AI-Based Estimation of Available Flexibility at Individual House Level

AI-Based Estimation of Available Flexibility at Individual House Level

Baraa Mohandes, Daniel Koster
Luxembourg Institute of Science and Technology
Baraa.Mohandes@list.lu; Daniel.Koster@list.lu

Phuong H. Nguyen
Eindhoven University of Technology
P.Nguyen.Hong@tue.nl

***Abstract*—This paper develops a data-driven model for assessing the availability of flexibility from individual household devices, at house level. The model predicts the potential shift, increase or decrease of the energy consumption of a particular type of device at a given time, in response to a price signal, and for each house separately. Therefore, the location of the flexibility source is known with accuracy. The model has an Auto-Encoder architecture based on Convolution Neural Network. The augmented model demonstrates good performance in terms of predicting the time-shift in load.**

***Index Terms*—Flexibility, deep learning, distribution level, data-driven model, curtailable, time-shift**

I. Introduction and Review of Literature

The importance of power system flexibility transcends the transmission system; flexibility is vital for the distribution network as well. The use-cases of flexibility at the distribution level include: relieving local congestions, suppressing reverse flow, and correcting voltage deviations. The use cases of flexibility at the distribution level require knowledge of the location of the flexibility source. For example, an estimate of the total flexibility obtainable from all users on a feeder may not be useful for correcting a voltage deviation on the feeder itself.

Aggregators who are bidding to sell flexibility need good estimates of the amount of flexibility available at their disposal [1]. Flexibility availability is defined in [1] as "the amount of load available for switching, by the control action". In a survey on Demand Response Programs (DRPs) at the distribution level, Li *et al.* [2] reveal that available flexibility is commonly estimated at the system-wide level, or sector level (e.g. residential, commercial, etc.), rather than at the individual unit level. Gottwalt *et al.* [3] highlight that the majority of literature considers only one type of responsive devices when quantifying available flexibility. Therefore, when the authors attempt to quantify flexibility of different device types, they use a dedicated linear model for each load category.

The models to quantify available flexibility can be deterministic, statistical (i.e. empirical), or hybrid [2, 4, 5]. Deterministic models rely on accurate physical models of the load. The deterministic model of thermostatically controlled loads (TCLs) (e.g. heat pump (HP), freezer (FR)) describes their thermal dynamics via a first-order differential equation [1, 4]. Deterministic models usually have poor scalability, and run on small sets of loads only [4]. A hybrid approach is proposed by [4]. A physical model is used with only a small group of buildings, and a regression model is trained to scale up the flexibility from the small group to the full population of buildings.

Haque *et al.* [5] quantify the flexibility potential from 200 houses participating in a Real Time Price (RTP) DRP; also known as dynamic tariffs. Each house contains six types of devices. When flexibility is needed, the aggregator composes a price profile for the whole day and conveys it to the end-users. Each household device optimizes its own consumption plan with respect to the new price profile, and its internal constraints. The adjusted load profile of each household device is submitted to the aggregator. The difference between the benchmark consumption profile and the adjusted load profile is deemed as flexibility. This process may repeat for a number of iterations until the desired flexibility is availed.

It can be inferred that any form of data aggregation, either at the device level or the location level, defeats the purpose of flexibility procurement at the distribution level. An estimate of flexibility potential at sector or system-wide level is informative to the Transmission System Operator (TSO); albeit, it has little value to the aggregator and the Distribution System Operator (DSO). The process in [5] requires several rounds of communication, which is time-consuming. Each communication session is also prone to cyber intrusions [6]. An accurate prediction of the response of a particular load to dynamic tariffs would reduce the number of rounds of communication; thus, reducing decision-making time and intrusion risk. Such a tool allows the aggregator in [5] to test candidate price profiles, internally, before officially transmitting them to loads. Such a tool can also serve inside a Reinforcement Learning model, trained and utilized by the aggregator. Lee *et al.* [7] build such a load predictor with a shallow Artificial Neural Network (ANN), which features one hidden layer. It is noteworthy that this type of ANN is not suited for processing time-series data.

This paper aims to develop a data driven model which can replicate the output of load optimization with respect to a dynamic energy tariff in [5]. The purpose of such model is to predict the availability of flexibility, for each type of device, and for each house, individually. The model generates a prediction faster than optimization solvers do. Consequently, system operators have a fast and reliable tool to test different price profiles. To this aim, we appropriate the consumer and tariff data in [5], and attempt to replicate the results of [5]. The model adopts an Auto-Encoder (AE) architecture composed of a number of Convolution Neural Network (CNN) layers. The

978-1-6654-6441-3/23/$31.00 ©2023 IEEE

proposed tool in this paper is part of an ongoing effort to develop smart autonomous agents in a local flexibility market.

