Final Project
Collaborative AI (236203)

Porter, Alon
porteralon@campus.technion.ac.il
CS Department
Technion - Israel Institute of Technology

Deshpande, Anushka
anushkad@campus@technion.ac.il
Medicine Department
Technion - Israel Institute of Technology

September 16, 2023

1 Introduction

The efficient management of energy resources is a pivotal challenge for modern urban environments. With growing populations and technological advancements, cities are increasingly adopting smart solutions for a range of utilities, one of the most crucial being energy consumption. The CityLearn challenge aims to provide a scalable and versatile environment to develop and test various machine learning and control algorithms for multi-agent demand response optimization in urban settings.

2 Related Work

In the realm of energy management, machine learning and control paradigms have been applied to various degrees of success. One particularly relevant example to our work is the use of Decision Transformers [1].

In addition to the more traditional machine learning methods, swarm and evolutionary algorithms have been employed for various optimization problems.

Genetic Algorithms (GAs) Genetic Algorithms mimic natural selection processes to evolve candidate solutions towards an optimum. In the context of energy management, GAs have been used for feature selection in demand prediction, optimal scheduling of energy resources, and more. [2].

Particle Swarm Optimization (PSO) PSO has been effective in real-time optimization and control of distributed energy resources, owing to its capability of efficiently searching high-dimensional spaces. PSO is especially useful when the objective function is expensive to evaluate, as it generally requires fewer function evaluations than other optimization techniques

3 Problem Statement

The CityLearn Challenge 2022 aims to address the issues of effective energy management in residential communities by focusing on battery storage devices and photovoltaics. The challenge involves developing machine learning agents responsible for the real-time control of battery charge and discharge in each of 17 single-family buildings in the Sierra Crest home development in Fontana, California. These agents must operate with dual goals:

1. Minimizing the monetary costs incurred from drawing electricity from the grid.

2. Reducing CO2 emissions when the grid supplies electricity to meet demand.

Performance is evaluated based on an equally weighted sum of the aforementioned goals, aggregated at the district level, over a simulation period of one year (8,760 time steps). The challenge thus asks for a holistic energy management solution that is not only cost-effective but also environmentally responsible.

4 Solution Approach

4.1 Single-Agent Case: DDPG Agent

In the single-agent scenario, we have used the DDPG algorithm [3]. Deep Deterministic Policy Gradient (DDPG) is an off-policy algorithm particularly well-suited for problems with continuous action spaces. It aims to concurrently learn an action-value function (Q-function) and a policy function.

4.1.1 Motivation

In a finite, discrete action space, one can directly compute the optimal action-value function $Q^*(s,a)$ and find the action that maximizes it. However, in continuous action spaces, computing $\max_a Q^*(s,a)$ is non-trivial. DDPG employs a gradient-based approach, approximating the optimal action by $\max_a Q(s,a) \approx Q(s,\mu(s))$, where $\mu(s)$ is a policy function.

4.1.2 The Q-Learning Side

The Bellman equation for the optimal action-value function $Q^*(s,a)$ is given by:

$$Q^*(s,a) = \mathbb{E}_{s' \sim P}\left[r(s,a) + \gamma \max_{a'} Q^*(s',a')\right]$$

The mean-squared Bellman error (MSBE) loss function to approximate $Q^*(s,a)$ is:

$$L(\phi, \mathcal{D}) = \mathbb{E}_{(s,a,r,s',d)\sim\mathcal{D}}\left[\left(Q_\phi(s,a) - \left(r + \gamma(1-d)\max_{a'} Q_\phi(s',a')\right)\right)^2\right]$$

4.1.3 Key Properties

- DDPG is off-policy.
- Designed for continuous action spaces.
- Does not support parallelization.

4.1.4 DDPG Algorithm

The DDPG algorithm consist of 3 different entities:

- Actor Critic with target for each one of them (to stabilize the proccess)
- OU-Noise - to make an exploration using the randomization of the noise in the action
- replay buffer - a buffer that stores all the transition and is used for training the agent - to avoid dependency on the sequence.

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

Figure 1: Pseudocode of DDPG Algorithm

4.2 Centralized Multi-Agent Case: Centralized CTCE DDPG Agent

For the centralized multi-agent scenario, we extend the DDPG framework [3]to create a centralized agent that manages multiple sub-agents. This centralized agent receives a global state and determines a joint action, which is then disseminated to each of the sub-agents.

