Making Energy Modeling Faster

A machine learning based approach to emulate physics based simulation for energy consumption prediction

June 2024

Emi Lee, Junghwan Park, PhD, Jeff Friesen

agile ELECTRIFICATION PROJECT | THE DESIGN LAB

Executive Summary

The usage of energy modeling software and research has in large provided practical applications and stakeholder support across multiple industries from policy making to contracting with models such as National Renewable Energy Laboratory's ("**NREL**") EnergyPlus. NREL's EnergyPlus modeling engine enables users to utilize this open source technology for multiple use cases such as building and retrofit design, code compliance, green building certifications, informing users of qualifications for tax and utility incentives, and helping produce large-scale analysis to assist in policy decisions among other things[1]. Now, with the advancement of computational systems, technology companies such as Radiant Labs are able to utilize the core physics-based algorithms of EnergyPlus to predict energy consumption and upgrade options for larger areas ranging from neighborhoods to entire states through their physics-based engine ("**PBE**").

More specifically, the PBE is able to take algorithmically derived insights from EnergyPlus and, in tandem with their own algorithms, produce customer targeting and analytics dashboards, lead tracking and consumer energy roadmaps, hourly energy modeling, grid impact reports, as well as perform market factor analysis. Similar to other state-of-the-art engines, the PBE takes a physics-based approach using EnergyPlus to optimize for accuracy by accounting for weather and seasonal changes, HVAC properties, heat transfer, and more. However, while accurate, the PBE, like many physics-based models, is largely computationally intensive which can result in longer calculation times. Moreover, the equipment required to run PBE may be cost prohibitive for many users.

The machine learning based methodology developed by the Agile Electrification ("**AE**") Project 1 team emulates the energy prediction capabilities of PBE while addressing the challenges previously identified by inferring results faster and improving scalability through use cases such as web-based applications and single home computing. This approach has demonstrated that by accounting for building characteristics, weather, and time parameters it is possible to emulate physics-based modeling engines while significantly increasing output speed several-fold and preserving accuracy. Through using machine learning algorithms such as multilayer perceptions, this approach presents promising improvements to current methods for investigating future model emulation cases.

This whitepaper introduces the protocols developed by AE Project 1, starting from preprocessing the building characteristic data to model experimentation, testing and evaluation, final model implementation, and finally, a single home use case. Additionally this paper also provides open source code for the AE's PBE-trained emulator model and an example of an interactive, web-based sensitivity analysis use case showcasing the implementability and accessibility of a model created under this established method.

Notably, all models developed based on our protocols heavily depend on the regionality of the dataset which includes variables such as climate, building characteristics (e.g., materials and structural footprint), and occupancy schedule. Moreover, the training data was collected from New York and northern California and thus, locality in the dataset may impact the model's generalizability in regions outside of those in the training dataset and must be accounted for when training and implementing the open-source code in other regions. However despite this the protocol may remain valid at large.

Introduction

From large organizations to independent contractors and consumers, informed decision-making is critical when evaluating choices related to energy consumption and upgrade scenarios with current industry practices often involving physics-based modeling engines, such as EnergyPlus, to produce these estimations. By accounting for factors such as weather and building characteristics, through direct calculations in accordance with physical properties, these engines can produce estimates across various industry use cases.

The physics based engine ("**PBE**") developed by Radiant Labs takes this a step further by utilizing the core of NREL's EnergyPlus engine alongside their algorithms to produce customer targeting and analytics dashboards, lead tracking and consumer energy roadmaps, hourly energy modeling, grid impact reports, as well as perform market factor analysis, with a large portion of its analytical power deriving from its capacity to model hourly energy consumption. Using technologies such as the PBE, users can receive state-of-the-art estimations to guide their decision-making and create robust mental models of their energy consumption.

While providing one of the most accurate options, the current state of the PBE, and physics-based modeling engines at large, face the challenge of accessibility. In this context, accessibility refers not only to the physical availability of computing resources but also to the awareness and willingness of individual users, such as consumers or independent contractors, to engage with the technology. Many lack the necessary infrastructure or the motivation to seek out and implement these advanced systems, creating a significant barrier to entry.

In upholding Agile Electrification's (“**AE”**) overarching mission to harness cutting-edge technologies to enhance industry practices, AE Project 1 targets this pain point by proposing a new machine learning-based methodology to estimate hourly, daily, weekly, and annual energy consumption while aiming to increase computation speed while preserving accuracy.

To achieve this, the PBE will be emulated by the machine learning algorithms trained on building characteristics, weather properties, and time inputs to predict energy, natural gas, fuel oil, propane, wood cord, wood pellet, coal and total MMBTU[1] consumption provided by Radiant Labs.

