Duelling DQNs for Datacenter Cooling

Joseph Islam, Peyton Gozon, Aadarsha Gopala Reddy

12 December, 2024

1 Introduction

With deep learning's popularity soaring in the 2010s, and with companies increasingly turning towards big data, data centers have increasingly become part of the functioning of the modern world. As data center usage has grown, so has the associated energy consumption – both in terms of operating the computers and cooling the environment. This growing energy consumption has highlighted the need to reduce operational costs and environmental impact. Although large-scale data centers have been studied extensively, small- to mid-sized data centers remain understudied – despite occupying 42.5% and 19.5% of the market share, respectively [2].

As machines within a data center complete their tasks, they generate heat; with machines occupying physical spaces, this creates a complex spatial cooling problem within the data center. Special care also must be taken to account for the external (changing by season) weather of the data center. Given the spatial distribution of machines and data centers often having multiple cooling elements distributed throughout the space, this becomes a complex control task.

Deep Reinforcement Learning (DRL) represents a promising technique to address these issues. With its ability to dynamically adapt to changing conditions, such as machine workload and external temperature, and its ability to learn from continuous feedback, DRL offers flexibility that's lacking in static rules-based methods. Furthermore, given DRL's capacity for robust decision-making, and its ability to account for non-linear relationships among sensor readings and uncertainties about future workloads, it seems poised to outperform control-based methods.

This project leverages Deep Reinforcement Learning (DRL) to optimize the energy efficiency of cooling small- to mid-sized data centers. We provide a comparative analysis of Proximal Policy Optimization (PPO), Soft Actor-Critic (SAC), and Dueling Deep Q-Networks (DDQN) in this domain, and compare their efficacy against static rules-based and control-theoretic baselines.

Key contributions include:

- A focus on small- to mid-sized data centers
- A novel exploration on DDQN within for data-center cooling
- Comparisons between DRL methods with a rules-based and control-theoretic controller

2 Related Works

The data center cooling research and practice currently seem immature. Small data centers occupy approximately **42%** of the data center market [2], which is expected to grow by **12.5%** in the next eight years [1]. While Google DeepMind [6] and DeepEE [10] have explored cooling optimization for large data centers, Dominkovic et al. [5] acknowledge that addressing market growth's threat to decarbonization goals needs more research in small-to-mid-sized data center cooling optimization. Contrastive studies featuring DQN, BDQ, and DDPG [12], model-based reinforcement learning, and PPO [15], SAC [4], as well as separate frameworks like Cool.Ai [9] have been researched, but overlook common methods like Duelling Deep Q Networks, which we explore. We leverage the EnergyPlus simulation environment [3] via the Sinergym Python wrapper [13], as demonstrated in prior work [7], to evaluate the applicability of SAC, DDQN [14], and PPO for small to mid-size data center cooling systems to review alternative DRL algorithm's applicability to data center cooling.

3 Background & Preliminaries

Data centers are large facilities housing computing devices that generate heat as they process workloads. Maintaining suitable cooling conditions is critical, both in terms of operational efficiency and hardware longevity, but also for occupant safety. Finding an energy-efficient cooling strategy, however, is a complex challenge due to the non-linear and time-dependent interactions between server workloads, environmental conditions, cooling elements, and the spatial relationship between them.

Reinforcement Learning (RL) provides a promising framework for addressing this challenge. In RL, an agent interacts with an environment, which is modeled as a Markov Decision Process (MDP). An MDP is defined as a 4-tuple, $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ defines the transition function mapping $(s, a) \mapsto s'$, and R defines the reward function. At time step t, the agent observes state $s_t \in \mathcal{S}$, and selects action $a_t \in \mathcal{A}$ according to the transition function, and receives reward r_t reflecting how good the resulting state is. We seek to find a policy, that is an optimal set of actions for each state, such that we maximize the cumulative rewards over time. Future state rewards are discounted by γ.

In the data center cooling problem, states correspond to the readings of sensors placed both in and around the facility – such as internal or external thermometers, humidity sensors, and wind direction and speed information. Additional information might also be added to this vector, such as (remaining) workloads of machines.

Actions, in this setting, correspond to interactions with cooling elements throughout the facility – such as turning on or off HVAC systems, or setting fan speeds. Ideally, the action taken a_t utilizes the cooling elements efficiently.

The problem then, at its core, is to uncover the model of the data center cooling problem, $T(s, a)$. This can be highly complex, especially as the spatial layout of the sensors, computers, and cooling elements are considered – not to mention the non-linear relationship with humidity, outdoor temperature, and the time-delay between activating an HVAC system and observing the temperature be corrected. Recovering T is thus the crux of the problem.

Many algorithms exist to facilitate this – in this paper, we will investigate Proximal Policy Optimization (PPO), Soft Actor-Critic (SAC), and Dueling Deep Q-Networks (DDQN) to

implicitly learn this model. Before we can investigate these methods, we must also consider how to evaluate the current state of the environment. This is defined as R_t, or the "Reward" at the current time step t. For the purposes of this exploration, we consider a reward function, $R(s_t)$, that balances between energy consumption and the difference between the current temperature and a "comfortable" temperature. The specifics of how these are balanced will be addressed in the Methods section.

