric are used where the left y-axis represents the RMSE and the right y-axis represents

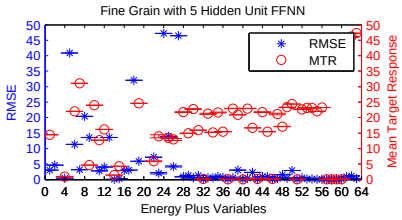
a response variable's mean (MTR). Normally, the ratio, the CV between these two values would be presented, but several response variable values are small which cause misleading CV values that show poor relative performance whereas the absolute performance is good. For variables in which RMSE (blue) exceeds the average value (red), predictions are off by more than 100%. Since non-load results are for variables with very different units (e.g. thickness, density, and specific heat for multiple materials), all non-load figures restrict the y-axis to $[0, 50]$ for RMSE and MTR. While RMSE error values in all figures never exceed the range, a few MTR values are significantly beyond this range and do not appear on the figure (meaning the prediction performance for this variable is excellent).

5.1. Fine Grain

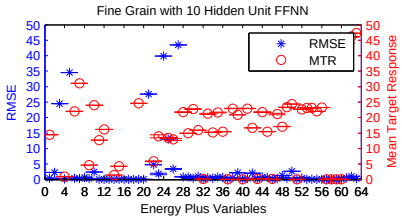
Figure 1 presents the FFNN non-load variable prediction results for the experiments on the FG dataset. Most models perform the same on the response variables, but a few variables do present noticeable differences across the different models. In particular, the environmental and electrical variables between 1 and 16, as well as, the envelopes heat gain and loss variables between 20 and 28. The variables between 1 and 16 present the best performance with 10 hidden units (Figure 1(b)). Analyzing the figure closely reveals variable 10 has a much better error rate with 15 hidden units. Even though the performance for the other variables between 1 and 16 produce similar performance with 15 hidden units (Figure 1(c)), 10 hidden units is considered the better model since it takes less time to calculate and is more likely to generalize to new data.

The 15 hidden unit model presents the best performance on the variables between 21 and 28. Since each network predicts 10 outputs, in practice, we allow the 15-node model to also estimate variables 20 and 29 since results are comparable to the 10-node model.

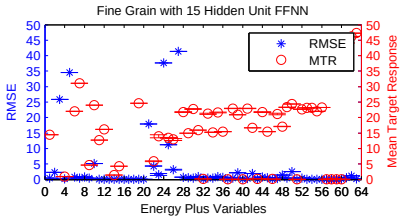
While the non-load prediction performance was considered to be excellent, the FFNN load prediction performance shows room for improvement. The load variables in Figure 2 physically represent the sensible heat, latent heat, sensible



(a) 5 Hidden Units



(b) 10 Hidden Units



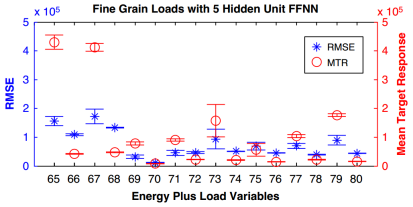
(c) 15 Hidden Units

Figure 1: Results for predicting the FG non-load output variables as a function of simulation inputs. RMSE (blue) and average (red) are normalized $[0, 50]$ to allow side-by-side comparison of prediction accuracy. Dividing RMSE by MTR gives the average percent error for predicting that variable. Physical description of each input variable (x-axis) is provided in supplementary material. Neural networks achieve good prediction accuracy of most simulation inputs for this dataset.

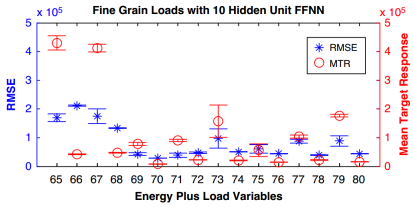
cooling, and latent cooling for four different thermal zones in the following order: living room (LR—variables 65–68), master bedroom (MB—69–72), basement (BM—73–76), and second floor (SF— 77–80).

Figure 2(a) and 2(c) shows the 5 and 15 hidden unit models represent the best prediction results overall. We considered the 5-node FFNN to best predict FG load variables since it is more stable than the 15-model FFNN (less variance in RMSE) and is less complex.

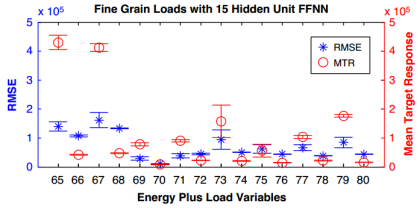
Figure 3 presents the Lasso linear regression results on the FG dataset. The model performs well on the non-load variables and is fairly competitive with the previous FFNN models. However, some error rates have much higher variance



(a) 5 Hidden Units



(b) 10 Hidden Units



(c) 15 Hidden Units

Figure 2: Results for predicting the FG load variables. Load variables share the same unit (Joules) and are not normalized. Load variables are more difficult to predict than non-load variables, with 7 of 16 variables having error rates above 100%, regardless of the complexity of the neural network utilized.

than the best FFNN model (Figure 1(c)) for variables 7 and 20–28. In addition, all error rates for these variables are statistically worse than the FFNN error rates. This means these variables have a non-linear relationship with their inputs, and the Lasso regression is not as capable a methodology for capturing these patterns.