The remaining of this paper is organized as follows: Section II provides a brief introduction about AE. The actual architecture of the developed model is demonstrated in Section III. The case-study and data preparation are described in Section IV. Section V recaps the work.

II. Auto Encoders

An AE consists of a chain of neural networks: 1) Encoder, 2) Intermediate layer 3) Decoder. The encoder converts the input data to another abstract format. The intermediate (i.e. second) layer carries out a specific function on the encoded data. This layer cannot operate directly on the raw data. Conversely, the decoder can convert the encoded data back to the original readable format, as shown below:

$$\text{Human Readable Data} \xrightarrow{\text{Encoder}} \text{Encoded Data}$$

$$\text{Encoded data} \xrightarrow{\text{Intermediate}} \text{Processed Encoded Data}$$

$$\text{Processed Encoded Data} \xrightarrow{\text{Decoder}} \text{Processed Human Readable Data}$$

An example of human readable data is a 2D photo. The interested reader can learn more about AEs in [8]. Some applications of AEs are:

- **Image Restoration:** A distorted image is fed to the encoder, then repaired in the intermediate layer. The decoder converts the processed data into a meaningful image.
- **Encrypting data:** The encoder conceals the original information. Only the right decoder can decrypt the data. No intermediate layer is needed in this application.
- **Data compression:** The encoded data have smaller dimension, and smaller memory requirements. The decoder can restore the original data.

III. Mining Flexibility With AE–CNN

Figure 1 demonstrates the price-based dynamic-tariff DRP adopted in [5]. The aggregator announces a full-day's profile of energy prices to its users. In response, the users update their next-day's load consumption plan, and transmit the consumption plans back to the aggregator. This process can repeat for several rounds, until the aggregate load profile complies with the aggregator's commitments. The adjusted consumption plan of each user, in response to the proposed dynamic-tariff profile, is the solution of a linear optimization problem. This particular part of the process is represented by the yellow box in Fig. 1. This optimization problem is solved for each of four flexible household appliances, in each of 200 houses. In this paper, we train an AE to reproduce the solution of this Linear Program (LP) (yellow box in Fig. 1), thus, eliminating the need for a LP solver. If the aggregator possesses such a tool, the aggregator can explore hypothetical dynamic tariff profiles, before actually announcing them to end-users. This can reduce the number of communication rounds between aggregators and end-users.

The goal of training a data-driven model is to reproduce the

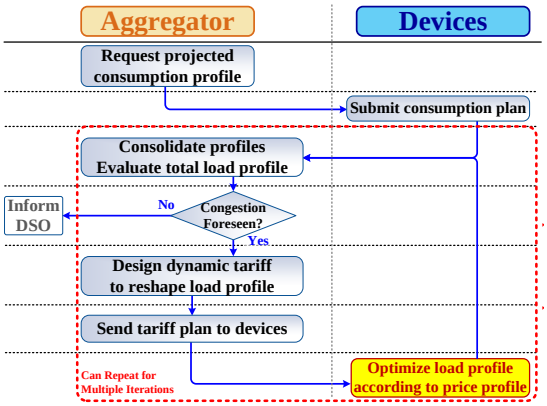
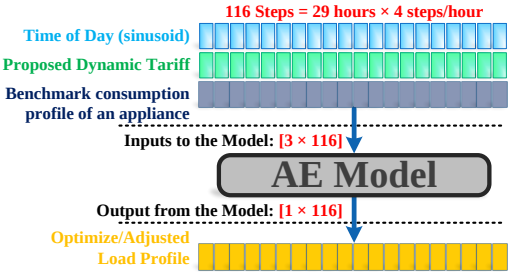


Fig. 1: Aggregator↔Consumer Interaction

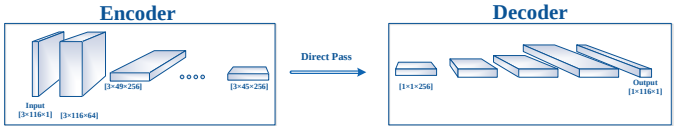
output of the optimization. This model would, later, be utilized to predict the adjusted consumption profile of an electric device in response to a dynamic tariff. The adjustment in the load profile represents the available amount of flexibility at each time period for the given price signal. The model's inputs and outputs, and the model's internal architecture are depicted in Fig. 2a. The working principle of the model is:

1) The input to the encoder is a set of three time-series: the time of the day, the benchmark load profile, and the proposed dynamic price profile. The encoder converts the input data to another higher dimensional space.
2) The encoded data pass through a number of fully-connected layers, which imitate optimization solvers and reproduce the optimization result.
3) The decoder converts the solution back from the high-dimensional space to a time-series which represents the adjusted load profile.

