Proceedings of the 53rd Hawaii International Conference on System Sciences | 2020

Improving End-Use Load Modeling Using Machine Learning and Smart Meter Data

Brandon Thayer
PNNL
brandon.thayer@pnnl.gov

Dave Engel
PNNL
dave.engel@pnnl.gov

Indrasis Chakraborty
PNNL
indrasis.chakraborty@pnnl.gov

Kevin Schneider
PNNL
kevin.schneider@pnnl.gov

Leslie Ponder
Duke Energy
Leslie.Ponder@duke-energy.com

Kevin Fox
Duke Energy
Lance.Fox@duke-energy.com

Abstract

An accurate representation of the voltage-dependent, time-varying energy consumption of end-use electric loads is essential for the operation of modern distribution automation (DA) schemes. Volt-var optimization (VVO), a DA scheme which can decrease energy consumption and peak demand, often leverages electric network models and power flow results to inform control decisions, making it sensitive to errors in load models. End-use load modeling can be improved with additional measurements from advanced metering infrastructure (AMI). This paper presents two novel machine learning algorithms for creating data-driven, time-varying load models for use with DA technologies such as VVO. The first algorithm uses AMI data, k-means clustering, and least-squares optimization to create predictive load models for individual electric customers. The second algorithm uses deep learning (via a convolution-based recurrent neural network) to incorporate additional data and increase model accuracy. The improved accuracy of the load models for both algorithms is validated through simulation.

Nomenclature

$+\!\!+_{k=1}^{N}$	Operator for row-wise concatenation of N row vectors of length x. Output dimension: $[N, x]$.
b_s	Batch size of deep network training.
d	Length of historic data window.
ϵ	Deep network training error tolerance.
f_k	Solar flux at interval k (W/ft.^{2}).
$\mathbf{f}$	$:= [f_1, f_2, \ldots, f_k, f_{k+1}, \ldots, f_N]$
$\mathcal{F}^i$	Neural network functional representation at step i.
F_c, F_p	Filter size of convolution, pool layer.
$I_\%$	Current fraction of ZIP load model.
I_θ	Current phase angle of ZIP load model.
k	Time interval index. $k \in (1, 2, \ldots, N)$.
n	Number of clusters used in k-means clustering.
$P_\%$	Power fraction of ZIP load model.
P_θ	Power phase angle of ZIP load model.
P_k	Load active power at interval k (W).
$\mathbf{P}$	$:= [P_1, P_2, \ldots, P_k, P_{k+1}, \ldots, P_N]$.
$\hat{P}_k$	Predicted active power at interval k (W).
Q_k	Load reactive power at interval k (var).
$\mathbf{Q}$	$:= [Q_1, Q_2, \ldots, Q_k, Q_{k+1}, \ldots, Q_N]$.
$\hat{Q}_k$	Predicted reactive power at interval k (var).
T_{a_k}	Dry bulb temperature at interval k (°F).
$\mathbf{T_a}$	$:= [T_{a_1}, T_{a_2}, \ldots, T_{a_k}, T_{a_{k+1}} \ldots, T_{a_N}]$.
u_k	Neural network input, either P_k or Q_k.
V_n	Load nominal voltage magnitude (V).
V_k	Load voltage magnitude at interval k (V).
$\mathbf{V}$	$:= [V_1, V_2, \ldots, V_k, V_{k+1}, \ldots, V_N]$
$\mathbf{v}_k^1$	Neural network row vector for 1st step. $:= [u_k, V_k, T_{a_k}, f_k, \ldots, u_{k+d-1}, V_{k+d-1}, T_{a_{k+d-1}}, f_{k+d-1}]$
$\mathbf{v}_k^2$	Neural network row vector for 2nd step. $:= [\mathcal{F}^1(\mathbf{v}_k), V_k, T_{a_k}, f_k, \ldots, \mathcal{F}^1(\mathbf{v}_{k+d-1}), V_{k+d-1}, T_{a_{k+d-1}}, f_{k+d-1}]$
$\mathbf{X}^i$	$:= +\!\!+_{k=1}^{N}\mathbf{v}_k^i$. Neural network input for step i.
$\mathbf{y}$	$:= [P_1, P_2, \ldots, P_N, Q_1, Q_2, \ldots, Q_N]$
$\hat{\mathbf{y}}$	$:= [\hat{P}_1, \hat{P}_2, \ldots, \hat{P}_N, \hat{Q}_1, \hat{Q}_2, \ldots, \hat{Q}_N]$
$\mathbf{Y}^i$	Neural network output for step i.
$Z_\%$	Impedance fraction of ZIP load model.
Z_θ	Impedance phase angle of ZIP load model.

The Pacific Northwest National Laboratory (PNNL) is operated by Battelle for the U. S. Department of Energy (DOE) under contract DE-AC05-76RL01830. This work was funded by DOE's Office of Electricity as part of the development of GridAPPS-D™: a standards-based platform for advanced distribution system application development [1].

1. Introduction

End-use load models are an essential element of distribution planning and many operational systems [2, 3]. Volt-var optimization (VVO) is one such operational distribution automation (DA) system that is dependent on accurate end-use models to reduce annual energy consumption, reduce peak power demand, and to manage power factor. Early VVO deployments relied on local measurements (voltage and/or current) at voltage regulators, shunt capacitors, and remote end-of-line points [4, 5]. Many modern VVO applications utilize measurements in conjunction with models of both the electric network and end-use loads in order to improve VVO performance [6–8].[1] However, model-based schemes are sensitive to the accuracy of both their network and load models, and the load models often have a high level of uncertainty. Model-based VVO schemes typically use a simple load allocation where the feeder peak load, service transformer nameplate ratings, and a limited number of system measurements are used to estimate power demand for each end-use load. The resulting demand for each load is combined with the network topology and then used in a power flow calculation to determine the voltage at each point on the feeder [9]. The sparse set of measurements in the load allocation, and the resulting poor load estimates, can lead to large errors in the power flow solution.

