A Prospective Human-in-the-Loop Experiment using Reinforcement Learning with Planning for Optimizing Energy Demand Response

Lucas Spangher [1] **Adam Bouyamourn** [1] **Manan Khattar** [1] **Akaash Tawade** [1] **Akash Gokul** [1] **Alex Devonport** [1] **Costas Spanos** [1]

Abstract

While reinforcement learning on humans has shown incredible promise, it often suffers from a scarcity of data and few steps. In instances like this, a planning model of human behavior may greatly help. We present an experimental setup for the development and testing of two different RL architectures with several different neural architectures for planning models. Our RL architectures are Batch Constrained Q-Learning and Soft Actor Critic. Our primary planning models are various autoML frameworks and neural architectures. We present an experiment by which we hope to verify the learning of the RL agents and the efficacy of the planning model.

1. Introduction

1.1. Motivation in Energy (Background)

The California grid (CAISO) of 2020 reported 33% of electricity was generated from renewable sources. While admirable, this comes with challenges. Specifically, at certain times in the year, system-wide generation of energy (specifically solar) was too large for electricity grid absorb, and so CAISO simply shut down transmission going to those lines. CAISO reported that 3-4% of its renewable energy was curtailed due to overproduction in 2019. Curtailment was spread unevenly throughout the year, with certain days in the spring reporting 20-30% curtailment. Without solutions, this problem is likely to grow non-linearly with respect to the amount of renewable energy on the grid.

1.2. Energy storage and demand response

Two solutions to curtailment commonly touted are energy storage and demand-response. Energy storage entails the deferment of the consumption from when energy was generated to when it is needed, and Demand-Response (DR) entails the deferment of energy demand from when it is demanded to when it is most opportune for it to be filled. DR is basically costless, as it requires no infrastructure, so it is important as a direct solution. Consumer facing DR is nearly exclusively implemented with a price signal, which is the focus of our research. DR can further help by: (1) alleviating curtailment during high generation months to stretch battery reserves farther, (2) shift seasonal loads (3) alleviate peak loads during the day, to shift demand to other generation sources.

1.3. Social Games in building energy consumptions

The share of buildings' energy demands makes up a significant component of US energy demand and is increasing. In residential and commercial buildings, plug loads represent 30% of total electricity use ((Lanzisera et al., 2013), (Srinivasan et al., 2011)). In addition, the quantity of energy used by plugs is increasing more quickly than any other load type in both residential and commercial buildings (Comstock & Jarzomski, 2012).

To this end, notable work has been performed in creating and administering "Social Games" – for our purposes, defined as competitions around energy use. Social Games tend to comprise of: (1) informing each player of their energy use in a friendly and easy to consume manner and (2) an accompanying competition in which higher energy savings relative to others is rewarded. Many studies have either created Social Games themselves ((Konstantakopoulos et al., 2017), (Ratliff et al., 2014), (Papaioannou et al., 2018), (Papaioannou & Stamoulis, 2018)) or studied existing Social Games ((Cowley et al., 2011), (Ayres et al., 2012)) to draw insight. To this end, the general finding has been that Social Games increase the extent to which an individual is motivated to and able to save energy in their daily functioning within their office or home. In addition, DR has been extensively employed and studied within Social Games.

1.4. Reinforcement Learning for DR

Reinforcement Learning is a technique in Machine Learning that trains an agent to choose actions that maximize its rewards in the in an environment. Although RL is revolutionizing ad-placement, robotics, and gaming, its reach in energy is relatively limited. We are unaware of a study

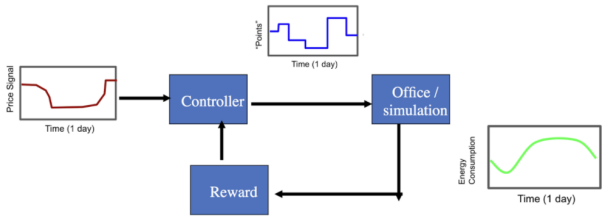

Figure 1. Diagram describing the flow of information in our experiment.

that attempts to aim a reinforcement learner at a DR price signal within a Social Game framework. Therefore, we aim to reproduce a Social Game experiment and quantify the difference in demand response from a variety of RL and RL augmentation techniques.