We specifically choose to explore the novel application of DDQN to datacenter cooling in the context of already applied DRL algorithms PPO and SAC to provide an elementary contrastive study accessible to industry practitioners trying to optimize their data center cooling.

4 Method

We first discuss the simulator setup (Section 4.1), then our RL problem formulation (Section 4.2), followed by implementation details for each algorithm (Section 4.3), baselines (Section 4.4), real-world training and computer-environment details (Section 4.5).

4.1 Simulator

We use the `Sinergym` Python package to simulate the dynamics of a small datacenter. `Sinergym` is a Gymnasium-compatible wrapper for `EnergyPlus`, which is "a whole building energy simulation program that engineers, architects, and researchers used to model ... energy consumption ... in buildings" [3]. Having this tool accessible through a Gymnasium-compatible wrapper allowed our group to adapt to this complex domain with a familiar programming context.

We chose to use `Sinergym`'s built in `Eplus-datacenter-mixed-continuous-stochastic-v1` environment. According to Sinergym's documentation [13], this environment simulates a "491.3 m^2 building, divided into two asymmetrical zones: the west zone and the east zone. Each zone is equipped with an HVAC system. The primary heat source in the building is generated by the hosted servers".

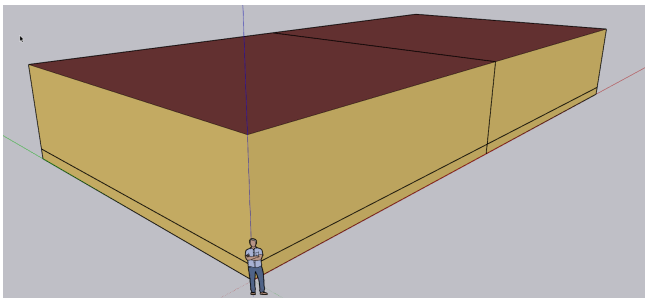

Figure 1: Sinergym 2-Zone Datacenter with HVAC

This environment, being "mixed," will choose between hot and cold temperature data to be simulated on a given run. This environment is also stochastic, meaning that weather fluctuations are amplified by an additional 1.5 standard deviations of normal conditions. This was chosen to ensure our DRL algorithms work across a robust set of conditions.

By default, `Sinergym` utilizes a linear reward function. This is defined as,

$$R(t) = -W \cdot \lambda_E \cdot \text{Power}_t + (1 - W) \cdot \lambda_T * (\max(T_t - T_{\text{low}}, 0) + \max(T_{\text{upper}} - T_t), 0)$$

where, $\lambda_E = 1$ is an energy-specific weighting term, $\lambda_T = 1$ is an energy-specific weighting term, and $W \in [0, 1]$ considers the balance. T_{lower} and T_{upper} are the lower- and upper-temperature bounds considered comfortable, with T_t being the current temperature.

As a simplifying assumption, we only train our algorithms across the period of June 1st to August 31st in a given year. This was chosen as it represents the period with the greatest volatility in terms of temperature, making rewards less sparse.

By default, the weather file used by `EnergyPlus` used weather details sampled from JFK International Airport.

4.2 Reinforcement Learning Setup

By default, the state space of this environment is 28-vector of different environment properties, including the internal temperature of both the west and east zones, as well as their humidities, and the outdoors conditions. See Appendix 8.1 for a complete overview of state-space variables.

During the ablation study, we also wrap the environment in `Sinergym`'s `WeatherForecastingWrapper`. This provides weather information three days into the future, with each day being represented as another six floats appended onto the original state space; thus, when the wrapper is enabled, our state space becomes a 46-vector.

The action space is a 2-vector, codified as the $\begin{bmatrix}\text{lower-threshold} & \text{upper-threshold}\end{bmatrix} \in [15, 22] \times [22, 30]$ sets the temperature control boundaries. If the indoor temperature (in a given zone) falls below the lower-threshold, heating initiates; if it rises above the upper-threshold, cooling is activated. Each of these values are temperatures in Celsius. When an action is applied to the environment, both the HVACs controllers are given the action's temperature control boundaries.

4.3 Algorithmic Details

We implemented PPO, SAC, and DDQN DRL algorithms for this problem. However, we made alterations to each algorithm to make it work within the environment.

4.3.1 PPO Implementation Notes

PPO is implemented with two implicit neural networks, a Policy Network and a Value Network. Both networks are feed-forward sequential networks with ReLU non-linearity, with the Policy Network returning a mean and standard deviation (defining a policy distribution) for a given state; the Value Network estimates the value of the state. Both are optimized using Adam.

Internally, each element of the action vector is scaled to $[-1, 1]$ to improve stability by tanh-squashing. Actions are rescaled to the bounds identified above whenever they are used to interact with the environment.

After consulting the literature, we identified Generalized Advantage Estimation (GAE) as a way to improve training stability. This method uses the TD(λ) algorithm to calculate a λ-return to compute the Generalized Advantage Estimator, GAE(λ) [11]:

$$\hat{\mathcal{A}}_t = \text{TD}(\lambda) - V(s_t)$$

We found the details in Appendix (A) of [8] particularly insightful.