Due to errors in load models, phenomena such as the conservation voltage reduction (CVR) may not be correctly quantified or fully leveraged by VVO systems [10], impacting the ability to reduce annual energy consumption and peak demand, two of the largest business drivers for VVO systems [8]. The use of customer "smart" electric meter data from advanced metering infrastructure (AMI) systems can improve the performance of model-based VVO schemes, and other DA systems that rely on accurate load models [11].

Since the performance of technologies such as VVO are dependent on distribution network parameters, topology, and load, current state-of-the-art model-based applications use voltage-dependent load models [6, 8]. Typically, a small set of models is derived for each customer class (e.g., industrial, single-family residential, etc.). These models may be changed seasonally and/or for weekdays vs. weekends. However, load composition varies on smaller timescales and depends on a variety of factors, such as ambient temperature [12]. For example, the voltage-dependent energy consumption of a customer's composite load will be different when an air conditioner is running versus when it is not. Additionally, load composition varies between customers within a class. When it is cold, the energy consumption of a large residential home with a natural gas furnace will respond differently to voltage variations than an apartment with resistive heating. The deployment of new sensors and systems, such as AMI, provide additional sources of information to develop voltage-dependent and time-varying end-use load models to support operations.

There is significant work in the literature investigating how to integrate new sources of sensor data with the goal of improving load models and/or system operation. In [13], AMI voltage magnitude data is used for a model-based VVO system, but not to generate load models. The authors of [6] vary ZIP load model[2] coefficients seasonally for their model-based VVO system and note that improved load modeling could improve VVO performance. Load decomposition from 15-minute AMI data is described in [14], and the challenges of using such a coarse data resolution are discussed. In both [15] and [16], customer demand is disaggregated into individual appliances, and then ZIP load models are fit for each assumed appliance. In [15], these appliance models are then used in a model-based VVO scheme. In [17], the need for data-driven, time-varying load models for distribution system applications such as VVO is discussed. The work uses AMI power and voltage measurements offline to create ZIP load models for individual buildings for different time ranges. While the work in the literature has shown that improved load models can increase DA/VVO performance, the majority of existing methods are not able to integrate AMI data to create near-real-time, frequently-updated, predictive, per-customer load models.

This paper presents two independent and novel algorithms for creating data-driven, time-varying ZIP load models for individual electric customers.

[1]Electric distribution feeders in North America are typically operated radially, but the term "network" is often used when describing feeder topology and attributes of connected elements (e.g., line impedance).

[2]ZIP load models are comprised of constant impedance (Z), constant current (I), and constant power (P) components and are discussed in detail in Section 3.

The first algorithm is light-weight from both a data and computation perspective and could readily be deployed to a modern DA system. The second algorithm provides more accurate predictions, but requires more data, extensive training, and significant computation power, making deployment on current DA systems more challenging. Both algorithms advance the state of the art for modeling and forecasting individual customer electric demand and provide better predictions than traditional load allocations, resulting in improved DA/VVO performance. The first algorithm uses *k*-means clustering [18], least-squares optimization [19], and nominally two weeks of historical data to create predictive load models, while the second leverages neural networks and deep learning to incorporate additional data into the model formulation [20,21]. Both algorithms use historic 15-minute average active power (**P**), reactive power (**Q**), and voltage magnitude (**V**) measurements from each customer meter, joined with feeder-level temperature ($\mathbf{T_a}$) and solar flux (**f**). These data are used to form data-driven, time-varying, predictive ZIP load models which can be used to obtain more accurate power flow results. Since operational DA technologies such as VVO often solve the power flow repeatedly while searching for an optimal solution, improved load models can help these technologies get closer to the true optimal solution.

The rest of the paper is organized as follows: Section 2 presents the meter data used by the two algorithms, Section 3 presents the ZIP load model, Section 4 presents the two algorithms in detail, Section 5 provides experimental results for both algorithms, and Section 6 concludes the paper and discusses future work.

2. Synthetic Meter Data

The work presented in this paper uses synthetic meter data created with the GridLAB-D™ simulation environment [22]. Annual time-series simulations are conducted using feeder "R2-12.47-2" from a publicly available set of prototypical distribution feeders [10], [23]. This prototypical feeder is representative of feeders in the North Central/Eastern U.S., has a nominal voltage of 12.47 kV, and represents a moderately-populated suburban area. The model uses detailed multi-state end-use load models, and the one-year time-series simulation (which generates meter data) is run with a 30-second minimum simulation time step.

In the simulation, a total of 898 individually metered residences are modeled in detail. The thermal properties of buildings and their heating, ventilation, and air-conditioning (HVAC) systems are represented by an equivalent thermal parameter (ETP) model to approximate HVAC response (electrical and thermal) to voltage, solar gains, temperature, humidity, and thermostatic set points [10]. The simulation's weather is driven by typical meteorological year (TMY) data [24]. Beyond the HVAC ETP models, each residence has additional end-use loads which are modeled as multi-state water heaters and a collection of ZIP loads, all of which operate on time-varying schedules. To represent variations between customers, all HVAC ETP, water heater, and ZIP load parameters are varied while ensuring total feeder load closely matches real-world, head-of-feeder power measurements. In addition to representing realistic feeder behavior, this modeling process provides representative models of individual loads [10].