Given the diversity of potential training datasets to which this model can be applied, developing a protocol to optimize the machine learning models serves to be worthwhile rather than just focusing on a single materialized model set; And thus, AE Project 1 aims to establish a protocol to build models for energy calibration emulation. In creating an open-source model and establishing a protocol, this project not only supports AE’s wider initiative by widening the accessibility of such technologies but also opens the door for future iterations of similar emulation scenarios and improvements. With this in mind, this paper will detail the methodology behind emulating a physics-based engine using machine learning and its results from data preprocessing to final implementation and wider use cases.

[1] A unit of energy measurement. BTU=British Thermal Unit, MMBTU=million BTU.

Methods

To approach emulating the PBE, the process was largely divided into two parts: data conversion/preprocessing and model design. This was in part due to the novel nature of this experiment as a whole as utility data is typically stored in a format focusing on human interpretability, resulting in representations that are not fully compatible with numeric models. Moreover, the recorded information must be standardized across all houses, further adding to challenges regarding nonlinearity and dimensionality, both of which are detailed in this section.

Data Conversion and Preprocessing

Overview

The dataset used for the model were sourced from the Radiant Labs' database, consisting of 1) anonymized building characteristics data of the houses in New York and California, 2) local weather station data, and 3) the hourly predicted energy use estimates using NREL's OpenStudio. Weather and hourly predicted energy use data were also accompanied with time information with the following data preparation protocols in the following section detailing this process in full.

Building Characteristics Data

A comprehensive dataset of 1.05 million houses was provided, encompassing every architectural detail potentially relevant to energy usage estimation. For example, this included wall thickness and azimuth, window size and material, light transmittance, number and type of appliances, energy efficiency of appliances and the type of energy they use, year built, lot size, floor area, building foundation materials and dimensions, whether each room is connected to air conditioning or heating, square footage of each room, connections between rooms, age, fuel, size, and energy efficiency rating of the heating and air conditioning system (“**HVAC”**). For the practical convenience of the training process, we randomly sampled 100,000 houses. Notably, this dataset may include imputed building characteristics by regional socio-demographic distribution using another RL’s product. However, regardless of whether the data was actually measured or imputed, they were entered into OpenStudio to synthesize the prediction results, and due to the primary object being to emulate the PBE, information source was not taken into account.

JSON-HPXML Conversion and Preprocessing for ML Training

The building characteristics were maintained in the industry-standard format: JSON-ified HPXML. Home Performance XML (“**HPXML**”) [2,3] is an XML-based building characteristics scheme developed by the Building Performance Institute in targeting a broad range of applications from auditing to retrofit simulation. For the ease of parsing, storing, and rendering, JSONified HPXML has been used in Radiant Labs as a lossless way to store information such as building characteristics.

This JSON-ified HPXML, while providing a robust representation of residential building details, inherently needs to be converted into numeric vectors which can be entered into numerical models including machine learning algorithms. A JSON-ified HPXML flattener was developed to streamline the process of converting JSON-ified HPXML files into a structured numerical array for training.

Key objectives and functionalities of this flattener include:

- **Flexible Design with Dynamic Output**: Takes JSON-ified HPXML and customize it to 1*N numerical vectors factoring in both the design file's variability and the exclusion of specific HPXML elements, enabling integration with a separate design file.
- **Standardized Data Representation:** Introduces a mechanism within the tool to cater to the variance posed by multiple instances of in-house entities in

JSON-ified HPXML. This would involve predetermining a maximum for each entity type to ensure consistent data structure for numerical processing.

- **Efficient Conversion & Future Compatibility:** Transforms JSON-ified HPXML files into a structured 1xN numerical vector suitable for numerical tools and algorithms accounting for future scalability
- **Easy-to-Use in Python Code:** Ensures the tool's design is intuitive, especially for Python developers. Such that even those with a basic grasp of Python should find the tool easy to navigate and operate.

To do this, the HPXML JSON Flattener operates through a two-step algorithm:

1. **Step 1: Transforming JSON to Ordered Tuples**
   Objective: To systematically organize HPXML JSON data into a sequenced collection of tuples, each representing a key-value pair from the original dataset. This structure ensures data consistency and prepares it for further conversion.
   Procedure*:*
    1. **Initialization:** Start by loading the HPXML JSON design and creating an empty list intended for tuple storage.
    2. **Data Processing:** Iterate through the design's predefined mapping order, capturing each data point (key and its associated value) for transformation.
    3. **Key Management:** Address unique and special keys by considering enumerated variables and default values, ensuring that all data points are accurately represented. Keys are used to order the key-value pairs in step 4.
    4. **Tuple Conversion:** Convert the processed data points into tuples and organize them into an ordered list, maintaining a consistent sequence of information.

2. **Step 2: Converting Tuples to a Numerical Vector**
   Objective: To translate the ordered tuples into a numerical vector, employing a binary (one-hot encoded) representation for each tuple, thereby facilitating a

consistent and computable data format.

***Procedure**:*

1. **Vector Initialization:** Start with an empty vector designed to hold the numerical representation of the data.
2. **Binary Conversion:** Transform each tuple into its binary representation, reflecting its unique position and value within the dataset. All possible values given by HPXML specification were used to encode. (See Appendix 1).
3. **Array Compilation:** Merge these binary representations into a comprehensive numerical array, ensuring a unified format ready for analysis.