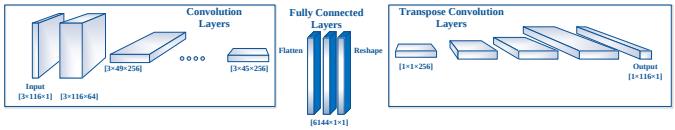
For this auto-encoder to work properly, the model is constructed and trained in two stages, depicted in Fig. 2b and 2c:



(a) Model's Input and Output



(b) Stage 1: Training Encoder-Decoder Pair Only



(c) Stage 2: Training AE to Reproduce the LP Solution

Fig. 2: CNN Model for Flexibility Quantification

978-1-6654-6441-3/23/$31.00 ©2023 IEEE

Stage 1) Create a preliminary model consisting of an encoder and decoder only. The encoder-decoder pair is trained to encode the benchmark load profile, and reconstruct it back as it is, without any optimization. It is important to keep in mind that, at this stage, the decoder should output the benchmark load profile, not the adjusted load profile. This stage can be characterized as follows:

- **Model**: Encoder output is fed directly to the decoder.
- **Trainable layers**: Encoder and decoder.
- **Input data**: three time-series: time, dummy price, benchmark load.
- **Prediction Target**: Reconstruct benchmark load as it is.

In principle, the encoder and decoder can be based on any type of ANN. Processing time-series data must consider the intercorrelations within the data. Two types of deep ANN are capable of handling intercorrelations: CNN, and Recurrent Neural Networks (RNNs). Training RNNs is slower than training CNNs. Therefore, we adopt a CNN-based AE model. The role of the encoder-decoder pair in the bigger model is only to convert data from one format to another, and convert it back.

Stage 2) Fully connected layers are inserted between the trained encoder and decoder from Stage 1. The new composite model is trained to produce the adjusted load profile. The primary scope of training is the fully-connected layers. The goal of training is for the fully-connected layers to imitate optimization solvers in the encoded space, in order to reproduce the true optimization solution.

- **Model**: Encoder output is fed to a chain of fully connected layers, which pass their output to the decoder.
- **Trainable layers**: Encoder and fully connected layers.
- **Input data**: three time-series: time, dynamic price of electricity, benchmark load profile.
- **Output data**: Adjusted load profile, in response to dynamic electricity price.

The input and output data of the physical deterministic model developed in [5] are used to train the model. The dataset includes a RTP profile and a benchmark energy profile for a full year, for each of four devices, in each of 200 houses. Therefore, a data-driven model must take in the price and load data for an individual house, and make predictions accordingly. At the same time, it is necessary to design a versatile model which can process the input data of any house, rather than building dedicated models for each individual house. In real-world applications, not all houses may have the same four responsive devices studied in [5]. Therefore, a separate ANN model is trained for each type of electric device. The ANN for a particular device (i.e. out of four models) must be usable (i.e. give reliable predictions) for any of the 200 houses.

Multiple hidden patterns (i.e. load-cycles with different lengths) may exist in a time-series, simultaneously. Detecting different possible patterns in a time series requires employing several CNNs with different kernel sizes. Each of the CNNs must operate on the raw input, before the patterns are destroyed. Therefore, a GOOGLE inception layer is implemented, where the data are sent through multiple parallel paths, simultaneously. The outputs from all paths are concatenated into one big stack, for further processing.

IV. CASE STUDY

A. Data Preprocessing

The three time profiles extend for 29 hours. This allows loads starting on hour 23 to be deferred to the first hours of the next day. The data have a time resolution of 15-minutes. This gives rise to data vectors with $4 \times 29 = 116$ time-steps. The three time series vectors form a matrix of size $[3 \times 116]$.