1.5. Paper Outline

This paper is intended to explain two scopes of work: the RL architecture and planning model search. We intend to give emphasis to the planning model search and de-emphasize the explanation of the RL techniques; for a full treatment of the RL component please see (Spangher et al., 2020). In Section 2, we will briefly describe the two RL architectures that we adapted and fine-tuned. We will then describe the planning models. After this, we will detail the reward specification and simulation that the first two items will be testing in. In Section 3, "Simulated Results", we will show results from the simulation that we use to argue success or failure for the experiment. In Section 4, we will describe the prospective experimental timeline, which will help the reader understand how we intend to actually the RL architectures "in the wild."

2. Methods

For a quick overview on the proposed experiment/simulation, please see figure 1. We will describe each of the boxes drawn in this flow chart in detail.

2.1. Reinforcement Learning techniques

Reinforcement Learning as a general technique came out of the physical simulators and computer gaming – arenas that are data rich in measurements and rewards. They might have access to many steps with millions of iterations to try various trajectories. We are dealing with a different scenario – so-called human-in-the-loop learning, where each step in the world is a day and data is very costly to attain.

As such, we need to consider off-policy RL: policies that do not rely solely on interactions with the environment, but with stored data that can be reused, with older data from other experiments, and with intelligent extrapolations. We describe three architectures that we are considering below.

2.1.1. Batch Constrained Q Learning (BCQL)

Q Learning is an model-free method of RL in which an agent learns Q-functions associated with each state, action pair. The prediction model serves to predict the reward the agent would obtain from each, and then picks the action corresponding to the maximum Q-value at each step. This is fundamental when you do not have the transition probabilities (i.e. you don't have an underlying dynamics model of the situation.)

In BCQL, the goal is to train the policy offline, but constrain the agent's actions to within the data distribution of the batch (Le et al., 2019). It uses Actor, Critic, and Deterministic Policy Gradient (DPG) to guide the learning, collecting pairs of states and actions in a replay buffer. It uses this replay buffer to train a variational auto-encoder (VAE), meaning it maps state action pairs down to a smaller latent space that it is able to search efficiently with the few actions. For this reason, we hypothesize that it would be good for small sample learning. It uses VAE actions with soft clipped double Q learning to get target values and train the critic. A soft clipped double Q-learning means using two Q networks to estimate the critic function by overlighting the smaller Q value, i.e.

$$Q_C = .75 * (\min(Q_1, Q_2)) + .25 * (\max(Q_1, Q_2))$$

Then, we generate and use the actions and the critic to calculate DPG and train the actor. Our BCQL agent emits a 10-dim points vector, each output a discrete integer value between 0 and 10.

2.1.2. Soft Actor Critic (SAC) V2

In traditional Actor Critic, we employ a form of Q learning with two different prediction networks. One network, the the "Critic", predicts the Q-value of the state and actions that are sampled from the policy distribution, whereas the "Actor" updates the policy distribution with critic's estimate (which functions as a gradient step.)

The SAC is an extension of the traditional Actor Critic to a continuous output space, with some improvements to function in an off-policy way (Haarnoja et al., 2018). Now, the actor updates the policy distribution with entropy as a regularizing term. maximizing the reward while taking the most "random" step possible. This makes the actions sampled along the trajectory cover the most ground possible, essentially exploring the space more effectively. Our SAC

V2 emits a 10-dim points vector, each output a continuous value between 0 and 10.

2.1.3. Additional technique: Overtraining

In addition to a more routine hyperparameter search, we would like to explore the effect of "overtraining" on the performance of the agent. Here, we will train the agent for successively more iterations before taking steps in the environment. As each time step is important in the training of our agent, we will use the action that produced the minimum reward in the environment. We will apply this technique to all architectures to improve performance.

2.2. Planning models: improving RL agent's data efficiency

Although the algorithms that we describe above are important advances in RL for efficient data, we still are concerned that a lack of steps within the environment will hinder any algorithm from effectively learning. We propose a way to address this: planning.

Constructing a good model for human behavior in the office – and specifically response to points – could provide a way for the algorithm to explore more than it could by interfacing with just the world. Each night, the agent will have stepped once in the real world and the planning model will train with an extra observation, but after the planning model updates, the agent can then train thousands of times on the updated model, essentially exploring many more points values.