In this work, results focus on a selection of 16 representative electric customers. These 16 customers were selected for their collective diversity of underlying device types, building models (size, insulation properties, etc.), and locations on the feeder. Table 1 presents details for four customers for which detailed results are presented later on.

Table 1. Details for Selected Customers

Meter	Housing Type	Cooling	Heating	EWH*	PP**
1	Single Family	Electric	Gas	N	N
3	Single Family	Electric	Gas	N	N
9	Apartment	Electric	Resistance	N	N
13	Single Family	None	Gas	Y	N

*EWH: Electric Water Heater; **PP: Pool Pump

Each customer's meter is assumed to be capable of measuring average active power, reactive power, and voltage magnitude on a 15-minute basis: a common AMI interval [25, 26]. In this work, it is assumed that the AMI data are available for use immediately after each interval; future work will investigate the effects of delayed and/or noisy measurements. The same feeder-level dry bulb temperature ($\mathbf{T_a}$) and solar flux (**f**) is used for all customers.

3. End-use Load Modeling

The ZIP load model represents an electric load's power consumption as a function of terminal voltage magnitude, representing the load as the sum of constant impedance (Z), current (I), and power (P) elements [27]. ZIP load models are available in most distribution simulators.

Oftentimes, ZIP model parameters are determined at the device level by varying input voltage, measuring active and reactive power consumption, and then performing a constrained curve-fit to determine model parameters [10]. In this work, 15-minute average **P**, **Q**, and **V** from simulated AMI data are used in the creation of time-varying ZIP models for each electric customer. The ZIP model equations are defined in (1)-(3).

$$P_k = S_n \left[\frac{V_k^2}{V_n^2} Z_{\%} \cos(Z_\theta) + \frac{V_k}{V_n} I_{\%} \cos(I_\theta) + P_{\%} \cos(P_\theta) \right] \quad (1)$$

$$Q_k = S_n \left[\frac{V_k^2}{V_n^2} Z_{\%} \sin(Z_\theta) + \frac{V_k}{V_n} I_{\%} \sin(I_\theta) + P_{\%} \sin(P_\theta) \right] \quad (2)$$

$$1 = Z_{\%} + I_{\%} + P_{\%} \quad (3)$$

To reduce the number of variables for algorithmic use, new variables are introduced in (4)-(6):

$$\bar{P} := \frac{P_k}{S_n},\ \bar{Q} := \frac{Q_k}{S_n},\ \bar{V} := \frac{V_a}{V_n} \quad (4)$$

$$a_1 := Z_{\%} \cos(Z_\theta),\ a_2 := I_{\%} \cos(I_\theta),\ a_3 := P_{\%} \cos(P_\theta) \quad (5)$$

$$b_1 := Z_{\%} \sin(Z_\theta),\ b_2 := I_{\%} \sin(I_\theta),\ b_3 := P_{\%} \sin(P_\theta) \quad (6)$$

The variables in (4)-(6) are substituted into (1)-(3) to derive the reduced ZIP equations shown in (7)-(9):

$$\bar{P} := \bar{V}^2 a_1 + \bar{V} a_2 + a_3 \quad (7)$$

$$\bar{Q} := \bar{V}^2 b_1 + \bar{V} b_2 + b_3 \quad (8)$$

$$\sqrt{a_1^2 + b_1^2} + \sqrt{a_2^2 + b_2^2} + \sqrt{a_3^2 + b_3^2} = 1 \quad (9)$$

4. Algorithms for Creating Data-Driven, Time-Varying ZIP Load Models

As previously discussed, historical 15-minute average **P**, **Q**, and **V** measurements from each of the 16 customer meters mentioned in Section 2 are used to derive a single, aggregate, predictive ZIP load model for each meter and each future time interval. ZIP models are used due to their ubiquitous presence in today's distribution simulators, the relative ease with which parameters can be computed, and the ease of which P and Q can be computed in a distribution simulation. Fifteen minute average $\mathbf{T_a}$ and **f** are used in the process. In the following sections, the two novel algorithms employed to create these ZIP models are discussed: *k*-means clustering coupled with least-squares optimization (KMLS), and deep learning (DL). Both algorithms make predictions every 15-minutes and are intended for use in near-real-time, power-flow-based DA applications, such as VVO. The KMLS algorithm requires significantly less data and computation time than the DL algorithm, but as discussed in Section 5, the DL algorithm can provide more accurate predictions.

4.1. Algorithm 1: *k*-Means Clustering and Least-Squares Optimization

Given a set of historical **P**, **Q**, and **V** measurements for a given meter, we solve for the set of variables a_1, a_2, a_3, b_1, b_2, and b_3 in (5)-(6) that minimizes the average sum of squares of the residual. This objective function is presented in (10), and is constrained by (9).

$$\frac{1}{N} \sum_{k=1}^{N} \left\{ \left[\bar{P}_k - \left(\bar{V}_k^2 a_1 + \bar{V}_k a_2 + a_3 \right) \right]^2 + \left[\bar{Q}_k - \left(\bar{V}_k^2 b_1 + \bar{V}_k b_2 + b_3 \right) \right]^2 \right\} \quad (10)$$

To derive a predictive ZIP model for an individual electric customer for the upcoming 15-minute interval, the KMLS algorithm works as follows:

1. The most recent two weeks of historic measurement data (**P**, **Q**, **V**, $\mathbf{T_a}$, and **f**) are obtained.
2. These data are then filtered to only include data with a similar day of the week (weekday vs. weekend), and a time of day that is ± 30 minutes from the prediction time in question.
3. Initialize $n = 1$ (number of clusters).
4. Filtered data are separated into n clusters via *k*-means clustering. Active and reactive power, dry bulb temperature, and solar flux are used as cluster features.