Although the other non-load variables are all statistically worse than the FFNN models, the Lasso method uses only an input subset[3] to make all the predictions, so the linear model is using less information than the FFNN to make only marginally worse predictions (i.e. 0.2-5.0 difference in RMSE, with only a few variables differing in RMSE by over 20). Lasso is fitting simpler models by reducing the number of input variables used within the model, which

[3]The subset refers to the inputs that have a non-zero weight in the model

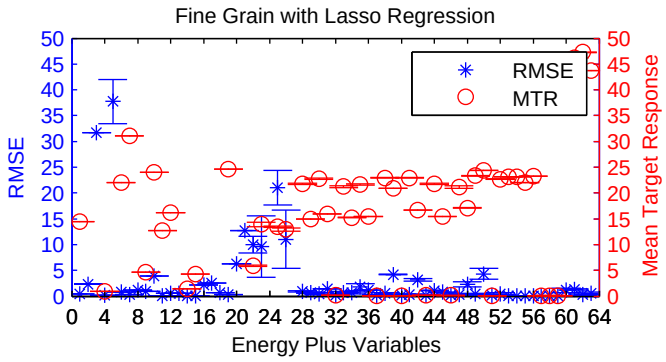


Figure 3: The Lasso regression generated model's performance on the FG dataset's non load variables.

results in a simpler model that is more likely to generalize to new datasets and can learn much faster than FFNNs, as previously discussed in Section 3.5.

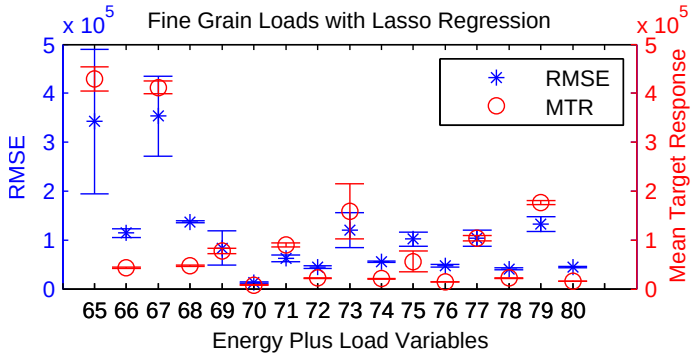
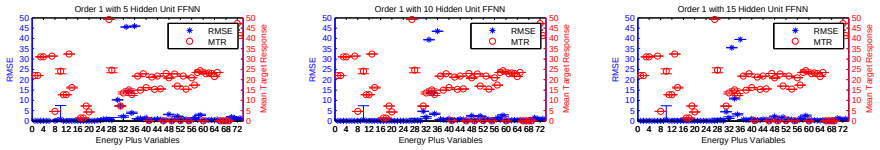


Figure 4: The Lasso regression generated model's performance on the FG dataset's load variables.

The Lasso regression load prediction results (Figure 4) are very similar to the FFNN results (Figure 2). Although the Lasso regression results are worse, the model performed best on the same variables that the FFNN models were able to predict—variables 65, 67, 71, 73, 77, and 79. However, the other load

variables were not fit well by either method, implying there is not sufficient information within the raw dataset to predict the other load variables.

5.2. Markov Order 1



(a) 5 Hidden Unit (b) 10 Hidden Unit (c) 15 Hidden Unit

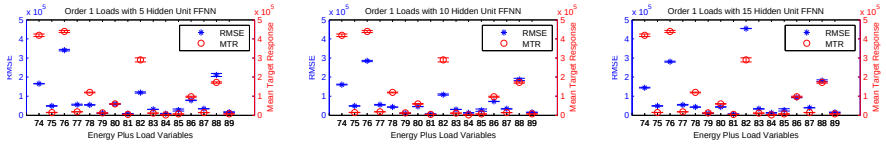
Figure 5: These figures present the results for predicting the MO2 non-load variables with 5 (Figure 5(a)), 10 (Figure 5(b)), and 15 (Figure 5(c)) hidden unit FFNNs.

Experiments with the MO1 dataset are more challenging than the the FG dataset, because we are testing how well models generalize when trained with a limited representation of the input space. While the MO2 dataset represents a relatively denser sampling than MO1, handling the larger data sizes can be a limitation to practical implementation of predictive analytics.

MO1 experimental results for non-load variables (Figure 5) and load variables (Figure 6) shows slightly better non-load variable prediction than with the FG dataset. All models produce about the same performance results on the non-load variables with the exception of the 15-node model which performs statistically better on variables 32 through 36.

Similar to the FG dataset, MO2's load variables remain difficult to predict. Additional analysis and improvement directions are discussed in Section 6. The FFNN models were able to predict variables 74, 76, 78, and 82 (Figure 6). However, it was observed that the 10-hidden-unit FFNN produced the best performance on the MO2 dataset based primarily on performance for variables 78, 82, and 86.

The Lasso regression results for MO2's non-load variables are shown in Figure 7. These results illustrate that many variables within the MO1 and MO2



(a) 5 Hidden Unit (b) 10 Hidden Unit (c) 15 Hidden Unit

Figure 6: These figures present the results for predicting the MO2 load variables with 5 (Figure 6(a)), 10 (Figure 6(b)), and 15 (Figure 6(c)) hidden unit FFNNs.

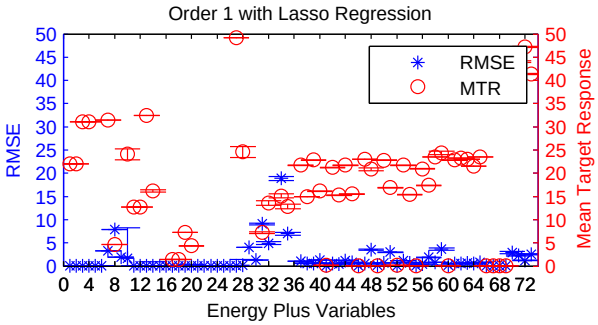


Figure 7: The Lasso regression generated model's performance on predicting the MO2 dataset's non-load variables.

dataset can be modeled using linear models. In addition, it highlights the variables that require a non-linear model as can be seen in Figure 5(c) as shown by variables 6–12, 28, and 36.