To this end, we propose different planning models. Each model is made and trained on a simulated dataset of 5 years worth of energy and points data. The predictors, i.e. the explanatory variables parameters are the following:

- Hour, 8h through 18h, modeling a normal length of workday in Singapore.
- Date, i.e. the calendar date stretching back to 2018.
- Day of week, i.e. a 5 category variable for the week day's name
- Points, a daily price signal generated throughout the day. For the purpose of this pre-experiment dataset, we used the point signal generated by the Soft Actor Critic V2 agent.
- Vicarious learning, i.e. the effect of the actions of others around the player. This is defined per timestep t as the sum of the energy use of people at the player's workstation: $(\frac{\sum_{p\in\mathcal{W}} E}{\sum_{p\in\mathcal{W}} 1})$ where $\mathcal{W}$ is the player's workstation.
- Out of Office indicator, i.e. a 1 if the player is in the office or 0 if the player is out of the office. This is sensed through a wifi enabled occupancy detection mechanism detailed in (Zou et al.).
- Out of Game Baseline Energy, i.e. the average energy consumption in the past, defined per hour $h \in$ day of week $d \in \{$Mon, Tues, wed, Thurs, Fri$\}$ as the average of three of the same $h \in d$ of the three week period *before* the Social Game started.
- In Game Basline, i.e. the average energy consumption in the past, defined as above, for the three week period before the value being queried.
- Email frequency, i.e. the time since an email was sent reminding the players of the points that they are to play with throughout the day
- Points, i.e. the intervention from the agent.

We faced some difficult design decisions in how to build signal into this variable, given all the parameters are synthetic. Ultimately, we compute energy, the direct output response, with the direct points-to-energy response detailed in Section 2.4.1, and we add on top of it: draws from $\mathcal{N}(\mu, \sigma)$ where σ is 10 throughout, and μ is for each parameters: is one-tenth the value of vicarious learning, -100 * Out of office indicator, one twentieth the values of both baselines, and -10 * email frequency. We also include a "latent work state": we assume that each person has a certain amount of intense work to do each day which corresponds to a greater, but finite pool of additional energy. We draw the value of this from a $\mathcal{N}(100, 10)$ and then sample from a bernouli to decide whether to distribute this randomly in either the morning hours or the afternoon hours.

We will explain our attempts to fit data to this in the results.

2.2.1. GPyOpt LSTM Search

There are numerous out-of-the-box automated network search packages for Python, and GPyOpt (Knudde et al., 2017) is a promising one that uses Bayesian Optimization to arrive at optimal network parameter configurations. We ran a GPyOpt function optimizing over network size, epocs, batch size, number of timesteps, number of LSTM layers, and validation split. The operation was allowed to run over 100 iterations, and the best performing model was kept for evaluation.

2.2.2. Genetically Optimized hybrid hard/soft Attention based LSTM

While recurrent networks have long held tration in language and audio processing communities, the addition of "attention" lights as a trainable input and output has revolutionized the success of various LSTM in applications. As presented

in class, soft attention (where Iights over input are continuous between 0 and 1) is currently favored over hard attention (where weights are either 0 or 1) as the derivatives are ill-behaved, so it may be outputted by the LSTM itself. HoIver, this technique has been criticized for requiring more training data than other methods of resolving the attention (Fernando et al., 2018). In addition, unlike hard attention, the accuracy of soft attention is subject to the assumption that the Iighted average is a good representation for the area of attention.

To address these, we propose a "hybrid" approach combining hard and soft attention outside network training: adding an optimization layer on top of a vanilla LSTM to deal with the Iights directly. We formulate it in terms of a genetic algorithm: an optimization technique that is loosely based on evolutionary principles. Our genetic algorithm uses the following mechanisms:

- Binary encoding: we encode the attention Iights into a binary string of twice the length of the time lag that the LSTM considers, allowing each value to take on six decimal values between 0 and 1. Each encoding becomes an "individual" and undergoes one training iteration in the test.
- Litter: Train the network for each of 10 individuals in one "generation", or "litter". Retain the highest performing 4 individuals as measured by root mean square error (RMSE).
- Mating: take the highest 4 performing individuals and mate them pairwise. Each mating samples a random integer between 0 and their lengths; at this point, produce individuals who cross over.
- Mutation: for 4 of the individuals each turn, sample random indices to switch to simulate genetic mutation.