5. Present (non-historic) temperature and solar flux measurements are used to compute Euclidean distance to each of the n clusters. The cluster with the smallest Euclidean distance from present temperature and solar flux is considered best.

6. $\mathbf{P}$, $\mathbf{Q}$, and $\mathbf{V}$ data within this best cluster are then used in a sequential least-squares program to derive the ZIP model, where $S_n = \text{median}(|\mathbf{P} + j\mathbf{Q}|)$ for $\mathbf{P}$ and $\mathbf{Q}$ within the best cluster.

7. Data from the best cluster and the corresponding ZIP model parameters are substituted into the objective function (10), and the result is tracked.

8. Steps 4-7 are repeated, with $n = n + 1$. If the length of the vectors in the best cluster is below a predefined threshold (four in this work), no ZIP fit is performed. Looping terminates when n is sufficiently large such that no cluster can possibly have more measurements than the predefined threshold.

9. After looping terminates, the ZIP model that results in the minimal value of (10) is considered best and is used as the predictive model for this electric customer and upcoming time interval.

The sliding two-week data window in Step 1 helps avoid using older data which may not be representative of the present. The choice of using a two-week window was made through experimentation, but other window sizes are possible, impacting both prediction accuracy and computation (time, memory, etc.).

The minimum number of measurement sets per cluster constraint in Step 8 helps avoid using too little data. E.g., fitting a ZIP model to only two measurement sets will result in a very small value of (10), but likely won't result in accurate predictions. Increasing this minimum measurements value can also decrease computation time by decreasing loop iterations.

While the components that make up the KMLS algorithm are not new (e.g., least-squares optimization), the end-to-end algorithm is novel. The combination of historical AMI data, environmental data, clustering, and least-squares optimization for ZIP model creation can help real-time or near-real-time DA applications such as VVO obtain more accurate power flow results, thus leading to improved operation.

4.2. Algorithm 2: Deep Learning with Neural Networks

One shortcoming of the KMLS algorithm is that if running in a near-real-time environment, the amount of historic data used should be limited to keep computation time reasonable. Additionally, the input data to the algorithm may or may not be representative of the future (e.g., if there is an impending heat wave, but the previous two weeks were mild), which can lead to inaccurate predictions. With deep learning (DL) techniques, many combinations of conditions are examined in training (e.g., $\mathbf{P}, \mathbf{Q}, \mathbf{V}, \mathbf{T_a}, \mathbf{f}$), and the algorithm "learns" how to predict given current and past conditions. The use of more data along with intensive training can allow DL techniques to produce more accurate predictions.

In this work, a deep neural network comprised of multiple sub-networks is implemented. A convolutional neural network (CNN) is used for feature extraction [28], and a recurrent neural network (RNN) with long short-term memory (LSTM) cells [29] is subsequently used for feature memorization and prediction. Finally, a series of dense (DENSE) [21] layers transform the LSTM predictions into one-dimension. To our knowledge, this is the first use of a deep neural network to create time-varying, predictive ZIP load models for individual electric customers. The following subsections describe the network layers and overall architecture.

CNNs are specialized networks for processing grid-type data (e.g., time series or images) [28], where each network layer transforms its input volume through a differentiable function. For this work, two types of layers are used in the CNN: convolution (CONV) and pooling (POOL).

A CONV layer computes the output of neurons that are connected to local regions of the input (dimension $[b_s \times d \times 1]$ for a single feature). Each neuron computes a dot product between its weights and a small connected region in the input. The resulting value is then input to the scaled exponential linear unit (SELU) activation function [21]. The output of a CONV layer (dimension $[b_s \times d \times F_c]$) is passed to a POOL layer, which performs down-sampling. The resulting output is of dimension $[b_s \times \left(\frac{(d-F_p)}{F_p} + 1\right) \times F_c]$ [28].

After the CNN extracts features from the time series AMI data, the LSTM cell architecture from [29] memorizes the extracted features.

DENSE layers are layers where each neuron in the layer is connected to every neuron in the next layer [21]. Successive DENSE layers of decreasing size reduce the final output to one dimension.

The network layers previously discussed are used to model a_1, a_2, $\tan(Z_\theta)$, $\tan(I_\theta)$, and $\tan(P_\theta)$. Table 2 gives an ordered overview of the deep network architecture. The first input dimension, 2880, is the number of 15-minute intervals in a 30-day month. The second input dimension, 240, comes from using a data window of $d = 60$ with the four-featured vector $\mathbf{v}_k^1$.

Table 2. Deep Network Architecture for Modeling a_x and $\tan(\theta)$

Layer Type	Input Dimension	Output Dimension	Activation Function
CONV	$[2880 \times 240 \times 1]$	$[2880 \times 240 \times 200]$	SELU
POOL	$[2880 \times 240 \times 200]$	$[2880 \times 120 \times 200]$	N/A
CONV	$[2880 \times 120 \times 200]$	$[2880 \times 120 \times 100]$	SELU
POOL	$[2880 \times 120 \times 100]$	$[2880 \times 60 \times 100]$	N/A
CONV	$[2880 \times 60 \times 100]$	$[2880 \times 60 \times 50]$	SELU
POOL	$[2880 \times 60 \times 50]$	$[2880 \times 30 \times 50]$	N/A
CONV	$[2880 \times 30 \times 50]$	$[2880 \times 30 \times 25]$	SELU
LSTM	$[2880 \times 30 \times 25]$	$[2880 \times 40]$	N/A
DENSE	$[2880 \times 40]$	$[2880 \times 20]$	SELU
DENSE	$[2880 \times 20]$	$[2880 \times 10]$	SELU
DENSE	$[2880 \times 10]$	$[2880 \times 1]$	SELU (for a_x) tanh (for $\tan(\theta)$)

The architecture for $\tan(\theta)$ ensures that $\tan(\theta) \leq 1$ by using the tanh activation function in the last DENSE layer. In order to satisfy (9), a_3 is reformulated in (11):

$$a_3 = \frac{1 - a_1\sqrt{1+\tan^2(Z_\theta)} - a_2\sqrt{1+\tan^2(I_\theta)}}{\sqrt{1+\tan^2(P_\theta)}} \tag{11}$$

In (11), the square root results in a vanishing gradient problem in the deep network framework. To circumvent this, three terms of the binomial expansion of (11) are used to approximate a_3. Then, the a_x and $\tan(\theta)$ terms are used to derive the b_x terms given in (6).

