Pacific Northwest
NATIONAL LABORATORY

PNNL-28905 ver 2

Forecasting Commercial Building Electricity Consumption, Zone Airflow and Zone Temperature: Update

Development of a Generalized Machine Learning Approach

January 2020

Elliott Skomski*
Ryan Haight*
Joon-Yong Lee
Sen Huang
Brian Hutchinson*
Srinivas Katipamula

U.S. DEPARTMENT OF ENERGY

Prepared for the U.S. Department of Energy
under Contract DE-AC05-76RL01830

DISCLAIMER

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC05-76RL01830

Printed in the United States of America

Available to DOE and DOE contractors from
the Office of Scientific and Technical Information,
P.O. Box 62, Oak Ridge, TN 37831-0062
www.osti.gov
ph: (865) 576-8401
fox: (865) 576-5728
email: reports@osti.gov

Available to the public from the National Technical Information Service
5301 Shawnee Rd., Alexandria, VA 22312
ph: (800) 553-NTIS (6847)
or (703) 605-6000
email: info@ntis.gov
Online ordering: http://www.ntis.gov

PNNL-28905

Forecasting Commercial Building Electricity Consumption, Zone Airflow and Zone Temperature: Update

Development of a Generalized Machine Learning Approach

January 2020

Elliott Skomski*
Ryan Haight*
Joon-Yong Lee
Sen Huang
Brian Hutchinson*
Srinivas Katipamula

Prepared for
the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Pacific Northwest National Laboratory
Richland, Washington 99354

Abstract

The power grid of the United States is being transformed to make it smarter, more efficient, and cleaner. This transformation is leading to the addition of a significant amount of energy generated by distributed, variable, and renewable resources. Because of the variable nature of renewable generation, the short- and long-term supply and demand imbalances are less predictable, and conventional approaches to mitigating the imbalances will be less efficient or cost effective.

To address this challenge and to support the mission and the vision of this transformation, the U.S. Department of Energy's (DOE's) Office of Energy Efficiency and Renewable Energy (EERE) Building Technologies Office has developed a Grid-Interactive Efficient Building Strategy.[1] The strategy focuses on simultaneously improving overall building energy efficiency and supporting the reliability and resilience of the electric grid more efficiently and at a lower cost. In addition, EERE and DOE's Office of Electricity created an initiative led by DOE and supported by the national laboratories under the Grid Modernization Lab Consortium structure to enhance grid modernization.

The work reported in this document is part of the first set of projects funded under the initiative to design, develop, and validate scalable transactive control technologies for the commercial buildings sector. Transactive controls requires the ability of individual end-use loads to express flexibility as a function of a transactive signal (e.g., price). Empirical grey- and black-box models have been widely used to express flexibility. Although this approach is generally easy to construct and simple to use, it does not capture non-linear behavior that some end-use loads represent. Therefore, Pacific Northwest National Laboratory (PNNL) with support from Western Washington University conducted this research to explore the use of deep machine learning (ML) techniques.

The work reported in this document is limited to forecasting whole building electricity consumption, the zone airflow, and the zone temperature predictions.

[1] Details on the Grid-Interactive Efficient Building Strategy can be found at: https://www.energy.gov/eere/buildings/grid-interactive-efficient-buildings

Acknowledgments

The authors acknowledge the Buildings Technologies Office (BTO) of the U.S. Department of Energy Office of Energy Efficiency and Renewable Energy for supporting the research and development effort. The authors also acknowledge the valuable guidance from Ms. Erika Gupta, the Sensors and Controls Technology Development Manager at BTO, Robert Lutes for peer review comments, and Colleen Winters for editing the report, both from Pacific Northwest National Laboratory.

Acronyms and Abbreviations

AHU	air handling units
CEF	Washington State Clean Energy Fund
CNN	convolutional neural network
CPU	central processing unit
DOE	U.S. Department of Energy
EERE	DOE Office of Energy Efficiency and Renewable Energy
GPU	graphic processing unit
GRU	gated recurrent unit
HVAC	heating, ventilation, and air conditioning
LSTM	Long Short-Term Memory
ML	machine learning
RMSE	root mean square error
RNN	recurrent neural network
ROM	reduced-order model
RTU	roof top unit
seq2seq	sequence-to-sequence (model)
VAV	variable air volume

Contents

Figures

Tables

1.0 Introduction

The power grid of the United States is being transformed to make it smarter, more efficient, and cleaner. This is leading to the addition of significant amounts of power generated by distributed, variable, and renewable resources. Because of the variable nature of renewable generation, short- and long-term supply-demand imbalances are less predictable, and conventional approaches to mitigating the imbalance will be less efficient and cost effective.

To address this challenge and to support the mission and the vision of the U.S. Department of Energy's (DOE's) Office of Energy Efficiency and Renewable Energy (EERE) Building Technologies Office has developed a Grid-Interactive Efficient Building Strategy[1] that focuses on simultaneously improving overall building energy efficiency and supporting the reliability and resilience of the electric grid more efficiently and at a lower cost. In addition, EERE and DOE's Office of Electricity created an initiative led by DOE and supported by the national laboratories under the Grid Modernization Lab Consortium structure to enhance grid modernization.

The work reported in this document is part of the first set of projects funded under the initiative to design, develop, and validate scalable transactive control technologies for the commercial building sector. Transaction-based controls, which exchange, negotiate, and respond to information, are the fundamental building blocks of the transactive energy vision. The GridWise® Architecture Council (GWAC 2019) defines transactive energy as "… techniques for managing the generation, consumption, or flow of electric power within an electric power system through the use of economic or market-based constructs while considering grid reliability constraints." To deploy transactive controls in buildings, individual controllable loads/devices must express their demand flexibility as a function of a transactive signal (e.g., price). To estimate demand flexibility requires end-use consumption models that are easy to use and accurate. Empirical grey- and black-box models have been widely used for such forecasts. Although these simplified empirical models are generally easy to construct and use, they typically do not capture the non-linear behavior that some end-use loads exhibit. Therefore, Pacific Northwest National Laboratory (PNNL) with support from Western Washington University started this research to explore the use of deep machine learning (ML) techniques for load forecasting. This report documents that effort.

1.1 Introduction to Transactive Controls[2]

Transaction (e.g., contract) networks (Smith 1980) and agent-based systems present an opportunity to implement strategies in which highly optimized control (both local and global) is an inherent attribute of the strategy rather than an explicitly programmed feature. The premise of transaction-based control is that interactions among various components in a complex energy system can be controlled by negotiating immediate and contingent contracts on a regular basis in lieu of or in addition to conventional command and control. Each device is given the ability to negotiate deals with its peers, suppliers, and customers to maximize revenues while minimizing costs. This is best illustrated by an example.

A typical building might have several chillers that supply a number of air-handling units (AHU) with chilled water on demand. If several AHUs require the full output of one chiller and another

[1] Details on the Grid-Interactive Energy Building Strategy can be found at: https://www.energy.gov/eere/buildings/grid-interactive-efficient-buildings

[2] This introduction has been adapted from Katipamula et al. (2003).

AHU suddenly also requires more cooling capacity, traditional building control algorithms simply start up a second chiller to meet the demand and the building's electrical load increases accordingly.

A transaction-based building control system behaves differently. Instead of honoring an absolute demand for more chilled water, the AHU requests such service in the form of a bid (expressed in dollars as a function of service level [e.g., comfort]), increasing its bid in proportion to its "need" (i.e., divergence of the zone or supply air temperature from its set point). With knowledge of the electric rate structure, the chiller controls can easily express the cost of service as the cost of the kilowatt-hours needed to run the additional chiller plus the incremental kilowatt demand charge (if it applies). If the zone served by this AHU just began to require cooling, its "need" is not very great at first, so it places a low value on its bid for service and the additional chiller stays off until the level of need increases.[1] Meanwhile, if another AHU satisfies its need for cooling, the cost of chilled water immediately drops below the bid price because a second chiller is no longer required, and the AHU awaiting service receives chilled water. Alternatively, a peer-to-peer transaction can take place in which an AHU with greater need for service displaces (literally outbids) another chiller whose thermostat is nearly satisfied.

In this way, the contract-based control system accomplishes several things. First, it inherently limits demand by providing the most "cost-effective" service.[2] In doing this, it inherently prioritizes service to the most important needs before serving less important ones. Second, it decreases energy demand and consumption by preventing the inefficient operation of an entire chiller to meet a small load (assuming no AHU is willing to pay the additional cost of service to start the second chiller).

Third, contract-based controls inherently propagate cost impacts up and down through successive hierarchical levels of the system being controlled (in this example, a chiller or a boiler that provides cooling or heating, an AHU that provides air circulation, and the zone). The impacts on the utility bill, which are easily estimated for the chiller operation, are used as the basis for expressing the costs of AHU and zone services. Using cost as a common denominator for control makes the expression of what is effectively a multi-level optimization much simpler to articulate than an engineered solution would be. It allows controls to be expressed in local, modular terms while accounting for their global impact on the entire system.

In effect, the engineering decision-making process is subsumed by a market value-based decision-making process that indirectly injects global information conveyed by market activity into the local engineering parameters that govern the behavior of individual systems over multiple time scales.

For contract-based controls to work, individual controllable loads that are participating in a market will have to express their flexibility in the form of a price vs. capacity (i.e., their willingness to consume varying quantities for energy based on the cleared price). If the price is low, they may be willing to consume more and if the price is high, they may curtail their consumption. To establish these price-capacity curves requires accurate models that can forecast energy consumption into the future. Some markets not only require curves for the immediate clearing period, but also curves into the future (for example, 24 hours ahead). Traditionally, researchers have relied on black- or grey-box empirical regression models

[1] This is true because the cost of starting another chiller will be greater than the bid placed by this zone for service.

[2] By "cost effective," we mean the best match for the level of service to the cost of service.

because of their simplicity. Although these models are reasonably accurate at forecasting, their accuracy decreases when behavior of the system is highly non-linear. There is an opportunity to improve the development and use of the models developed using ML techniques.

1.2 Why Machine Learning?

Machine learning means that a computer "learns" to automatically find relationships in a dataset to solve the problem, even if there is no logic instruction provided by a human. The goal of ML is to find an accurate relationship between the given data and the actual corresponding output by identifying a function that can best explain the given data. Overall, ML is the process of setting up a problem, identifying a model family, and estimating the model from the training data.

Over the past two decades, ML has made significant strides in health care, manufacturing, education, financial modeling, policing, and marketing (Jordan and Mitchell 2015). Machine learning techniques have also been applied to problems in the building domain since the 1990s (Kreider et al. 1995). However, many of the earlier ML techniques had several limitations and constraints. For example, many ML techniques use “supervised” learning to infer from past patterns. However, this approach requires a significant amount of data as well as data that span all modes of building operations. While it is possible to generate test data for some applications to help in the development of an ML model, that is not always the case. Therefore, the application of ML in the building domain has seen limited success over the past two decades. New hybrid approaches that rely on limited data are needed for application in the building domain.

1.3 Application of Machine Learning in the Building Domain

Advancements in ML made from other domains also have a significant potential in the building domain, especially in the improvement of advanced and transactive controls. Many advanced and transactive control applications require an accurate load forecast. Currently, statistical grey- and black-box models are widely used for load forecasting. These models are generally reliable if the relationship between dependent variables (load) and independent variables (outdoor temperature, hour of day, etc.) is linear. However, if the relationship between the dependent and independent variables is complex and/or non-linear, traditional modeling approaches usually have high levels of uncertainty. Machine learning techniques are better at handling complex and non-linear relationships.

Machine learning also has significant potential in enhancing advanced and transactive controls. Many advanced and transactive control applications are generally open-loop controls, which means there is often no feedback to correct/adjust the control parameters to achieve the desired outcome within the control interval. For example, in a transactive control application that manages the electricity consumption of a whole building to a desired target, certain control actions are taken based on the rated energy consumption information or by using a model prediction of the consumption. Because of measurement and modeling errors, the transactive control actions based on these assumptions generally do not result in the exact desired result. The use of ML techniques that learn from past experience to make future decisions will significantly improve transactive controls. While there are other areas within the building domain in which ML techniques can be useful as well, for this effort we focus on advanced and transactive controls and more specifically forecasting whole building electricity needs. More details on prior applications of ML in the building domain can be found in Section 2.0.

1.4 Why is Machine Learning Taking Off Now?

Machine learning has proved to be very effective in a number of areas, such as speech recognition[1,2] (Hinton et al. 2012; Sainath et al. 2013; Chan et al. 2015), natural language processing (Collobert et al. 2011), and image classification[3,4,5] (Krizhevsky et al. 2012; Simonyan and Zisserman 2014; He et al. 2016; Tompson et al. 2014). Recently, deep learning has emerged as one of the most popular and powerful concepts within ML (LeCun et al. 2015). The building domain is not left untouched by the current wave of interest in ML, especially deep-learning ML. Machine learning is well-suited for the building domain, given the complexity and non-linear operations and the increasing availability of data. However, several challenges also exist when applying ML learning that are unique to the building domain. In the following discussion, we outline some factors that have contributed to the recent popularity of ML techniques (Jordan and Mitchell 2015), and explain how these techniques apply to the building domain.

1.4.1 Availability of Data

Widespread deployment of both traditional and newly emerging, low-cost sensor technologies has led to the ubiquitous availability of data in several domains. We also have entered the era of “big data.” The exponential growth of big data has enabled researchers to develop and train ML algorithms that are powerful in revealing non-linear and complex patterns. This has resulted in the ability to mine useful and actionable information. The building domain has not been untouched by this trend. The increasing use of building automation systems to manage buildings, especially modern buildings, has resulted in much more available data that spans long time windows (e.g., multiple years). While data is certainly increasing in “breadth” (time window), it is still limited from a “depth” perspective; sensing is limited to measuring certain key quantities of interest, such as zone temperatures and building energy consumption. Hence, there is a need to develop ML methods that can navigate this data depth challenge in buildings.

1.4.2 Higher-Performance Computation

Significant improvements in hardware affordability have accelerated the adoption of ML in various domains. The advances in hardware (i.e., central processing units [CPUs] and graphics processing units [GPUs]) and new computational models have created a large opportunity for ML to handle a high degree of parallel operations and perform matrix multiplications in an efficient manner. In particular, GPUs provide a parallel architecture that has been especially powerful when applied to the types of calculations needed for neural networks. Additionally, CPUs have made advances for better parallelization and more efficient matrix computations. However, these developments have not yet penetrated the building domain, and the computational infrastructure used within buildings is still quite limited. New computational approaches (e.g., Cloud computing), which provide access to high-performance computational resources in an inexpensive manner using pay-per-use business models, can potentially be leveraged.

---

[1] https://www.cs.toronto.edu/~gdahl/papers/deepSpeechReviewSPM2012.pdf
[2] https://arxiv.org/abs/1508.01211
[3] https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
[4] https://arxiv.org/abs/1512.03385
[5] https://arxiv.org/abs/1409.1556

1.4.3 Advanced Algorithms

The advent of large datasets and high-performance computing hardware makes it easier to test and develop advanced algorithms because hardware limitations that existed in the past are no longer a constraint. These advances in learning computing and algorithms have made training deep architectures feasible; it has been shown that such advanced ML models can provide superior or comparable performance with state-of-the-art methods. For example, deep-learning algorithms have been shown to perform better than human experts in various fields (e.g., some learning algorithms have become world champions at a variety of games from Go to Atari games[1] [Mnih et al. 2013]) and in tasks such as lip reading[2] (Chung et al. 2016). These advanced algorithms appear promising in the building domain as well.

1.4.4 Software Libraries

Deep ML algorithms can be very easily built and customized today using available open-source software libraries such as *Tensorflow*, *Caffe*, *Torch*, and *Theano*. Various ML algorithms are made available as "black boxes" within these libraries. The advent of such software packages is unlocking new possibilities that were unimaginable a few years ago. These libraries allow very powerful learning algorithms to be built with a few lines of code. The availability of these libraries also allows researchers to build, implement, and maintain ML systems for a variety of real domains. This is promising in the context of building systems, where such libraries can be used to build control applications in the Cloud using less time and effort.

1.4.5 Scalability/Portability/Transferability

In the past, a major limitation of ML was that models trained on one system were not easily generalizable to other systems, especially in the building domain. This creates a challenge with regard to scaling the ML algorithms to a large set of systems. For example, a model trained on one building can be quite inaccurate when applied to a different building. In recent years, transfer learning has been proposed as a potential solution to address this problem, where some learned attributes of a model trained on one system are re-used to build a model for a different but related system (Pan and Yang 2010). Without transfer learning, ML can require a large amount of training data to achieve satisfactory accuracy. Although transfer learning has been demonstrated to address the generalizability and scalability challenges of ML models in other domains, it has been investigated only to a limited extent for buildings.

1.5 Challenges with Machine Learning in the Building Domain

Some of the key challenges in applying ML in the building domain are summarized in this section.

1.5.1 Lack of "True" Deep Learning

Applications of deep learning seem largely unexplored. Very few publications are available, and most of those limit the architecture to a single hidden layer in a feed-forward mode. In contrast, it has been rigorously shown in other domains (e.g., natural language processing) that architectures such as Long Short-Term Memory (LSTM) used in stacked/deep recurrent neural networks (RNN) can significantly improve modeling accuracy.

[1] https://deepmind.com/research/publications/playing-atari-deep-reinforcement-learning/
[2] https://www.robots.ox.ac.uk/~vgg/publications/2016/Chung16/chung16.pdf

1.5.2 Lack of Generalizability

The buildings considered in each of the papers surveyed (see Section 2.0) were of different sizes, ranging from one to 24 zones. The ML model developed for each of these buildings was a standalone effort, meaning that the model architecture was optimized and tuned to provide satisfactory accuracy for the single building in question. Consequently, the models varied quite drastically. For example, in the case of single hidden layer models the number of neurons ranges from four to 128. There is a need to develop a generalized modeling framework that can work across buildings. This has not yet been attempted in the building domain. In the absence of such a framework, the modeling problem for each building becomes tedious, and the appropriate architecture and parameters must be determined separately for each building, which becomes a time-consuming effort. Also, the lack of generalizability impedes the use of these models for large-scale investigations, such as those at the level of a distribution grid involving tens or thousands of buildings. In the context of integrating buildings and the grid, this is a severe limitation.

1.5.3 Data Requirements

Each of the buildings modeled in the state-of-the-art modeling approach were well instrumented, with sensors providing time-series data of physical parameters such as supply airflow rates, humidity, and carbon dioxide concentrations, orchestrated using a sophisticated building automation system. However, most existing buildings do not have such a fine-grained sensing infrastructure. In fact, over 85% of buildings in the United States do not use building automation systems (Rawal 2019). Therefore, there is a need to develop ML algorithms that can work with limited data.

1.5.4 Challenges and Potential Solutions

Although deep architectures can learn more powerful models than shallow networks, simply adding layers in neural networks will not provide significant improvement for load forecasting. Several researchers (Amarasinghe et al. 2017; Shi et al. 2017) indicate that increasing the depth of the network by increasing the number of layers and units reduces training dataset error. However, increasing the depth of the network can only improve performance on testing data up to a limited number of layers, which suggests the occurrence of overfitting due to the lack of data diversity and network parameter redundancy, and the difficulty of optimizing very deep models. Deep learning is limited by the quality and quantity of the data inputs. It is important to increase data diversity to accurately train the load-forecasting model. When increasing the number of layers in a deep neural network, model complexity will increase exponentially and eventually become excessive for limited training data. As a result, the model will begin to capture the noise and fit the training data too well, hence negatively impacting the predictive performance. To improve the predictive power of deep-learning algorithms, dropout can be used as a regularization methodology. The dropout at the input and hidden layers specifies the proportion of neurons to be randomly ignored during model training and is introduced to ensure the robustness of deep-learning models (Srivastava et al. 2014). Bootstrap aggregation has been proposed as an effective way to avoid overfitting (Goodfellow et al. 2013).

Also, most deep ML networks used for load forecasting usually have only one or two hidden layers (Marino et al. 2016; Amarsinghe et al. 2017; Fan et al. 2017). This indicates deep learning has limited advantages when compared to neural networks with a shallow architecture for the load-forecasting problem. To develop robust and reliable estimations of these model coefficients, a substantial amount of data is needed. It is possible that a few hidden layers are good enough only when input features are limited.

There is a paucity of literature exploring the impact of network architecture on the performance of deep-learning models for energy forecasting. To address this limitation, we undertook a detailed investigation into how hyperparameters of deep-learning architectures can impact the energy-forecasting capabilities of the models. We used real field data from several commercial buildings spanning multiple years in our analysis. Hyperparameters are parameters that are set before learning begins. Therefore, they differ from other parameters (e.g., weights, biases) that are derived and updated as part of the learning process. Hyperparameter tuning involves selection of optimal hyperparameters for the ML algorithms.

1.6 Project Purpose and Scope

Key goals of the Clean Energy and Transactive Campus Project include demonstrating that significant energy savings are possible in commercial buildings and that the reliability of the power grid can be maintained even under large-scale integration of energy generated by renewable resources at a regional scale by using the transactive control technology to coordinate many distributed energy resources. These distributed energy resources include controllable building loads (e.g., rooftop units, AHUs, chillers, fans, lights), energy storage systems (electric and thermal), smart inverters for photovoltaic solar systems, and electric vehicles.

Deployment of transactive controls requires various models (e.g., zone temperatures, loads); this report details the development of whole building electricity models, as well as models for temperature and airflow forecasting in individual building zones. To inform the process of developing these models, this work aims to provide insight into the data and model requirements for designing and training generalized deep-learning models for whole building electricity, zone airflow, and and zone temperature forecasting in commercial buildings.

1.6.1 Whole Building Load Forecasting Objectives

For the whole building load forecasting task, the main objectives are to investigate how our models respond to changes in training set characteristics and model hyperparameters, and to measure generalization performance. In this work, we aim to:

1. Identify suitable ML techniques that can be trained using limited “depth” and “width” of data for the building domain.
2. Establish the minimum amount of data needed for a reasonably accurate (within 10% of actual) deep load forecasting model.
3. Identify how conditions during each season affect model parameters.
4. Determine the effect of data acquisition frequency (i.e., 15 or 60 minutes) on model accuracy.
5. Explore the use of transfer learning methods for the building domain.

1.6.2 Zone Forecasting Objectives

For the zone forecasting task, the main objectives are to create models that are effective at predicting the airflow rate of zones that operate with a variable air volume unit (VAV) and predict the temperature of zones that operate with a roof top unit (RTU). In this work, we aim to:

1. Identify and implement a deep-learning model architecture that is suitable for both temperature and airflow predictions.

2. Create a deep ML model that outperforms a model previously implemented by PNNL engineers for airflow prediction.
3. Explore embedding techniques for effectively handling missing values in zone measurements.