We coded the EA-LSTM using Keras to optimize a bi-layer LSTM of hidden state 128, batch size 1024 and epoch 50 and auto-regressive lag of 18 in all variables, explanatory and response. We performed most of the neural training on Savio-GPUs, a high performance computing cluster available to the Berkeley computing community.

Our idea loosely draws from (Li et al., 2019), who perform this mostly for different kinds of data and do not provide a codebase. However, it is substantially different to warrant individual merit. Implementing it required considerable time, effort, and coding.

2.2.3. LSTM Recurrent Neural Networks

We compare the Attention-based network to vanilla LSTM for quantification of the difference in output. The vanilla LSTM is comparable to the base LSTM in all aspects: it is implemented in Keras, with hidden state 128, batch size 1024 and epoch 50 and auto-regressive lag of 18 in all variables.

2.2.4. AutoKeras Structured Regressor

AutoKeras (Jin et al., 2019) is another AutoML package that, like GPyOpt, uses Bayesian optimization to guide a network architecture search. The framework further uses a tree-structured acquisition function to efficiently explore the search space. We ran a vanilla structured neural regression on the test data, allowing it to run for a maximum of 1000 iterations with early stopping on network training. Although all networks were allowed to train for 1000 epochs, due to early stopping they generally halted training by 60-70 epochs.

2.2.5. Fourier Prediction

We are interested in exploring Fourier Analysis and spectral decomposition. As far as we can tell, serious work has not yet been done in incorporating Fast-Fourier-transorms *within* the inner layers of a neural network; the closest is (Mishra et al., 2020), who present a method applied to wind-turbine prediction to incorporate fourier signal as an input into a neural network. Here, we present the results of a Fourier Analysis that will mimic the first component of the paper. It is our future intention to use this as a predictor in the neural net as they suggest.

Here, we perform and present a fast fourier transform to decompose the Energy signal into frequency components. We then take the 5 frequency components with the highest magnitudes and formulate a item cosine series with each of these frequencies as a component. We will use the output directly as a prediction for energy.

2.2.6. Baseline Model: Ordinary Least - Squares Regression and Autoregression

In developing these more complex models, we endeavor to also provide have baseline models to compare against. To this end, we propose a brief search into linear regression models. Our best linear regressions, which we will present in the results, consisted of the following: $y_t \sim \vec{u}_t$, where y_t is the energy at time t and $\vec{u}_t$ is the vector of exogenous variables at time t. We searched various degrees of autocorrelation, but found that (for hour i) the previous hour (i.e. $i-1$) and the same hour in the previous day (i.e. $i-24$) were most strongly autocorrrelated, any inclusion of previous time steps worsened the outcome. Therefore, we omit the others to save both on paper space and our readers' brain space.

2.3. Reward

All agent use the same reward calculation. This reward is defined as the difference between the day's total cost of energy and the ideal cost of energy. The ideal cost of energy is obtained using a simple convex optimization. If $\vec{d}$ are the actual demand of energy computed for the day, $\vec{g}$ is the vector of the grid prices for the day, E is the total amount of energy, and d_{min}, d_{max} are 5% and 95% values of energy observed over the past year, then the ideal demands are calculated with the following optimization:

$$\begin{aligned} d^* = \min_d \quad & d^T g \\ \text{s.t.} \quad & \sum_{t=0}^{10} d = E \\ & d_{min} < d < d_{max} \end{aligned}$$

Then, the reward becomes:

$$R(d) = \frac{d^{*T} g - d^T g}{d^{*T} g}$$

I.e. taking the difference and scaling by the total ideal cost to normalize the outcome.

2.4. Simulation design

In order to test the aforementioned techniques, we will use a simulation that can provide some level of insight into the superiority of some methods over others. We define three deterministic responses to points, and populate our office with an equal mixture "people" who exhibit each response type. We will describe the linear response here and then refer the reader to (Spangher et al., 2020) for definition of sinusoidal and threshold exponential response, as they are simple variations of the linear response.