- **Unified Actor-Critic**: Both the actor and critic are centralized. The critic evaluates the global utility of state-action pairs for all agents, and the centralized actor selects actions for each agent based on the global state.
- **End-to-End Control**: Unlike decentralized approaches, our CTCE setup allows the centralized DDPG agent to have control over individual agents both during training and execution. This ensures coordinated multi-agent behaviors.

4.3 Semicentralized Multi-Agent Case: Deep Implicit Graph Based Communication Network

In the decentralized multi-agent scenario, we propose a hybrid approach that incorporates elements of both local and global decision-making. This approach utilizes a Deep Implicit Communication Network [4] that includes a Graph Convolutional Network (GCN) and a Transformer-based encoder for generating communication signals.

- **Local Decision Making**: Each sub-agent employs its local DDPG algorithm for decision-making.
- **Coordination Graphs**:For several multi-agent domains, the outcome of an agent's action often depends only on a subset of other agents in the domain. This locality of interaction can be encoded in the form of a coordination graph (CG). A CG is often represented as an undirected graph $G = \langle V, E \rangle$

and contains a vertex $v_i \in V$ for each agent i and a set of undirected edges $\{i, j\} \in E$ between vertices v_i and v_j.

The action-value function can be factored as:

$$q_{\mathrm{CG}}(s_t, a) = \sum_{v_i \in V} f_i(a_i|s_t) + \sum_{i,j \in E} f_{ij}(a_i, a_j|s_t)$$

In this work, we forgo explicitly computing the joint action through inference over factored representation with a given coordination graph. Instead, we use attention to learn the appropriate agent observation-dependent coordination graph structure with soft edge weights and then use message passing in a graph neural network to compute appropriate values or actions for the agents.The GCN has been implemented as found in [5]

- **Communication Network**: The network generates communication signals that are rich in mutual information and serve to coordinate the actions of the sub-agents.

  1. **Transformer Encoder**: Takes observations from all agents and processes them to generate essential feature embeddings. The self-attention mechanism can be mathematically represented as:

  $$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

  In detail, we first pass n observations $\{o_i\}_{i=1}^n$ of the n agents through a parameter sharing encoder parameterized by θ_e. The encoder outputs n embedding vectors $\{e_i\}_{i=1}^n$, each with size d:

  $$e_i = \mathrm{Encoder}(o_i; \theta_e), \quad \text{for } i = 1, \ldots, n. \quad (3)$$

  We then compute the attention weights from agent i to j using these embeddings as:

  $$\mu_{ij} = \frac{\exp(\mathrm{Attention}(e_i, e_j, W_a))}{\sum_{k=1}^n \exp(\mathrm{Attention}(e_i, e_k, W_a))}. \quad (4)$$

  where the attention module is parameterized by W_a, which is a trainable $d \times d$ weight matrix. The attention score function we adopt is general attention:

  $$\mathrm{Attention}(e_i, e_j, W_a) = e_j^\top W_a e_i.$$

  2. **Graph Convolutional Network (GCN)**: Takes these embeddings and processes them to produce output embeddings that serve as communication signals. A single layer of a GCN can be represented as:

  $$H^{(l+1)} = \sigma\left(D^{-1/2} A D^{-1/2} H^{(l)} W^{(l)}\right)$$

  where A is the adjacency matrix, D is the degree matrix, $H^{(l)}$ are the node features at layer l, $W^{(l)}$ are the layer parameters, and σ is the activation function.