By converting HPXML JSON data through these structured steps, the HPXML JSON Flattener tool achieves a standardized and numerically consistent format, suitable for training with numerical models.

Feature Selection

This newly converted dataset consists of a total of 7,697 variables describing the building characteristics of the house(s). Due to the similarities of the houses, there exists an inherent abundance of zeros of one-hot-encoded vectors and highly unlikely characteristics (e.g., coal-fueled appliances) leaving the majority of the input features consistently valued as 0 or other constant values. To remove recurring constant input features, a variance test was performed to only select variables with a variance above an arbitrary threshold, ensuring that the selected variables hold enough variance to possess significant predictive value.

Thus, to preserve as many building characteristic combinations as possible, 0 was selected as the threshold reducing the total number of features from 7,697 to 295 features with non-zero variances. (See appendix 2 for the list of the final input features)

Weather Data

RL’s dataset contains the local weather stations. It has 6 dimensions:

1. Temperature measured by a dry bulb thermometer,
2. Temperature measured by a wet bulb thermometer,
3. Relative humidity,
4. Wind speed,
5. Direct solar radiation, and
6. Diffuse solar radiation.

Each weather data point is recorded on an hourly time scale. When building models at other time scales (e.g., annual), weather information was aggregated to the required time interval using the minimum, maximum, mean, and variance.

Hourly Predicted Energy Use Data

RL conducted a series of simulations to calculate hourly energy use using OpenStudio. The output contains a large amount of data ranging from indoor airflow, conditioned temperature, temperature of each space, ventilation air speed per type, heating load, cooling load, and energy use per energy type. We only focused on 1) total energy use (million BTU), and 2) electricity use (kWh). Energy use was measured at an hourly frequency and when building models at other time scales, energy use information was simply summed to the required time interval.

Final Input Parameters

Post conversion and feature reduction, since we are targeting emulation at various time intervals (i.e., year, month, week, day, hour), inputs were aggregated accordingly. Building characteristic feature columns remained the same, but weather and prediction results were aggregated as described above.

The final the cleaned dataset consists the following features:

Variable Category	**Type**	**Description**
Time	Predictor	Time variables regardless of dataset interval measured in units of ● hour of the year ● hour of the day ● day of month ● day of year
Weather	Predictor	Weather variables in TMY3 format measuring: ● Drybulb temperature ● Wetbulb temperature ● Relative Humidity ● Diffused solar radiation ● Direct solar radiation
Building Characteristics	Predictor	Variables measuring building characteristics such as: ● Square footage, orientation, floors and foundations ● HVAC characteristics ● Household appliances and fuel type For the full list, see Appendix 2.
Natural Gas Consumption	Target	Variable measuring total gas consumption in Therm
Total Electricity Consumption	Target	Variable measuring total electricity consumption in kWh
Total MMBTU Consumption	Target	Variable measuring total electricity consumption in MMBTUs

Figure 1: Breakdown of variable categories used in the training and testing datasets

Model Development

Model Candidates

For model candidate selection, to maximize the speed while retaining accuracy, the nature of the dataset (heavily skewed with high dimensionality) was largely considered when selecting models, with criteria rooted in the models ability to handle non linearity and high dimensional data, inference and training speed, as well as past proven success in similar modeling scenarios. Through literature review and examining industry use cases, the preliminary white and black box model candidates were then selected and further narrowed in initial testing.

Model Evaluation and Final Selection

The initial experimentation phase evaluated model performance based on its ability to model annual MMBTU energy consumption. Each candidate was evaluated on their base model, then their model after hyperparameter tuning. The process is further detailed below:

- **Data Preparation:**
    - Test on a sample of 320 houses to effectively manage the dataset.
    - The process includes downloading all related information for these houses and loading it into a local database for HPXML conversion.
    - The datasets encompass various metrics, such as electricity and natural gas totals, and environmental conditions within and around the house.
    - HPXML documents undergo parsing and conversion.
- **Model experimentation and evaluation:**
    - Model is trained on this dataframe at an 80:20 split and evaluated on:
        - Performance: Mean Squared Error ("**MSE"**[2]) for MMBTU
        - Speed: Inference time for 1,000,000 houses sampled from 20,000 houses from the test data set with replacement.

[2] An evaluation metric to measure the distance between two sets of scalar values. It is defined as MSE = $(\text{sum}(x1 - x2))^2/n$.