In the Linear response, we define a very simple person who decreases their energy consumption linearly below a baseline with respect to points given. Therefore, if b_t is the baseline energy consumption consumed at time t and p_t are the points given, the energy demand d is $d_t = b_t - p_t$, clipped at d_{min} and d_{max} as defined in Section 2.3.

3. Simulated Results

3.1. RL Architectures

3.1.1. BCQL

In the RL architecture front, we report limited success with BCQL. Trained over a thirty day period, we see limited evidence that BCQL can learn, and it seems to exhibit little exploration; indeed, the reward seems to be largely constrained to a small bit of the range. Please see Figure 2.

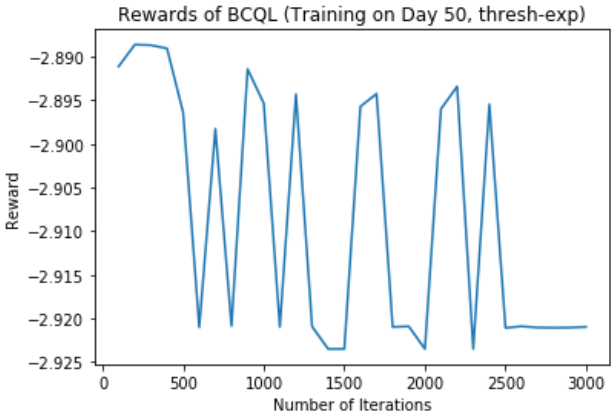

Figure 2. Learning rate of the BCQL algorithm, trained over 3000 iterations.

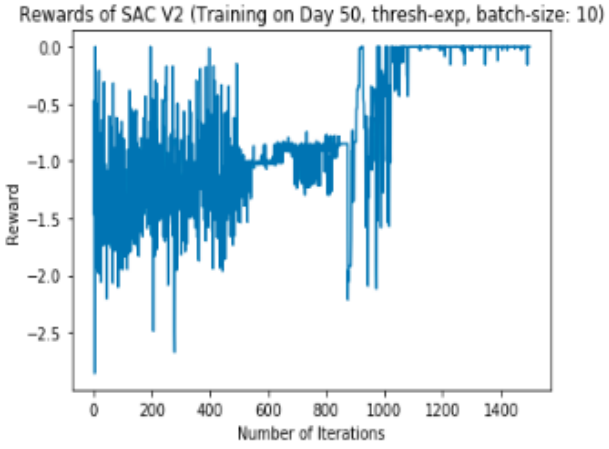

Figure 3. One example run of SAC V2 with threshold exponential response.

3.1.2. SAC

However, SAC V2 appears to be performing better. We report that it generally appears to learn across the three deterministic responses. Please see Fig 3 for a learning curve that exhibits some of the basic properties of a good learning curve: it spends time in the beginning exploring a large part of the domain, which then helps it improve. Interestingly, it tightens its search as it believes it has found an optimum; this turns out to be a local optimum and it breaks into newer parts of the domain, at which point it converges to 0, or near perfect match to the ideal demands.

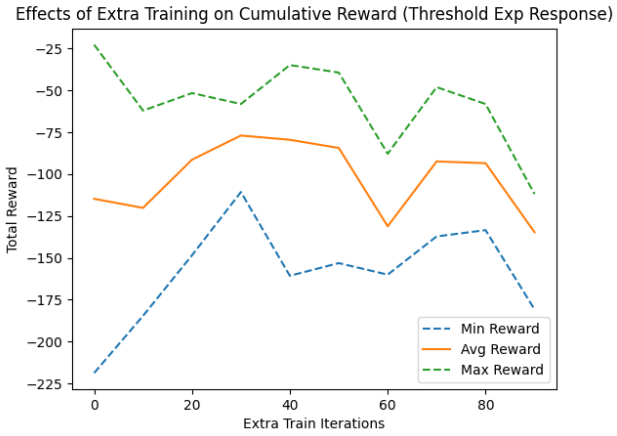

Figure 4. A summary of ten runs of the SAC V2 in an overtraining example. Here, we plot the middle green line as an average and the dotted lines as the minimum and maximum rewards noted in the runs.