This report describes the efforts of designing, developing, and testing ML methods that address the objectives listed above. Prior ML work in the building domain is summarized in Section 2, followed by an overview of ML methods used in Section 3 to forecast whole building electricity loads. Section 4.0 documents the experimental setup for zone temperature and airflow forecasting and how the analysis was conducted. Conclusions and future work are presented in Section 5, and references cited in the report are listed in Section 6.

2.0 Prior Machine Learning Work in the Building Domain

Machine learning has proved to be effective in several areas, such as speech recognition (Hinton et al. 2012; Sainath et al. 2013; Chan et al. 2015), natural language processing (Collobert et al. 2011), and image classification (Krizhevsky et al. 2012; Simonyan and Zisserman 2014; He et al. 2016; Tompson et al. 2014). Recently, deep learning has emerged as one of the most popular and powerful concepts within the ML field (LeCun et al. 2015). The building domain is not untouched by the current wave of ML, especially deep learning. The availability of large data streams (a.k.a. big data) from today's building automation systems, significant improvements in high-performance computation infrastructure and its affordability, advanced algorithms, and the easy access to them provided by software libraries have contributed to the popularity of ML. Machine learning has been applied to address a variety of different problems such as modeling energy demand, modeling of zone temperatures, estimation of thermal properties of a building, and model-free optimal control using reinforcement learning. In this section, we provide a summary review of the state of the art in these applications of ML to buildings, with a focus on modeling energy demand.

2.1 Machine Learning for Forecasting Energy Demand

Machine learning techniques have been applied in the context of building energy modeling for a long time. Previous ML approaches can be grouped into two categories—parametric and nonparametric. Parametric techniques include regression models (Braun et al. 2014; Priyadarsini et al. 2009; Dong et al. 2005a), time-series models (Espinoza et al. 2005), and grey-box models (Zhou et al. 2008). Nonparametric methods include support vector machines (Dong et al. 2005b) and falling rule lists (Marasco et al. 2016), artificial neural networks (Huang et al. 2015; Kalogirou et al. 1997; Mocanu et al. 2014; Kumar at. al. 2013), and regression trees (Fan et al. 2014). The models presented in these papers help us understand energy consumption as a function of a reduced number of variables, usually related to weather or building materials. Although several energy-forecasting methods have been developed, most of them use shallow energy-forecasting models and are unsatisfactory for many practical applications. It is difficult to say that one method is clearly superior over other methods in any situation. One reason for this is that the proposed models are developed with a small amount of data from one building. Also, the accuracy of the energy-forecasting methods is significantly influenced by weather and occupancy behavior, which are stochastic in nature and hence difficult to predict. These limitations inspire us to think about ML methodologies based on deep architectures supplied with rich amounts of building data.

There have been a few recent developments in the application of deep learning for energy prediction, with promising results showing good predictability and robustness. Google recently implemented a neural network with five hidden layers and 50 nodes per hidden layer to increase power user efficiency in their data centers (Gao et al. 2014). Their approach used 19 features, consisting of parameters such as number of chillers running and outside air enthalpy, and control inputs such as cooling tower water temperature set point. Recurrent neural networks with very few features have also been tested to predict heating gas consumption (Macas et al. 2014). Conditional Restricted Boltzmann Machines and Factored Conditional Restricted Boltzmann Machines have been applied to forecast energy in buildings (Mnih et al. 2012; Taylor et al. 2011). Deep RNNs have also been recently designed to tackle energy disaggregation problems, wherein individual consumption of power appliances is estimated from a single measure (Kelly et al. 2015). Data needed for these kinds of studies can now be obtained from sources that have been made publicly available in the last few years (Kolter et al. 2011;

Monacchi et al. 2014; Gisler et al. 2013). Currently, most deep neural networks for energy forecasting are supervised; thus, the neural network is trained against a true value of the output (Chae et al. 2016; Deb et al. 2016). The selection of the input features is usually the most important step in developing a reliable energy forecasting base.

One of the primary challenges with applying deep learning for energy forecasting is determining the architecture of the network. Although deep architectures can learn more powerful models than shallow networks, simply adding layers in neural networks does not provide significant improvement for energy forecasting. Several researchers (Amarasinghe et al. 2017; Shi et al. 2017) indicate that increasing the depth of the network by increasing the number of layers and units only reduces training dataset error. However, increasing the depth of the network can only improve performance on testing data up to a limited number of layers, which reflects the occurrence of overfitting due to the lack of data diversity and network parameter redundancy. Consequently, most deep neural networks used for load forecasting usually have only one or two hidden layers (Marino et al. 2016; Amarasinghe et al. 2017; Fan et al. 2017). This indicates deep learning has limited advantages when compared to neural networks with a shallow architecture for the load-forecasting problem.

To develop robust and reliable estimations of the coefficients in deep-learning models, a substantial amount of data is needed. Previous studies have usually focused on using up to one year of data with a data acquisition frequency of 30 or 60 minutes. This data volume may not be large enough to guarantee the reliability of a deeper architecture. It is possible that a few hidden layers are good enough only when input features are limited.

2.2 Machine Learning for Forecasting Building Zone Temperatures

Zone temperature modeling using ML techniques, especially deep learning, for both single and multiple time steps has been investigated in the literature since the early 2000s. Some of the earlier works in this domain looked at the problem of estimating physical parameters that describe the thermal characteristics of a building, using neural networks. Olofsson and Andresson (2002) estimated steady state heat transfer and heat gain coefficients for a building using a neural network trained using data for that building. This work was later extended to estimate the thermal capacitance parameter that is associated with the transient thermal dynamics (Lundin et al. 2004). The estimation technique involved indirect techniques such as perturbing inputs and analyzing perturbations in outputs for the trained models. For the specific building considered, an architecture with three hidden layers and 60 hidden units was used.

Mechaqrane and Zouak (2004) implemented a three-layer neural network architecture to predict the indoor temperature of a small apartment building for three timesteps into the future. Approximate data for ten weeks were used for training the model. A fully connected architecture was implemented, which was then pruned using the Optimal Brain Surgeon strategy—for computational and accuracy benefits—to result in a sparse architecture with ten units in the hidden layer, and only 30 connections between the units in the input, hidden, and output layers. A somewhat similar approach was adopted by Mustafaraj et al. (2011) for predicting indoor temperature and humidity for an open office zone for short- to medium-term predictions (30 minutes to three hours). Here again, the authors started with a fully connected three-layer network and pruned it using the Optimal Brain Surgeon strategy to four units in the hidden layer. However, a key difference from Mechaqrane and Zouak (2004) was the use of carbon dioxide concentration as an indicator of occupancy as one of the model inputs, which provided better accuracy.

Huang et al. (2012, 2013, 2015) described their use of a single hidden layer neural network to model a four-zone commercial building in Australia for long-term (up to six days) predictions of indoor temperature. A key aspect considered by the model was inter-zone coupling, meaning temperatures of neighboring zones were used as inputs in the model for each zone. The size of the hidden layer (number of hidden units) was optimized using an exhaustive evaluation of accuracy over a finite set of choices, resulting in a 14-unit architecture. A similar analysis procedure was used to optimize the choice of inputs. Indoor temperature prediction for a multi-zone building using feed-forward neural networks also was investigated by Mateo et al. (2013) in which they looked at one time step ahead predictions. They invoked clustering methodologies, where data points were re-arranged according to their similarities, and independent models were used for each cluster. However, each of these models had the same architecture consisting of a single hidden layer with 20 nodes.

A recent work by Rubio-Herreo et al. (2017) used an RNN with the LSTM architecture to model the zone temperatures and thermal demands for a 24-zone building in Eastern Washington. The model used data from 224 sensors for a time window of more than eight months. One of the key benefits of deep learning over other ML methods is that careful selection of inputs (features) is not required. Therefore, the proposed modeling framework in Rubio-Herreo et al. (2017) did not involve rigorous feature selection, and data from all sensors was used for the modeling. Through an exhaustive evaluation, similar to Mateo et al. (2013), it was observed that a single hidden layer RNN with 128 units provided the highest accuracy among all architectures evaluated.

2.3 Machine Learning for Estimating Building Properties

Researchers have investigated how to characterize the thermal characteristics of a building and include that information in cost-minimization control problems so that the operation of heating, ventilation, and air conditioning (HVAC) systems and other electric loads within the building is optimized, while the environmental conditions are kept within a reasonable comfort range (Ma et al. 2012; Radecki et al. 2013; Taylor et al. 2015; Alizadeh et al. 2015). The state evolution of such loads, especially thermostatically-controlled loads, have been represented using models, given that the loads are usually not observed by agents such as demand aggregators, and therefore their state must be inferred via Kalman filters or Markov Models (Mathieu et al. 2013; Taylor et al. 2014). This approach is particularly convenient for demand response applications in which these agents must decide which loads to deploy at any time instance. Neural networks have also been studied for over two decades within this context. Kredier et al. (1995) used RNNs to estimate the equivalent thermal capacitance and resistance of a building using hourly energy and temperature data. Olofsson et al. (2002) and Lundin et al. (2004) trained neural networks to estimate the thermal properties of a building using just indoor/outdoor temperature difference and delivered heating energy.

A few researchers have developed unsupervised methods for identifying the relationships between sensor data streams in buildings. For example, Trindler et al. (2003) used neural networks and clustering techniques to identify the temporal relationships between related sensors. They then organized them into a hierarchical tree that represented the network's functional structure. Fontugne et al. (2012) used a different method called empirical mode decomposition to identify related sensors based on usage patterns. They found that the method was able to correctly identify their physical locations in relation to each other. The techniques in Trindler et al. (2003) and Fontugne et al. (2012) establish a foundation for automatically identifying how sensors may be functionally and physically related, which has applications to the automated configuration of control applications. Additionally, they may provide a mechanism for

verifying sensor location, which could be used to identify building automation system configuration errors and sensor faults. These methods may also be used to help uncover anomalies or inefficient operation; for example, Fontugne et al. (2013) builds upon the authors' previous work and provides an example of using empirical mode decomposition to detect changes in building energy patterns. Furthermore, these methods may allow deeper insights that could potentially improve building control in non-obvious ways. For example, the conditioning of adjacent zones may be better balanced by a control system that is able to learn their thermal relationship and adjust damper positions to provide equivalent conditioning at reduced airflows or duct static pressure, thus saving energy.

2.4 Machine Learning for Model-Free Control (Reinforcement Learning)

Some researchers have recently started developing reinforcement learning for building control (Ruelens et al. 2015; Li et al. 2017). Reinforcement learning methods, such as Q learning, can be used to generate optimal control policies in the absence of a complete building dynamic model. The learning agent's role is to define action and state, update its understanding of the thermal environment, and make better control decisions in accordance with its defined objectives of optimizing both thermal comfort and energy costs. In this context, Reinforcement learning can be used for passive thermal storage (building mass) control, which shifts the heating, ventilation, and air conditioning (HVAC) energy demand while still meeting the requirements for zone temperature (Liu et al. 2016). It also has been shown that deep reinforcement control algorithms can be used to determine the optimal airflow set point for each zone to maintain zone temperature within a desired range (Barrett et al. 2015; Wei et al. 2017).

3.0 Short-Term Load Forecasting with Sequence-to-Sequence RNNs

In our approach to whole building short-term electrical load forecasting, we make use of deep neural networks due to their powerful non-linear modeling capabilities and minimal need for manual feature engineering. For time-series modeling, two popular deep-learning methods are commonly used: RNNs and convolutional neural networks (CNNs). For this work, only RNNs with sequence-to-sequence configuration were used. In follow-up work, use of convolutional neural networks will be explored. In this section, RNNs and sequence-to-sequence (seq2seq) models are described.

3.1 Recurrent Neural Networks

An RNN is a family of standard feed-forward neural networks adapted to use sequential information. Recurrent neural networks have been applied successfully to various sequential data such as machine translation (Jean et al. 2014; Sutskever et al. 2014; Bahdanau et al. 2014), image captioning (Karpathy et al. 2015), sentiment classification (Socher et al. 2013; Tang et al. 2015), video classification (Donahue et al. 2015), and time-series prediction (Che et al. 2018). In RNNs, a sequential model can account for a probability distribution of the next input (or output) in a sequence given the current input and context from previous time steps, while all inputs (or outputs) are assumed as independent of each other in a traditional feed-forward neural network. The RNN architecture enables models to learn how to transform the input sequences into output sequences by remembering and processing past information. To track arbitrary long-term dependencies in the input sequences, RNNs introduce the hidden state (or context) computed in the previous time step. This is additionally considered to predict the output of the current time step as well as the current input.

Given a sequence of inputs $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T)$, where T is a length of input time steps and each data point $\boldsymbol{x}_t$ is a real-valued vector, a standard RNN computes a sequence of target outputs $(\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T)$, where a target output $\boldsymbol{y}_t$ at each time step is a vector of the length L, and hidden units by iterating the following equations:

$$\boldsymbol{h}_t = f(W_{hx}\boldsymbol{x}_t + W_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h) \quad (1)$$

$$\boldsymbol{y}_t = \mathrm{g}\left(W_{yh}\boldsymbol{h}_t + \boldsymbol{b}_y\right) \textbf{Error! Bookmark not defined.} \quad (2)$$

where f and g are activation functions. In RNNs, the hidden activation function f is typically a saturating function such as the logistic sigmoid or hyperbolic tangent, while the output activation function g is task-specific: the identity function is typically used for regression tasks, while the logistic sigmoid and softmax functions are used for binary and multi-class classification tasks, respectively. W_{hx} and W_{hh} indicate the matrices of feed-forward weights between the input and the hidden layers and of recurrent weights between the hidden layers in adjacent time steps, respectively. Unlike the conventional neural network, Eq. (1) shows that the hidden layers enable tracking the information in previous time steps; in this way, hidden units play a role as the memory of the network by capturing information about what happened in the previous time steps. The output $\hat{y}_t$ at time step t is calculated based on the hidden states at time t. Note that RNNs reuse the same weight matrix at every time step. This parameter sharing makes it possible to extend and apply the model to examples of different lengths and generalize across them.

3.1.1 Extensions to Recurrent Neural Networks

Despite many promising features introduced in standard RNNs, the difficulty of effectively training models due to the instability of the gradients (the so-called vanishing/exploding gradient problem) had hampered its early success (Bengio et al.1994). As the layers are made deeper to model long-term dependencies, training a model gets more difficult.

Given the limitations of a standard RNN, the LSTM architecture (Hochreiter et al. 1997) and the gated recurrent unit (GRU) (Cho et al. 2014) were introduced to overcome the vanishing gradient problems in long-term sequence modeling.

The LSTM architecture introduces a memory cell state c_t (capturing the dependencies between time steps) and three gates such as input i_t, output o_t, and forget f_t gates as shown in Eq (4).

$$
\begin{aligned}
f_t &= \sigma\left(W_{h_f x} x_t + W_{h_f h} h_{t-1} + b_{h_f}\right) \\
i_t &= \sigma\left(W_{h_i x} x_t + W_{h_i h} h_{t-1} + b_{h_i}\right) \\
o_t &= \sigma\left(W_{h_o x} x_t + W_{h_o h} h_{t-1} + b_{h_o}\right) \\
g_t &= \tanh\left(W_{h_g x} x_t + W_{h_g h} h_{t-1} + b_{h_g}\right) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot tanh\,(c_t)
\end{aligned}
\tag{4}
$$

where ⊙ indicates the Hadamard (i.e., element-wise) product between two vectors. A memory cell state c_t facilitates gradients in the back propagation to flow through cell states farther without interruptions.

As a variation on the LSTM, GRU has been introduced (Cho et al. 2014). All the gates in LSTM are compactly represented into a reset gate and an update gate to capture the short-term and longer-term dependencies, respectively.

$$
\begin{aligned}
r_t &= \sigma\left(W_{h_r x} x_t + U_r h_{t-1} + b_{h_r}\right) \\
z_t &= \sigma\left(W_{h_z x} x_t + U_z h_{t-1} + b_{h_z}\right) \\
h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot \hbar_t \\
\hbar_t &= \tanh\left(W_{\hbar x} x_t + U(r_t \odot h_{t-1})\right)
\end{aligned}
\tag{5}
$$

3.1.2 Sequence-to-Sequence Modeling

The sequence-to-sequence model was proposed to more effectively deal with variable-length input and output sequences (Sutskever et al. 2014). Modeling sequences with neural networks can be done effectively by adding recurrent connections to the hidden layers to propagate intermediate representations of an input sequence across time (Figure 3.1). This framework consists of an encoding model (encoder) and a decoding model (decoder) as shown in Figure 3.1.

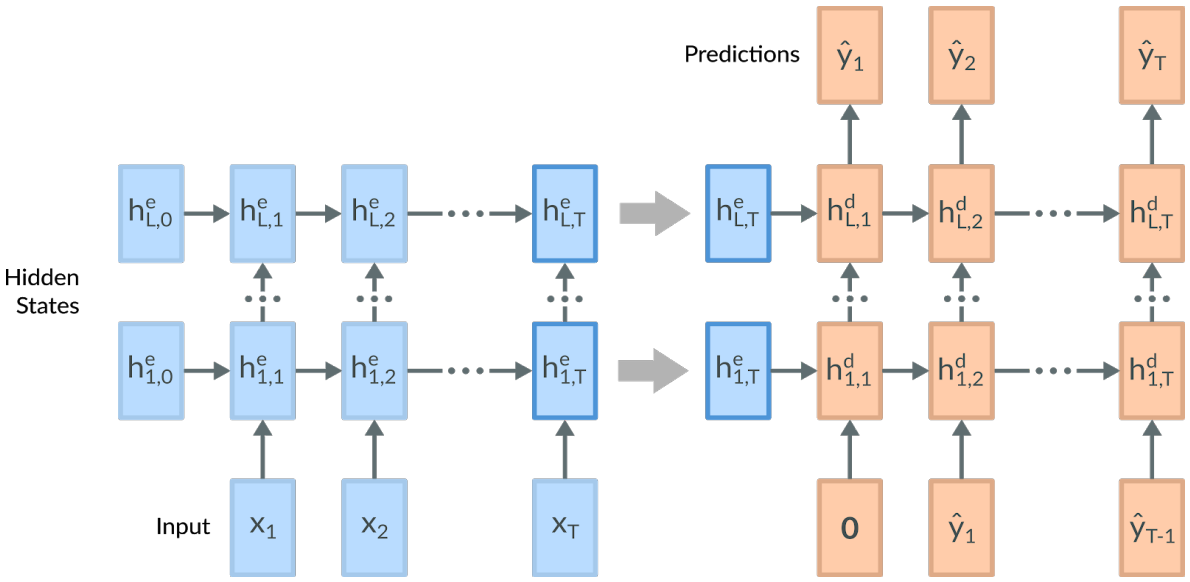


Figure 3.1. Encoder and decoder in the sequence-to-sequence model.

An encoder learns to map a variable-length sequence into a fixed-length vector representation, and a decoder in the same manner learns to map a given fixed-length vector representation back into a variable-length sequence. In other words, variable-length input sequences can be mapped to variable-length output sequences by encoding the input as a fixed-length vector using an RNN, then using this representation to seed another RNN to generate an output sequence. This framework is known as seq2seq. It has demonstrated impressive performance on a variety of sequence modeling tasks such as machine translation.

Given an input sequence $\boldsymbol{x}_{\mathbf{1}:\mathrm{T}_{enc}} \in \mathbb{R}^{D\times \mathrm{T}_{enc}}$, we define the encoder state for each input as $h^e_{l,t} = f_{enc}\left(x_t, h^e_{l,t-1}\right)$, then define the output at time t from sequence $\hat{\boldsymbol{y}}_{\mathbf{1}:T_{dec}} \in \mathbb{R}^{C\times T_{dec}}$ as $\hat{y}_t, h^d_{l,t} = f_{dec}\left(\hat{y}_{t-1}, h^d_{l,t-1}\right)$, where D and C indicate the dimension sizes of input and output data, respectively. Also, T_{enc} and T_{dec} indicate the sequence lengths of input and output data, respectively.

In this work, we chose f_{enc} and f_{dec} to be either LSTMs or GRUs, owing to their ability to handle long-term dependencies. We employ these seq2seq models to predict the power consumption in various buildings.

To learn the parameters of our seq2seq model, we use stochastic gradient descent methods to minimize the mean squared error of the model's predictions compared to ground-truth measurements of building electricity use. Expressed mathematically, we aim to minimize the

average loss over the training set: $\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}(y_{i,t}-\hat{y}_{i,t})^2$, where Y and $\hat{Y}$ are the ground truth and predicted building electricity use, respectively. To do so, we use the squared error loss function, $L(y_{i,t},\hat{y}_{i,t}) = (y_{i,t}-\hat{y}_{i,t})^2$.

3.2 Experimental Setup

This section describes our data preparation, experimental configurations, and the training procedure.

3.2.1 Data Preparation

Data from four office buildings from Eastern Washington, labeled in this report as Buildings A, B, C, and D, are used for the analysis. Buildings A and C are less than five years old and use VAV air handling units (AHUs) to condition building zones. The chilled water for these buildings is provided from a central plant so the whole building electricity consumption excludes chiller consumption. Buildings B and D are over 30 years old and use packaged rooftop units so the whole building energy consumption includes electricity consumption by the rooftop units. Figure 3.2 shows the outdoor air temperature (top) and whole building electricity consumption (bottom) from July 2017 through June 2018 for all four buildings. All four buildings are typically occupied from 8 a.m. to 5 p.m. but the HVAC systems are run from 6 a.m. to 6 p.m.

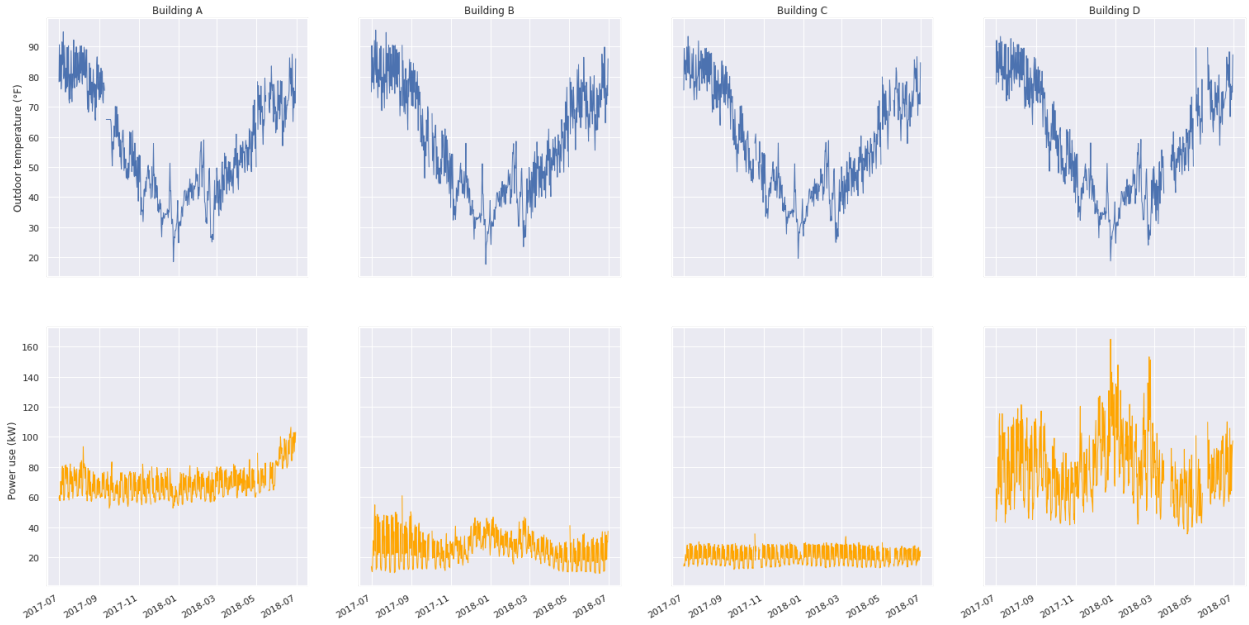


Figure 3.2. Outdoor temperature and power consumption in each building during a year.