- Model then undergoes hyperparameter tuning and reevaluation under the same metrics

It must be noted that despite having the option to train models tailored for all target variables such as total electricity consumption or fuel oil consumption, MMBTU was selected as the target variable across all experiments due to its ability to measure total consumption of all fuel types in a standardized unit. The equation used to calculate the MMBTU target column is as follows:

$$\begin{gathered}\text{MMBTU} = 0.003412 \times \text{ELECTRICITY TOTAL} + \\ 0.1 \times (\text{NATURAL GAS TOTAL} + \text{FUEL OIL TOTAL} + \text{PROPANE TOTAL} + \\ \text{WOOD CORD TOTAL} + \text{WOOD PELLETS TOTAL} + \text{COAL TOTAL})\end{gathered}$$

Figure 2: MMBTU equation used to calculate MMBTU target columns

By standardizing the total consumption to MMBTUs, this approach provides a robust and efficient way to evaluate overall accuracy and speed, especially when training on large datasets across several models.

We compared the model's accuracy and computation time across 5 common machine learning algorithms: 1) linear regression (baseline), 2) decision tree, 3) random forest, 4) XGBoost, and 5) MultiLayer Perceptron (“**MLP”**). The computation time estimation was done in a single machine to keep the estimates comparable.

For MLP, an “elimination” style hyperparameter tuning approach was taken with the Ray Tune Library (Ray Tune)[4,5]. Using Ray Tune, a three step tuning method was implemented.

The first step of elimination involved concurrently running 500 different parameter configurations of MLP with early stopping. By using early stopping, all parameter configurations without significant improvement in accuracy are dropped every 20 training rounds, leaving behind only the models showing substantial improvement across the training process, seen below. The early-dropping process was conducted across the training epochs. For each series of training, the best model up to that point was saved, for the later comparison.

The second step was sampling. The Ray Tune library initially performs training and evaluation by sampling random data points in the hyperparameter search space. Then, based on the evaluation results of these models, a Bayesian Search [6] is performed to select better hyperparameters. For this process, we used an implementation called AsyncHyperBandScheduler [7], which is included in the Ray Tune package.

The third step took the top 10 configurations and mapped out their mean absolute error[3] (“**MAE”**) and MSE. In this phase, the focus was placed on the range of MSE values per model rather than MAE; as by doing so, the overall accuracy and resilience of the model is prioritized, emphasizing its ability to predict with consistency rather than just closeness to the actual values.

In comparing the model with the overall range of MSE values, the final model with minimum MSE values for the validation set (i.e., random 10% samples of training dataset which is not utilized for the training) was selected and reevaluated similarly to the other models.

[3] Another metric to measure the distance between two sets of scalars. MAE = (abs(x1-x2))/n

Results

By using this method for initial experimentation, a total of 5 model candidates were evaluated in accuracy and speed. In both metrics, XGBoost showed the best performance.

Model	**MMBTU MSE**	**Recall Time (s/1M random inferences)**
Linear Reg (Baseline)	1.79E+06	2.63
Decision Tree	4.36E+05	4.09
Random Forest	1.46E+05	4.41
XGBoost (**XGB**)	9.22E+04	4.48
Multi-Layer Perceptron (**MLP**)	3.85E+05	7.01

Figure 3: Performance metrics across the 5 models tested, based on MMBTU

As seen above, overall, all advanced ML models such as XGB, MLP, and Random Forest largely outperformed baseline models (i.e., linear regression). MLP showed comparable results with XGB and Random Forest and it was observed during experimentation that to handle the nonlinearity of the data, especially for black box models, further calibration through hyperparameter tuning was an essential step when modeling high dimensional, nonlinear data.

The figure below shows the experiments conducted during the hyperparameter tuning process and the MSE accuracy at each of their experimental points (called epochs in the ML field). You can see the early-stopping and continuous performance improvement described above.