ANNs require the input data to be normalized to a range [8] (e.g. $[-1, 1]$). Similarly, the output vector consists of numerical values within a range. Different houses have different consumption patterns and characteristics, such as: peak load, cycle length, total connected load, and total daily energy consumption. It is not realistic to normalize the data for 200 houses using the same set of parameters. The profile for each device in each house is normalized separately based on its own characteristics over the full-year. For example, in house #1, the electric vehicle (EV)'s consumption profile for the whole year is normalized based on the extreme values in this particular profile (whole year's profile, for the EV in house #1 alone). When a load profile is normalized by its absolute minimum and maximum values, as depicted in (1), rare load spikes expand the range, and the average consumption level appears to be close to zero. To mitigate this problem, data outliers or extreme events are neglected when we choose normalization parameters.

We normalize every profile by the range [2.5^{th} percentile – 97.5^{th} percentile] of all its points. Consequently, the normalized load profile provides an accurate representation of the original load profile. We emphasize that extreme events above the 97.5^{th} percentile are not removed from the data, but truncated to the boundaries $\{-1, 1\}$.

Some household devices are used intermittently or occasionally, such as washing machines and dish washers. In some cases, the device is not switched ON at all, and the profile is zero for a full day. No flexibility can be anticipated in this particular instance, and it is justified to eliminate such instances from the data. Finally, it is important to note that representing the time of the day as a number between 1 and 23 provides the ANN with a false impression that there is huge difference between 11:45PM and 12:15AM. Therefore, time-stamps are mapped to a sinusoidal wave using (2).

$$\widetilde{P} = \frac{P - P_{\text{min}}}{P_{\text{max}} - P_{\text{min}}} \tag{1}$$

$$q = \tfrac{1}{2} \left(1 - \cos \frac{2\pi t}{24}\right) \tag{2}$$

B. Training Process

The same network architecture is used with all four devices. However, a separate copy of the network is trained for each device. The network architecture for stage 1 includes more than 1.6 million trainable weights. Two fully-connected layers are added in stage 2. These layers have 512 and 256 neurons,

978-1-6654-6441-3/23/$31.00 ©2023 IEEE

respectively, and comprise more than 1.8 million trainable parameters. The training hyper-parameters are given in Table I. More information about training AEs is available in [8].

The dataset is split into a training set, a validation set, and a test set. The actual training process is carried out using the training set only. The purpose of the validation set is to monitor the training progress of the ANN on extrinsic data, during training, rather than after training. Validation error decreases with training error during the first epochs. If the validation error plateaus or starts rising again, then it is likely that the ANN is overfitting the training data, and training should be stopped. If validation error is much lower than testing error, then the validation set is not a good representative of the whole dataset.

TABLE I: Auto-Encoder Training Settings

Parameter	Value
Leaky ReLU activation slope	0.1
Learning Rate (LR)	9×10^{-4}
Batch Size	32
LR Exponential Decay Factor	0.8 every 100 Batches

Stage 1) The training algorithm aims to minimize the loss function, the Mean Squared Error (MSE). The algorithm is also set to stop if the relative improvement in the validation error is below 0.001 for 15 consecutive epochs. The progress and the performance of training the model for the EV are illustrated in Fig. 3. The Training progress of the other devices is not shown, for brevity.

The left-side plot in Fig. 3 depicts the MSE value at the end of each epoch of training. Similarly, the right-side plot in Fig. 3 depicts the Mean Absolute Error (MAE). The MAE metric is for observation only, and does not affect the training progress. Training is halted because validation error stabilizes for 15 epochs. The final MSE values for the training set and the validation set are 0.004 and 0.02, respectively. This indicates that the model does not overfit the data. The final MAE value for the training set and the validation set are 0.012 and 0.022, respectively. In general, training concludes within less than 30 epochs for all cases.

Fig. 4 illustrates the performance of the AE model with the EV and washing machine (WM) data. The left-side plots depict the benchmark load, and the reconstructed profile by the auto-encoder. The right-side plots depict the reconstruction error (i.e. the difference between the benchmark load and the reconstruction). The preliminary AE model demonstrates good ability to encode and decode the raw data, with negligible deviations. The next step is to introduce the intermediate layers and retrain the model.