The first step of training involves learning $\mathcal{F}^1$, using the input-output pair $(\mathbf{X}^1, \mathbf{Y}^1)$. Training runs until $\|\mathbf{Y}^1 - \mathcal{F}^1(\mathbf{X}^1)\|_2 \leq \epsilon$ Likewise, the second training step learns $\mathcal{F}^2$ using the input-output pair $(\mathbf{X}^2, \mathbf{Y}^2)$, and runs until $\|\mathbf{Y}^2 - \mathcal{F}^2(\mathbf{X}^2)\|_2 \leq \epsilon$. Each training step is conducted via batch processing with all relevant data for one month.

Since training is performed offline, but predictions are performed in near real time, each successive prediction relies on more predicted values in its input, $\mathbf{X}$ (specifically, $\hat{P}$ and $\hat{Q}$). The inclusion of predictions in the input leads to error accumulation, which is mitigated by the novel second training step. The full training and prediction process is described with an example:

Given historical data for months 1 and 2, predictions are made for month 3. The first step of training uses month 1's data, and the resulting network is used to formulate predictions for month 2. The second training step uses both the predicted and actual data for month 2 to train the network on how error is accumulated. The network then formulates predictions for month 3. In an operational setting, the network would need to be retrained monthly.

5. Results, Analysis, and Discussion

The goal of this work is to develop ZIP load modeling algorithms that use AMI and environmental data to create predictive models in near real time. The resulting models can then be used by power-flow-dependent DA applications.

As an initial validation step, both algorithms are given **V**, **P**, and **Q** for simple, static ZIP models. The resulting predicted active and reactive power for each model match the expected **P** and **Q**, validating the basic functionality of both algorithms. The results presented in this section focus on the algorithms' predictions for each individual customer's aggregate demand, rather than this simple validation procedure.

To evaluate both the KMLS and DL algorithms for individual customer aggregate demand, different ZIP models are derived for each of the 16 meters described in Section 2 for every 15-minute interval in the simulation year. The meter data for deriving the predictive ZIP models is purely historical, i.e. the training and testing data are kept strictly separate. For each prediction interval, $\hat{P}$ and $\hat{Q}$ are computed by substituting the actual V and derived ZIP model parameters into (1) and (2). These predicted power values are then compared with the actual power values from the meter data.

5.1. Definition of Evaluation Metrics

Table 3 describes the metrics used to evaluate the performance of the ZIP modeling algorithms [30]. Mean bias error (MBE), mean absolute error (MAE), root mean quadratic error (RMQE), root mean square error (RMSE), and accuracy (Acc) are all used. For accuracy, values closer to one are better. For all other metrics, closer to zero is better. For a single meter, $\mathbf{y}$ and $\hat{\mathbf{y}}$ are comprised of both active power (Watts) and reactive power (var) and have length $2N$, where N is the number of 15-minute intervals in a year.

Table 3. Performance metrics for load modeling algorithms

Metric	Equation	Description
MBE	$\frac{1}{2N}\sum_{k=1}^{2N}(\hat{\mathbf{y}}_k - \mathbf{y}_k)$	Assess prediction bias. Range: $[-\infty, \infty]$.
MAE	$\frac{1}{2N}\sum_{k=1}^{2N}\lvert\hat{\mathbf{y}}_k - \mathbf{y}_k\rvert$	Evaluate uniform prediction errors. Range: $[0, \infty]$.
RMQE	$\left(\frac{1}{2N}\sum_{k=1}^{2N}(\hat{\mathbf{y}}_k - \mathbf{y}_k)^4\right)^{0.25}$	Evaluate prediction accuracy, higher penalty for large errors than RMSE. Range: $[0, \infty]$.
RMSE	$\left(\frac{1}{2N}\sum_{k=1}^{2N}(\hat{\mathbf{y}}_k - \mathbf{y}_k)^2\right)^{0.5}$	Evaluate prediction accuracy while penalizing large errors. Range: $[0, \infty]$.
Acc	$\frac{1}{2N}\sum_{k=1}^{2N}\lvert\hat{\mathbf{y}}_k - \mathbf{y}_k\rvert / \mathbf{y}_k$	Mean absolute percent correct. Range: $[0, 1]$.

5.2. Aggregate Results Compared with Load Allocation

To compare the KMLS and DL algorithms with current modeling practices, a simple proxy for load allocation is implemented. Each of the 16 electric customers is represented by a ZIP model with static coefficients from [31]. For every meter and time interval, S_n is updated with the value computed for KMLS in Step 6 of Section 4.1, and then $\hat{P}$ and $\hat{Q}$ are computed. The use of the S_n term from the KMLS algorithm may result in increased modeling accuracy compared to current load allocation algorithms, which typically use a small set of feeder-level measurements and secondary transformer ratings to estimate power demand for each electric customer. Unfortunately, the load allocation algorithms used today are proprietary and built into distribution management systems, making them unavailable for true comparison.