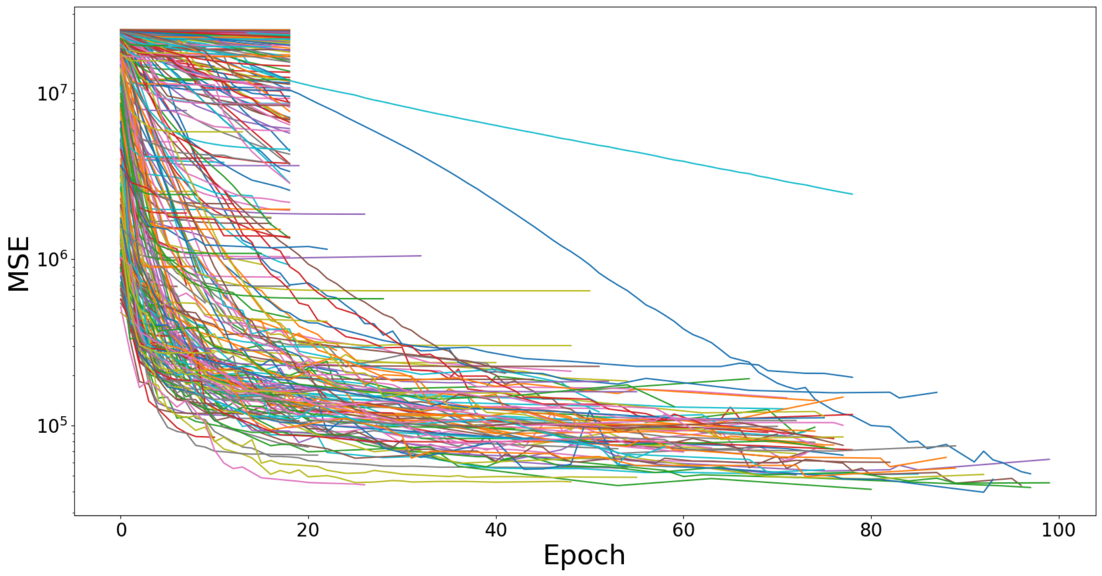


Figure 4: Hyperparameter tuning comparison of early-stopping across the configurations in MLP

Among the models and hyperparameters evaluated, the best parameters for the MLP model for annual MMBTU was chosen as below:

- Number of hidden layers: 4
- Number of neurons in each hidden layers: 86, 135, 11, 12
- Activation function: ReLU [8]
- Dropout rate: 0.0101
- Optimizer Type: Adam
- Learning Rate: 0.02, 76.8% every 9 epochs
- Batch Size: 32
- Kernel Initializer: Glorot Uniform
- Batch Normalization: Applied

The comparison between the actual value of the dataset (i.e., RL's OpenStudio) and predicted value by MLP is also shown below. The dashed diagonal line is the x=y line, signifying that the actual and the predicted are identical. Thus for interpretation, the closer to the line, the more accurate the prediction is to the actual value.