For each building, we first partition the data into three disjoint datasets: *training*, to estimate model parameters; *validation*, to perform model selection and identify optimal hyperparameters; and *testing*, to evaluate model generalization. To avoid seasonal biases between sets and ensure fair estimates of model generalization, we split the building data by first partitioning each month into four weeks, then randomly selecting two weeks for the training set, one week for the validation set, and one week for the testing set. Any leftover days in a month are added to the training set. Moreover, we perform these splits over the same date range for each building (i.e., July 1, 2017, to June 30, 2018) to make comparisons between buildings fair. Both outdoor temperature and electricity consumption are standardized according to statistics from the full 12-

month training set to improve training stability and to make results across variations in dataset timing more comparable. All experiments were evaluated using the full validation and testing sets.

3.2.2 Choice of Hyperparameters

Nearly all ML models have hyperparameters; that is, parameters or design decisions that cannot be learned from the training set. These parameters can relate to the model structure and architecture or to how it is optimized and regularized. The large number of hyperparameters used to optimize our model are described below:

- Architecture hyperparameters
  - Number of stacked hidden recurrent layers in the encoder and decoder.
    - Range: 1-4.
  - Number of hidden units in each hidden layer.
    - Range: 32 to 512, incrementing by powers of two.
  - Whether the encoder is unidirectional or bidirectional. In addition to the standard RNN with recurrent connections going forward in time, a bidirectional RNN also adds reverse recurrent connections going backward in time. Both the forward and backward models are used to encode the input.
    - Range: {unidirectional, bidirectional}.
  - Recurrent cell type used in the encoder and decoder.
    - Range: {LSTM, GRU}.
  - Whether to include residual connections between layers. These connections provide additive connectivity between layers and allow for improved training stability by mitigating vanishing gradients between recurrent layers during backpropagation.
    - Range: {residual connection, no residual connections}.
  - Whether to make residual predictions; that is, at each time step, the model predicts the expected difference between the current and previous time step.
    - Range: {residual predictions, normal predictions}.
- Optimization hyperparameters
  - Minibatch size. The number of data points to randomly sample for each gradient computation during training.
    - Range: 32 to 1024, incrementing by powers of two.
  - Learning rate (i.e., step size) for each gradient update during training.
    - Range: 0.0001 to 0.01.
  - Optimization algorithm to use during training.
    - Range: {stochastic gradient descent, Adam (Kingma and Ba 2014), RMSprop (Tieleman and Hinton 2012)}.
- Regularization hyperparameters

- Dropout probability. During training, each hidden unit is zeroed out randomly with the dropout probability. This encourages the model to distribute its representation of the input across all hidden units.
    - Range: 0.0 to 0.9.
- L2 regularization coefficient. The L2 norm of model weights (but not biases) is penalized during training, weighted by this coefficient. Increasing the coefficient reduces the ability of the model to over-fit the training set (Bishop 2016).
    - Range: 0.0 to 5.0.

3.2.2.1 Random Search

The first two experimental comparisons described below aim to systematically compare experimental factors such as the training duration (in months) the model undergoes and the number of hours upon which each prediction is conditioned. To avoid conflating the effects of hyperparameter search with these experimental factors, we established a fixed set of 50 randomly selected hyperparameter configurations. We refer to this set of configurations as $\mathcal{H}$. For simplicity, in each of the configurations in $\mathcal{H}$, the optimizer was fixed to Adam, dropout probability was 0.5, L2 regularization factor was 0.00001, the encoder was unidirectional, the recurrent cell type was GRU, and neither residual connections between layers nor between model predictions was used.

3.2.2.2 Directed Search

Our third experimental comparison explores the effect of the hyperparameters themselves. We take a directed search approach in which the successive hyperparameter configurations are strategically chosen given the performance of the hyperparameter configurations tried thus far. We made use of the hyperparameter tuning tool Chocolate[1] to perform this search. Chocolate provides a directed search interface that allows us to specify a search space of hyperparameters and their ranges from which an arbitrary optimizer will sample and incrementally build a model of the hyperparameter space. As the model develops stronger priors over its search space, it can begin to converge toward regions of the space where corresponding hyperparameters minimize the validation loss of the model. In our experiments, we use the covariance matrix adaptation evolution strategy (CMA-ES) algorithm (Hansen 2016) to perform our search because we found that it progresses considerably faster than its Bayesian search strategy.

3.2.3 Model Inputs and Output

The ML model used only three measured inputs or independent variables: outdoor dry-bulb temperature (°F), whole building electricity consumption (kW) for the previous hour(s), hour of day, and day of week. Temporal features are represented using sine and cosine encodings (to show continuity; for example, hour 0, follows hour 23). The output variable (dependent variable) is the whole building electricity consumption (kW).

As mentioned in Section 3.1, the output of our model for a single time step t can be expressed as follows:

[1] https://chocolate.readthedocs.io

$$h^e_{l,t} = f_{enc}(x_t, h^e_{l,t-1})$$

$$\hat{y}_t, h^d_{l,t} = f_{dec}(\hat{y}_{t-1}, h^d_{l,t-1})$$

where x_t is a vector comprised of the features described previously, $\hat{y}_t$ is the predicted building electricity consumption, and f_{enc} and f_{dec} are each LSTMs or GRUs (see Section 3.1). To predict multiple time steps of electricity consumption, we repeatedly apply f_{dec} to each successive prediction and decoder hidden state, leading to the following simplified notation:

$$\hat{y}_{1:T_{dec}} = f_{enc-dec}(x_{1:T_{enc}})$$

Here $\hat{y}_{1:T_{dec}}$ denotes the output sequence of length T_{dec} time steps and $x_{1:T_{enc}}$ denotes the input sequence of length T_{enc} time steps. Later experiments present the effect on performance when T_{enc} and T_{dec} are varied.

3.2.4 Experimental Comparisons

In this section, the distinct experimental comparisons are described.

3.2.4.1 Effect of Training Set Length, Date Range, and Time Resolution

To determine empirically the effect of varying the overall duration of the training set, the month in which the training set begins, and the time resolution of the model's input and predictions, we conducted a grid search. Specifically, we varied training set duration from one month to 12 months, start months from July to June in 2-month intervals, and time resolutions at 1-hour and 15-minutes. For a 12-month duration, the start month was fixed to July and the random seed incremented over six separate experiments to ensure that we had enough samples for these durations. After conducting preliminary experiments using a 1-minute time resolution, we observed that such results often were noisy, that the models tended to diverge and fail to train, and that the longer sequences incurred additional computation time that proved to be prohibitive. Therefore, we have deferred development of methods that work for fine temporal resolutions (<15-minutes) to the future. For each training set length, starting month, and resolution scenario, we train a model with each hyperparameter configuration in $\mathcal{H}$, and report the results for the configuration in $\mathcal{H}$ that gives the best performance on the validation set.

3.2.4.2 Effect of Context and Forecast Lengths

In these experiments, we assess the effect of longer context lengths (i.e., the number of timesteps predictions are conditioned on) and forecast lengths (i.e., the number of timesteps we aim for forecasting ahead). All these experiments were performed using fixed training set length (six months), training set start month (July), and temporal resolution (one hour). Because varying the context and forecast lengths changes the effective number of data points in the training set and therefore the number of weight updates per epoch (pass through the training set), we conduct these experiments using full-batch gradient descent to train each model. Input durations tested were one hour, three hours, six hours, 12 hours, and 24 hours. Output durations ranged from one hour to six hours in 1-hour increments. For each context and forecast length combination, we train a model with each hyperparameter configuration in $\mathcal{H}$ and

report the results for the configuration in $\mathcal{H}$ that gives the best performance on the validation set.

3.2.4.3 Effect of Model Hyperparameters

Determining an optimal hyperparameter configuration for our models is an important step toward ensuring that our models can achieve the best performance and generalization. Further, it also is important that we find a hyperparameter configuration that yields good performance across a variety of buildings to mitigate the need for additional tuning when training models using different buildings. Here we use Chocolate as described in Section 3.2.2.2 to optimize all our model hyperparameters. To isolate the effect of model hyperparameters for these experiments, the training set length was fixed to six months starting in July at a 1-hour time resolution, and we only predict one hour of building load conditioned on the previous six hours of outdoor temperature and building load measurements along with the usual temporal encoding features.

3.2.4.4 Transferability of Models Across Buildings

Finally, we explore the issue of model transferability; that is, how well a model trained on one building works for another. Generalization to unobserved buildings is another important axis of analysis for our models. For scenarios in which sensor data for a building are limited, it may be useful to employ a model trained on a similar building to generate forecasts. It also may be advantageous to use these pre-trained models to warm-start training for an unobserved building. To test how our model generalizes between buildings, we take the best generic hyperparameter configuration obtained from the directed hyperparameter search, use it to train a model for each building, and then evaluate each buildings model with data from other buildings. We also train a model on all buildings and evaluate its performance on each individual building.

3.2.5 Training

All models were implemented using PyTorch.[1] Recurrent models were trained using scheduled sampling (Bengio et al. 2015) during training at a fixed probability of 0.5 to ensure that the model would use its own outputs as targets approximately half the time; this mitigates any confusion the model may encounter when generating sequences outside the distribution of the training set. Finally, gradient clipping was applied to ensure a maximum gradient norm of 5.0. Models were run for a maximum of 100 epochs for the data and sequence timing experiments and a maximum of 200 epochs for the directed search experiments and stopped if performance on the validation set did not improve after five consecutive epochs (i.e., we employed early stopping).

Training the final models and optimizing the best models for each study requires significant computational resources and time. Model training was performed using a mix of CPU and GPU workstations. The GPU workstations were reserved for the most computationally intensive experiments. These workstations were configured as follows: Intel Xeon ES-1620v3 (3.5 GHz) CPU, 64 GB RAM, and either Nvidia GTX 980Ti with 6 GB VRAM or Titan Xp with 12 GB VRAM. Using this hardware, training for the data timing experiments averaged approximately 3.5 minutes for trials at 1-hour resolution and 12 minutes for trials at 15-minute resolution.

[1] https://pytorch.org/ is an open source machine learning software library for the programming language Python, based on the Facebook Torch library.

3.3 Results and Analysis

In this section, we summarize the results of the four experimental comparisons and provide one analysis of model behavior.

3.3.1 Effect of Training Set Length, Date Range, and Time Resolution

In Eastern Washington and in many parts of the United States, buildings experience four distinct seasons: winter, spring, summer and fall. The energy consumption of a building can vary significantly between summer (cooling) and winter (heating) seasons. Consumption during spring and fall is generally similar but it is significantly different from either summer or winter. Furthermore, the building energy consumption is primarily a function of outdoor air temperature, time of day, and day of week, commonly accessible regardless of building types. Therefore, the training dataset must include summer, winter, and at least one of the shoulder seasons (spring or fall). Figure 3.3 shows the seasonal variation of the outdoor air temperature for Eastern Washington.

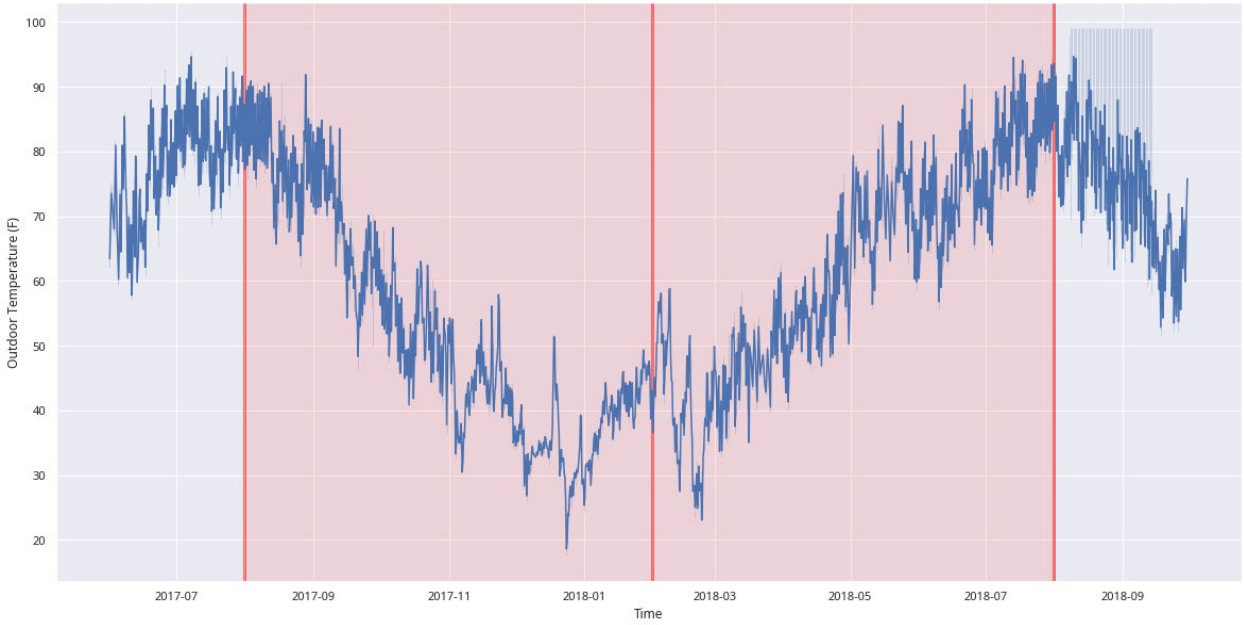


Figure 3.3. Mean outdoor temperature in degrees Fahrenheit for all buildings from June 2017 to October 2018. The highlighted region covers one year of measurements starting on August 1, 2017. Lines denote best observed date ranges for training data.

Figure 3.4 shows the effect of the number of months of training data (y-axis) and start month (x-axis) for each of the four buildings (columns) for two-time resolutions of one hour (top row) and 15 minutes (bottom row), respectively. The values shown in this figure are the root mean square error (RMSE) and color represents the magnitude of the RMSE: high (yellow) to low (dark blue). A start month of five implies that the training data began in May, while a start month of 11 implies that the training data began in November. A start month of one and number of months of 11 implies the training data started in January and had 11 months of data.

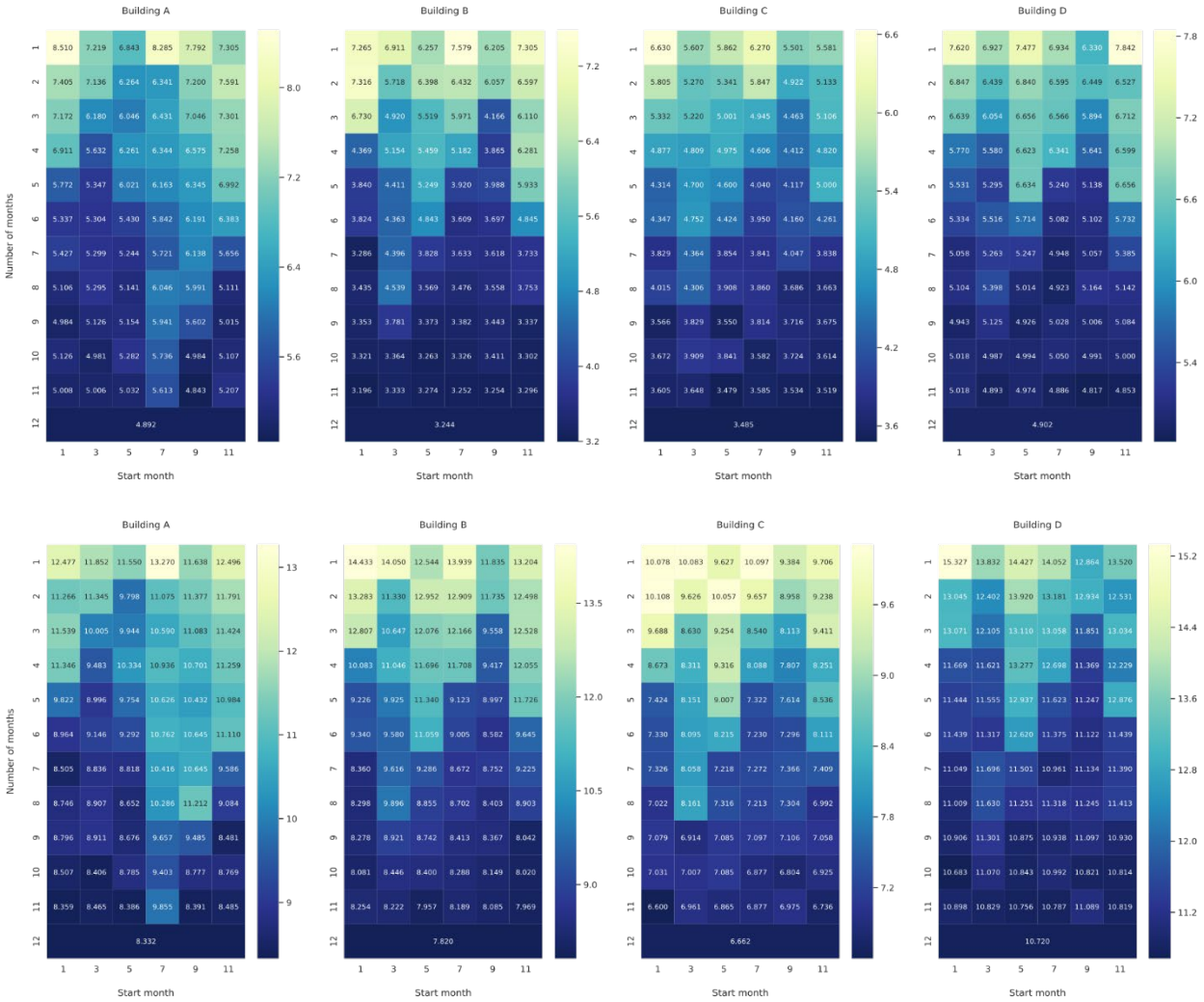


Figure 3.4. Normalized test RMSE as training set start month and duration vary. Top row shows results for data with 1-hour time resolution and the bottom row shows results for 15-minute time resolution. Start month was not varied for year-long experiments.

We observe that in the average case, one can reasonably expect good model performance with at least six months of training data starting in the middle of a heating or cooling season, somewhere within a month of February or August. This is conceivably because within half a year of these months, close to the entire range of outdoor air temperatures is captured. When the training data acquisition is started in spring or fall, at least nine months of training data typically is needed to get predictive performance. As would be expected, closer to a year or more of training data is preferable, but in scenarios in which only a limited amount of data has been collected, six months of training data that captures part of both a heating and cooling season appears to suffice for good predictive performance.

Despite our finding that it is possible for these models to obtain good forecasting performance with limited data, our best performing models used at least 11 months of training data. Table 3.1 and Table 3.2 outline the performance of our best models and the hyperparameters used to train them, respectively. We observed that in the general case, one can use shallow architectures with relatively low parameter counts and still achieve good results.

Table 3.1. Normalized validation and test normalized and RMSE for two-time resolutions. Values reported are nRMSE (%) and RMSE (kW, parenthesized).

	1 Hour		15 Minutes	
	Validation	**Test**	**Validation**	**Test**
A	5.1544 (3.9818)	4.8426 (3.7409)	8.7644 (8.9881)	8.3912 (8.6054)
B	3.1062 (2.4661)	3.2440 (2.5755)	7.8568 (7.3717)	7.8205 (7.3376)
C	3.7844 (1.4278)	3.4787 (1.3125)	7.1964 (2.9533)	6.6621 (2.7340)
D	4.9420 (10.7541)	4.9024 (10.6677)	11.5999 (28.9819)	10.7199 (26.7834)

Table 3.2. Best hyperparameter of each building and timing configurations for 1-hour and 15-minute time resolutions, as well as the complexity of each model.

1-Hour Resolution

	Start	**Span**	**Batch Size**	**Layers**	**Hidden Dim.**	**Learning Rate**	**No. Params**
A	Sept	11	1024	1	256	0.010245	401,921
B	July	12	1024	1	256	0.010245	401,921
C	May	11	64	1	64	0.026042	26,753
D	July	12	128	4	128	0.001786	697,089

15-Minute Resolution

	Start	**Span**	**Batch Size**	**Layers**	**Hidden Dim.**	**Learning Rate**	**No. Params**
A	Sept	11	1024	1	256	0.010245	401,921
B	July	12	1024	1	256	0.010245	401,921
C	July	12	1024	3	256	0.007301	1,980,929
D	July	12	1024	1	256	0.010245	401,921

3.3.2 Effect of Context and Forecast Lengths

Figure 3.5 shows the normalized RMSE values as a function of input and output sequence length, T_{enc} and T_{dec}, respectively. An interesting result emerges from our sequence timing experiments: in most cases, increasing the input sequence length does not yield considerable performance improvements, although one hour is rarely optimal. One possible explanation for this is the information bottleneck of the input sequence's fixed-length encoding; as the input sequence length increases, more information has to be encoded in the fixed-length representation, which may result in losing important information.[1] We also observe that the

[1] https://arxiv.org/pdf/1409.1259.pdf

optimal context length is different for different buildings: Building B favors very long context, while Building D favors very little context.