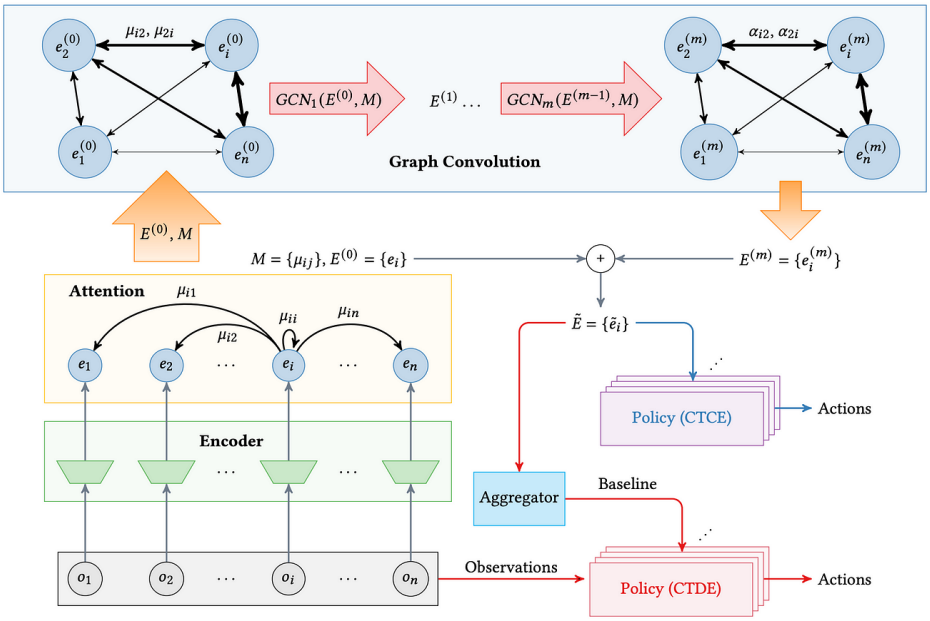

Figure 2: actor loss after running a lot of training steps

5 Formal Analysis

In this section, we examine the performance of three distinct types of agents: decentralized, centralized, and semi-centralized. The aim is to understand the convergence behavior of the actor and critic losses.

5.1 Decentralized Agents

In the case of decentralized agents, we observe that each actor's loss does not converge to a specific value but shows a tendency to stabilize.While the actor and critic losses converge towards zero, indicating effective learning, the Key Performance Indicators (KPIs) provide a more nuanced picture of the agent's performance.

The Load Factor, Carbon Emissions, Cost, Electricity Consumption, and Zero Net Energy metrics are all relatively close to 1. This suggests the agent is performing near the baseline levels for these metrics, indicating moderate performance with some room for improvement.

The Load Factor is at an impressive 0.99, very close to the ideal value of 1. This indicates that the electrical system is being used very efficiently, with minimal wastage of resources.

Similarly, the Peak Demand metric is almost 1, suggesting that the agent has effectively managed peak electricity demand to align with the grid's capacity, thus minimizing the risk of grid failures or the need for peaker plants.

In summary, while the agent shows promise in several areas, there are specific metrics that indicate the need for further refinement to achieve optimal performance.

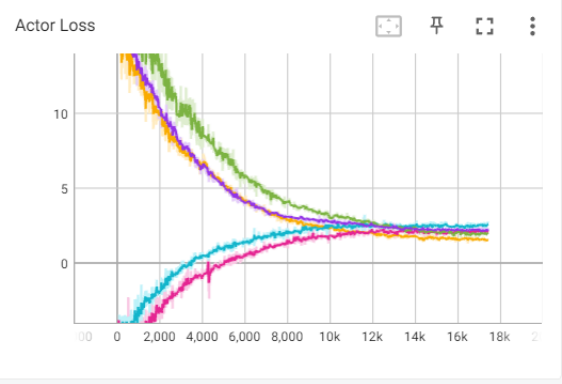

Figure 3: Actor Losses for Decentralized Agents

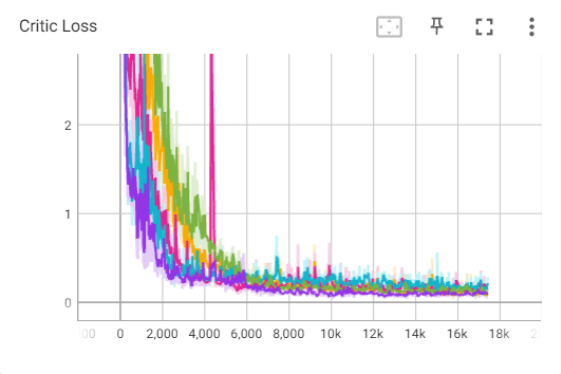

Figure 4: Critic Losses for Decentralized Agents

5.2 Centralized Agents

For our centralized agent, we observe distinct behavior between the actor and critic networks during the learning process. While the critic loss converges towards zero, suggesting an effective learning of the value function, the actor loss starts from a negative value and eventually stabilizes at 2. This divergence in loss behaviors could imply that while the critic network successfully learns to evaluate the state-action pairs, the actor network faces challenges in policy optimization.