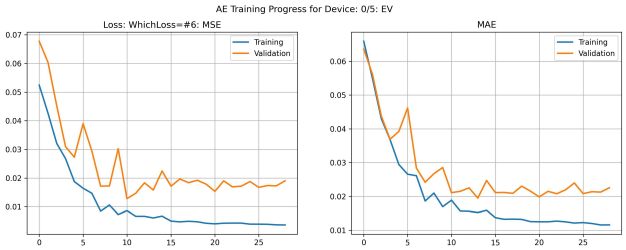
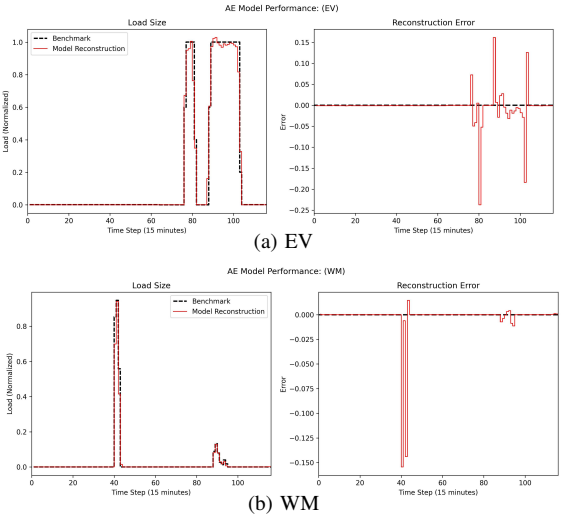


Fig. 3: Training Progress of Stage 1 AE for EV



(a) EV

(b) WM

Fig. 4: Stage 1 AE Performance

Stage 2) Two fully connected layers with 512 and 256 neurons are inserted between the encoder and decoder. The same hyper-parameters provided in Table I are used in the second training stage, as well. The augmented model is trained, and the training progress for the WM model is illustrated in Fig. 5. The final MSE values for the training set and the validation set are 0.0084 and 0.0069, respectively.

The performance of each trained AE model for the four device types (i.e. EV, WM, HP, and dish washer (DW)) is shown in Fig. 6. It is observable that the trained network succeeds at predicting the time-shift in the load event successfully. The network, however, is less accurate at predicting the actual magnitude of the load, especially for long continuous events such as that in Fig. 6a. The largest error in predicting the magnitude of the load is 30% of the load size.

These results prove the validity of the design concept. Better performance is certainly attainable with a larger dataset. In fact, the data appropriated from [5] suffer from the following limitations: 1) A significant number of samples where the load does not respond to the price profile, and does not provide any flexibility; 2) The benchmark load profile responds to only one proposed price profile. Optimizing the same benchmark load profile for different price profiles would help the model observe the isolated effect of price on the load. 3) The benchmark electricity tariff associated with the benchmark load profile is unknown. 4) Weather conditions are unknown. HP load is dependent on ambient temperature.

LPs can be solved efficiently (i.e. in polynomial time). How-

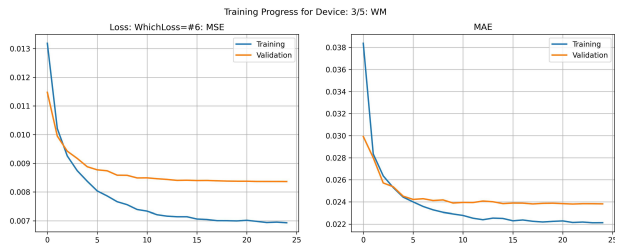
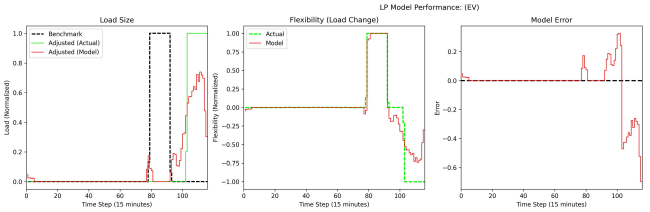
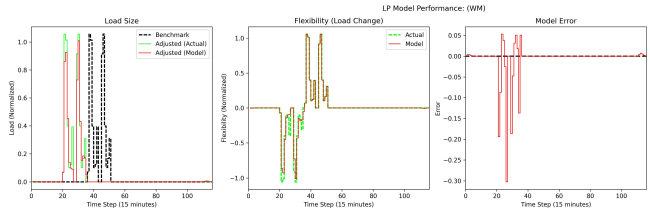