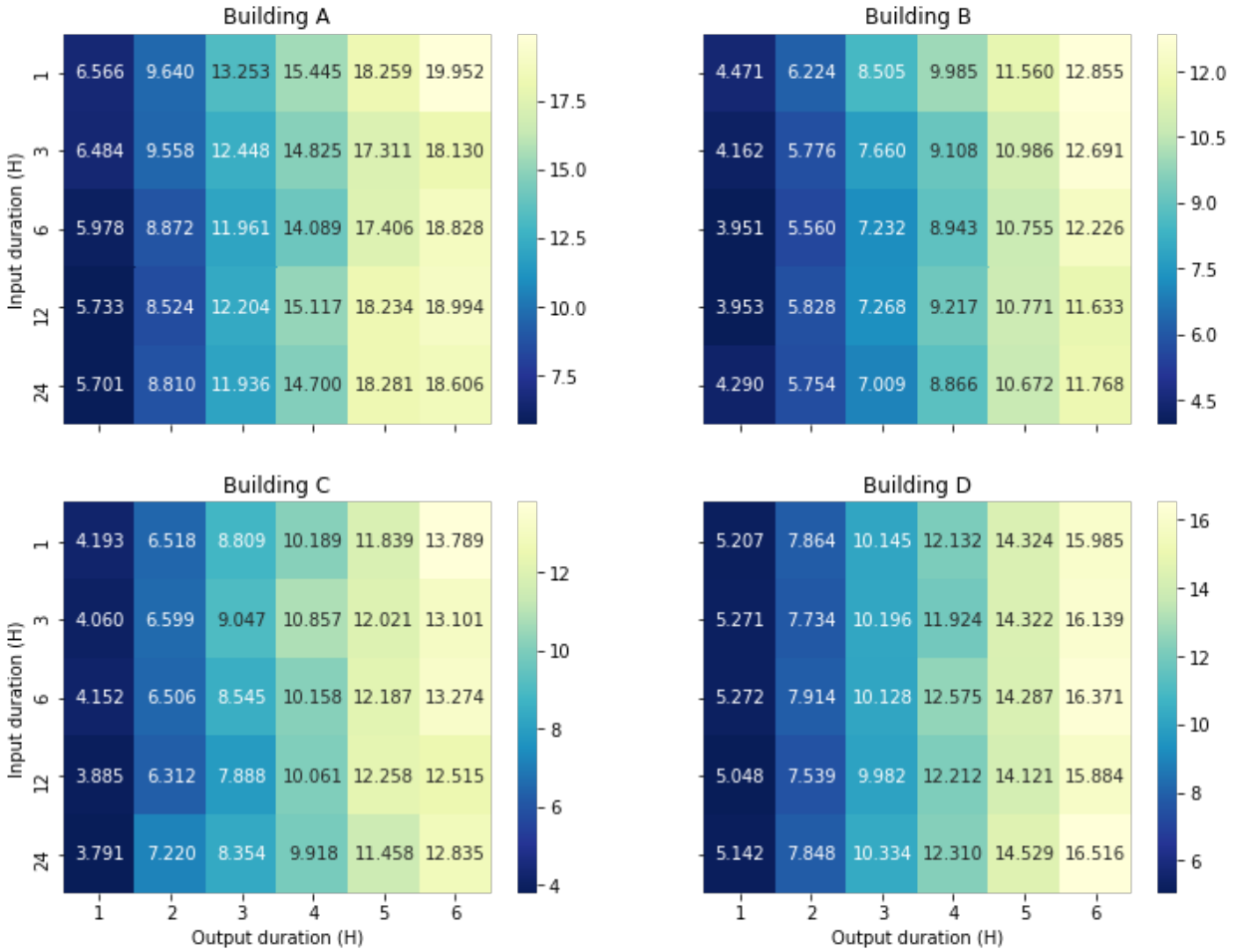


Figure 3.5. Normalized test RMSE as input and output sequence length vary for each building. Models for each building were obtained using the minimum validation set error over 50 random hyperparameter configurations. All models are trained with an equal number of weight updates.

3.3.3 Effect of Model Hyperparameters

Our random hyperparameter search trials indicated that on average, our models performed best with fewer recurrent layers, narrower hidden representations, large minibatch sizes, and small learning rates. However, the results overall were very noisy as indicated in Figure 3.6, and the models which achieved minimal validation set loss tended toward wider hidden representations, a result consistent with the model hyperparameters we list in Section 3.3.1.

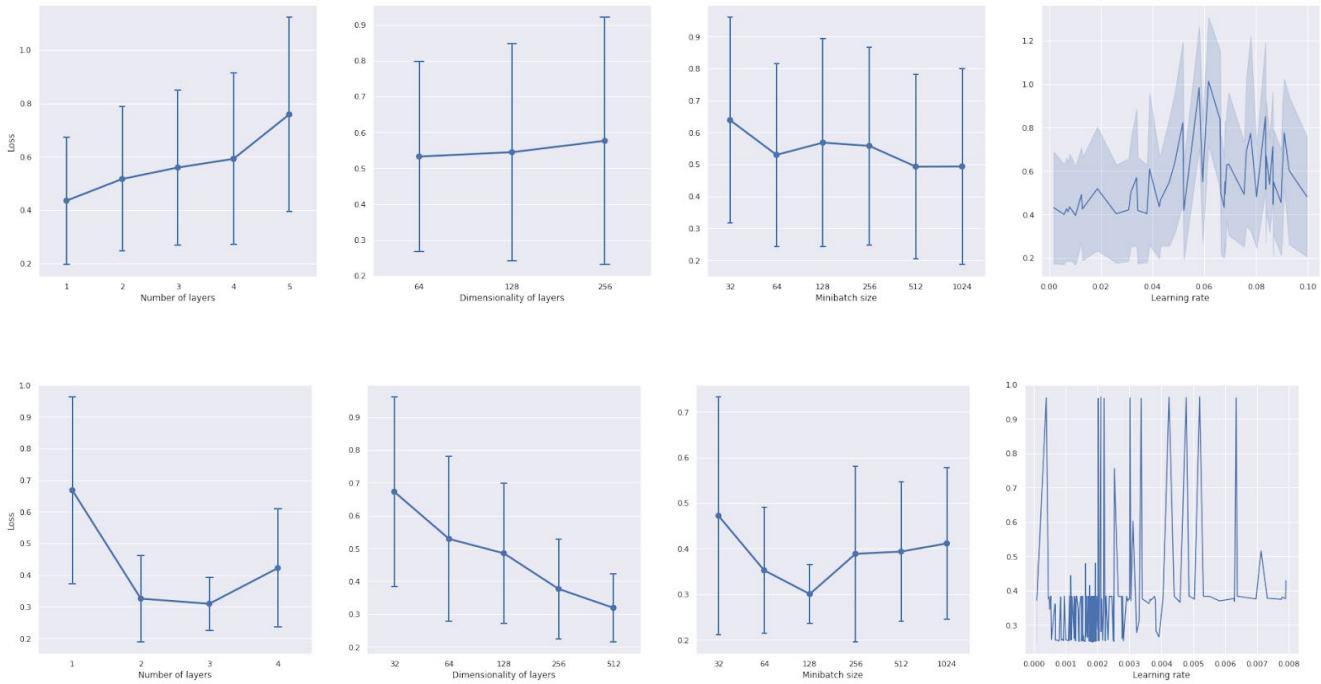


Figure 3.6. Loss response to changes in hyperparameters with error bars indicating standard deviation. Results in top row are from random search, while bottom row results are from directed search using CMA-ES.

The directed CMA-ES hyperparameter tuning experiments help give a clear picture of the effect of model hyperparameters as shown in Figure 3.7. We performed directed searches to optimize the standard error averaged over all four buildings. After approximately 450 trials, the best observed model uses three layers with 512 hidden nodes each. The Adam optimizer proved to be most effective at optimizing these models rather than other optimization algorithms, and it seems as though lower learning rates produce better results. The average learning rate among the top 20 models is 0.00174. Residual predictions seem to help the model, but the search did not result in configurations using residual connections between recurrent layers. The search converged on GRU as the recurrent cell type, but LSTM cells show comparable performance. Further, the best models obtained from this search do not use any L2 weight regularization but instead incorporate a relatively high degree of dropout. The top 20 results used a dropout probability of 0.8 on average. These findings are illustrated in Figure 3.7, and the top ten configurations are listed in Table 3.3.

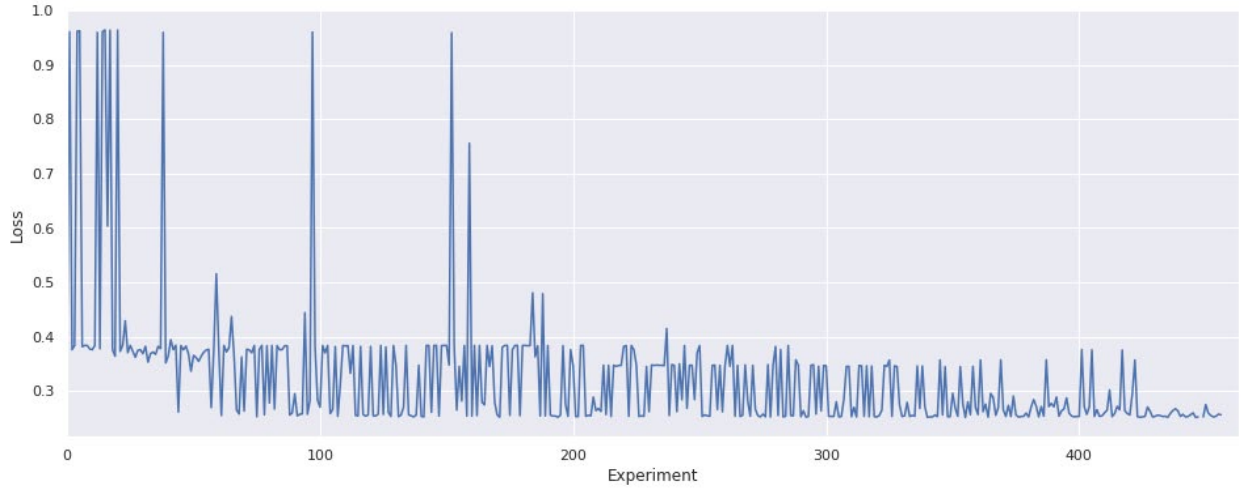


Figure 3.7. Model performance as CMA-ES optimization progressed and converged on a solution.

Table 3.3. Top ten configurations.

Number of Layers	Layer Dimension	Optimization	Learn Rate	Batch Size	Cell Type	Bidirectional Encoder	Residual Layers	Residual Out	Dropout Probability	L2 Regularization	Standard Error
3	512	Adam	0.001843	128	GRU	False	False	True	0.82381	0	0.25120
3	512	Adam	0.001843	128	GRU	False	False	True	0.82381	0	0.25138
3	512	Adam	0.001849	128	GRU	False	False	True	0.82392	0	0.25141
3	512	Adam	0.001843	128	GRU	False	False	True	0.82383	0	0.25164
3	512	Adam	0.001843	128	LSTM	False	False	True	0.82376	0.000015	0.25167
3	512	Adam	0.001842	128	GRU	False	False	True	0.82388	0	0.25179
3	512	Adam	0.001844	128	GRU	False	False	True	0.82391	0	0.25182
3	512	Adam	0.001832	128	GRU	False	False	True	0.80237	0	0.25185
3	512	Adam	0.001844	128	GRU	False	False	True	0.82265	0	0.25202
3	512	Adam	0.001844	128	GRU	False	False	True	0.82380	0	0.25202

3.3.4 Transferability of Models across Buildings

Figure 3.8 shows the performance of models trained on single buildings or combinations thereof, and then evaluated on each of the buildings and combinations. We include data from four additional office buildings in this analysis, hereafter collectively referred to as the off-site buildings: Building W, located in New Mexico and collected from January 2011 to 2012; Building X, located in California and collected from July 2001 to 2002; Building Y, collected from July 2011 to 2012 (location unknown); and Building Z, located in Colorado and collected from January 2012 to 2013. First, we observe that some buildings are easier to make predictions on in general (e.g., Buildings B and Y), while others are challenging (e.g., Buildings A and D). Intuitively, we see the best transferability among individual models for the on-site buildings, since they are all located in the same region. Individual models trained on on-site buildings also demonstrate poor transfer performance when transferring to buildings in other regions, though curiously certain individual models for off-site buildings are more capable of transferring to on-site buildings. In all cases, combination models perform adequately on the individual buildings they were trained on but do not outperform individual models. One promising direction to improve transferability would be to initialize a model on a source building, but rather than deploy it cold to a new target building, adapt its parameters with a limited amount of data from the target building. We leave that topic for future works.

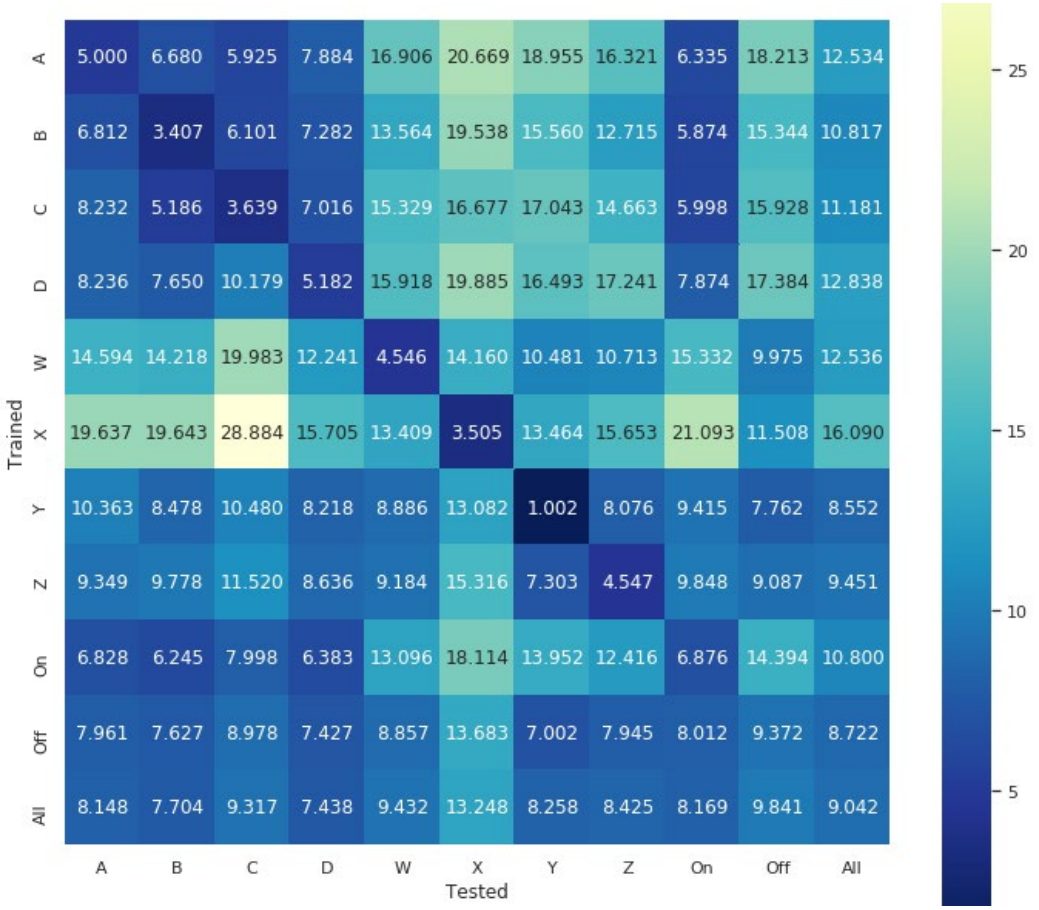


Figure 3.8. Comparison of standard error on the validation and test sets for single models trained on each building and tested on all buildings. Top row shows standard error for model trained on all buildings.

3.3.5 Prediction Analysis

Figure 3.9 presents a visual representation of the predictions generated by our sequence-to-sequence RNN model compared to ground truth for an optimal model configuration found by the directed search. It shows that our model can capture general electrical usage patterns; for instance, it seems to predict the onsets and offsets of peak electricity usage periods reasonably well. However, during periods of quiescence such as off-hours and weekends, the model tends to lag the ground-truth building load (note that it exhibits behavior unlike a persistence model due to its use of residual predictions). This is likely due to the high degree of irreducible noise in these periods. Because some appliances draw power irrespective of outdoor temperatures, a certain degree of this consumption is uncorrelated with environmental conditions. Moreover, the prediction quality again degenerates to persistence modeling when the model is evaluated on unseen buildings, though for certain unseen buildings the model generalizes peak usage and load profile patterns well.

Figure 3.10 shows how the hyperparameters for our best generic on-site model generalize to models trained on the off-site buildings. Like the models trained on buildings included in the directed search, these models are able to forecast general usage patterns, suggesting that the hyperparameters found by our directed search can generalize and be used to train models on buildings in other regions provided that a similar quantity of data are available for these buildings.

Finally, Figure 3.11 and Figure 3.12 provide examples of how models trained on a building or combination of buildings generalize to unseen buildings. Consistent with the results presented in Section 3.3.4, Figure 3.11 demonstrates that models generalize to other on-site buildings well, while Figure 3.12 suggests the same cannot be said of on-site model generalization to off-site buildings.

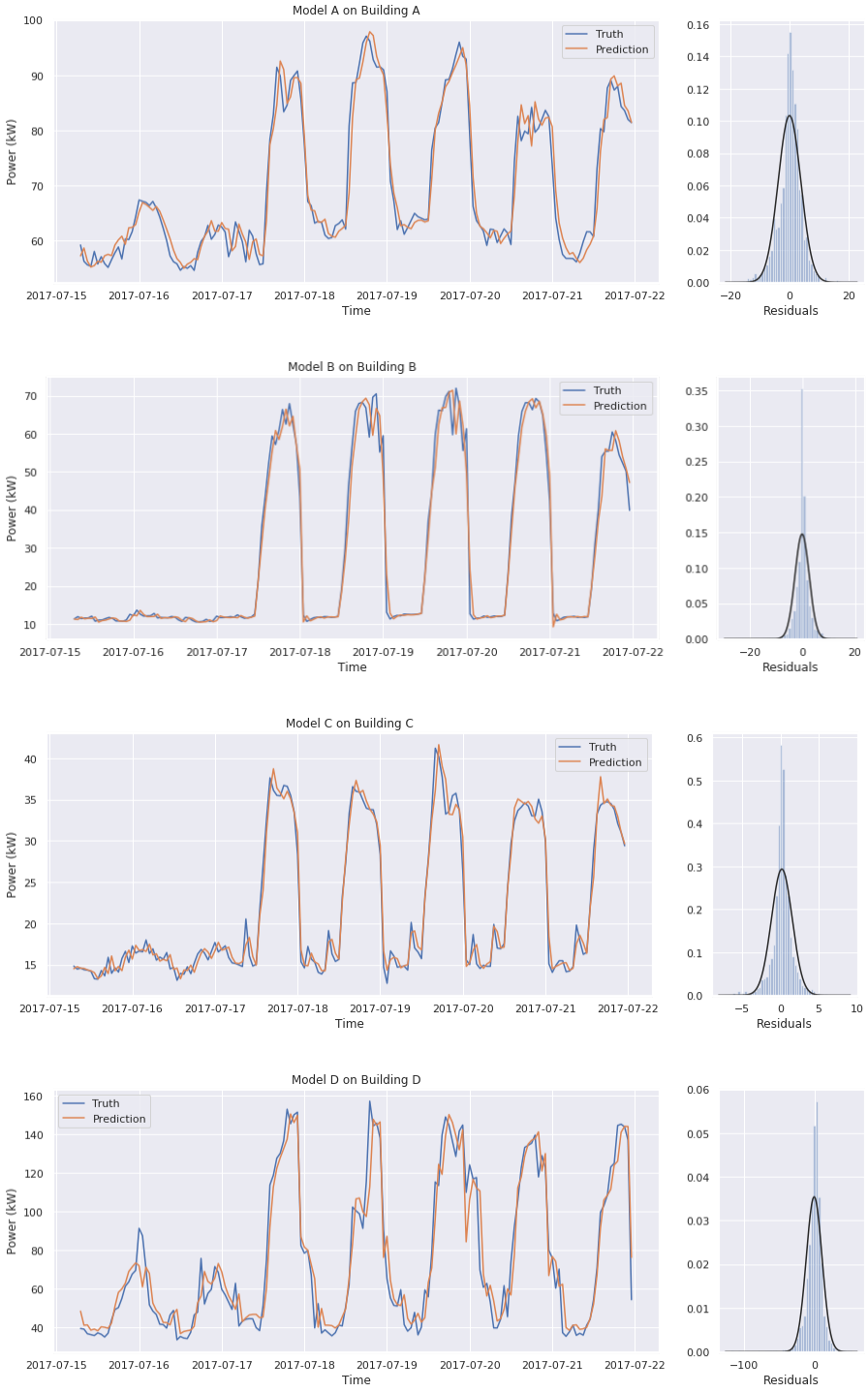


Figure 3.9. Predicted vs. ground truth electrical load measurements and residual histograms for each on-site building. Line plots show model performance on a week-long section of each building's test set, while residual histograms encompass the entire test set for each building.

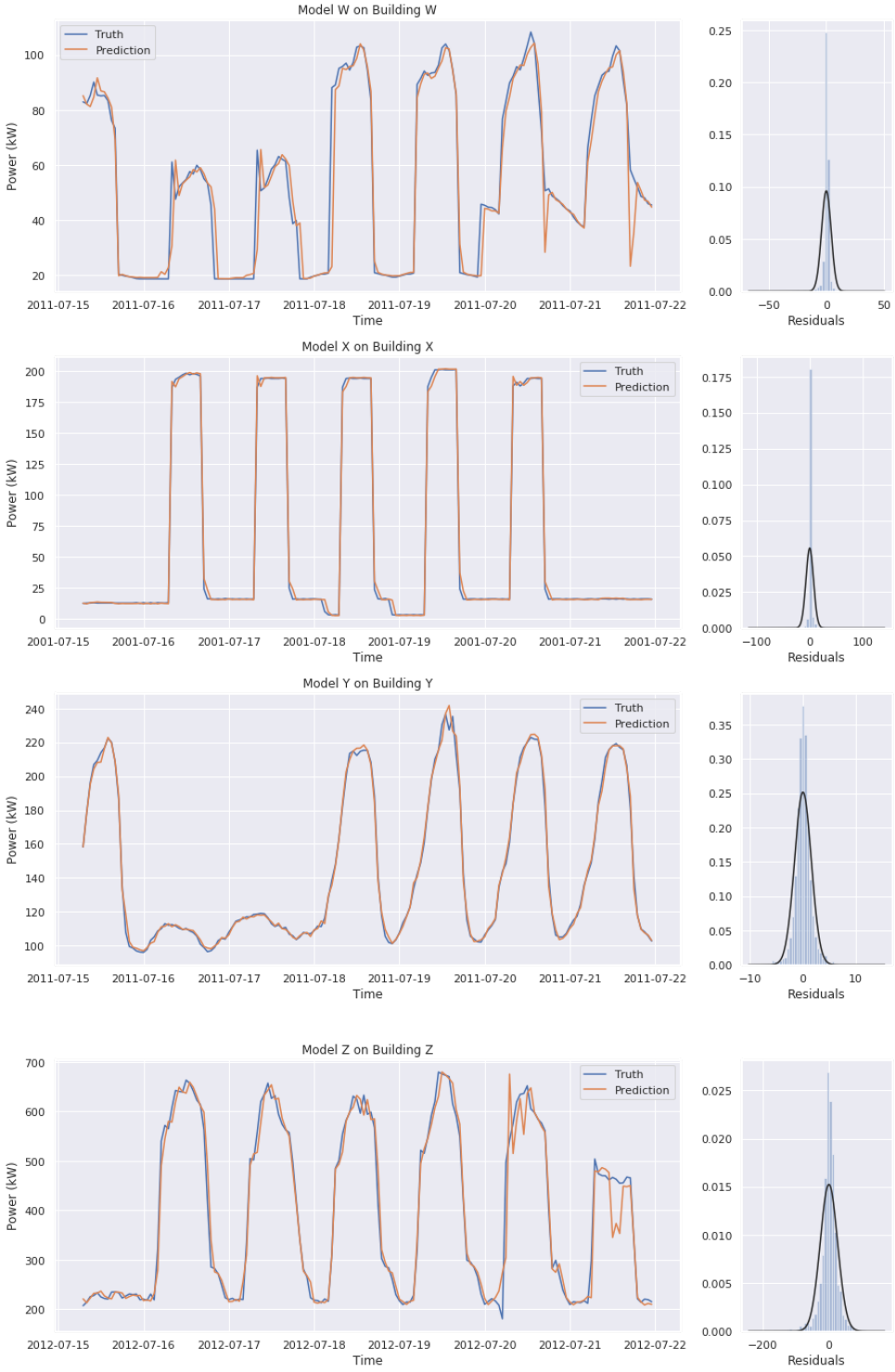


Figure 3.10. Predicted vs. ground truth electrical load measurements and residual histograms for each off-site building.