Both models present a high variance on variable 10. The high variance is directly associated with the sparse parameter sampling found in the MO1 dataset. This particular variable has instances in which it produces different response behaviors, as seen in Figure 8. This means that coarse parameter sampling used to generate MO1 may have limited abilities to produce meaningful general purpose models. However, it also implies that models created from the MO2 dataset will have similar deficiencies, because a limited sampling can only represent a fraction of the entire domain's behavior. It may be beneficial

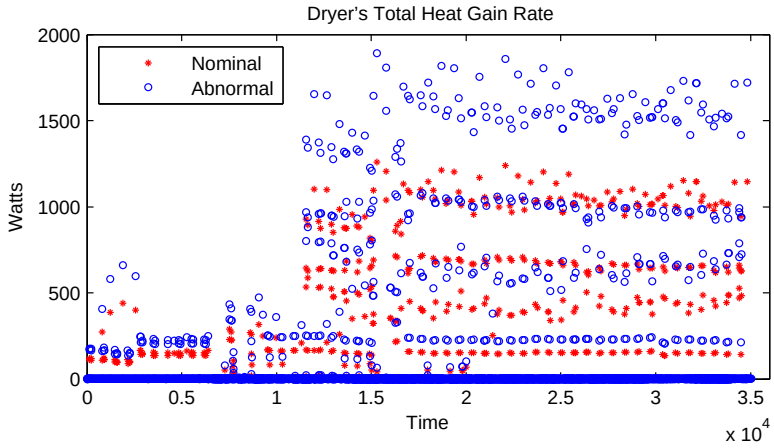


Figure 8: Comparison between the average MO2 dryer heat gain response (red) and an observed, scale-shifted response (blue), every 15 minutes for a year, shows variables with high variance can present a challenge for prediction.

to explore simulation parameter sampling strategies that consider the learner's fitting capabilities, as well as implement methods that can estimate the variance associated with each prediction.

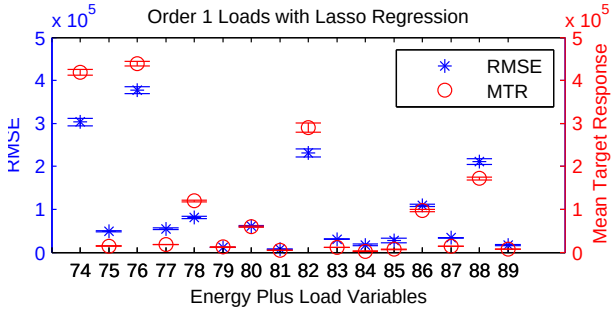


Figure 9: The Lasso regression model's performance on predicting the MO2 dataset's load variables.

Similar to the FG dataset results (Figure 4), the load predictions with Lasso regression (Figure 9) are all worse than the best FFNN results (Figure 6(b)). However, the Lasso regression method was able to fit the same load variables

Model	RMSE (W)	MEAN (W)	CV (%)
FFNN 5	9.125±0.00	1756.8±0.00	0.519±0.00
FFNN 10	1.164±0.00	1756.8±0.00	0.066±0.00
FFNN 15	1.061±0.00	1756.8±0.00	0.060±0.00
Lasso	4.797±1.61	1756.8±0.00	0.273±0.09

Table 1: Whole Building Energy consumption (MO1 Variable 90) can be predicted very accurately.

as the FFNN model—variables 74, 76, 78, and 82.

The MO1 dataset's 90^{th} simulation output variable represents whole building energy (WBE) consumption, which is not present in the FG dataset. Table 1 presents the WBE prediction results for all models. These results illustrate that the Lasso model provides a better fit than the 5 hidden node FFNN model, but provides a worse fit than the 10 and 15 hidden node models. It should be noted that the accuracy indicator, i.e. RMSE, of FFNN 15 model is barely better than that of FFNN 10 model. In general, we found that 20+ neurons with 1 hidden layer yielded minimal returns in predictive accuracy. Although the Lasso model does not perform as well, its overall training time is substantially better than that of the FFNN 10 and 15 hidden node models (Table 2). These performance characteristics indicate that it is best to use the Lasso regression model to predict all variables that can be sufficiently represented using a linear model. This is especially true when one has sufficient computational resources to run the learning algorithms in parallel. The total training time represents the execution time associated with training each individual model in serial. A more parallel approach will converge to the single model training time. Finally, the overall execution time represents the parallel execution speed for running the nine MO1 FFNN models in parallel and running the entire Lasso regression model as a single matrix multiplication. This indicates that the Lasso regression method's testing speed scales better than the FFNN when parallel execution is not possible.

Model	Training Time (Hr)	Total Training Time (Hr)	Execution Time (sec)
FFNN 5	∼2	∼18	∼2.70
FFNN 10	∼8	∼72	∼2.85
FFNN 15	∼24	∼216	∼2.93
Lasso	∼0.2833	∼25.50	∼2.90

Table 2: The first column represents the single MO1 model training time, and the second column is the necessary time to train all MO1 models in serial. The execution time represents single model execution time.

6. Analysis and Discussion

Several interesting findings were observed with regard to both datasets and prediction accuracy. First, a simulation clustering phenomenon was observed, which may provide insight into EnergyPlus approximation efforts for specific variable sand illustrates how prediction accuracy varies as a function of simulation data. Second, HVAC schedule features for improving overall heating and cooling load predictions are discussed.