The Load Factor for the centralized agent is at a near-ideal 0.99, indicating high efficiency in electrical system usage. Other KPIs, such as Average Daily Peak, Carbon Emissions, Cost, and Zero Net Energy, are closely aligned with those of the decentralized agent and hover around values indicating near-baseline performance.

Interestingly, despite the non-converging actor loss, the centralized agent still achieves similar KPIs as the decentralized agent. This could suggest that the non-zero stabilization of the actor loss may not significantly impede the system's overall performance in the CityLearn environment. Nonetheless, the divergence between the actor and critic loss behaviors suggests that further investigation may be beneficial for understanding the agent's policy optimization challenges.

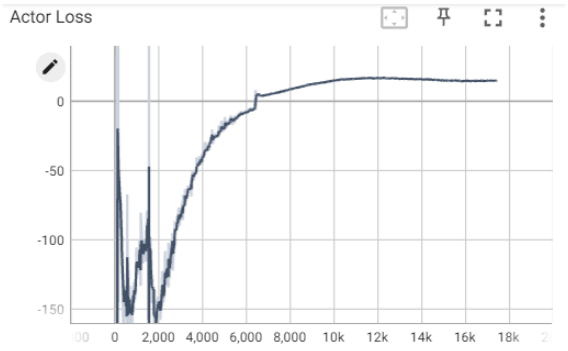

Figure 5: Actor Losses for Centralized Agents

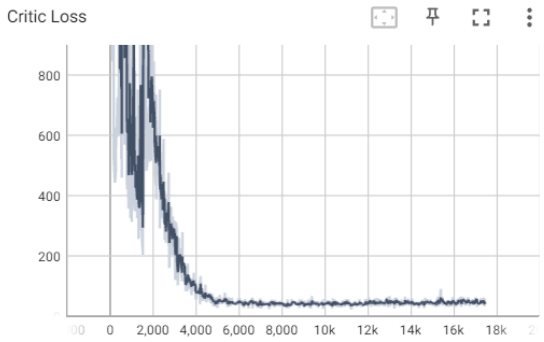

Figure 6: Critic Losses for Centralized Agents

5.3 Semi-Centralized Agents

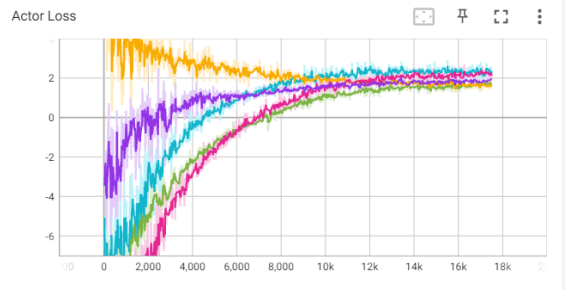

Figure 7: Actor Losses for Semi-Centralized Agents

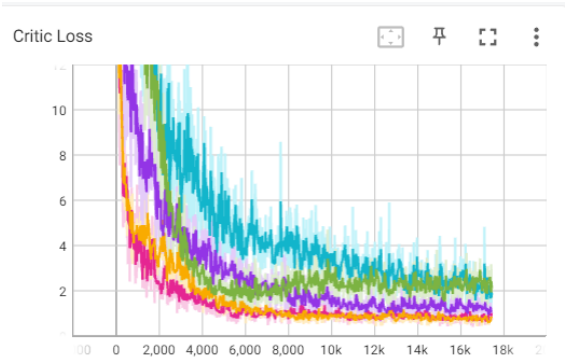

Figure 8: Critic Losses for Semi-Centralized Agents

In the semi-centralized scenario, we observe an interesting pattern where all actor losses stabilize at a value of 2, while the critic losses converge towards zero. This behavior contrasts with the fully decentralized setting, where both actor and critic losses converge to zero, and the fully centralized case, where the actor loss also stabilizes but at different values.

The Key Performance Indicators for the semi-centralized agent are quite similar to those for both the centralized and decentralized agents, hovering around near-ideal or baseline values.