Fig. 5: Training Progress of Augmented AE for WM

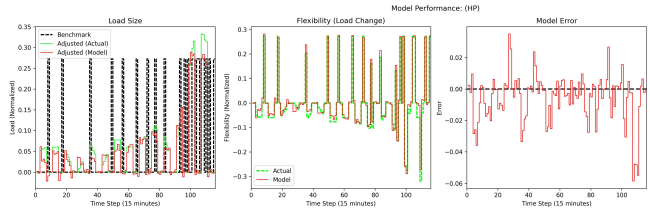
978-1-6654-6441-3/23/$31.00 ©2023 IEEE



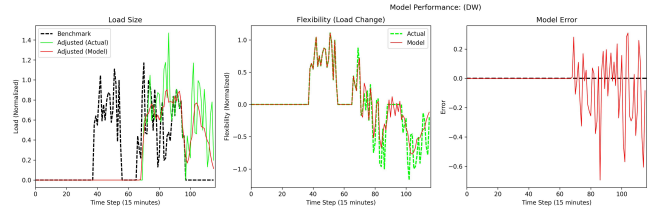
(a) EV



(b) WM



(c) HP



(d) DW

Fig. 6: Performance of Augmented AE

ever, the flexibility quantification problem tackled in this paper involves solving a separate optimization problem for each of 4 electric devices in 200 houses. Each optimization problem involves an iterative search for the optimum, utilizing primal-dual solution methods. With data-driven models, training the model is a one-time process that requires a substantial amount of time. However, once the model is trained, the model is capable of processing input data to produce results immediately. At the same time, since the distribution network observes the same time-stamp and price profile, the data-driven model can process a batch of multiple houses simultaneously.

Test runs are conducted on the flexibility model in [5], for the purpose of recording the execution time in different cases. After training the data-driven model, we also take note of both, the training time, and the evaluation time of a batch of 20 houses. Table II compares the execution times for each model. It is clear that evaluating a trained CNN-based AE model is multiple times faster than solving the LP. Implementing the trained AE on even larger systems makes the AE training-time worthwhile.

V. Conclusion

This paper develops a data-driven model for quantifying available flexibility. A model based on Convolution Neural Networks (CNNs) and an Auto-Encoder (AE) is developed in two stages. First: an AE learns to convert the data from its raw format to a latent space, and back to a readable format. Second: the AE is augmented with a number of fully-connected layers, inserted in the middle of the AE. These layers learn to imitate optimization solvers to reproduce the optimum solution. Building data-driven models involves a lengthy training process carried out once only. However, we demonstrate that evaluating an input instance with a trained model is much faster than solving the actual optimization problem. The final AE model demonstrates good performance in terms of predicting the time-shift in a load event. The magnitude of the adjusted load is predicted with less accuracy. This issue can be rectified if the dataset contained certain features, and more training data were available.

TABLE II: Training and Evaluation Time Comparison

	Convolutional Net		Linear Program
	Training (H:M)	**Evaluation** (milliseconds)	**Solution** (milliseconds)
EV	00:25 – 00:35	2.7 - 3.5	8.3 - 35.9
WM	00:09 - 00:10	1.17 – 1.89	13 - 32.6
HP	00:37 – 01:18	0.895 – 3.84	0.175 - 6.27
DW	00:16 – 00:20	0.5 - 2	0.196 - 4.27

Acknolwedgements

The work leading to this paper is from Work-Package 3 of FLEXIGRID project that is funded by the *European Community's Horizon 2020 Framework Programme* under grant agreement No. 86404

References

[1] M. Heleno *et al.*, "Availability and flexibility of loads for the provision of reserve," *IEEE Trans. on Smart Grid*, vol. 6, no. 2, pp. 667–674, 2015.

[2] H. Li *et al.*, "Energy flexibility of residential buildings: A systematic review of characterization and quantification methods and applications," *Advances in Applied Energy*, vol. 3, 2021.

[3] S. Gottwalt *et al.*, "Modeling and valuation of residential demand flexibility for renewable energy integration," *IEEE Trans. on Smart Grid*, vol. 8, no. 6, pp. 2565–2574, 2017.

[4] R. Yin *et al.*, "Quantifying flexibility of commercial and residential loads for demand response using setpoint changes," *Applied Energy*, no. 177, pp. 149–164, 2016.

[5] A. N. Haque *et al.*, "Integrating Direct and Indirect Load Control for Congestion Management in LV Networks," *IEEE Trans. on Smart Grid*, vol. 10, no. 1, pp. 741–751, 2019.

[6] B. Mohandes *et al.*, "Advancing cyber-physical sustainability through integrated analysis of smart power systems; a case study on electric vehicles," *Int. Jrnl of Critic. Infr. Prot.*, 2018.

[7] S. Lee and D. H. Choi, "Federated Reinforcement Learning for Energy Management of Multiple Smart Homes with Distributed Energy Resources," *IEEE Trans. on Indust. Info.*, vol. 18, no. 1, pp. 488–497, 2022.

[8] A. Geron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2nd. O'Reilly Media, Inc., 2019.