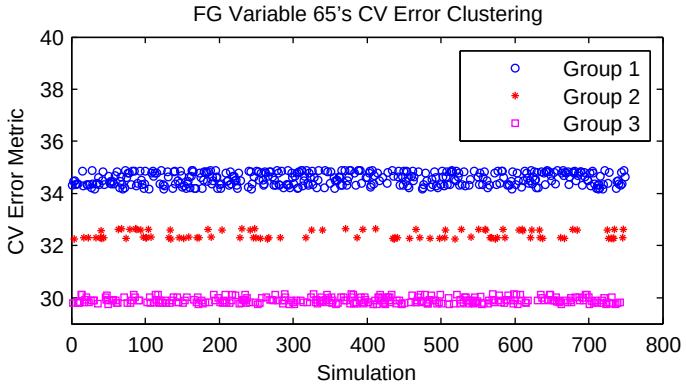
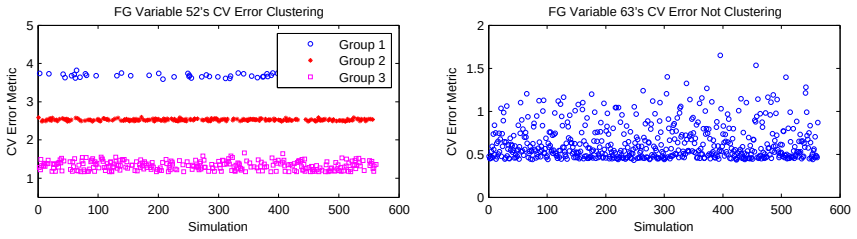


Figure 10: Illustrates the FFNN model's CV error clustering into distinct clusters on the Fine Grain dataset.



(a) Lasso Cluster

(b) Lasso No Cluster

Figure 11: Illustrates that the Lasso regression models can produce distinct clusters when a linear model captures the full relationship between inputs and outputs (Figure 11(a)). When a non-linear model is required, Lasso regression fails to produce the distinct clustering (Figure 11(b)) necessary for accurate prediction.

There are several properties worth mentioning that were exhibited by the predictive models related to clustering, representing simulations equally misinterpreted by the predictive model. For example, the CV error metric measured for FFNN prediction on FG's variable 65, constructs well defined clusters (Figure 10) and were seen for multiple variables. However, clustering with Lasso regression occurs only if the variable is sufficiently well predicted by a linear model (Figure 11(a) and 11(b)). Neither model exhibits the same clustering behavior in the MO1 experiments. The MO1 experiments show a single group (Figure 12). This clustering property suggests that as EnergyPlus parameter sampling density (i.e., sample the data more finely) increases, so will the number of clusters. In such cases, ensemble learning for specialization in predicting each cluster may be fortuitous as shown in previous work predicting future hourly residential electrical consumption via clusters determined by C-means and Hierarchical Mixture of Experts [36].

While clustering can improve predictive results, it becomes increasingly difficult due to the curse of dimensionality [58]. Each simulation contains 35,040 simulation output vectors, and the FG dataset contains 80 outputs and the MO1 dataset contains 90 outputs. These directions are discussed in more detail in Section 7.

In an effort to improve overall heating and cooling load predictions, we added features related to HVAC operation schedule and temperature gradients to the input set. The temperature gradient features include the inside and outside temperature gradient. The inside temperature gradient represents the average temperature change across the building's thermal zones. In the experiments, the building thermal zones correspond to the LR, MB, BM, and SF. The outside temperature gradient represents the change in dry bulb temperature. Using these temperature gradient features, we manually constructed a heuristic indicator function that limits heating to October through March and cooling otherwise (based on region).Finally, we use the gradient direction to estimate the on or off state for each 15-minute timestep. If the inside gradient is increasing and the outside gradient is decreasing, then the heat is activated, provided the time corresponds with a heating month. The inverse is used to establish when cooling is active.

Using these additional features with the FFNN model, we repeated the FG and MO1 experiments on the heating and cooling load variables. The FG load results are shown in Table 3. Improved performance is highlighted green, diminished performance is highlighted blue, and no change is highlighted yellow. The FG table illustrates that the HVAC operation features and temperature gradient features produce statistically better prediction results with 95% confidence on the LR, MB, and BM sensible heating loads (variables 65,69, and 73) as well as LR's latent heating and cooling loads, and MB's latent cooling load (variables 66, 68, and 72). Finally, the LR's sensible cooling load and MB's latent heating load were unchanged (variables 67 and 70). All other FG variable predictions are worse.

The MO1 experiments (shown in Table 4) with the HVAC operation and temperature gradient features produce statistically better LR, BM, and SF sensible heating load predictions (variables 74, 82, and 86). In addition, the features produce statistically better MB latent heating load predictions (variable 79). The load predictions for variables 75, 77, 81, 83, 85 and 89 were not statistically different. All other load predictions were statistically worse (variables 76, 78,

80, 84, 86, 87, 88).

Incorporating these additional features into the learning systems clearly provides mixed results since not all load predictions improved. In addition, the improved variables did not reach prediction rates similar to the better sensible load predictions (e.g., LR's sensible heating and cooling). Incorporating these findings with the Lasso regression results from Section 5, which suggest that necessary information for predicting latent loads is missing, shows that improving overall load predictions remains a challenging problem. To improve prediction, we suggest two complimentary directions: (1) bottom-up feature synthesis from existing data; or (2) top-down analysis, through continued interaction with domain experts, to determine additional EnergyPlus information that could improve approximations.

Figures 13(a) and 13(b) show the HVAC heating and cooling (on/off) features and the MO1 latent cooling and heating loads for the LR zone. Figure 13(a) shows that the heating is on mostly when the latent loads are non-zero, and the same is true for latent cooling loads in Figure 13(b). The current HVAC features correlate well with the MO1 sensible and latent loads, so improvement in predictive accuracy can be achieved through additional information.

Figures 14(a) and 14(b) illustrate that the FG's LR latent loads are uniformly distributed throughout the year. These latent loads are not necessarily indicative of HVAC operation. This issue is discussed further in Section 7.