What is particularly interesting is that despite the actor losses stabilizing at 2, the system still achieves strong performance metrics. This could indicate that the value to which the actor loss is converging may not be a critical factor for system performance in this specific environment. It also suggests that the actor network, despite not reaching a zero loss, has found a policy that works well in practical terms, as evidenced by the KPIs.

Another noteworthy development was observed when additional layers were added to the transformer encoder within the communication network. This architectural change led to a substantial decrease in the loss for all agents, both in the actor and critic networks.

The lowering of loss across all agents signifies several potential advantages:

- **Enhanced Learning Capability**: Additional layers in the transformer encoder likely contribute to a more expressive model, enabling more complex representations of the state and action spaces, and consequently better policy and value function approximations.
- **Uniform Improvement**: The universal decrease in loss across all agents implies that the improvements are not localized to specific agents but benefit the system as a whole. This is critical for maintaining system-wide performance and stability.
- **Synergistic Effects**: Given that the transformer is part of the communication network, the improvement in loss metrics may indicate not just better individual performance but also improved coordination and information sharing among agents.

	Decentralized agents	Centralized agent	Semi-Centralized agnet
1 - load factor	0.999	1.003	1.000
Carbon emmisions	1.003	1.016	1.000
Cost	1.002	1.013	1.000
Electricity consumption	1.004	1.015	1.000
Peak demand	1.000	1.035	1.000
Ramping	1.008	1.065	1.000
Average daily peak	1.002	1.016	1.000
Zero net energy	1.006	1.013	1.001

Table 1: comparing between the values of the evaluation of each method

6 Empirical Evaluation

Since most of the evaluation for the previous winner of city learn challenge is not published, we can compare our evaluation score to the ones that were published in the NeurIPS website - Emmision cost, Price cost. As we saw the values for the Emission was about 0.7-0.8 for the first 10 participants and in our 3 settings was around 1.0 we can assume we have more work to do in order to make it more efficient and not so close to the base line. as for the Price we see that the first 10 place of last year challenge got values of 0.6-0.7 and for us also around 1.00. We still want to make a comparison between the 3 methods we ran in the KPI's that related to the assignment (all are district level values from env.evaluate()) on the table above.

We can see from the table above that the semi centralized made a better performance from the Decentralized that had better performance from then the Centralized which is due to the inter agent communication.

7 Future Work

While we have made substantial progress, there are several promising avenues for future research. Exploring other architectural nuances within the transformer model might yield even more optimized results. This could involve experimenting with different types of attention mechanisms or position encodings. As urban environments grow in complexity, scaling the multi-agent system to accommodate a greater number of agents while maintaining performance is a vital challenge. Implementing models that can dynamically adjust their architecture based on the task or environment could be a groundbreaking direction. This might involve meta-learning or other forms of self-optimizing models.

A crucial extension to the current work would involve incorporating non-homogeneous agents, possessing different capabilities or objectives, to better simulate real-world complexity. Further research can also be dedicated to examining the communication network more closely. This would include identifying whether more informative signals could be passed between agents, and perhaps even incorporating these signals directly into the reward function. Employing advanced optimization techniques to fine-tune the reward parameters could also be invaluable for improving agent behavior and system performance. Finally, employing the CTDE approach in the centralized case could provide new insights into the scalability and efficiency of such systems.

In conclusion, our findings underscore the importance of both model architecture and the nuances of agent communication in multi-agent systems. Future advancements in these domains have the potential to further revolutionize urban energy management in environments like CityLearn.

Bibliography

[1] Emilio Parisotto and Kavosh Asadi. "Decision Transformers". In: URL: https://proceedings.neurips.cc/paper/2021/hash/7f489f642a0ddb10272b5c31057f0663-Abstract.html.

[2] Ramin Torkan, Adrian Ilinca, and Milad Ghorbanzadeh. "A genetic algorithm optimization approach for smart energy management of microgrids". In: 2022. DOI: 10.1016/j.renene.2022.07.055.

[3] Timothy P Lillicrap et al. "Continuous control with deep reinforcement learning". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2016.

[4] Sheng Li et al. "Deep Implicit Coordination Graphs for Multi-agent Reinforcement Learning". In: 2021. arXiv: 2006.11438 [cs.LG].

[5] Thomas N. Kipf and Max Welling. *Semi-Supervised Classification with Graph Convolutional Networks*. 2017. arXiv: 1609.02907 [cs.LG].