Table 4 presents aggregated performance metrics for all meters. Here, $\mathbf{y}$ and $\hat{\mathbf{y}}$ contain all actual and predicted values for all meters for the entire year. Table 4 shows that the KMLS and DL algorithms outperform the simple load allocation method for all metrics.

Table 4. Combined performance metrics for all meters

Algorithm	MBE	MAE	RMQE	RMSE	Acc
Load Allocation	50	814	2619	1575	0.16
KMLS	-27	258	1892	776	0.82
DL	3	87	1836	403	0.91

5.3. KMLS and DL Algorithm Results

Fig. 1 visually compares both algorithms' performance metrics (defined in Table 3) for individual meters for all predictions over the entire simulation year. The KMLS results have a negative MBE for all meters except meter 13, indicative of the KMLS' tendency to miss new peaks due to its sliding two-week data window. The DL algorithm predictions result in a negative MBE for half of the meters and a positive MBE for the other half: for this subset of meters the DL algorithm does not have a strong tendency to underpredict or overpredict.

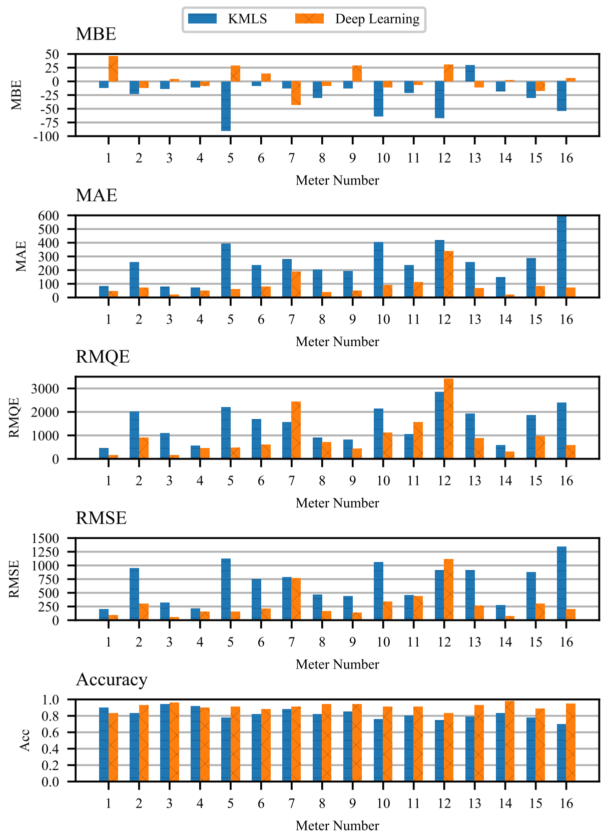

Figure 1. Metric comparison between KMLS and DL for one year of 15-minute predictions.

For MAE, Fig. 1 shows that DL performed better for all meters. However, examining RMQE and RMSE shows that KMLS outperforms DL for some meters. Note that for meter 7, the DL RMSE is slightly below that of KMLS, but DL's RMQE is significantly higher than KMLS'. This indicates DL had more large prediction errors than KMLS for this meter.

With respect to accuracy, Fig. 1 shows that DL outperforms KMLS for 14 meters. DL accuracy ranges from 83% to 98%, while KMLS ranges from 70% to 94%.

Actual and predicted power for meter 1 in June are shown in Fig. 2 and 3 for KMLS and DL, respectively. In Fig. 2, the predictions closely track the actual values but tend to miss abrupt periods of increased demand. This is a direct result of the sliding two-week data window used by KMLS. If a large peak has not been seen, the algorithm is less likely to predict a future large peak. This under-prediction could possibly be improved by adding more weight to $\mathbf{T_a}$ and $\mathbf{f}$ in clustering and/or including historic data from a previous year. Fig. 3 shows that the DL algorithm's active power predictions track the actual values more closely than the KMLS algorithm. However, in this case, the DL algorithm tends to overpredict reactive power.

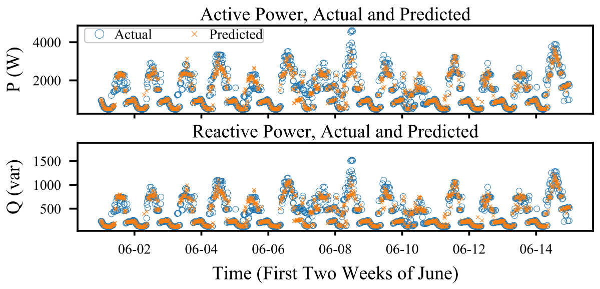

Figure 2. KMLS actual and predicted power for meter 1 in June.

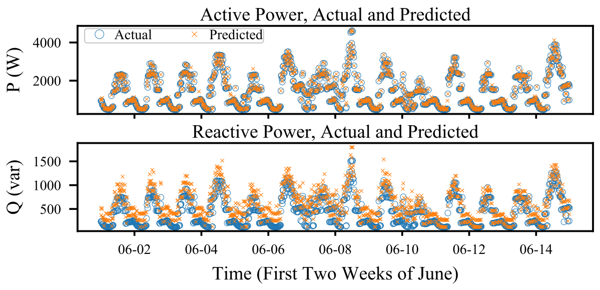

Figure 3. DL actual and predicted power for meter 1 in June.

Fig. 4 and 5 show actual and predicted power for meter 9 in January for KMLS and DL, respectively. Similar to Fig. 2, Fig. 4 shows that while KMLS does well at predicting overall, it underpredicts most peaks. The DL algorithm predictions in Fig. 5 closely match for both active and reactive power, but reactive power mins/maxes are under/overestimated.