7. Conclusions and Future Works

Using FFNN and Lasso regression, the ability to produce EnergyPlus approximation models for a residential building was demonstrated. The models use building envelope parameters selected by domain experts, an operation schedule, and weather data totaling 7–156 inputs for 3 benchmark datasets. These models were able to predict 15-minute values for most of the 80–90 simulation outputs deemed most important by domain experts within 5% (whole building electrical energy within 0.07%). While EnergyPlus can take 5 minutes to run,

the predicted outputs are calculated in 3 seconds, greatly reducing the simulation runtime required. In addition, variables requiring more complex non-linear models were identified by comparing the FFNN and Lasso models directly. However, these models had only moderate success at predicting sensible heating and cooling loads, and were unsuccessful at predicting the latent cooling and heating loads.

In an effort to improve the load predictions, we incorporated HVAC operation heating and cooling features, which indicated the on and off states for these respective operating conditions with mixed results. Based on Lasso regression's ability to automatically select relevant inputs, it can be concluded that either better use of existing information or additional information would be necessary to better predict the latent load variables. It is left as future work to analyze additional features and internal EnergyPlus variable information that can be incorporated into the prediction process without diminishing the EnergyPlus approximation's generality.

The Lasso model is able to predict an entire yearly simulation in $\sim$3 seconds, and the FFNN models can achieve the same execution time when run in parallel. These runtimes are considerably faster than the average EnergyPlus runtime ($\sim$2-3 minutes). During the process of calibrating a building model to utility data for creation of a legally useful model, or calculating an optimal retrofit, such performance increases could dramatically improve the speed at which such iterative simulation use cases can be completed.

Finally, three datasets (FG, MO1 and MO2) were constructed to determine the best EnergyPlus approximation model and typically requires multiple models. While attempts were made to generally represent residential buildings with these datasets, and efforts were made to quantify the robustness of prediction through testing and validation datasets, it is left to the reader to ascertain the robustness of surrogate models to other use cases requiring different simulation inputs (building descriptors) and outputs.

Given the supercomputing capabilities leveraged to generate the benchmark datasets, new ones could be generated relatively quickly to extend both the ro-

bustness and predictive accuracy of surrogate models generated from such data. There are two approaches to improving predictive accuracy. First, domain experts could further identify and isolate internal simulation variables that can be used with care to ensure the variables' robustness to allow a derivative predictive model's relevance to other buildings. Second, explore features synthesized from the existing variables. This first approach would require new benchmark datasets to be simulated whereas the second requires some expert time and computational cost for synthesizing general features.

8. Acknowledgements

This work was funded by field work proposal CEBT105 under the Department of Energy Building Technology Activity Number BT0305000. We would like to thank Amir Roth for his support and review of this project. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the US Department of Energy under Contract No. DE-AC05-00OR22725. Our work has been enabled and supported by data analysis and visualization experts at the National Science Foundation–funded RDAV (Remote Data Analysis and Visualization) Center of the University of Tennessee–Knoxville (NSF grant no. ARRA-NSF-OCI-0906324 and NSF-OCI-1136246).

Oak Ridge National Laboratory is managed by UT-Battelle, LLC, for the US Department of Energy under contract DE-AC05-00OR22725. This manuscript has been authored by UT-Battelle, LLC, under Contract Number DEAC05-00OR22725 with the US Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

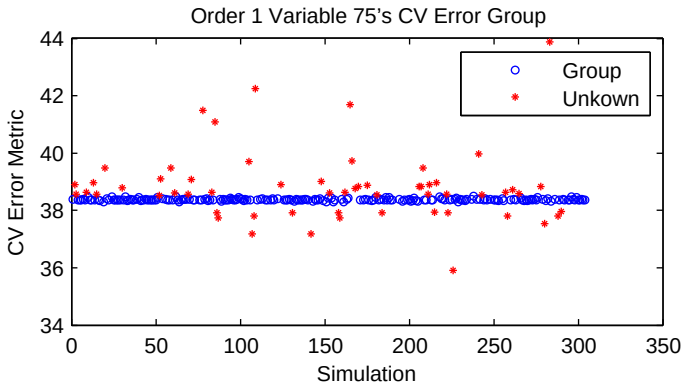
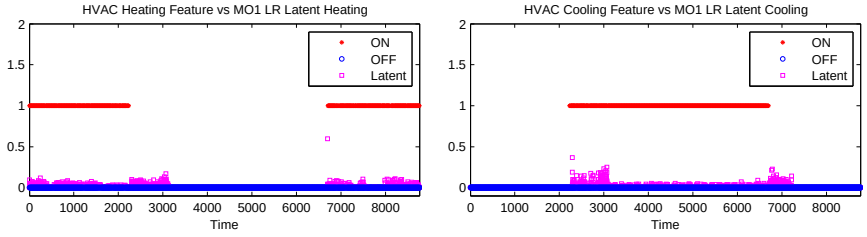
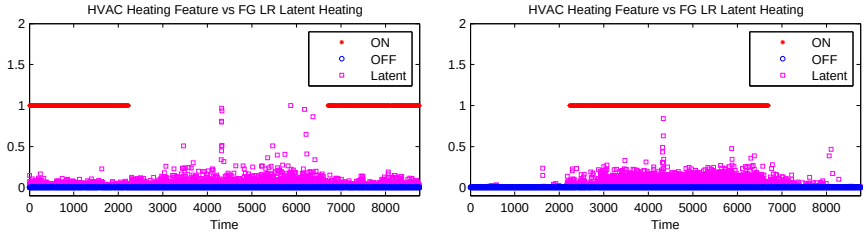


Figure 12: Illustrates the FFNN model's CV error clustering into a distinct cluster on the Markov Order 2 dataset.



(a) MO1 LR Latent Heating (b) MO1 LR Latent Cooling

Figure 13: The HVAC on and off operating feature overlayed onto a sample MO1 LR latent heating (Figure 13(a)) and sample MO1 LR latent cooling (Figure 13(b)).