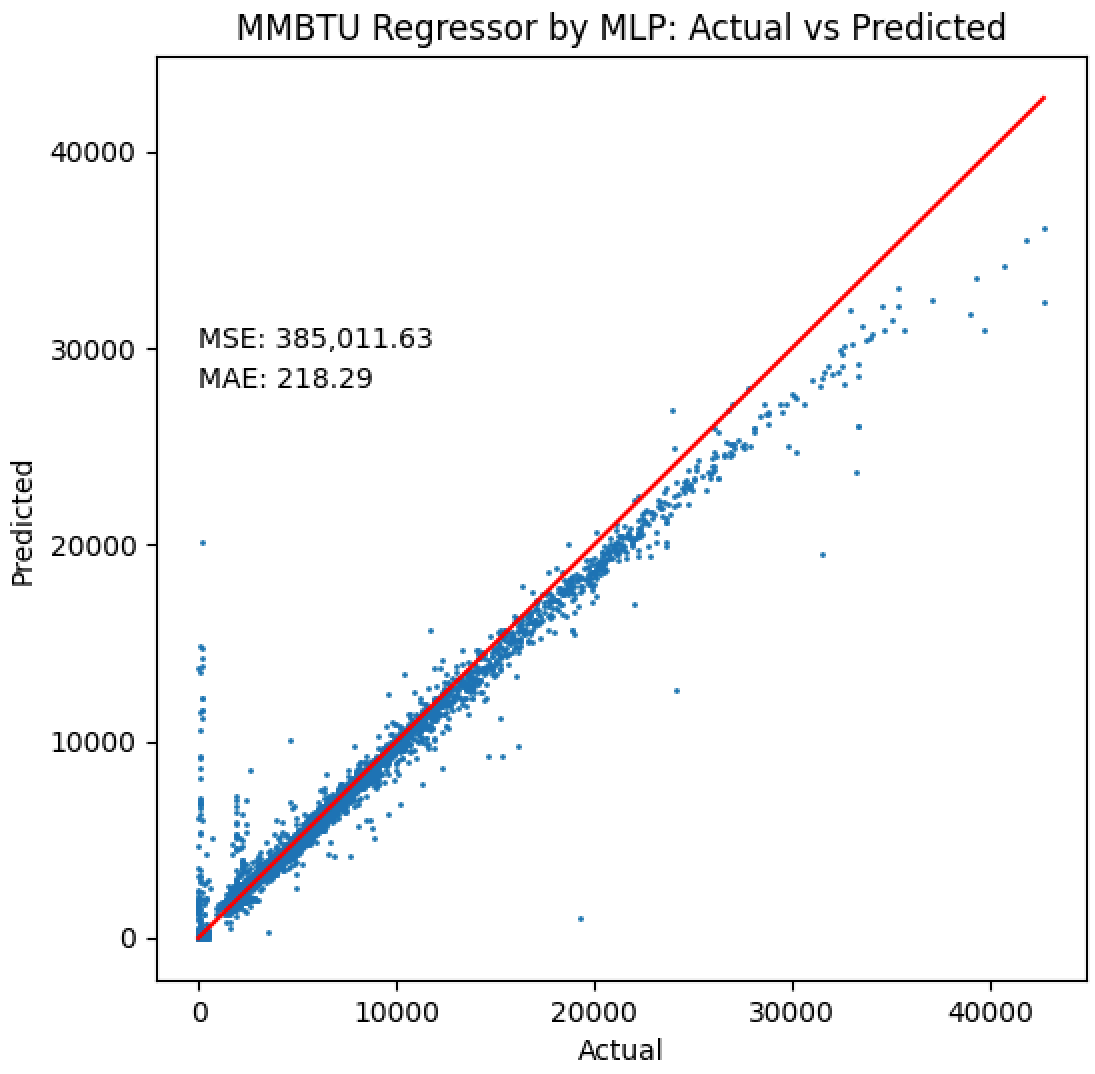


Figure 5: Comparison of predictions and actual values of MLP for MMBTU values

With a final MSE of 3.85E+05, the final model was able to make predictions for a million inferences in 7.01 seconds. This speed is equivalent to roughly 507 million houses per hour. In predicting MMBTU, the machine learning algorithms can finish the calculation considerably faster than RMLE's average calculation time (i.e., less than 1,000 houses per hour).

In conclusion, these findings highlight the effectiveness of using machine learning for energy estimation tasks. An example MLP model was produced with highest accuracy among model candidates, to be published in the public repository[4] along with the inference code.

In addition, the three-step hyperparameter tuning process has demonstrated the comparative advantages of machine learning models, particularly black box models like the MLP, in processing complex, high-dimensional, and nonlinear data sets. This approach, characterized by an initial "elimination" phase followed by a detailed analysis based on MSE values, was able to identify and refine the most efficient MLP model configuration. This process is also published in the open source project github repository for reference.

Overall, these outcomes highlight the potential of machine learning methodologies in enhancing energy estimation processes and suggest that such techniques can contribute significantly to advancements in energy analysis and sustainability efforts; And with further model experimentation and calibration, future iterations under this method show promise in increasing the overall efficiency and effectiveness of models for both energy estimation and use case scenarios.

[4] https://github.com/AgileElectrification/bill-calibration-public

Use Case 1 – Single Home Predictions

When increasing computational speed through a lighter weight, machine learning based approach, there are several use case scenarios that arise.

In the first use case, the MLP model can be used to predict hourly energy consumption of a single home, most closely resembling a potential application at the contractor and consumer level. Displayed below as an illustrative example, given the same data as the training set (weather, time, and building characteristics), the model was able to produce predictions for a single house with a low error while maintaining high inference speed compared to the PBE.