5.4. Algorithm Alternatives

Both algorithms can be altered in a multitude of ways that can affect accuracy and/or computation time. This section presents some relevant algorithm alternatives.

At times, the cluster selection described in Section 4.1 selects clusters that do not lead

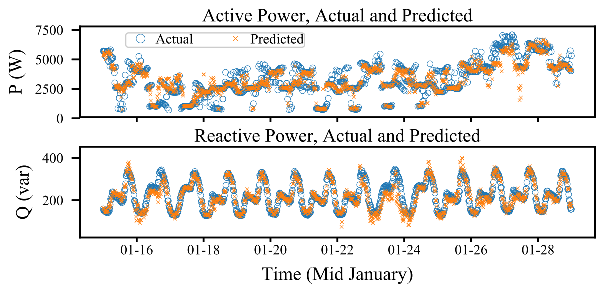

Figure 4. KMLS actual and predicted power for meter 9 in January.

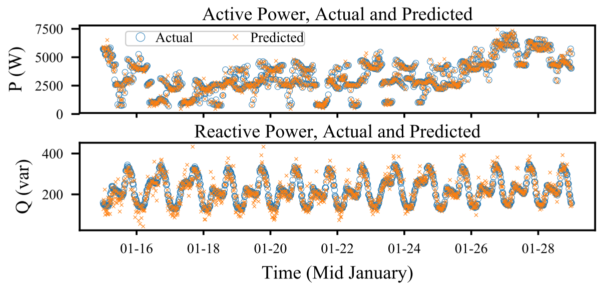

Figure 5. DL actual and predicted power for meter 9 in January.

to a model that makes accurate predictions. An alternative cluster selection method can improve accuracy: Instead of only using present temperature and solar flux in cluster selection, average active and reactive power for the historical data are also used as cluster selection parameters. In computing these averages, $\mathbf{P}$ and $\mathbf{Q}$ from the historic data are further filtered before means are computed. Filtering is performed such that the only times that remain match the upcoming prediction interval. I.e., if predicting ZIP coefficients for 09:30 a.m., the average P and Q used in cluster selection will only come from historical measurements timestamped 09:30 a.m.

Fig. 6 shows the predicted vs. actual active power for meter 13 for both cluster selection methods. The best linear fit to the data is presented as a line. The standard cluster selection results in a coefficient of determination (COD, also known as R^2) between the regression line and the data of 0.05, and the alternative cluster selection results in a COD of 0.89 (closer to 1 is better). While this alternate cluster selection showed a significantly improved prediction for meter 13, it does not perform well for all meters at all times. Future work may investigate when it's best to use each clustering method.

Two alternative training methods for DL are explored: classification-based two-stepped training and one-stepped training. In classification-based two-stepped training, the

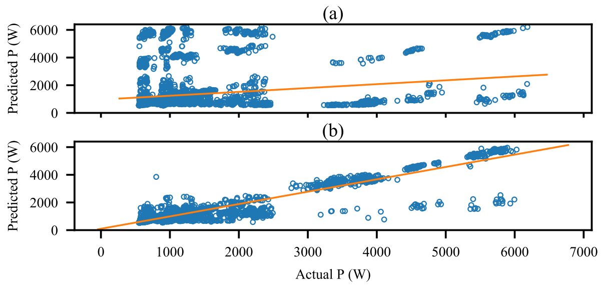

Figure 6. Predicted vs. actual active power for meter 13 using the KMLS algorithm. (a): Standard clustering (b): Alternative clustering

training process is the same as the two-stepped training process, except the first training step is performed with all data from an initial classification of meters based on their properties (column labels in Table 1), instead of for every meter. The second training step is carried out for each individual meter. The one-stepped training process is the first step of training without using the second step.

Table 5 illustrates a comparison of the three training methods, applied to meter 3. Parallel training is performed on an 8 core, 16 thread, 2.1GHz processor. While the two-stepped training described in Section 4.2 performs the best, the training is the most computationally intensive. The classification-based, two-stepped training strikes a balance between accuracy and training time, while the one-stepped training does not perform well due to error accumulation.

Table 5. Training Methods for Meter 3 Comparison

Method	Trainable Parameters	Time (hrs)	Acc	RMSE
Two-stepped	339,768	7.23	0.96	54
Classification-based two-stepped	169,884	3.34	0.71	1312
One-stepped	169,884	3.38	0.43	3987

5.5. Computational Considerations

One advantage to the KMLS algorithm is that it does not require large amounts of data or significant processing time to derive a predictive ZIP model. After acquiring two weeks of data, the average time to compute the ZIP coefficients for a single electric customer is roughly one second (time includes entire algorithm in Section 4.1). Since each customer's fit is independent, this is a highly parallelizable problem. The fitting can be performed in a distributed fashion, possibly even at the meters themselves.

As shown in Table 5, the two-stepped DL algorithm can take more than seven hours to train per meter. In practice, DL computational requirements would likely necessitate centralized training. Once trained, the time to make a prediction is on the same order of magnitude as KMLS, approximately one second.

Overall the presented DL algorithm outperforms the KMLS algorithm. However, deep learning requires more data, training time, and computational power.

6. Conclusions and Future Work

As DA control schemes such as VVO move toward model-based operation, it becomes imperative to have accurate predictive load models to ensure control decisions are made based on accurate power flow results. This paper presents two algorithms for performing data-driven, time-varying load modeling on a per-customer basis in the electric distribution system. The first uses two weeks of historical data, *k*-means clustering, and least-squares optimization to derive a predictive ZIP load model, and the second uses a convolution-based recurrent neural network with long short-term memory.