(a) FG LR Latent Heating (b) FG LR Latent Heating

Figure 14: The HVAC on and off operating feature overlayed onto a sample FG LR latent heating (Figure 14(a)) and sample FG LR latent cooling (Figure 14(b))

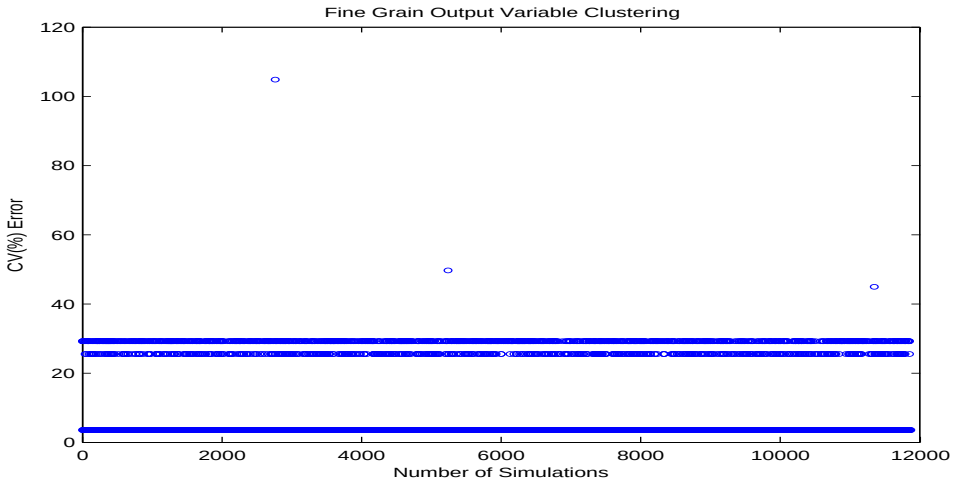


Figure 15: Identifies three potential outlier simulations within the Fine Grain dataset

Variable	RMSE (#×10^5J)	MEAN (#×10^5J)	CV (%)
65-Old	1.4022±0.1624	4.2998±0.2505	32.522±2.175
65-New	1.3596±0.1721	-	31.516±2.420
66-Old	1.0741±0.0334	0.4286±0.1343	250.657±4.533
66-New	1.0618±0.0333	-	247.796±4.536
67-Old	1.6204±0.2621	4.1195±0.1343	39.170±5.088
67-New	1.6216±0.2683	-	39.195±5.238
68-Old	1.3304±0.0158	0.4729±0.0111	281.406±3.914
68-New	1.3182±0.0148	-	278.826±4.124
69-Old	0.2922±0.0624	0.7780±0.0539	37.366±6.505
69-New	0.2764±0.0652	-	35.308±6.922
70-Old	0.1081±0.0043	0.0816±0.0076	133.239±9.119
70-New	0.1075±0.0042	-	132.564±9.041
71-Old	0.3848±0.0742	0.8996±0.0404	42.490±6.344
71-New	0.3913±0.0789	-	43.190±6.835
72-Old	0.4452±0.0295	0.2237±0.0050	198.806±8.934
72-New	0.4374±0.0309	-	195.306±9.626
73-Old	0.9445±0.3328	1.5752±0.5593	63.423±21.675
73-New	0.9123±0.3572	-	60.302±19.348
74-Old	0.5032±0.0064	0.2076±0.0082	242.661±8.171
74-New	0.5129±0.0064	-	247.333±8.410
75-Old	0.6210±0.1473	0.5637±0.2172	131.310±87.927
75-New	0.6621±0.1437	-	137.680±81.744
76-Old	0.4475±0.0061	0.1470±0.0032	304.523±4.892
76-New	0.4495±0.0060	-	305.913±4.947
77-Old	0.6698±0.0972	1.0371±0.0537	64.311±6.454
77-New	0.6681±0.1022	-	64.131±6.947
78-Old	0.3876±0.0105	0.2231±0.0092	173.867±5.115
78-New	0.3925±0.0105	-	176.058±5.192
79-Old	0.8449±0.1765	1.7647±0.0457	47.654±8.738
79-New	0.9044±0.1788	-	51.024±8.785
80-Old	0.4372±0.0070	0.1573±0.0037	277.937±2.613
80-New	0.4366±0.0068	-	277.518±2.708

Table 3: Comparison between the best FG FFNN model results, without HVAC features, and the best FG FFNN model with HVAC operation features.

Variable	RMSE (#×10^5J)	MEAN (#×10^5J)	CV (%)
74-Old	1.6084±0.0165	4.1829±0.0670	38.461±0.654
74-New	1.5322±0.0173	-	36.641±0.723
75-Old	0.4831±0.0132	0.1470±0.0048	328.746±9.685
75-New	0.4815±0.0130	-	327.706±9.755
76-Old	2.8421±0.0290	4.3868±0.0512	64.792±0.6371
76-New	2.9102±0.0290	-	66.356±0.605
77-Old	0.5497±0.0229	0.1812±0.0049	303.245±8.912
77-New	0.5519±0.0226	-	304.489±8.8140
78-Old	0.4313±0.0056	1.1951±0.0175	36.092±0.597
78-New	0.4653±0.0068	-	38.934±0.557
79-Old	0.1044±0.0026	0.1238±0.0043	84.380±1.939
79-New	0.1032±0.0025	-	83.396±2.042
80-Old	0.4558±0.0187	0.5947±0.0091	76.640±1.551
80-New	0.4713±0.0129	-	79.241±1.679
81-Old	0.0665±0.0021	0.0512±0.0017	130.159±5.149
81-New	0.0664±0.0021	-	129.885±5.121
82-Old	1.0850±0.0371	2.902±0.1096	37.476±2.840
82-New	1.0372±0.0371	-	35.820±2.678
83-Old	0.3082±0.0025	0.1178±0.0021	261.775±8.171
83-New	0.3080±0.0026	-	261.556±2.642
84-Old	0.1228±0.0155	0.0225±0.0071	564.939±111.943
84-New	0.0798±0.0175	-	364.840±81.744
85-Old	0.2670±0.0590	0.0693±0.0038	383.244±47.206
85-New	0.2669±0.0589	-	383.040±47.123
86-Old	0.7264±0.0101	0.9751±0.0256	74.531±1.1813
86-New	0.6800±0.0114	-	69.761±1.483
87-Old	0.3379±0.0038	0.1372±0.0023	246.345±3.786
87-New	0.3405±0.0037	-	248.260±3.814
88-Old	1.8742±0.0609	1.7166±0.0337	109.169±2.408
88-New	1.9058±0.0666	-	111.002±2.753
89-Old	0.1583±0.0111	0.0728±0.0013	217.361±14.703
89-New	0.1600±0.0111	-	219.708±14.657