Model	RMSE
GPyOpt LSTM	28
Genetic attention LSTM	30
Vanilla LSTM	31
AutoKeras Structured Regressor	50
Fourier Prediction	100
Regression	100

Table 1. Summary of results for the four planning models. (a) Refers to the synthetic Energy (a), which is the energy with several signals built in to it additively, whereas (b) refers to the synthetic Energy (b), which is the energy with additive signals as well as a latent state.

3.1.3. Overtraining Results

We believe that overtraining holds some promise in helping the RL policies train better. Although we noted general improvements to models following the implementation of more overtraining, we sought to formalize this on one of forerunners of the RL search. We present a summary of multiple training runs presented in figure 4 on the SAC of the threshold exponential response. The experiment seems to indicate that the region of 40 to 60 overtraining runs helps to produce a uniformly more positive reward in SAC on the threshold exponential response.

3.2. Planning models

Here we present the results of each individual planning model. In summary, please see the RMSE values reported in Table 1.

Parameter	Value
Validation Split	0
Network Size	512
Epochs	10
Batch Size	100
Timesteps	40
Number of LSTM Layers	2

Table 2. Optimal parameters after GPyOpt fitting completed.

3.2.1. GPyOpt LSTM

The GPyOpt LSTM produced the best RMSE of any model that we evaluated. The parameters of the optimal LSTM can be seen in Table 2.

3.2.2. Genetic Attention LSTM

For the attention LSTM, the genetic search drove a general improvement, which we noted was largely trending in the right directions.

The parameter with the greatest success was which, decoded, is an attention vector of `[0.75, 0.51, 0.87, 0.05, 0.1, 0.27, 0.68, 0.83, 0.73, 0.73, 0.95, 0.67, 0.68, 0.79, 0.16, 1.0, 0.81, 0.79]`. The orientation of this vector is: left to right, further in time to closer in time. Therefore, this implies that the attention of the net is optimized when it's directed to the timesteps that are largely closer to it, with some exceptions.

Overall the best RMSE was 30, a slight improvement to vanilla LSTM, but not one that justifies the large computational expense.

3.2.3. Vanilla LSTM

The Vanilla LSTM had a relatively high success. It trained in a small amount of time, and produced a relatively low RMSE of 31.

Please see figure 5 of sample predictions.

3.2.4. AutoKeras Structured Regressor

The AutoKeras Structured Regressor was surprisingly ineffective given the success of the other neural methods. It returned an RMSE of 50. One possible reason for the lack of performance was the structure of the nets as being simple feed-forward networks. While the algorithm was able to push down the RMSE in successive iterations, they were not able to match the LSTMs for dealing with the time series.

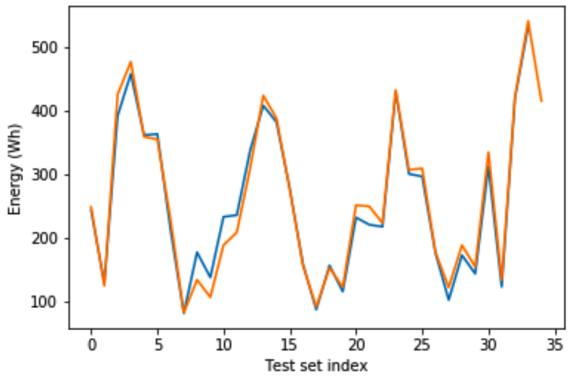

Figure 5. Predictions (blue) from the test set of the vanilla LSTM against test Energy values (orange). Plotted is Energy (a).

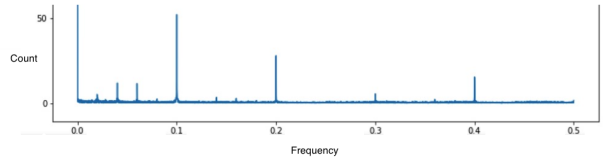

Figure 6. Frequency domain of the Fourier transform.

3.2.5. Fourier Analysis

The Fourier Analysis was surprisingly effective given the small amount of input it required. With just the Energy variable as an input, it was able to reconstruct the signal with relative ease. Please see figure 6 for a printout of the spectral analysis for the reading ease of the reader. With an RMSE of 99 on the energy (a) set and 166 on the energy (b) set, it is not nearly enough to use as a prediction by itself (plus, insofar as planning would go, incorporating points into the prediction would be a challenge) but it would be interesting to try to incorporate this into a neural model.