To evaluate both algorithms, customer power demand is predicted with each algorithm's derived ZIP models, and compared with the actual customer demand. For this work, both algorithms derive a new predictive ZIP model for every 15-minute interval in a one-year simulation. The results demonstrate that both algorithms can provide accurate predictions and perform better than a simple load allocation method.

The KMLS algorithm, while novel, is relatively simple and very practical. It could be deployed on modern advanced distribution management or AMI systems to improve model-based DA applications, such as VVO. The DL algorithm, also novel, is more accurate but requires more data, computing power, and training time. Deployment would likely require centralized training and would be more difficult to implement.

Future work will focus primarily on two areas. First, the presented algorithms will be validated with utility data, and methods to increase deployability will be explored, such as transfer learning. Second, these algorithms will be integrated into near-real-time VVO schemes that are being developed as part of the GridAPPS-D program under the U.S. Department of Energy's Advanced Grid Research Program [1].

References

[1] R. Melton, K. Schneider, E. Lightner, *et al.*, "Leveraging standards to create an open platform for the development of advanced distribution applications," *IEEE Access*, vol. 6, 2018.

[2] M. Vadari, *Electric System Operations*. Artech House, 2013.

[3] K. P. Schneider, E. Sortomme, S. S. Venkata, *et al.*, "Evaluating the magnitude and duration of cold load pick-up on residential distribution using multi-state load models," *IEEE Transactions on Power Systems*, vol. 31, pp. 3765–3774, Sep 2016.

[4] K. P. Schneider and T. F. Weaver, "Volt-var optimization on American Electric Power feeders in northeast Columbus," in *PES T D 2012*, pp. 1–8, May 2012.

[5] M. E. Baran and M.-Y. Hsu, "Volt/var control at distribution substations," *IEEE Transactions on Power Systems*, vol. 14, pp. 312–318, Feb 1999.

[6] V. Dabic and D. Atanackovic, "Voltage var optimization real time closed loop deployment - BC Hydro challenges and opportunities," in *2015 IEEE Power Energy Society General Meeting*, July 2015.

[7] S. Rahimi, S. Massucco, and F. Silvestro, "Coordinated closed-loop voltage control by using a real-time volt/var optimization function for MV distribution networks," in *2015 IEEE 15th International Conference on Environment and Electrical Engineering (EEEIC)*, pp. 1222–1228, June 2015.

[8] "Avista utilities' conservation voltage reduction program," tech. rep., Navigant Consulting, Inc., May 2014.

[9] R. Moghe, D. Tholomier, D. Divan, *et al.*, "Grid edge control: A new approach for volt-var optimization," in *2016 IEEE/PES Transmission and Distribution Conference and Exposition (T D)*, pp. 1–5, May 2016.

[10] K. Schneider, J. Fuller, F. Tuffner, *et al.*, "Evaluation of conservation voltage reduction (CVR) on a national level," tech. rep., Pacific Northwest National Laboratory, Jul 2010.

[11] M. Vadari, *Smart Grid Redefined*. Artech House, 2018.

[12] K. P. Schneider and T. F. Weaver, "A method for evaluating volt-var optimization field demonstrations," *IEEE Transactions on Smart Grid*, vol. 5, pp. 1696–1703, July 2014.

[13] C. Zhang, B. Dai, T. Wu, *et al.*, "Model-based volt-var optimization using advanced metering infrastructure in distribution networks," in *2015 IEEE Eindhoven PowerTech*, pp. 1–5, June 2015.

[14] "End-use load composition estimation using smart meter data," Tech. Rep. 1020060, EPRI, Dec. 2010.

[15] M. Manbachi, H. Farhangi, A. Palizban, *et al.*, "Quasi real-time ZIP load modeling for conservation voltage reduction of smart distribution networks using disaggregated AMI data," *Sustainable Cities and Society*, vol. 19, no. Supplement C, pp. 1 – 10, 2015.

[16] A. Bokhari, A. Alkan, R. Dogan, *et al.*, "Experimental determination of the ZIP coefficients for modern residential, commercial, and industrial loads," *IEEE Transactions on Power Delivery*, vol. 29, pp. 1372–1381, June 2014.

[17] X. Zhang, S. Grijalva, and M. J. Reno, "A time-variant load model based on smart meter data mining," in *2014 IEEE PES General Meeting — Conference Exposition*, pp. 1–5, July 2014.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[19] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," 2001–.

[20] M. Abadi, A. Agarwal, P. Barham, *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[21] F. Chollet, "Keras." `https://github.com/fchollet/keras`, 2015.

[22] "GridLAB-D™." `https://www.gridlabd.org/`.

[23] K. Schneider, Y. Chen, D. Chassin, *et al.*, "Modern grid initiative distribution taxonomy final report," tech. rep., Pacific Northwest National Laboratory, Nov 2008.

[24] "TMY." `https://nsrdb.nrel.gov/tmy`.

[25] S. Parker, W. Hunt, K. Fowler, *et al.*, "Metering best practices: A guide to achieving utility resource efficiency, release 3.0," tech. rep., Pacific Northwest National Laboratory, Mar 2015.

[26] "Advanced metering infrastructure and customer systems," tech. rep., U.S. Department of Energy, Sep. 2016.

[27] W. Kersting, *Distribution System Modeling and Analysis*. CRC Press, 3 ed., 2012.

[28] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[30] J. Zhang, A. Florita, B.-M. Hodge, *et al.*, "A suite of metrics for assessing the performance of solar power forecasting," *Solar Energy*, vol. 111, pp. 157–175, 2015.

[31] B. Shah, A. Bose, and A. Srivastava, "Load modeling and voltage optimization using smart meter infrastructure," in *2013 IEEE PES Innovative Smart Grid Technologies Conference (ISGT)*, Feb 2013.