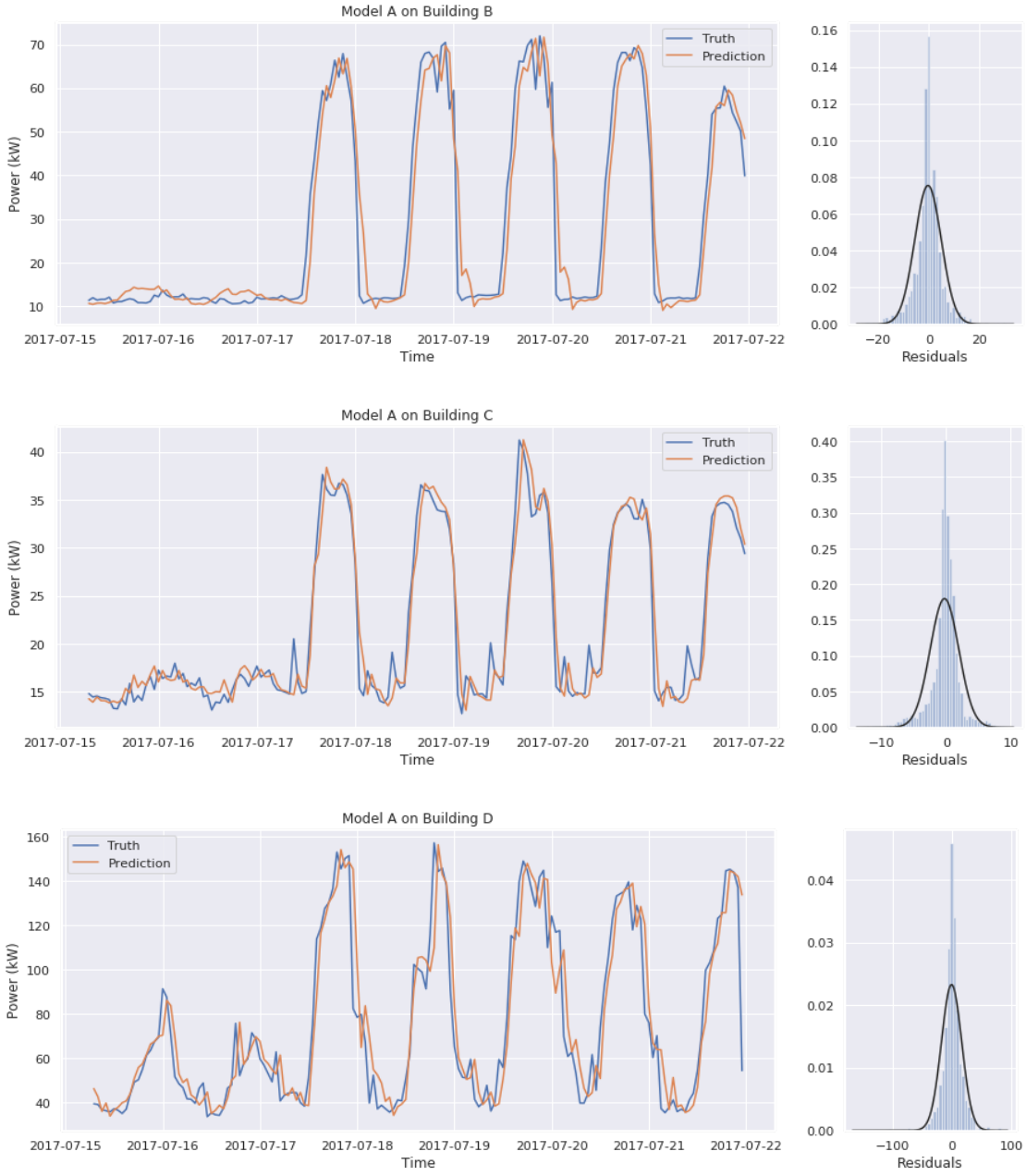


Figure 3.11. Predicted vs. ground truth electrical load measurements and residual histograms for a model trained on Building A, then tested on other on-site buildings.

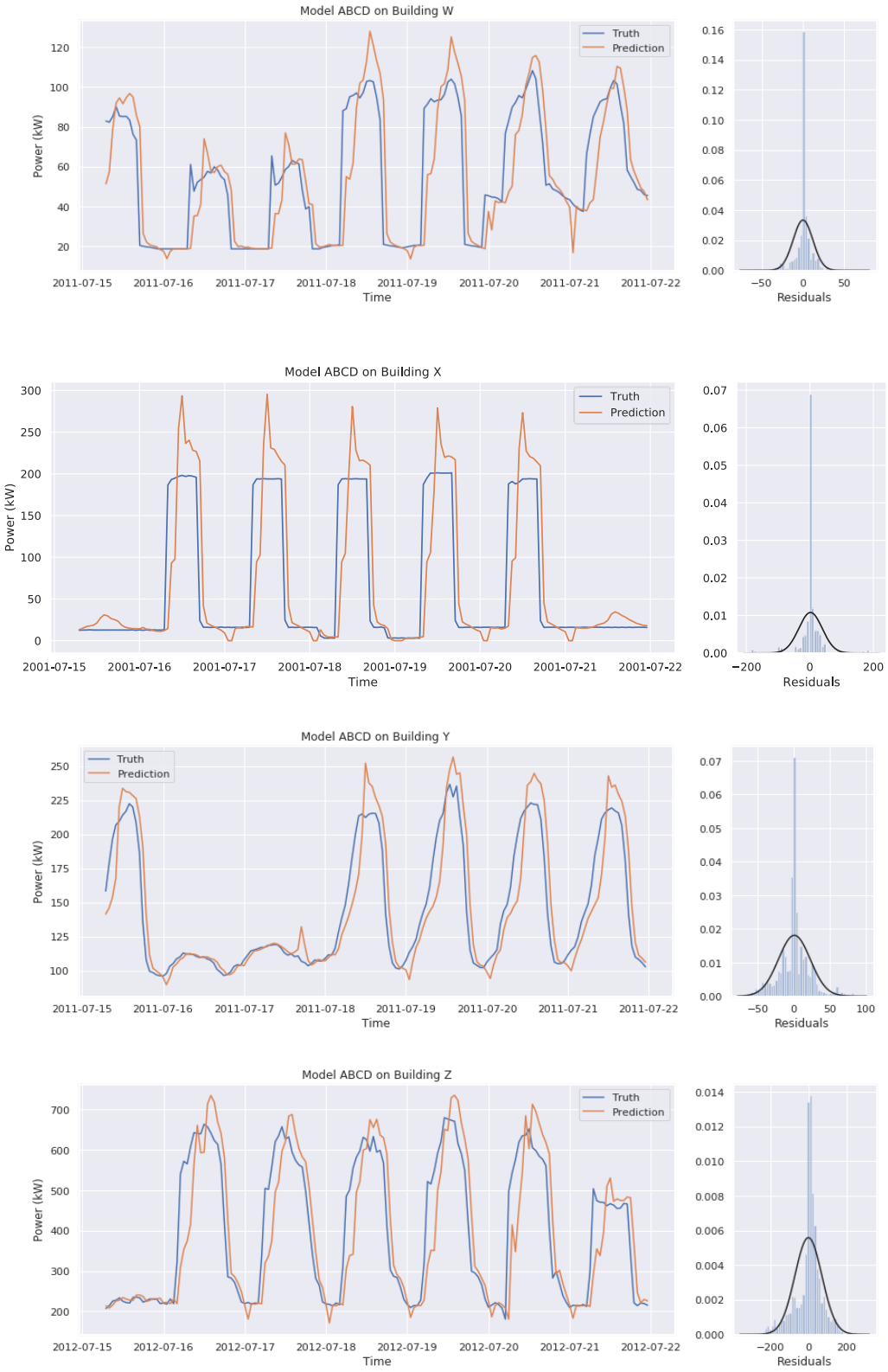


Figure 3.12. Predicted vs. ground truth electrical load measurements and residual histograms for a model trained on all on-site buildings, then tested on each off-site building.

4.0 Temperature and Airflow Forecasting

This section describes data preparation, experimental configurations, and the training procedure for both temperature and airflow forecasting. Two factors drive the design choices made in both data preparation and model selection. The primary factor being the fine-grained prediction resolution (minute-level) and the duration of the prediction window (24 hours). In contrast to the 15 minutes previously used for temperature prediction (Yimin Chen et al. 2019), we must predict out a total of 1440 time steps. A secondary challenge that factors into the first challenge is the existence of missing data; we propose a few methods to compensate for missing data.

4.1 Data Preparation

We must first preprocess the data; these steps are identical for the RTU (temperature forecasting) and VAV (airflow forecasting) building zones. We first split 12 months of data into training, validation, and test sets using the strategy described in Section 3.2.1. Next, we create several periodic time features: day of year, time of day, and day of week. Each of these three temporal inputs is represented by a pair of features (with the cyclical encoding by sine and cosine transformations).

We then standardize the measured features by subtracting the mean value of each feature and dividing by the standard deviation. The mean and standard deviation is calculated on the training set. We apply the same transformations to the validation and test sets (i.e., using the training set mean and standard deviation).

4.1.1 Handling Missing Data

Due to measurement errors in the devices used to collect data there are some gaps in the measured data. Roughly three percent of the year of data is missing at random intervals. We have explored three separate methods to address the problem of missing data. The first method is to linearly interpolate over the missing values (for non-binary variables) as a preprocessing step.

The next two methods do not fill the missing data during preprocess, but instead modify the model architecture to support missing values.

The first of these two methods, which we will refer to as “simple embeddings”, requires us to create a two-dimensional encoding for each feature that contains missing values. The first dimension is a binary variable that encodes whether the value of the feature that we are encoding is missing or not. The second dimension of this encoding is the actual value of the feature. If the feature is missing, then the second dimension is set to zero. This effectively allows the model to learn custom interpolation strategies, and to potentially weight implicitly imputed data differently than real data.

We denote the final method as “full embeddings”. We discretize the variable, mapping it into a set of equally spaced “bins” that span the range of observed values for the feature. Missing values are mapped to a distinct bin. As an example, for zone temperatures we discretize into 200 evenly spaced bins ranging from 60°F to 80°F, where the first and last bins include any temperature measurements below 60°F and above 80°F, respectively. We treat the bin identifier for a temperature as a categorical label, which is used to index into a matrix of temperature embeddings. That is, each discrete temperature is represented by a vector; the contents of the

vector are learned jointly with all model parameters during training. The advantage of this strategy is that it is extremely flexible. The major disadvantage to this approach is that all temperatures are initially treated as equidistant from each other, and the similarity, for example, between 70.1 degrees and 70.2 degrees must be learned from scratch from the data.

4.2 Model Structure and Model Inputs and Outputs

This section describes the model structure, inputs, and outputs. These include airflow and temperature forecasting.

4.2.1 Model Structure

Both temperature and airflow forecasting utilize the same model architecture. Since these tasks require us to model many time steps into the future, we have chosen to employ the Wavenet (van den Oord et al 2016)[1] architecture. Wavenet, developed by DeepMind, was made specifically for modeling long-term dependencies required for generating audio. We are using a slightly modified version known as Fast Wavenet (Le Paine et al. 2016)[2] that utilizes a queueing system to store the results of past calculations for future use, speeding up predictions. The primary architectural building block used in Wavenet is dilated temporal convolution.

Illustrated in Figure 4.1, these stacked dilated temporal convolutions enable the model to use information far in the past with an efficient number of parameters. In this diagram the nodes represent a learnable parameter and the arrows indicate which information a node uses as input. This efficiency is aided by increasing the dilation on deeper layers, enabling the model to leverage a long input sequence when making predictions, without an excessive number of model parameters.

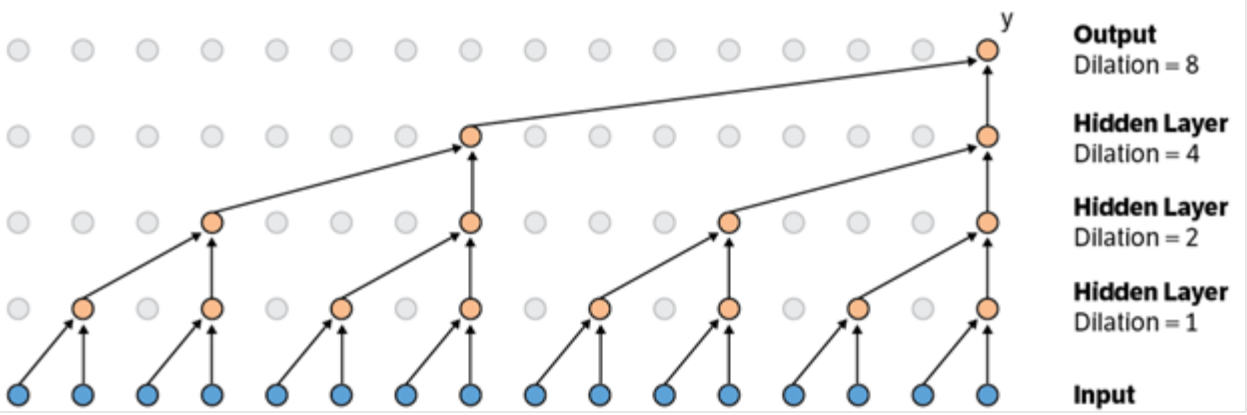


Figure 4.1. Stacked dilated temporal convolutions[3].

[1] https://arxiv.org/abs/1609.03499
[2] https://arxiv.org/abs/1611.09482
[3] https://arxiv.org/abs/1611.09482

4.2.2 Airflow Forecasting Inputs and Outputs

Features (for both airflow and temperature predictions) were selected to be plausibly available at prediction time. For instance, we do not use zone temperature for input to the airflow forecasting models as we cannot accurately know the zone temperature for 24 hours into the future.

We use the following features as input at each timestep to the airflow forecasting model:

Time encoding features (sine and cosine features for each):

- Minute of day
- Day of week
- Day of year

Standardized features (zero mean and unit variance):

- Zone cooling airflow set point (predicted at each timestep and fed back in auto-regressively)
- Zone cooling airflow set point (24 hours ago)
- Outdoor air temperature
- Zone cooling temperature set point

Binary encoded features (True or False)

- Holiday

Lastly, we include a feature that increments from zero to one, linearly, as prediction proceeds from one minute to 1440 minutes into the future. We call this feature the “minute of prediction” encoding as we predict at a one-minute resolution. The inspiration for this feature is that the autoregressive zone cooling airflow set point feature is likely to be more reliable earlier in the prediction than later.

The model outputs the zone cooling airflow set point, which is fed back in at the next timestep autoregressively.

4.2.3 Temperature Forecasting Inputs and Outputs

We use the following features as input to the temperature forecasting model:

Time encoding features (sine and cosine features for each):

- Minute of day
- Day of week
- Day of year

Standardized features (zero mean and unit variance):

- Zone temperature (predicted at each timestep and fed back in autoregressively)
- Zone temperature (24 hours in the past)

- Outdoor air temperature
- Heating temperature set point
- Cooling temperature set point

Binary encoded features (True or False)

- Holiday

We include the minute of prediction variable as input to the temperature forecasting as well.

The model then outputs the zone temperature, and feeds this in auto-regressively as we predict a sequence of values.

4.3 Performance Metrics

We evaluate and compare the performance of our models using the coefficient of determination (R^2) between the predicted and true values.

For RTU buildings (zone temperature forecasting), the R^2 is defined as

$$R^2 = 1 - \frac{\sum_{k=1}^{n}\left(T^k - \hat{T^k}\right)^2}{\sum_{k=1}^{n}(T^k - \bar{T})^2}$$

where $\hat{T^k}$ is the predicted temperature at the kth discrete time, T^k is the actual temperature at the kth discrete time and n is the number evaluated time steps. For VAV buildings, the temperature is replaced by the airflow rate. We also report RMSE.

4.4 Baseline Method

When evaluating the airflow prediction of our Wavenet we compare it to the performance of the previously developed method of using reduced-order models (ROMs). Experiments with ROMs for temperature prediction is left to the future work.

4.4.1 Input/Output of ROMs

For buildings that have VAV systems, the airflow rate of a VAV terminal (zone) is the desired dependent variable of interest and it is a function of the following:

$$m_i^k = f\left(T_i^{k-1}, T_a^k, T_{i,s}^k, h^k, d^k\right) \quad (4.1)$$

where $m_i^k$is the airflow rate for the ith terminal at the kth discrete time, T_i^{k-1} is the zone temperature for the ith terminal at the $(k-1)$th discrete time, T_a^k and $T_{i,s}^k$ are the ambient temperature and the zone temperature set point for the ith terminal, respectively, at the kth discrete time, h^k and d^k are the hour index of a day and day index of a year, respectively. Typically, the measured airflow rate contains noise or disturbances that are not measured or measurable; we use the airflow rate set point as a replacement of the airflow rate when training the model. Although the airflow rate is dependent on many other variables (such as internal gains, solar gains, etc.), we have reduced the model to use parameters typically measured in a building.

It is noted that there are usually an upper bound (design maximum) and a lower bound (design minimum) for the airflow rate set point. When estimating the output from equation (4.1), we must use the following formula:

$$m_i^k = min\{m_{i,max}, max\{m_i^k, m_{i,min}\}\} \tag{4.2}$$

where $m_{i,max}$ and $m_{i,min}$ are the design maximum and design minimum airflow rate for the ith terminal. In addition, when applying equation (4.1) in a long-term prediction, e.g., 24-hour prediction, as T_i^{k-1} is unknown, one can make the following assumption:

$$T_i^{k-1} = T_{i,s}^{k-1} \tag{4.3}$$

In such case, $T_{i,s}^{k-1}$ will be provided as one of the inputs.

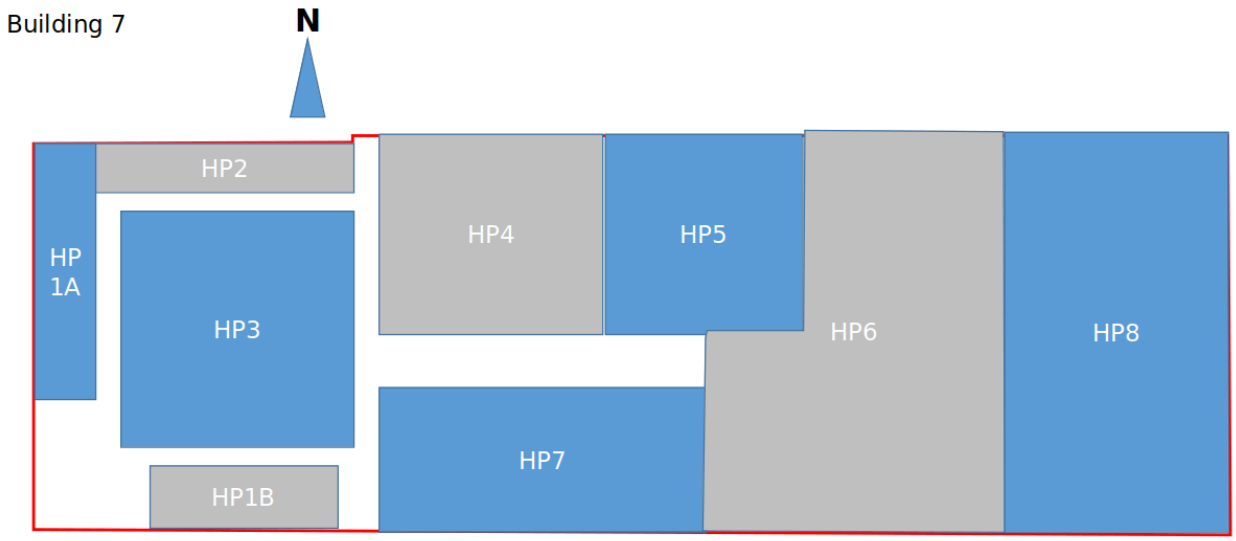
For a building with RTUs, the temperature of the zone served by the RTU is the desired dependent variable and it is a function of the following:

$$T_i^k = f\left(T_i^{k-1}, T_a^k, S_i^k, h^k, d^k\right) \tag{4.4}$$

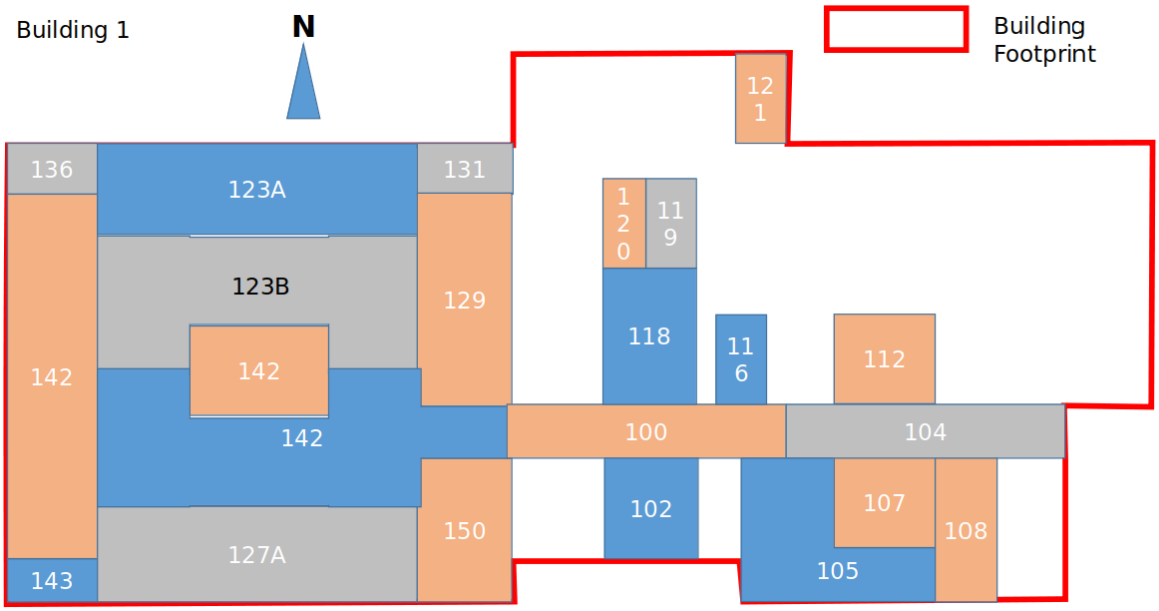
The parameters for the ROM (Equations 4.1 and 4.4) are estimated from one week of data, and the model is evaluated on the next week. To compare with the Wavenet model, we always evaluate on the same validation weeks.