3.2.6. Regressions

We have results from linear regressions to serve as a baseline above which to improve. Here, our best model appears to be $y_t \sim b_t + p_t$, (please see Figure 7) and inclusion of an autoregressive term seems to strictly worsen the predictions. We believe that this is because predicting a longer horizon propogated uncertainty through the end of the prediction.

Interestingly, the regression suffered little from the transition

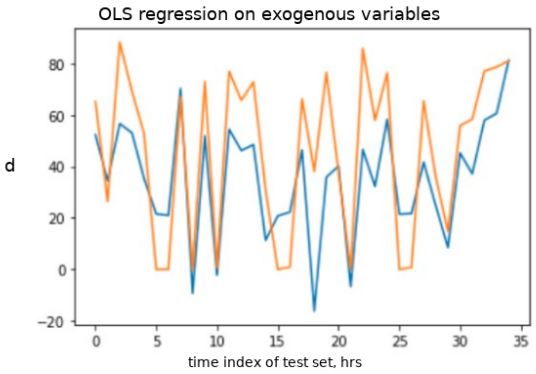

Figure 7. Prediction (blue) with actual (orange) for test set of energy demand.

from Energy (a) to Energy (b). Both register an RMSE of nearly 100.

4. Proposed Experimental timeline

We observe two experimental units for a period of five months, from August to December. We am interested in estimating the causal effects of two distinct reinforcement learning architectures, RL_j, for $j \in \{1, 2\}$, in addition to the causal effect of combining reinforcement learning with behavioural feedback from the Social Cognitive Model.

We estimate seasonality effects at period t (δ_t) and improvements in learning due to the accumulation of observations (Ω_t), by taking the average difference between performance across all conditions at time t and t_{-1}. We also estimate the effect of incorporating parameters from the social cognitive model by comparing observations *within* a single RL architecture, controlling for seasonality. Finally, we causally identify the effect of each RL_j, which is the difference between the score for RL_j versus RL_{-j}, and the difference in scores between RL_j and RL_{-j}, *conditional on incorporating the social cognitive model*, controlling for seasonality in each case.

Month	Group 1	Group 2	Control
July	— System ID —		
August	RL_2	RL_1 + plan	
September	RL_1	RL_2 + plan	

Table 3. Experimental Timeline in which we compare two different RL architectures and the effect of a planning model.

We then use later experimental periods to train the model on smaller subsets of our experimental subjects: smaller groups and then at the individual scale.

5. Discussion and Conclusion

5.1. RL Architectures

Of the three architectures, BCQL seems clearly far behind the others. We believe that an aspect of BCQL that we thought was a strength is actually a weakness: the batch that it fills to train off-policy. Because it needs to fill up the batch, it needs to explore the environment a bit before stepping or learning. Small sizes of the batch might prediscpose it to parts of the action space that are underexplored, and large sizes of the batch might mean that it is more data hungry than we thought it would be.

Our neural architecture search in the RL policies was relatively ad-hoc: larger hidden states and smaller batch sizes seems to improve the outcomes, and so we tended to gravitate towards them. However, we did not have the bandwidth to do a formal parameter search.

5.2. Planning model

In terms of technical lessons, it is unclear to us whether the computational investment in the Genetic Attention LSTM was worth it. We believe that it might have continued to trend in the decreasing direction, but the vanilla LSTM was nearly as good, and probably more flexible.

The experiments in out of the box AutoML were mixed, with the GPyOpt returning a huge improvement over the vanilla models, and AutoKeras returning an score in the middle of the pack. We hypothesize that the recurrent models that GPyOpt was optimizing with were able to push the RMSE down far, whereas the feed-forward networks in AutoKeras had a threshold of performance that was difficult to push forward from. Ultimately, we do not know how similar to real data our synthetic data was, but we hope that this project lent at least some insights on models to carry forward.

References

Ayres, I., Raseman, S., and Shih, A. Evidence from Two Large Field Experiments that Peer Comparison Feedback Can Reduce Residential Energy Usage. *The Journal of Law, Economics, and Organization*, 29(5):992–1022, 08 2012. ISSN 8756-6222. doi: 10.1093/jleo/ews020. URL https://doi.org/10.1093/jleo/ews020.