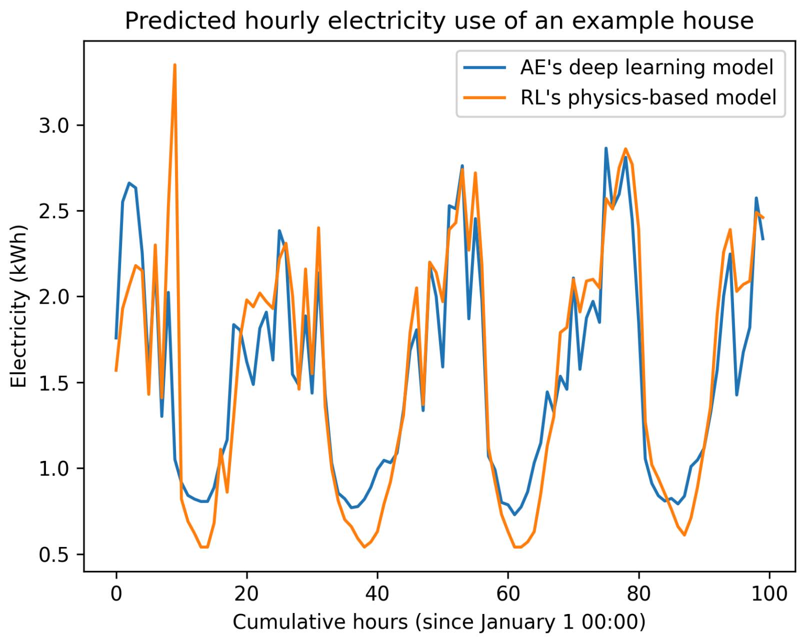


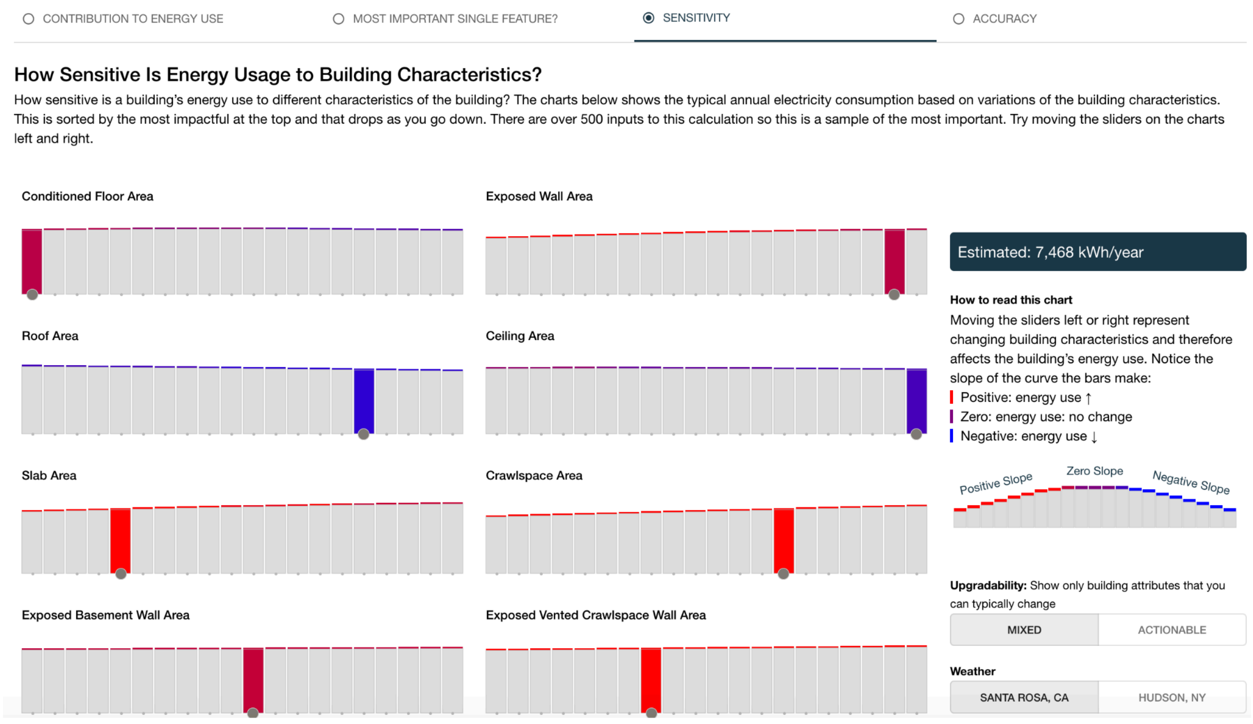
Figure 7: An illustrative figure for the predicted hourly electricity use of a single sample house across 100 hours

It is worth noting that the largest fluctuations in accuracy are present in extrema during or prior to larger local minimas in energy consumption. Currently, this is hypothesized to be largely related to occupancy schedule throughout the day such as peak usage times in the morning and evening, introducing complexities that may not be fully captured by the predictive models. Moreover, deep learning models, while adept at detecting patterns, may not fully account for the stochastic nature of human presence or absence, leading to discrepancies when compared to the PBE which may follow more deterministic rules set by an algorithm or the user.

Additionally, another limitation arises from the data itself. While data is imperative for training, after deployment, it is unlikely for users, especially at the individual level, to know the full extent of data to achieve the level of accuracy seen above. This can be mitigated by performing a sensitivity analysis which can reveal the most salient features the model uses when making a prediction and help guide users to prioritize collecting data on the features that significantly impact model performance. Moreover, improving data preprocessing methods, such as outlier detection and feature engineering can help in fine-tuning predictions. For future iterations, these insights can guide model refinements, ensuring that subsequent versions are more robust against the variabilities intrinsic to individual household electricity usage patterns.

Use Case 2 – Web Based Sensitivity Analysis

The second use case showcases an interactive, web based sensitivity analysis page[5] which highlights feature importances established by the model when predicting energy consumption. Aiming to demystify building energy usage, the page is broken into four main sections: the quantity of data necessary for accurate calibration, features by order of importance, feature interaction (sensitivity analysis), and an overall calibrated vs uncalibrated accuracy comparison.



How Sensitive Is Energy Usage to Building Characteristics?

How sensitive is a building's energy use to different characteristics of the building? The charts below shows the typical annual electricity consumption based on variations of the building characteristics. This is sorted by the most impactful at the top and that drops as you go down. There are over 500 inputs to this calculation so this is a sample of the most important. Try moving the sliders on the charts left and right.

Estimated: 7,468 kWh/year

How to read this chart

Moving the sliders left or right represent changing building characteristics and therefore affects the building's energy use. Notice the slope of the curve the bars make:

- Positive: energy use ↑
- Zero: energy use: no change
- Negative: energy use ↓

Upgradability: Show only building attributes that you can typically change

[✓ MIXED] ACTIONABLE

Weather

[✓ SANTA ROSA, CA] HUDSON, NY

Figure 8: Screenshot of the current web based sensitivity analysis website showcasing interactive updates to feature bars respective to their contribution to building energy consumption

[5] https://agileelectrification.org/playbook/blogs/demo-bill

By using a machine learning model, developers can extract feature importance from the MLP model through sensitivity analysis. However, it is well known that sensitivity analysis is difficult to interpret, particularly in the case of multidimensional input data. Based on the improved speed of our model, sensitivity analysis can be performed in real time on the website. This specific use case utilizes the tensorflow.js library to make real time inference calculations of feature interactions and estimated energy consumption when the user adjusts a feature value. As the user moves the slider, all bars, with interactive heights representing an output from the model for the respective building characteristic, update with the user's slider movement showing the impact of a feature on the overall building’s energy consumption.