Table 4: Comparison between the best MO2 FFNN model results, without HVAC features, and the best MO2 FFNN model with HVAC operation features.

References

[1] U.S. Dept. of Energy, Building Energy Data Book, D&R International, Ltd., 2010.
URL http://buildingsdatabook.eren.doe.gov/docs%5CDataBooks%5C2008_BEDB_Updated.pdf

[2] B. L. S. P. Ellis, Epx, a modified version of energyplus (2012).
URL http://bigladdersoftware.com/epx/

[3] T. Hong, F. Buhl, P. Haves, S. Selkowitz, M. Wetter, Comparing computer run time of building simulation programs 1 (3) (2008) 210–213.

[4] U. D. of Energy, Doe releases new version of energyplus modeling software (2011).
URL http://apps1.eere.energy.gov/news/progress_alerts.cfm/pa_id=651

[5] H. Sanyal, Y. H. Al-Wadei, M. S. Bhandari, S. S. Shrestha, B. Karpay, A. L. Garret, R. E. Edwards, L. E. Parker, J. R. New, Poster: Building energy model calibration using energyplus, supercomputing, and machine learning, Proceedings of the 5th National SimBuild of IBPSA-USA.

[6] Z. Qian, C. Seepersad, V. Joseph, J. Allen, C. Wu, Building surrogate models based on detailed and approximate simulations, American Society of Mechanical Engineers Journal of Mechanical Design 128 (4) (2006) 668.

[7] E. A. Institute, C. S. Group, Energy performance score 2008 pilot: Findings and recommendations report.

[8] B. P. M. H. S. C. D. Roberts, N. Merket, J. Robertson, Assessment of the u.s. department of energy's home energy scoring tool.

[9] H.-x. Zhao, F. Magoulès, A review on the prediction of building energy consumption, Renewable and Sustainable Energy Reviews 16 (6) (2012) 3586–3592.

[10] D. Popescu, F. Ungureanu, A. Hernández-Guerrero, Simulation models for the analysis of space heat consumption of buildings, Energy 34 (10) (2009) 1447–1453.

[11] A. Kusiak, G. Xu, Modeling and optimization of hvac systems using a dynamic neural network, Energy 42 (1) (2012) 241–250.

[12] A.-T. Nguyen, S. Reiter, P. Rigo, A review on simulation-based optimization methods applied to building performance analysis, Applied Energy 113 (2014) 1043–1058.

[13] K. Sun, T. Hong, S. C. Taylor-Lange, M. A. Piette, A pattern-based automated approach to building energy model calibration, Applied Energy 165 (2016) 214–224.

[14] W. Tian, R. Choudhary, A probabilistic energy model for non-domestic building sectors applied to analysis of school buildings in greater london, Energy and Buildings 54 (2012) 1–11.

[15] Y. Heo, V. M. Zavala, Gaussian process modeling for measurement and verification of building energy savings, Energy and Buildings 53 (2012) 7–18.

[16] M. Manfren, N. Aste, R. Moshksar, Calibration and uncertainty analysis for computer models–a meta-model based approach for integrated building energy simulation, Applied energy 103 (2013) 627–641.

[17] W. Tian, Q. Wang, J. Song, S. Wei, Calibrating dynamic building energy models using regression model and bayesian analysis in building retrofit projects, Ottawa, Canada (May 7–10, 2014).

[18] W. Tian, J. Song, Z. Li, P. de Wilde, Bootstrap techniques for sensitivity analysis and model selection in building thermal performance analysis, Applied Energy 135 (2014) 320–328.

[19] W. Tian, S. Yang, Z. Li, S. Wei, W. Pan, Y. Liu, Identifying informative energy data in bayesian calibration of building energy models, Energy and Buildings 119 (2016) 363–376.

[20] Y. Kang, M. Krarti, Bayesian-emulator based parameter identification for calibrating energy models for existing buildings, in: Building Simulation, Vol. 9, Springer, 2016, pp. 411–428.

[21] J. Hester, J. Gregory, R. Kirchain, Sequential early-design guidance for residential single-family buildings using a probabilistic metamodel of energy consumption, Energy and Buildings.

[22] X. Chen, H. Yang, K. Sun, Developing a meta-model for sensitivity analyses and prediction of building performance for passively designed high-rise residential buildings, Applied Energy.

[23] Z. ONeill, C. ONeill, Development of a probabilistic graphical model for predicting building energy performance, Applied Energy 164 (2016) 650–658.

[24] W. Tian, A review of sensitivity analysis methods in building energy analysis, Renewable and Sustainable Energy Reviews 20 (2013) 411–419.

[25] G. C. Rodríguez, A. C. Andrés, F. D. Muñoz, J. M. C. López, Y. Zhang, Uncertainties and sensitivity analysis in building energy simulation using macroparameters, Energy and Buildings 67 (2013) 79–87.

[26] C. J. Hopfe, G. L. Augenbroe, J. L. Hensen, Multi-criteria decision making under uncertainty in building performance assessment, Building and environment 69 (2013) 81–90.