4.5 Experiments

In this section, we analyze the results of our zone forecasting experiments. We selected a subset of zones to evaluate airflow prediction (zones VAV143, VAV131, and VAV129) and to evaluate temperature prediction (zones HP3 and HP4). They consist of both interior (VAV129 and HP3) and exterior (VAV143, VAV131, and HP4) zones. Below is the layout of the buildings these zones were sampled from.



(a)



(b)

Figure 4.2. Selected building zones to evaluate temperature predictions (a) and airflow prediction (b).

4.5.1 Implementation and Model Training

The Wavenet model was implemented with PyTorch[1], using a significantly modified version of the implementation found on Github[2]. Models were trained on a maximum 75 batches of 128, 24-hour sequences at a 1-minute resolution. Workstations used for training were configured as follows: Intel Xeon ES-1620v3 (3.5 GHz) CPU, 64 GB RAM, and either Nvidia GTX 980Ti with 6 GB VRAM or Titan Xp with 12 GB VRAM. Training over one batch took approximately 664 seconds (~11 minutes) to compute, leading the total training time of a single model to be roughly 13.75 hours. The model is 7.4 megabytes in size when saved.

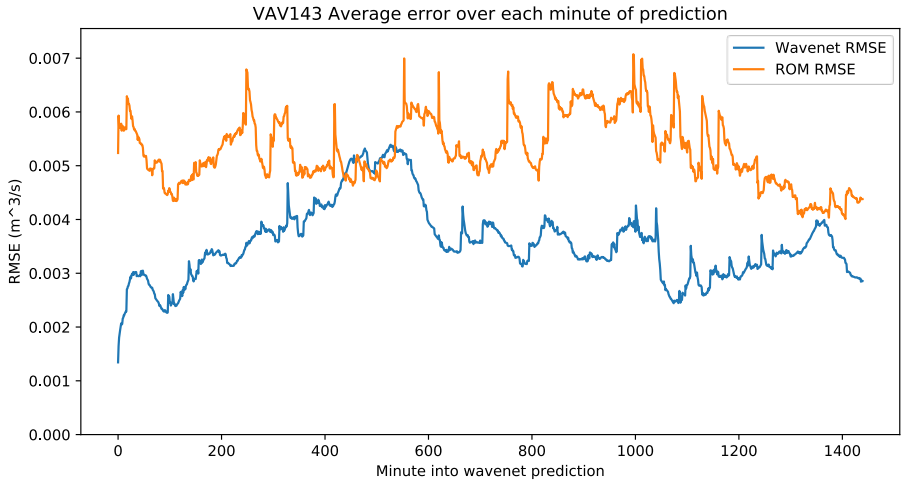
4.5.2 Comparing Wavenet and the Reduced-Order Model for Airflow Predictions

In this section we first compare R^2 values of the ROM and the Wavenet to directly evaluate their performance over the same 128 sequences in the validation set. We then compare error evaluated over each minute of the predicted 24-hour sequences. This allows us to evaluate how the Wavenet performs as it predicts further out. We then present a few randomly selected traces of prediction vs target to qualitative evaluate performance. In Table 4.1 we can see that our Wavenet outperforms the ROM on all three selected airflow prediction zones. We also notice that VAV143 is significantly harder to predict than the other selected zones.

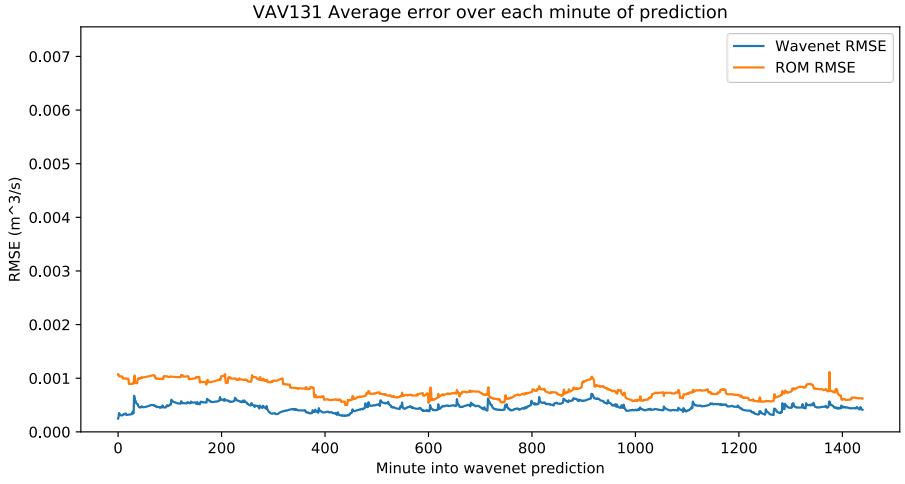
Table 4.1.Wavenet vs ROM R^2 comparison.

	VAV143	VAV131	VAV129
ROM R^2	0.4718	0.7297	0.7016
Wavenet R^2	**0.7480**	**0.9053**	**0.8784**

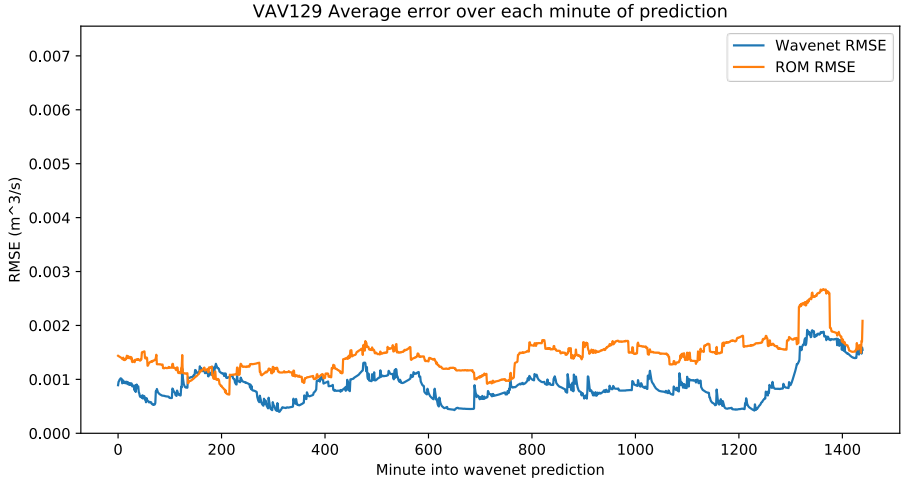
[1] PyTorch is an open source machine learning software library for the programming language Python, based on Facebook's Torch library.
[2] https://github.com/vincentherrmann/pytorch-wavenet, (Herrmann 2017, 2019).



(a)



(b)

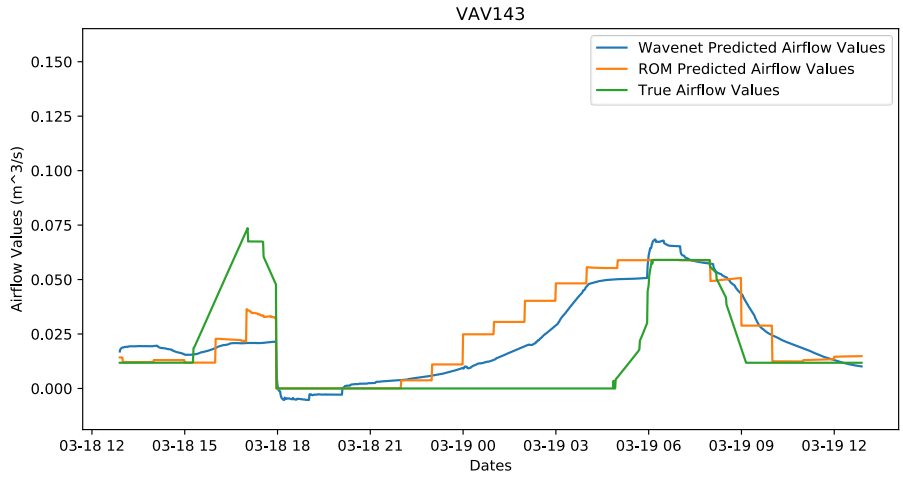


(c)

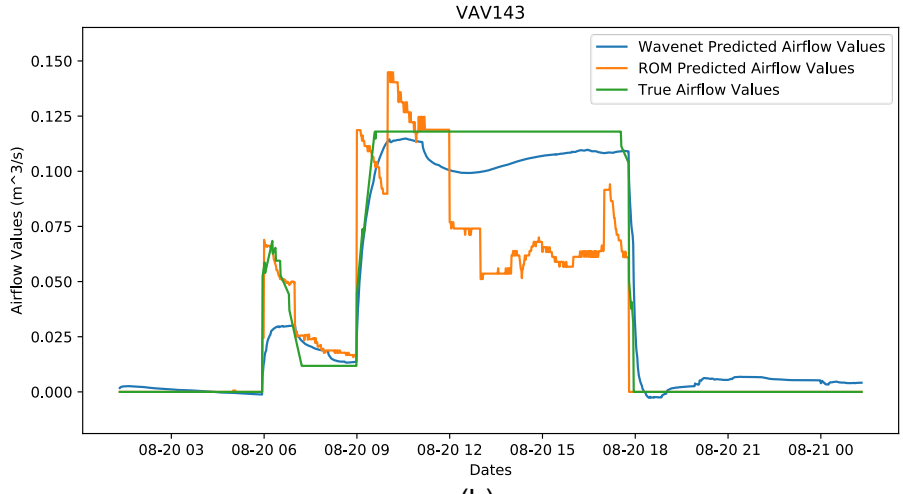
Figure 4.3. Root Mean Squared Error (RMSE) over the 24-hour window.

With Figure 4.3 we can see how the Wavenet performs over its 24-hour window and compare this to ROM performance over the same set of sequences. We initially expected to see Wavenet error increase over the 24-hour window as it autoregressed out 24 hours. However, it seems the error is relatively constant, and we think that the reason for this is that airflow is very sporadic. This erratic behavior may prevent the model from relying too much on the ground truth that it has seen previously. Another trend we notice is that the errors of the ROMs and Wavenet are somewhat correlated in these plots indicating that we may need to select more sequences (128 sequences are currently being selected) to evaluate. Selecting more sequences may prevent individual sequences from affecting the average error.

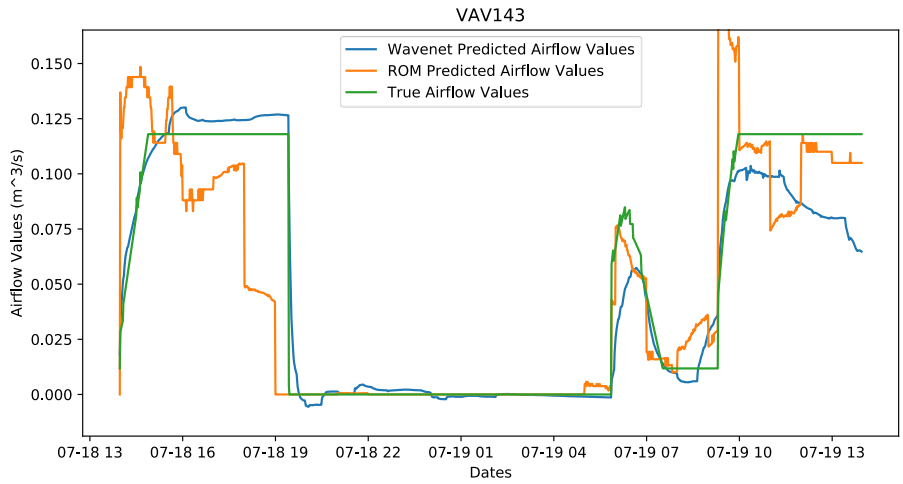
Finally, we present some randomly selected examples of model prediction vs ground truth in Figure 4.4 [(a) through (k)].



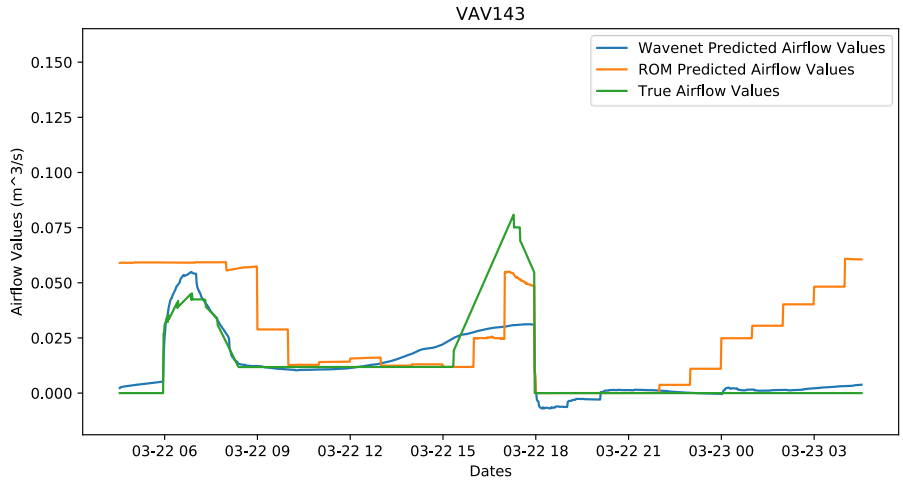
(a)



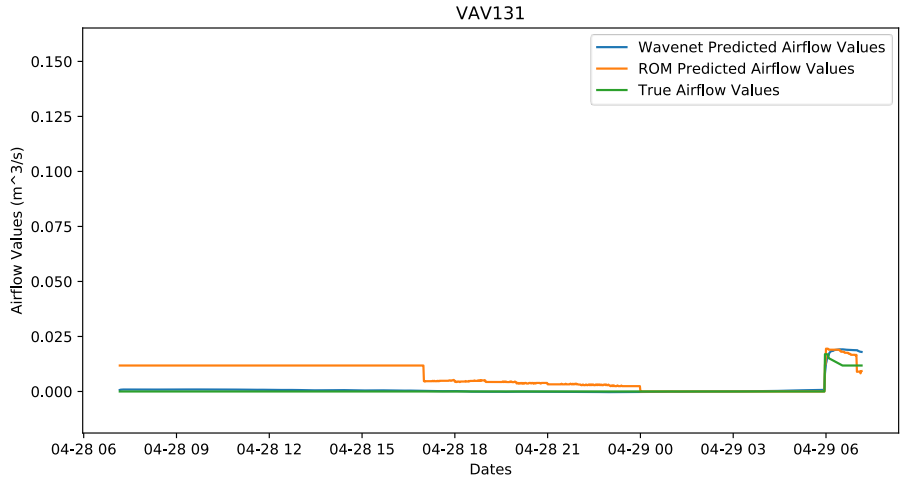
(b)


VAV143
Wavenet Predicted Airflow Values
ROM Predicted Airflow Values
True Airflow Values
Airflow Values (m^3/s)
0.150
0.125
0.100
0.075
0.050
0.025
0.000
07-18 13
07-18 16
07-18 19
07-18 22
07-19 01
07-19 04
07-19 07
07-19 10
07-19 13
Dates

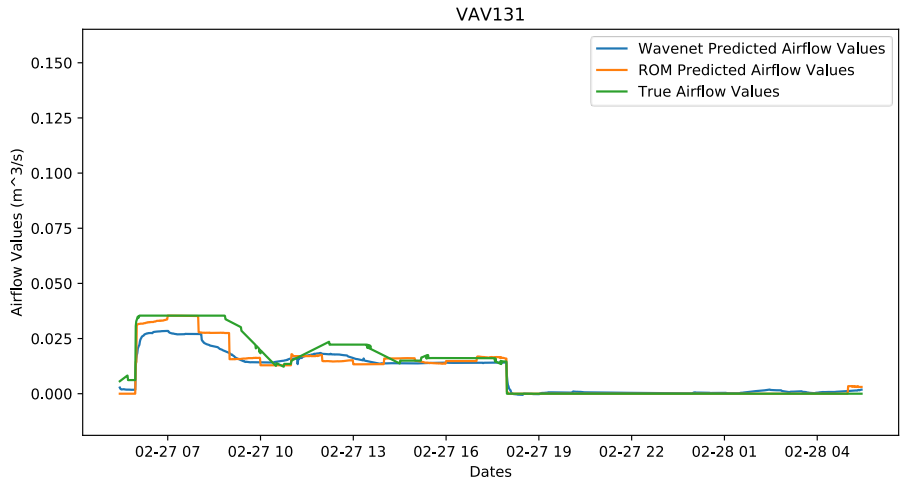
(c)


VAV143
Wavenet Predicted Airflow Values
ROM Predicted Airflow Values
True Airflow Values
Airflow Values (m^3/s)
0.150
0.125
0.100
0.075
0.050
0.025
0.000
03-22 06
03-22 09
03-22 12
03-22 15
03-22 18
03-22 21
03-23 00
03-23 03
Dates

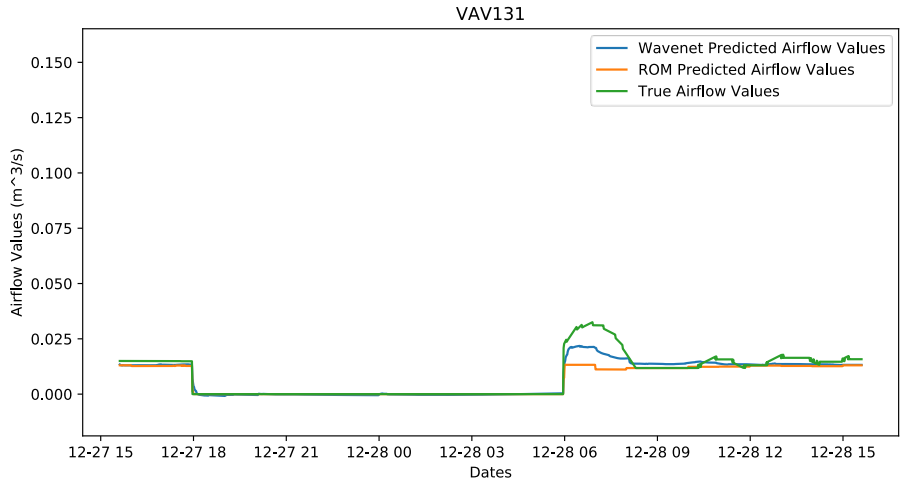
(d)


VAV131
Wavenet Predicted Airflow Values
ROM Predicted Airflow Values
True Airflow Values
Airflow Values (m^3/s)
0.150
0.125
0.100
0.075
0.050
0.025
0.000
04-28 06
04-28 09
04-28 12
04-28 15
04-28 18
04-28 21
04-29 00
04-29 03
04-29 06
Dates

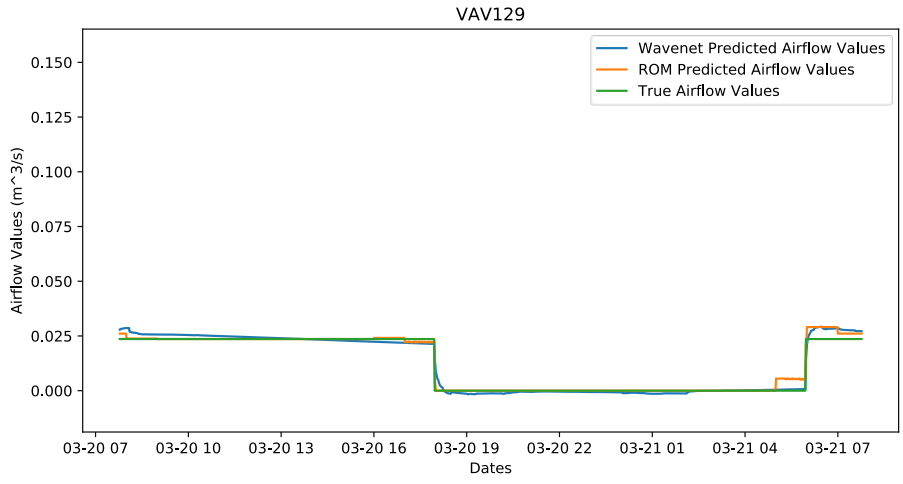
(e)


VAV131
Wavenet Predicted Airflow Values
ROM Predicted Airflow Values
True Airflow Values
Airflow Values (m^3/s)
0.150
0.125
0.100
0.075
0.050
0.025
0.000
02-27 07
02-27 10
02-27 13
02-27 16
02-27 19
02-27 22
02-28 01
02-28 04
Dates

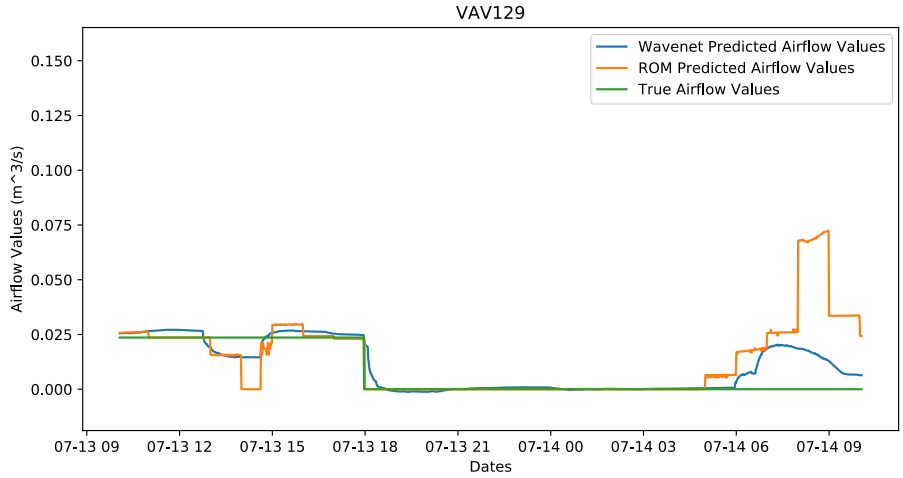
(f)