Comstock, O. and Jarzomski, K. Consumption and saturation trends of residential miscellaneous end-use loads. *ACEEE Summer Study on Energy Efficiency in Buildings, Pacific Grove, CA, USA*, 2012.

Cowley, B., Moutinho, J. L., Bateman, C., and Oliveira, A. Learning principles and interaction design for "green my place": A massively multiplayer serious game. *Entertainment Computing*, 2(2):103 – 113, 2011. ISSN 1875-9521. doi: https://doi.org/10.1016/j.entcom.2011.01.001. URL http://www.sciencedirect.com/science/article/pii/S1875952111000024. Serious Games Development and Applications.

Fernando, T., Denman, S., Sridharan, S., and Fookes, C. Soft+ hardwired attention: An lstm framework for human trajectory prediction and abnormal event detection. *Neural networks*, 108:466–478, 2018.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Jin, H., Song, Q., and Hu, X. Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1946–1956, 2019.

Knudde, N., van der Herten, J., Dhaene, T., and Couckuyt, I. Gpflowopt: A bayesian optimization library using tensorflow. *arXiv preprint arXiv:1711.03845*, 2017.

Konstantakopoulos, I. C., Ratliff, L. J., Jin, M., and Spanos, C. J. Leveraging correlations in utility learning. In *2017 American Control Conference (ACC)*, pp. 5249–5256, May 2017. doi: 10.23919/ACC.2017.7963770.

Lanzisera, S., Dawson-Haggerty, S., Cheung, H. Y. I., Taneja, J., Culler, D., and Brown, R. Methods for detailed energy data collection of miscellaneous and electronic loads in a commercial office building. *Building and Environment*, 65:170–177, 2013.

Le, H. M., Voloshin, C., and Yue, Y. Batch policy learning under constraints. *arXiv preprint arXiv:1903.08738*, 2019.

Li, Y., Zhu, Z., Kong, D., Han, H., and Zhao, Y. Ea-lstm: Evolutionary attention-based lstm for time series prediction. *Knowledge-Based Systems*, 181:104785, 2019.

Mishra, S., Bordin, C., Taharaguchi, K., and Palu, I. Comparison of deep learning models for multivariate prediction of time series wind power generation and temperature. *Energy Reports*, 6:273–286, 2020.

Papaioannou, T. and Stamoulis, G. Teaming and competition for demand-side management in office buildings. volume 2018-January, pp. 332–337, 2018. doi: 10.1109/SmartGridComm.2017.8340734. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85041942675&doi=10.1109%2fSmartGridComm.2017.8340734&partnerID=40&md5=1566057fd32f9fc730fb497d9eee4c17.

Papaioannou, T., Dimitriou, N., Vasilakis, K., Schoofs, A., Nikiforakis, M., Pursche, F., Deliyski, N., Taha, A., Kotsopoulos, D., Bardaki, C., Kotsilitis, S., and Garbi, A. An iot-based gamified approach for reducing occupants' energy wastage in public buildings. *Sensors (Switzerland)*, 18(2), 2018. doi: 10.3390/s18020537. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85041956116&doi=10.3390%2fs18020537&partnerID=40&md5=b8a4c19294b488934e08a6356c33065c.

Ratliff, L. J., Jin, M., Konstantakopoulos, I. C., Spanos, C., and Sastry, S. S. Social game for building energy efficiency: Incentive design. In *2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1011–1018, Sep. 2014. doi: 10.1109/ALLERTON.2014.7028565.

Spangher, L., Gokul, A., Khattar, M., Palakapilly, J., Tawade, A., Bouyamourn, A., Devonport, A., and Spanos, C. Prospective experiment for reinforcement learning on demand response in a social game framework. In *Proceedings of the 2nd International Workshop on Applied Machine Learning for Intelligent Energy Systems (AMLIES) 2020*, 2020.

Srinivasan, R. S., Lakshmanan, J., Santosa, E., and Srivastav, D. Plug load densities for energy analysis: K-12 schools,. *Energy and Buildings*, 43:3289 – 3294, 2011.

Zou, H., Das, H. P., Yang, J., Zhou, Y., and Spanos, C. Machine learning empowered occupancy sensing for smart buildings.