[27] C. Hopfe, J. Hensen, Uncertainty analysis in building performance simulation for design support, Energy and Buildings 43 (2011) 2798–2805.

[28] P. Heiselberg, H. Brohus, A. Hesselholt, H. Rasmussen, E. Seinre, S. Thomas, Application of sensitivity analysis in design of sustainable buildings, Renewable Energy 34 (9) (2009) 2030–2036.

[29] R. Enríquez, M. Jiménez, M. Heras, Towards non-intrusive thermal load monitoring of buildings: Bes calibration, Applied Energy 191 (2017) 44–54.

[30] J. J. Robertson, B. J. Polly, J. M. Collis, Reduced-order modeling and simulated annealing optimization for efficient residential building utility bill calibration, Applied Energy 148 (2015) 169–177.

[31] G. R. Ruiz, C. F. Bandera, T. G.-A. Temes, A. S.-O. Gutierrez, Genetic algorithm for building envelope calibration, Applied Energy 168 (2016) 691–705.

[32] T. Yang, Y. Pan, J. Mao, Y. Wang, Z. Huang, An automated optimization method for calibrating building energy simulation models with measured data: Orientation and a case study, Applied Energy 179 (2016) 1220–1231.

[33] E. Fabrizio, V. Monetti, Methodologies and advancements in the calibration of building energy models, Energies 8 (4) (2015) 2548–2574.

[34] J. Kreider, J. Haberl, Predicting hourly building energy use: the great energy predictor shootout- overview and discussion of results, ASHRAE Transactions 100 (2) (1994) 1104–1118.

[35] K. Li, H. Su, J. Chu, Forecasting Building Energy Consumption using Neural Networks and Hybrid Neuro-Fuzzy System: a Comparative Study, Energy and Buildings.

[36] R. E. Edwards, J. New, L. E. Parker, Predicting future hourly residential electrical consumption: A machine learning case study, Energy and Buildings 49 (2012) 591–603.

[37] B. Dong, C. Cao, S. Lee, Applying support vector machines to predict building energy consumption in tropical region, Energy and Buildings 37 (5) (2005) 545–553.

[38] F. Chlela, A. Husaunndee, C. Inard, P. Riederer, A new methodology for the design of low energy buildings, Energy and Buildings 41 (9) (2009) 982–990.

[39] S. Filfli, D. Marchio, Parametric models of energy consumption based on experimental designs and applied to building-system dynamic simulation, Journal of Building Performance Simulation 5 (5) (2012) 277–299.

[40] L. Van Gelder, H. Janssen, S. Roels, Metamodelling in robust low-energy dwelling design, in: 2nd Central European Symposium on Building Physics, Vienna University of Technology, 2013, pp. 93–99.

[41] G. M. Laslett, Kriging and splines: an empirical comparison of their predictive performance in some applications, Journal of the American Statistical Association 89 (426) (1994) 391–400.

[42] R. Jin, W. Chen, T. Simpson, Comparative studies of metamodelling techniques under multiple modelling criteria, Structural and Multidisciplinary Optimization 23 (1) (2001) 1–13.

[43] L. Wei, W. Tian, E. A. Silva, R. Choudhary, Q. Meng, S. Yang, Comparative study on machine learning for urban building energy analysis, Procedia Engineering 121 (2015) 285–292.

[44] W. Tian, R. Choudhary, G. Augenbroe, S. H. Lee, Importance analysis and meta-model construction with correlated variables in evaluation of thermal performance of campus buildings, Building and Environment 92 (2015) 61–74.

[45] W. Tian, Y. Liu, Y. Heo, D. Yan, Z. Li, J. An, S. Yang, Relative importance of factors influencing building energy in urban environment, Energy 111 (2016) 237–250.

[46] E. Tresidder, Y. Zhang, A. Forrester, Acceleration of building design optimisation through the use of kriging surrogate models, 2012.

[47] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Foundations and Trends® in Machine Learning 3 (1) (2011) 1–122.

[48] W. Miller, et al., Zebralliance: An alliance maximizing cost-effective energy efficiency in buildings (2012).

[49] M. Deru, K. Field, D. Studer, K. Benne, B. Griffith, P. Torcellini, B. Liu, M. Halverson, D. Winiarski, M. Rosenberg, et al., Us department of energy commercial reference building models of the national building stock.

[50] R. Dodier, G. Henze, Statistical analysis of neural networks as applied to building energy prediction, Journal of solar energy engineering 126 (2004) 592.

[51] J. Yang, H. Rivard, R. Zmeureanu, On-line building energy prediction using adaptive artificial neural networks, Energy and buildings 37 (12) (2005) 1250–1259.

[52] P. Gonzalez, J. Zamarreno, Prediction of hourly energy consumption in buildings based on a feedback artificial neural network, Energy and buildings 37 (6) (2005) 595–601.

[53] S. Karatasou, M. Santamouris, V. Geros, Modeling and predicting building's energy use with artificial neural networks: Methods and results, Energy and buildings 38 (8) (2006) 949–958.

[54] H. Akaike, Information theory and an extension of the maximum likelihood principle, in: Proceedings of the Second International Symposium on Information Theory, B.N. Petrov and F. Caski, eds., Akademiai Kiado, Budapest, Hungary, 1973, pp. 267–281.

[55] G. Schwarz, Estimating the dimension of a model, The annals of statistics 6 (2) (1978) 461–464.

[56] H. Bozdogan, D. Haughton, Informational complexity criteria for regression models, Computational Statistics & Data Analysis 28 (1) (1998) 51–76.

[57] G. Cawley, N. Talbot, On over-fitting in model selection and subsequent selection bias in performance evaluation, The Journal of Machine Learning Research 11 (2010) 2079–2107.

[58] A. Hinneburg, D. Keim, et al., Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering, Citeseer, 1999.