VAV131
Wavenet Predicted Airflow Values
ROM Predicted Airflow Values
True Airflow Values
Airflow Values (m^3/s)
0.150
0.125
0.100
0.075
0.050
0.025
0.000
12-27 15
12-27 18
12-27 21
12-28 00
12-28 03
12-28 06
12-28 09
12-28 12
12-28 15
Dates

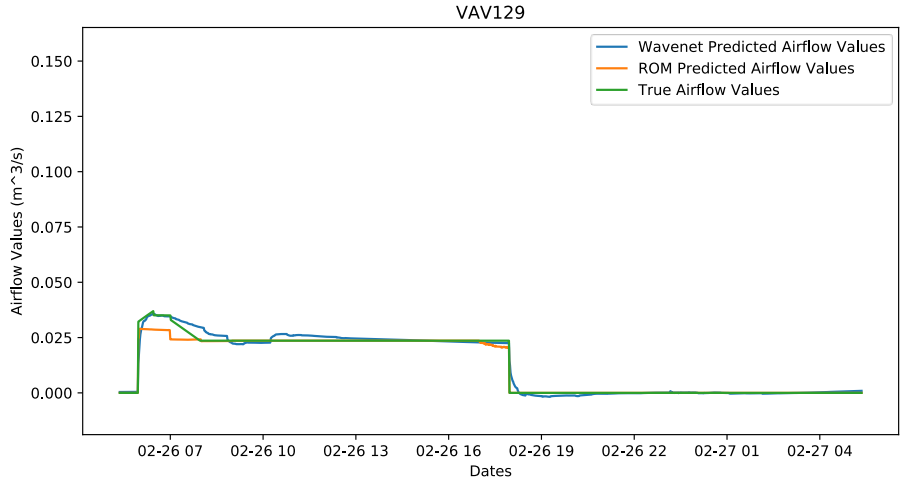
(g)


VAV129
Wavenet Predicted Airflow Values
ROM Predicted Airflow Values
True Airflow Values
Airflow Values (m^3/s)
0.150
0.125
0.100
0.075
0.050
0.025
0.000
03-20 07
03-20 10
03-20 13
03-20 16
03-20 19
03-20 22
03-21 01
03-21 04
03-21 07
Dates

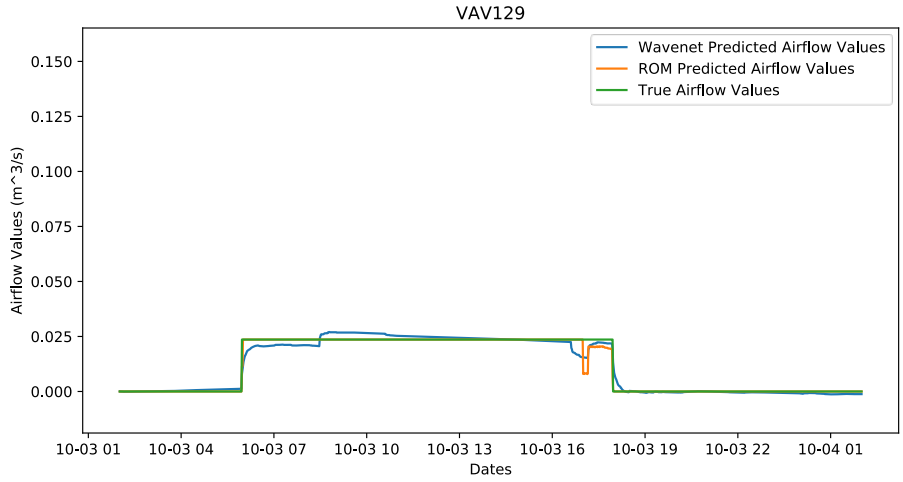
(h)



(i)



(j)



(k)

Figure 4.4. Randomly selected 24-hour sequences of predictions from zones VAV143, VAV131, and VAV129.

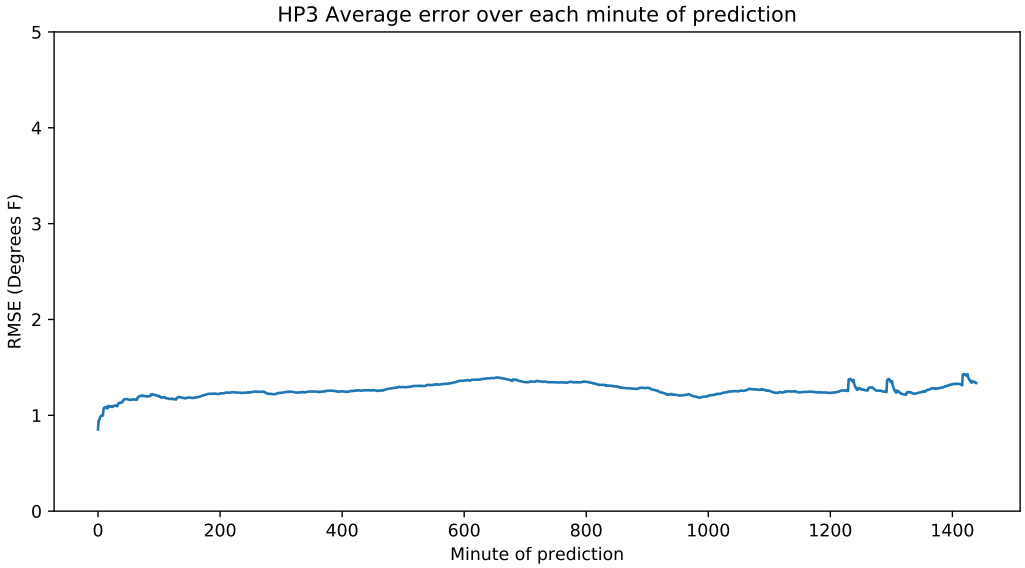
Analysis of the predictions compared to the ground truth allows us to make further insight into the performance of each model. One thing that is confirmed by these plots is the difficulty of predicting airflow for VAV143. This zone has a wider range of airflow values that are consistently encountered as well as more variability in its values over time. Also, it notes that there is the "plateau" in some areas of the ROM predictions. This plateau results in ROM predictions exactly following the ground truth in some instances. However, the Wavenet model remains more consistent in general as demonstrated by Figure 4.4 and Table 4.1.

4.5.3 Wavenet Temperature Predictions

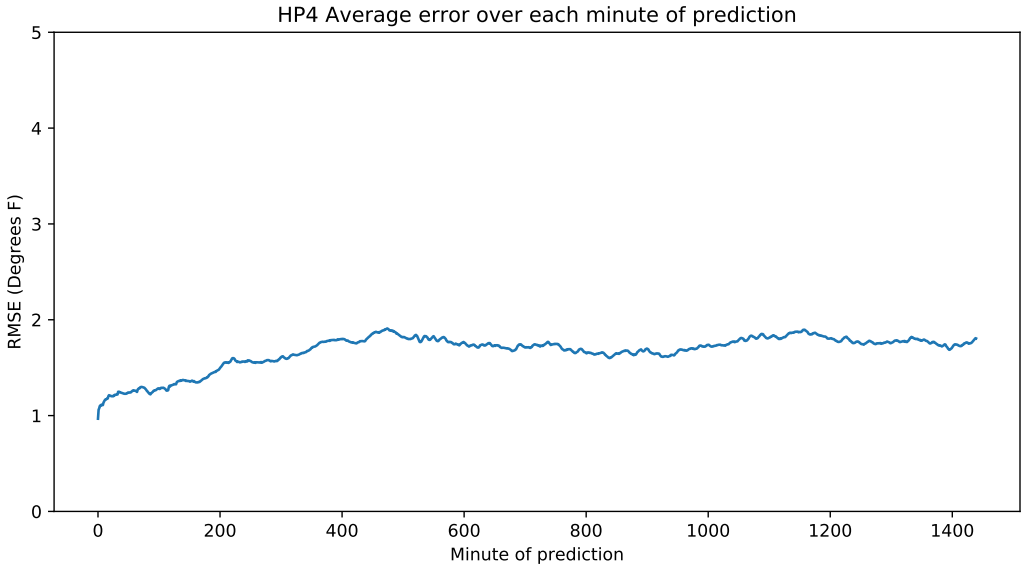
Here we will evaluate Wavenet temperature predictions, calculated on 128 randomly selected 24-hour sequences in the validation set. These results are relatively preliminary as we have yet to explore the methods of missing value compensation on temperature prediction (all results utilize linear interpolation over missing values). As shown in Table 4.2, we evaluate temperature predictions on two zones (HP3 and HP4) that use RTUs.

Table 4.2.R^2 of Wavenet temperature predictions ROM comparison).

	HP3	HP4
Wavenet R^2	0.5928	0.7795



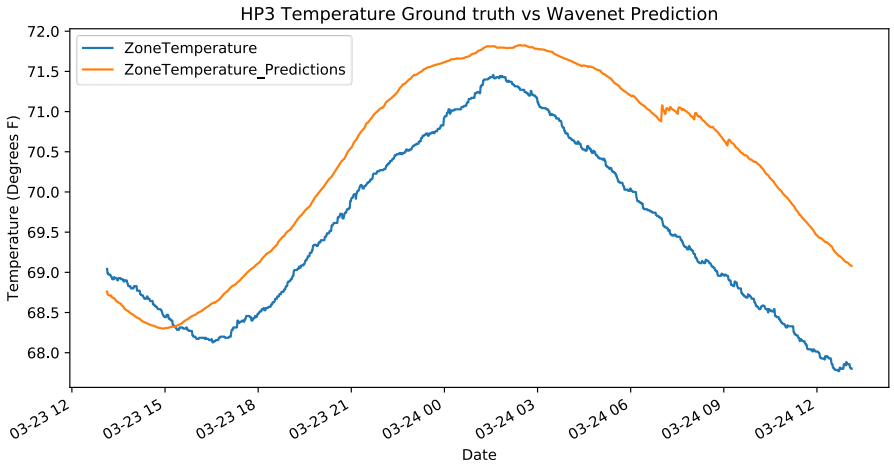
(a)



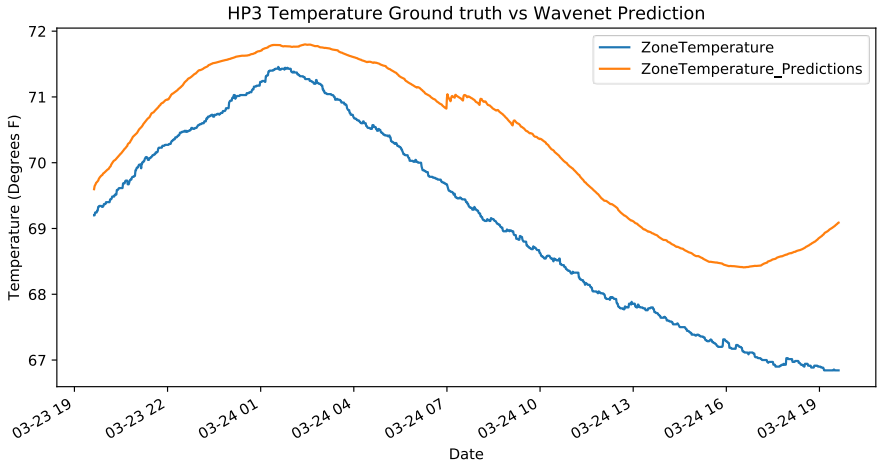
(b)

Figure 4.5. Average error over each minute of temperature prediction on 128 randomly selected sequences from validation set (a) HP3 and (b) HP4.

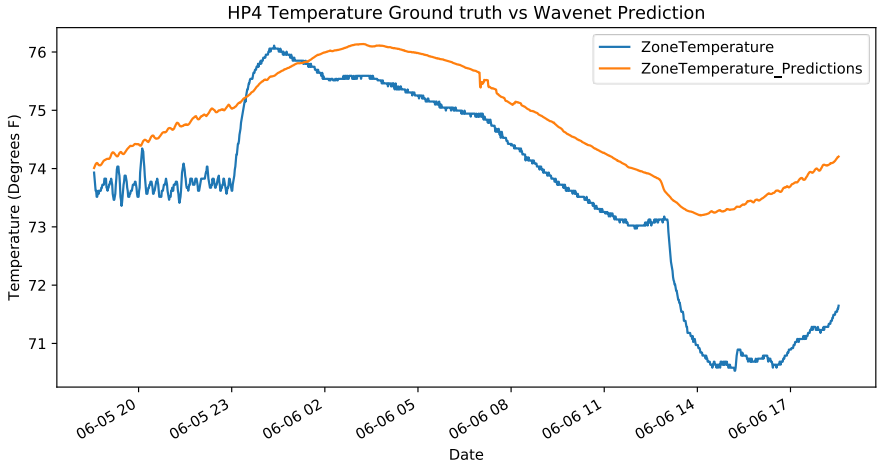
With Figure 4.5 we can see expected performance as the predictions generally worsen overtime. To summarize, at 24 hours our predictions are about 1.5 degrees off for HP4 and about 1.3 degrees off for HP3 on average.


HP3 Temperature Ground truth vs Wavenet Prediction
ZoneTemperature
ZoneTemperature_Predictions
Temperature (Degrees F)
72.0
71.5
71.0
70.5
70.0
69.5
69.0
68.5
68.0
03-23 12
03-23 15
03-23 18
03-23 21
03-24 00
03-24 03
03-24 06
03-24 09
03-24 12
Date

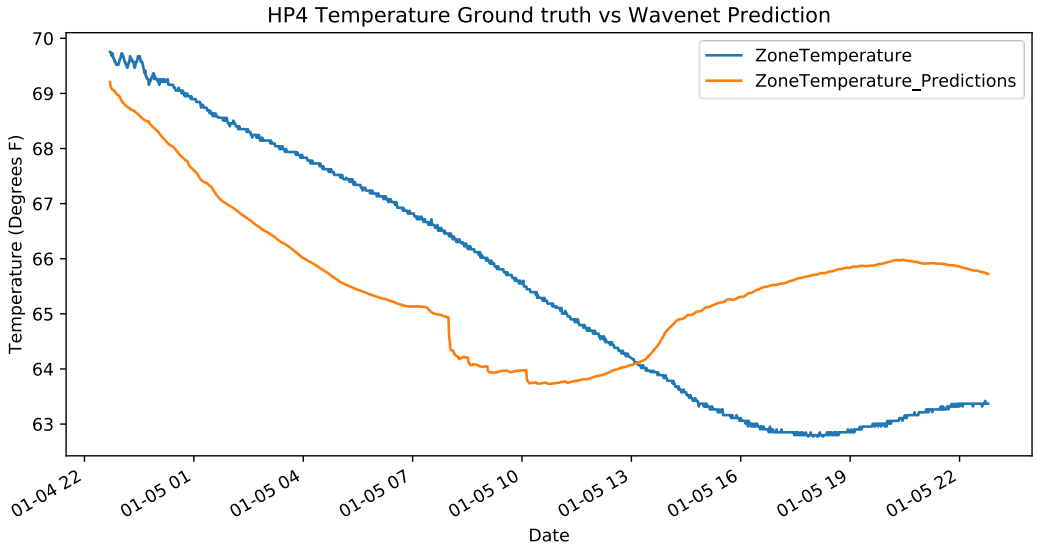
(a)


HP3 Temperature Ground truth vs Wavenet Prediction
ZoneTemperature
ZoneTemperature_Predictions
Temperature (Degrees F)
72
71
70
69
68
67
03-23 19
03-23 22
03-24 01
03-24 04
03-24 07
03-24 10
03-24 13
03-24 16
03-24 19
Date

(b)


HP4 Temperature Ground truth vs Wavenet Prediction
ZoneTemperature
ZoneTemperature_Predictions
Temperature (Degrees F)
76
75
74
73
72
71
06-05 20
06-05 23
06-06 02
06-06 05
06-06 08
06-06 11
06-06 14
06-06 17
Date

(c)



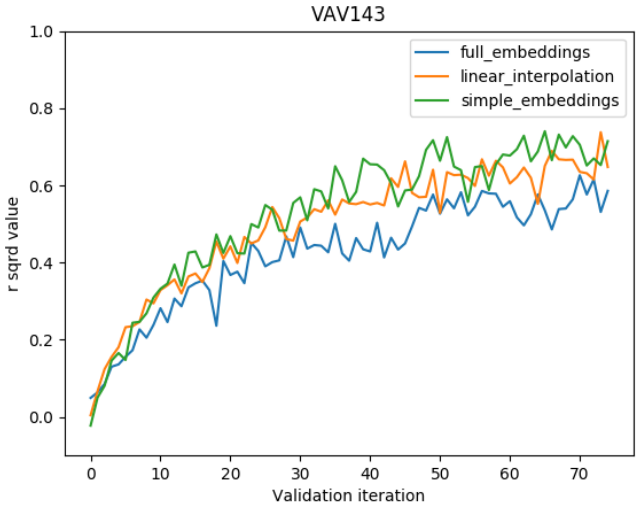
(d)

Figure 4.6. Randomly selected sequences of Wavenet temperature predictions vs ground truth.

4.5.4 Handling Missing Values

Here we compare our three methods of dealing with missing values (described in data preparation): linear interpolation, “simple embeddings”, and “full embeddings”. We compare these methods by training three distinct airflow forecasting models on separate zones and comparing R^2 values. In addition to this we perform some qualitative analysis of predictions made by these methods.

Figure 4.7 demonstrates that the full embeddings perform generally worse than the simple embeddings or linear interpolation.



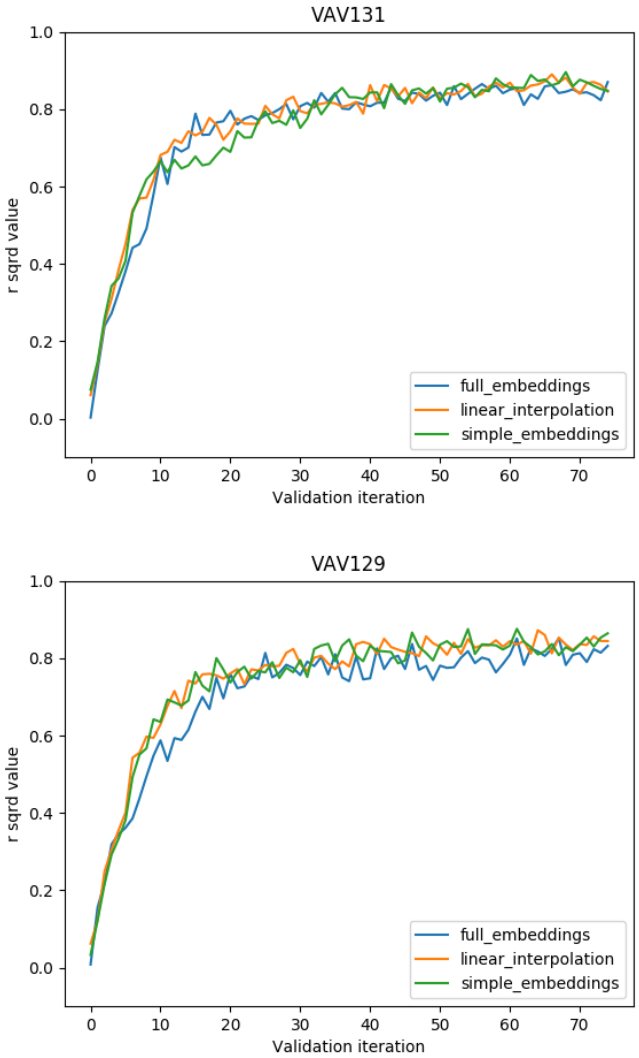


Figure 4.7. R^2 Comparisons with different missing value handling.

Of note in Figure 4.8 is the noise present in the full embeddings predictions. This noise is present in all tested zones and makes the use of full embeddings less desirable. We predict that this noise is caused by variables that are similar in value having very different embeddings. This could cause the model to behave inconsistently while receiving only small changes in a features value, possibly explaining the noise in prediction. Overall it seems that having to learn embeddings makes the task of predicting more difficult, leading us to believe that linear interpolation and our simple embeddings are superior. We wish to further explore these results by: testing on more zones, predicting temperature, and varying the number of missing data points.

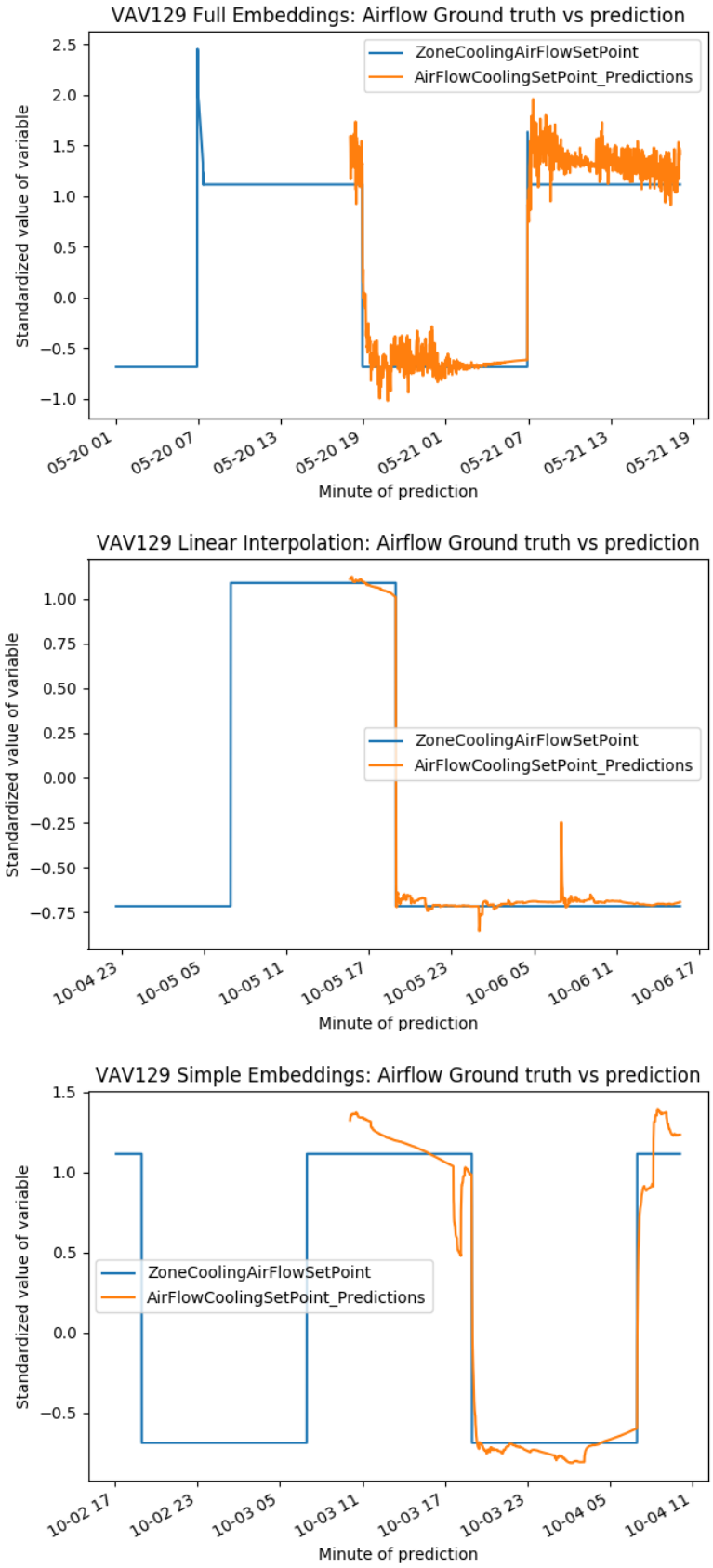


Figure 4.8. Sample predictions.