Through interactive visualizations, this web integration use case can empower users to pinpoint cost-effective energy upgrades and understand the interactions between building characteristics and energy usage, demonstrating the merits of utilizing a computationally lighter, machine learning based method. Additionally, through the MLP models compatibility with Tensorflow.js , potential future implementations may include direct single house prediction through a similar web based application, further demonstrating its potential accessibility to a wider audience.

Conclusion

In an effort to increase computational speed from that of physics based models, the AE project has developed an alternative, machine learning based methodology to predict household energy consumption through emulation. By emulating PBE, this method provides a robust guideline for model experimentation and implementation across several use cases , ultimately, showcasing the potential of machine learning emulation for energy consumption estimation.

By using the methods outlined in this paper, from file conversion to final model implementation, the MLP model trained on building characteristics, weather, and time data was able to predict energy consumption for houses with significantly higher speed while maintaining accuracy compared to the PBE. These results ultimately demonstrate the potential and merits of this machine learning based method and shed an optimistic light towards investigating increasingly accurate models under this method.

Additionally, by utilizing machine learning to generate these predictions, several use case scenarios are now possible. From quick single house estimates, to interactive web deployment showcasing sensitivity analysis and feature interaction, machine learning based emulators are able to tackle the pain point of accessibility and scalability by performing these computations quickly and cost effectively across several platforms.

While there exists potential limitations regarding the specificity of regional data inputs, availability of data, as well as occupancy schedule affecting the training and application of such models, we are optimistic that with further training, feature engineering, or even custom regularization algorithms these concerns can be further mitigated to produce a more robust model increasingly more capable of modeling these behaviors and regionalities. At large, the groundwork laid by the AE project paves the way for further refinement of training protocols, data processing techniques, and model adaptability. Ongoing research will further extend these methodologies to a more diverse array of environments, augmenting the robustness and versatility of energy modeling tools.

Model and Code Availability

The final models and relevant example source code are available in: https://github.com/AgileElectrification/bill-calibration-public.

Acknowledgements

Made possible through the generous gifts directed to the Agile Electrification (“**AE”**) project within the Design Lab, this work involved contributions from student researchers, including Junghwan Park and Emi A. Lee. This project is part of broader efforts conducted under the umbrella of the AE project led by Andrew Krause (“**NPPS**”). The UCSD Design Lab has been the foundation of this research, and this effort was designed and led by UCSD faculty member Eric Hekler. Employees of Radiant Labs, including Adam Stenftenagel, Kelly O'Connell, Jill Pinsky and Joshua Mendoza, shared invaluable insights and domain knowledge at every step of the project. Mark Hansen (NPPS) and Chantelle Domingue (NPPS) have been exceptionally helpful in managing this project. Junghwan Park is an employee of the Ministry of Health and Welfare, Republic of Korea.

References

1. Building Energy Modeling. Accessed March 30, 2024. https://www.nrel.gov/buildings/building-energy-modeling.html

2. HPXML Specifications. Accessed March 30, 2024. https://www.hpxmlonline.com/specifications/

3. HPXML - HPXML Data Dictionary v3.1.0. Accessed March 30, 2024. https://hpxml.nrel.gov/datadictionary/3.1.0/

4. Liaw R, Liang E, Nishihara R, Moritz P, Gonzalez JE, Stoica I. Tune: A Research Platform for Distributed Model Selection and Training. *arXiv [csLG]*. Published online July 13, 2018. http://arxiv.org/abs/1807.05118

5. Ray Tune: Hyperparameter Tuning — Ray 2.10.0. Accessed April 1, 2024. https://docs.ray.io/en/latest/tune/index.html

6. Wu J, Chen XY, Zhang H, Xiong LD, Lei H, Deng SH. Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization. *Dianzi Keji Daxue Xuebao*. 2019;17(1):26-40. doi:10.11989/JEST.1674-862X.80904120

7. Li L, Jamieson K, Rostamizadeh A, et al. A System for Massively Parallel Hyperparameter Tuning. *arXiv [csLG]*. Published online October 13, 2018. http://arxiv.org/abs/1810.05934

8. Maas, Andrew L Hannun, Awni Y Ng, Andrew Y. Rectifier nonlinearities improve neural network acoustic models. *Proceedings of the 30th International Conference on Machine Learning*. 2013;28(JMLR: W&CP). http://robotics.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf

Appendix

Appendix 1. A Conversion Scheme from HPXML JSON to Numeric Vectors

https://github.com/AgileElectrification/bill-calibration-public/blob/main/r1/data/design.json

Appendix 2. The Final Input Feature List

https://github.com/AgileElectrification/bill-calibration-public/blob/main/r1/data/non_zerostd_column_list.json