5.0 Conclusions and Future Work

The work described in this report demonstrates the effective use of sequence-to-sequence RNN models for predicting electrical load in commercial buildings in which outdoor temperature and temporal features are the driving conditions. In addition to this we demonstrated that the Wavenet architecture is effective at predicting both the zone airflow and the zone air temperature.

5.1 Whole Building Short-Term Load Forecasting

We found intuitive heuristics for selecting the number of training months and start month to produce performant models: starting training in the middle of a heating or cooling season, then training for at least six months. We further showed that models perform best when predictions are conditioned on three to 12 hours of prior data, with a decrease in performance for shorter and longer contexts. In directed hyperparameter search experiments, we identified recommended ranges for common hyperparameters that could be used by practitioners applying similar models to their own tasks. Finally, we found that transferability of models across buildings is highly dependent on the building pairs, but in the best case, models are highly transferable.

As currently configured, our recurrent models have difficulty with generating forecasts longer than one timestep into the future. Further, conditioning the models on longer input does not seem to improve performance beyond 12 hours. One possible way to improve our model's performance on longer output sequences and make better use of longer inputs would be by incorporating an attention mechanism into the decoder. Attention mechanisms in sequence transduction models compute a weighted average of the encoder’s hidden states as a function of the current hidden state of the decoder and then use this representation to condition the output of the decoder. This allows the model to learn to “attend” to different regions of the input sequence that are most relevant to the decoder's current state (Bahdanau et al. 2014). As discussed in prior work, recent literature has increasingly made use of CNNs for sequence modeling; this is a result of their ability to model longer sequences than RNNs, their improved time complexity, and parameter efficiency. Exploring the use of temporal convolutional networks (Bai et al. 2018) for our forecasting task is one promising future direction.

5.2 Zone Airflow and Temperature Forecasting

Our approach is moderately effective at predicting zone airflow and zone temperature forecasting. We show that a deep neural network architecture (Wavenet) employed in this report outperforms an empirical method previously developed by PNNL for airflow prediction (the ROMs). The Wavenet model has lower error and is more usable as it is only trained once on the entire zone. The ROM approach, as currently utilized, must be trained new for each week of prediction.

We showed promising initial results for temperature prediction on RTU zones. Further, we explored methods of compensating for missing data, finding several successful methods.

We plan to deepen our understanding of the missing value handling methods in the future by performing experiments with an increasing percentage of missing data. Evaluating the performance of these experiments will allow us to draw more definitive conclusions about the effectiveness of these methods. We plan to configure the ROM approach to perform

temperature predictions so that we can evaluate our Wavenet temperature prediction performance against it. Hyperparameter tuning is a part of our upcoming future work as well, it is of great importance for increasing both zone temperature and zone airflow prediction performance. Finally, we intend to explore the effectiveness of our approach when training and evaluating multiple zones.

6.0 References

Alizadeh M, A Scaglione, A Applebaum, G Kesidis, K Levitt. 2015. “Reduced-order load models for large populations of flexible appliances.” *IEEE Transactions on Power Systems* 30(4):1758–1774.

Amarasinghe K, DL Marino, M Manic. 2017. “Deep neural networks for energy load forecasting.” In Industrial Electronics (ISIE), 2017 IEEE 26th International Symposium on Industrial Electronics, 1483-1488.

Bahdanau D, K Cho, Y Bengio. 2014. “Neural machine translation by jointly learning to align and translate.” arXiv preprint arXiv:1409.0473.

Bai S, JZ Kolter, V Koltun. 2018. “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling.” arXiv preprint https://arxiv.org/abs/1803.01271.

Barrett E and S Linder. 2015. “Autonomous HVAC control, a reinforcement learning approach.” In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 3-19). Springer, Cham.

Bengio Y, P Simard, P Frasconi. 1994. “Learning long-term dependencies with gradient descent is difficult.” *IEEE Transactions on Neural Networks* 5(2):157-166.

Bengio S, O Vinyals, N Jaitly, N Shazeer. 2015. “Scheduled sampling for sequence prediction with recurrent neural networks.” *Advances in Neural Information Processing Systems* pp.1171-1179.

Bishop CM. 2006. *Pattern Recognition and Machine Learning*. Springer.

Braun MR, H Altan, SBM Beck. 2014. “Using regression analysis to predict the future energy consumption of a supermarket in the UK.” *Applied Energy* 130:305-313.

Chae YT, R Horesh, Y Hwang, YM Lee. 2016. “Artificial neural network model for forecasting sub-hourly electricity usage in commercial buildings.” *Energy and Buildings* 111:184-194.

Chan W, N Jaitly, QV Le, O Vinyals. 2015. “Listen, attend and spell.” arXiv preprint arXiv:1508.01211.

Che Z, S Purushotham, K Cho, D Sontag, Y Liu. 2018. “Recurrent neural networks for multivariate time series with missing values.” *Scientific Reports* 8(1):6085.

Chen Y, V Chandan, Y Huang, MJE Alam, O Ahmed, L Smith. 2019. “Coordination of Behind-the-Meter Energy Storage and Building Loads: Optimization with Deep Learning Model.” Proceedings of the Tenth ACM International Conference on Future Energy Systems, ACM, 2019, pp. 492–499. ACM Digital Library, doi:10.1145/3307772.3331025.

Cho K, B Van Merriënboer, D Bahdanau, Y Bengio. 2014. “On the properties of neural machine translation: Encoder-decoder approaches.” arXiv preprint arXiv:1409.1259.

Chung J, Gulcehre C, Cho K, Bengio Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

Chung JS and A Zisserman. 2016. “Lip reading in the wild.” In Asian Conference on Computer Vision (pp. 87-103). Springer, Cham.

Collobert R, J Weston, L Bottou, M Karlen, K Kavukcuoglu, P Kuksa. 2011. “Natural language processing (almost) from scratch.” *Journal of Machine Learning Research* 12(Aug):2493-2537.

Deb C, LS Eang, J Yang, M Santamouris. 2016. “Forecasting diurnal cooling energy load for institutional buildings using artificial neural networks.” *Energy and Buildings* 121:284-297.

Donahue J, L Anne Hendricks, S Guadarrama, M Rohrbach, S Venugopalan, K Saenko, T Darrell. 2015. "Long-term recurrent convolutional networks for visual recognition and description." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2625-2634.

Dong B, SE Lee, MH Sapar. 2005a. “A holistic utility bill analysis method for baselining whole commercial building energy consumption in Singapore.” *Energy and Buildings* 37(2): 167–174.

Dong B, C Cao, SE Lee. 2005b. “Applying support vector machines to predict building energy consumption in tropical region.” *Energy and Buildings* 37(5):545–553.

Espinoza M, C Joye, R Belmans, B De Moor. 2005. “Short-term load forecasting, profile identification, and customer segmentation: A methodology based on periodic time series.” *IEEE Transactions on Power Systems* 20(3):1622-1630.

Fan C, F Xiao, S Wang. 2014. “Development of prediction models for next-day building energy consumption and peak power demand using data mining techniques.” *Applied Energy* 127:1-10.

Fan C, F Xiao, Y Zhao. 2017. “A short-term building cooling load prediction method using deep learning algorithms.” *Applied Energy* 195:222-233.

Fontugne R, J Ortiz, D Culler, H Esaki. 2012. “Empirical mode decomposition for intrinsic-relationship extraction in large sensor deployments.”

Fontugne R, J Ortiz, N Tremblay, P Borgnat, P Flandrin, K Fukuda, D Culler, H Esaki. 2015. “Strip, bind, and search: A method for identifying abnormal energy consumption in buildings.” 12th International Conference on Information Processing in Sensor Networks, pp. 129–140.

Gao J and R Jamidar. 2014. “Machine learning applications for data center optimization.” Google White Paper.

Goodfellow IJ, D Warde-Farley, M Mirza, A Courville, Y Bengio. 2013. “Maxout networks.” arXiv preprint arXiv:1302.4389.

Gisler C, A Ridi, D Zufferey, OA Khaled, J Hennebert. 2013. “Appliance consumption signature database and recognition test protocols.” In Systems, Signal Processing and their Applications (WoSSPA), 2013 8th International Workshop on. IEEE, 2013, pp. 336–341.

GWAC (GridWise Architecture Council). 2019. https://www.gridwiseac.org/about/transactive_energy.aspx. Last visited May 2019.

Hansen N. “The CMA Evolution Strategy.” 2016. http://cma.gforge.inria.fr.

He K., X Zhang, S Ren, J Sun. 2016. “Deep residual learning for image recognition.” In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, p. 770-778.

Herrmann V. 2017. Vincentherrmann/Pytorch-Wavenet. 2017. 2019. GitHub, https://github.com/vincentherrmann/pytorch-wavenet.

Hinton G, L Deng, D Yu, G Dahl, AR Mohamed, N Jaitly, A Senior, V Vanhoucke, P Nguyen, B Kingsbury, T Sainath. 2012. “Deep neural networks for acoustic modeling in speech recognition.” *IEEE Signal Processing Magazine*, 29.

Hochreiter S and J Schmidhuber. 1997. “Long short-term memory.” *Neural Computation* 9(8):1735-1780.

Huang H, L Chen, E Hu. 2015. “A neural network-based multi-zone modelling approach for predictive control system design in commercial buildings.” *Energy and Buildings* 97:86-97.

Huang H, L Chen, M Mohammadzaheri, E Hu, M Chen. 2013. “Multi-zone temperature prediction in a commercial building using artificial neural network model.” In Control and Automation (ICCA), 2013 10th IEEE International Conference, pp. 1896-1901.

Huang H, L Chen, M Mohammadzaheri, E Hu. 2012. “A new zone temperature predictive modeling for energy saving in buildings.” *Procedia Engineering* 49:142-151.

Jean S, K Cho, R Memisevic, Y Bengio. 2014. “On using very large target vocabulary for neural machine translation.” arXiv preprint arXiv:1412.2007, 2014.

Jordan MI and TM Mitchell. 2015. “Machine learning: Trends, perspectives, and prospects.” *Science* 349(6245):255-260.

Kalogirou S, C Neocleous, C Schizas. 1997. “Building heating load estimation using artificial neural networks.” In Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques, vol. 8, p. 14.

Karpathy A and F-F Li. 2015. "Deep visual-semantic alignments for generating image descriptions." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Katipamula S, DP Chassin, DD Hatley, RG Pratt, DJ Hammerstrom. 2003. *Transactive Controls: Market-Based GridWise™ Controls for Building Systems*. PNNL-15921, Pacific Northwest National Laboratory, Richland, Washington.

Kelly J, and W Knottenbelt. 2015. “Neural NILM: Deep neural networks applied to energy disaggregation.” In Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments, pp. 55–64.

Kingma DP and J Ba. 2014. “Adam: A method for stochastic optimization.” arXiv preprint arXiv:1412.6980.

Kolter JZ and MJ Johnson. 2011. “Redd: A public data set for energy disaggregation research.” in Workshop on Data Mining Applications in Sustainability (SIGKDD), San Diego, CA, vol. 25, 2011, pp. 59–62.

Kreider JF, DE Claridge, PP Curtiss, RR Dodier, JS Haberl, MM Krarti. 1995. “Building energy use prediction and system identification using recurrent neural networks”. *ASME. J. Sol. Energy Eng* 117(3):161-166. doi:10.1115/1.2847757.

Krizhevsky A, I Sutskever, and GE Hinton. 2012. “Imagenet classification with deep convolutional neural networks.” In Advances in Neural Information Processing Systems, pp. 1097-1105.

Kumar R, R Aggarwal, J Sharma. 2013. “Energy analysis of a building using artificial neural network: A review.” *Energy and Buildings* 65:352–358.

LeCun Y, Y Bengio, G Hinton. 2015. “Deep learning.” *Nature* 521(7553):436-444.

Le Paine T, P Khorrami, S Chang, Y Zhang, P Ramachandran, MA Hasegawa-Johnson, TS Huang. 2016. “Fast Wavenet Generation Algorithm.” ArXiv:1611.09482 [Cs], Nov. 2016. arXiv.org, http://arxiv.org/abs/1611.09482.

Li Y, Y Wen. K Guan, D Tao. 2017. “Transforming cooling optimization for green data center via deep reinforcement learning.” arXiv preprint arXiv:1709.05077.

Lundin M, S Andersson, R Östin. 2004. “Development and validation of a method aimed at estimating building performance parameters.” *Energy and Buildings* 36(9):905-914.

Ma J, J Qin, T Salsbury, P Xu. 2012. “Demand reduction in building energy systems based on economic model predictive control.” *Chemical Engineering Science* 67(1):92–100.

Macas M, F Lauro, F Moretti, S Pizzuti, M Annunziato, A Fonti, G Comodi, A Giantomassi. 2014. “Sensitivity based feature selection for recurrent neural network applied to forecasting of heating gas consumption.” In International Joint Conference SOCO?14-CISIS?14-ICEUTE?14. Springer, pp. 259–268.

Marasco DE and CE Kontokosta. 2016. “Applications of machine learning methods to identifying and predicting building retrofit opportunities.” *Energy and Buildings* 128: 431–441.

Marino DL, K Amarasinghe, M Manic. 2016. “Building energy load forecasting using deep neural networks. In Industrial Electronics Society, IECON 2016-42nd Annual Conference of the IEEE, pp. 7046-7051).

Mateo F., JJ Carrasco, A Sellami, M Millán-Giraldo, M Domínguez, E Soria-Olivas. 2013. “Machine learning methods to forecast temperature in buildings.” *Expert Systems with Applications* 40(4):1061-1068.

Mathieu JL, S Koch, DS Callaway. 2013. “State estimation and control of electric loads to manage real-time energy imbalance.” *IEEE Transactions on Power Systems* 28(1): 430–440

Mechaqrane A and M Zouak. 2004. “A comparison of linear and neural network ARX models applied to a prediction of the indoor temperature of a building.” *Neural Computing & Applications* 13(1):32-37.

Mnih V, H Larochelle, GE Hinton. 2012. “Conditional restricted Boltzmann machines for structured output prediction.” arXiv preprint arXiv:1202.3748.

Mnih V, K Kavukcuoglu., D Silver, A Graves, I Antonoglou, D Wierstra, M Riedmiller. 2013. “Playing Atari with deep reinforcement learning.” arXiv preprint arXiv:1312.5602.

Mocanu E, PH Nguyen, M Gibescu, WL Kling. 2014. “Comparison of machine learning methods for estimating energy consumption in buildings.” in Probabilistic Methods Applied to Power Systems (PMAPS), 2014 International Conference. IEEE, 2014, pp. 1–6.

Monacchi A, D Egarter, W Elmenreich, S D’Alessandro, AM Tonello. 2014. “Greend: An energy consumption dataset of households in Italy and Austria.” In Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference, 2014, pp. 511–516.

Mustafaraj G, G Lowry, J Chen. 2011. “Prediction of room temperature and relative humidity by autoregressive linear and nonlinear neural network models for an open office.” *Energy and Buildings* 43(6):1452-1460.

Olofsson T and S Andersson. 2002. “Overall heat loss coefficient and domestic energy gain factor for single-family buildings.” *Building and Environment* 37(11):1019-1026.

Pan SJ and Q Yang. 2010. “A survey on transfer learning.*” IEEE Transactions on Knowledge and Data Engineering* 22(10):1345-1359.

Pham V, T Bluche, C Kermorvant, J Louradour. 2014. “Dropout improves recurrent neural networks for handwriting recognition.” In Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference, pp. 285-290).

Priyadarsini R, W Xuchao, LS Eang. 2009. “A study on energy performance of hotel buildings in Singapore.” *Energy and Buildings* 41(12):1319–1324.

Sainath TN, AR Mohamed, B Kingsbury, B Ramabhadran. 2013. “Deep convolutional neural networks for LVCSR.” In Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference, pp. 8614-8618).

Socher R, A Perelygin, J Wu, J Chuang, CD Manning, A Ng, Cr Potts. 2013. "Recursive deep models for semantic compositionality over a sentiment treebank." In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 1631-1642.

Srivastava N, G Hinton, A Krizhevsky, I Sutskever, R Salakhutdinov. 2014. “Dropout: a simple way to prevent neural networks from overfitting.” *The Journal of Machine Learning Research* 15(1):1929-1958.

Sutskever I, O Vinyals, QV Le. 2014. "Sequence to sequence learning with neural networks." *Advances in Neural Information Processing Systems*, pp. 3104-3112.

Radecki P and B Hencey. 2013. “Online thermal estimation, control, and self-excitation of buildings.” in 52nd IEEE Conference on Decision and Control. IEEE, 2013, pp. 4802–4807.

Rawal G. 2019. Costs, Savings, and ROI for Smart Building Implementation, blog, viewed May 2019, https://blogs.intel.com/iot/2016/06/20/costs-savings-roi-smart-building-implementation/.

Rubio-Herrero J, V Chandan, C Siegel, A Vishnu, D Vrabie. 2017. “A learning framework for control-oriented modeling of buildings.” In Proceedings of IEEE International Conference on Machine Learning and Applications.

Ruelens F, S Iacovella, BJ Claessens, R Belmans. 2015. “Learning agent for a heat-pump thermostat with a set-back strategy using model-free reinforcement learning.” *Energies* 8(8):8300-8318.

Shi H, M Xu, and R Li. 2017. “Deep learning for household load forecasting–A novel pooling deep RNN.” *IEEE Transactions on Smart Grid*.

Simonyan K and A Zisserman. 2014. “Very deep convolutional networks for large-scale image recognition.” arXiv preprint arXiv:1409.1556.

Smith RG. 1980. “The Contract Net Protocol: High-Level Communication and Control in Distributed Problem Solver.” *IEEE Transactions on Computers* C-29(12):1104-1113.

Tang D, B Qin, T Liu. 2015. “Document modeling with gated recurrent neural network for sentiment classification.” In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1422-1432.

Taylor JA and JL Mathieu. 2014. “Index policies for demand response.” *IEEE Transactions on Power Systems* 29(3):1287–1295.

Taylor JA, and JL Mathieu. 2015. “Uncertainty in demand response – Identification, estimation, and learning.” *Tutorials in Operations Research*. https://doi.org/10.1287/educ.2015.0137

Taylor GW, GE Hinton, ST Roweis. 2011. “Two distributed-state models for generating high-dimensional time series.” *Journal of Machine Learning Research* 12(March):1025–1068.

Tieleman T and G Hinton. 2012. Lecture 6.5-rmsprop: “Divide the gradient by a running average of its recent magnitude.” *COURSERA: Neural Networks for Machine Learning* 4(2):26-31.

Tompson JJ, A Jain, Y LeCun, C Bregler. 2014. “Joint training of a convolutional network and a graphical model for human pose estimation.” In Advances in Neural Information Processing Systems, pp. 1799-1807.

Trindler J, R Zwiker, U Rutishauser, R Douglas, J Joller. 2003. “Discovery of logical structures in a multisensor environment based on sparse events.” 8th National Congress of Italian Association for Artificial Intelligence, pp. 1–11.

van den Oord A, S Dieleman, H Zen, K Simonyan, O Vinyals, A Graves, N Kalchbrenner, A Senior, K Kavukcuoglu. “WaveNet: A Generative Model for Raw Audio.” ArXiv:1609.03499 [Cs], Sept. 2016. arXiv.org, http://arxiv.org/abs/1609.03499.

Wei T, Y Wang, and Q Zhu. 2017. “Deep reinforcement learning for building HVAC control.” In Proceedings of the 54th Annual Design Automation Conference 2017, p. 22. ACM.

Zhou Q, S Wang, X Xu, F Xiao. 2008. “A grey-box model of next-day building thermal load prediction for energy-efficient control.” *International Journal of Energy Research* 32(15):1418-1431.

Additional Bibilography

Djukanovic M, B Babic, DJ Sobajic, YH Pao. 1993. “Unsupervised/supervised learning concept for 24-hour load forecasting” In IEE Proceedings C (Generation, Transmission and Distribution), Vol. 140, IET, p. 311–318.

Gasparin A, S Lukovic, C Alippi. 2019. “Deep learning for time series forecasting: The electric load case.” arXiv preprint arXiv:1907.09207.

Hooshmand A and R Sharma. 2019. “Energy predictive models with limited data using transfer learning.” In Proceedings of the Tenth ACM International Conference on Future Energy Systems, p. 12–16.

Kim K. 2018. “WaveNet: A Network Good to Know.” Medium, 6 Oct. 2018, https://medium.com/@kion.kim/wavenet-a-network-good-to-know-7caaae735435.

Kim S, S Kang, KR Ryu, G Song. 2019. “Real-time occupancy prediction in a large exhibition hall using deep learning approach.” Energy and Buildings 199:216–222.

Kuo PH and CJ Huang. “A high precision artificial neural networks model for short-term energy load forecasting.” Energies 11(1):213.

Ribeiro M, K Grolinger, HF ElYamany, WA Higashino, MA Capretz. 2018. “Transfer learning with seasonal and trend adjustment for cross-building energy forecasting.” Energy and Buildings 165:352–363.

Pacific Northwest National Laboratory

902 Battelle Boulevard
P.O. Box 999
Richland, WA 99354
1-888-375-PNNL (7665)

www.pnnl.gov