Exact learning rate, clip ratios, mini-batch sizes, and neural network architectures are modified in each experiment.

4.3.2 SAC Implementation Details

Our implementation of SAC utilizes a single actor with two critic networks to improve stability with training. Just like in PPO, the policy network computes the mean and standard deviation of the policy distribution for a given state through a feed-forward network with two heads. The critic networks are also feed-forward networks Q-networks with ReLU activation functions. All three networks are optimized using Adam.

SAC, similar to our PPO implementation, uses tanh-squashing to improve training stability internally; all interactions with the environment are scaled to the environment's action space.

We also implemented automatic entropy tuning to balance exploration vs exploitation, making SAC perform more effectively in our environment. (In fact, we found SAC performed poorly without it). We optimized this parameter using Adam as well.

We use soft-updates with $\tau = 0.005$ for each minibatch update.

4.3.3 DDQN Implementation Details

This is perhaps the most interesting. DDQNs, given a state, choose between a series of discrete actions; however, our environment is continuous.

To overcome this, we discretize the action space, converting the domain of the lower-threshold and upper-threshold into a series of five bins of equal width; this means $|\mathcal{A}| = 5 \times 5 = 25$ for the DDQN.

The DDQN is defined very similarly to its original paper [14]. However the original paper is designed to operate on the Atari Gym environment. To this end, we took liberty with designing the feature stream as a feed-forward neural network with one hidden layer, rather than the series of 2D-convolutional layers depicted in the original paper. Faithful to the original implementation, we then use the feature-stream to compute two separate streams – a value stream and an advantage stream – before combining the values together again to compute the Q-values for each action.

We used a linearly decaying epsilon GLIE policy to handle the exploration vs exploitation trade off.

4.4 Baseline Details

We explored three baselines, each available through `Sinergym` by default.

As is a common baseline, we evaluated the `RandomController`, which samples a random action from the environment's action space at each time step.

For a static rule-based baseline, we utilized the `RBCDatacenter` controller (Rules-based Controller for the Datacenter environment). This always selects an action of $(18, 27)$ celsius, corresponding to The American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) guidelines.

For a control-theoretic approach, we used the `RBCIncrementalDatacenter` controller. This begins by using the `RBCDatacenter` Controller's action of $(18, 27)$, but varies the upper- and lower-thresholds according to the average temperature of the west- and east-zones in the data center. If the mean indoor temperature exceeds the 'upper-threshold', both the lower- and upper-thresholds are decreased by one degree centigrade; conversely, if the lower-threshold exceeds the mean indoor temperature, both the lower- and upper-thresholds are increased by one. This controller stabilizes internal temperature in an energy-efficient way through a series of one-degree increments to the temperature set points

4.5 Experimental Infrastructure

We implemented all our models using PyTorch, performed model convergence analysis using Tensorboard. Our environment specifics are a fresh Conda 3.1.2 environment running Python 3.12, torch 2.5.1, tensorboard 2.18.0, wandb 0.19.0, gymnasium 1.0.0, sinergym 3.7.0 (installed directly via GitHub [13], and EnergyPlus 24.2.0.

We ran our models across 14 M2 Mac Minis, usually overnight. We analyzed the average per-step reward for evaluation, but looked at the total episode reward to understand model performance while training; as one full episode is consistently defined as 6624 steps, we could convert between these values as necessary.

Figure 2: Training Overnight. Ironic how we made a data center, while learning how to cool them.

5 Experiments

We conceptualized the experiments as three separate investigations into data center cooling, one for PPO, SAC, and the DDQN. Our primary goal in running these experiments was to explore whether they would outperform the Random Baseline, the static Rules-Based Baseline, and the control-theoretic Rules-Based Incremental Baseline.

All experiments were conducted in the simulated data center environment, across the time period of June 1 to August 31.

We began by performing a partial grid search with refinements to identify key model parameters that would lead to model convergence. Once these models were identified, we performed

hyperparameter tuning on the models. We evaluated models in terms of their average per-timestep reward during an evaluation setting. Model performance was averaged across 20 evaluation episodes. Models were evaluated every 10 episodes while training.

These are the parameters domains evaluated for each model:

5.0.1 PPO

- learning rate $\in [10^{-4}, 3 \cdot 10^{-4}, 10^{-5}]$
- learning rate scheduler $\in [\text{Linear}, \text{Exponential}]$
- number of hidden layers $\in [2, 3, 4]$
- nodes per hidden layer $\in [64, 128, 256]$
- batch size $\in [64, 128]$
- clip ratio $\in [0.1, 0.2, 0.3]$
- Include 3-day Weather Forecasting? $\in [\text{Yes}, \text{No}]$

We also specified $\lambda = 0.95$ for computing GAE. PPO was evaluated for 50 epochs, at 6624 steps per episode.

5.0.2 SAC

We chose to limit our exploration of SAC to neural networks with two hidden layers, kept the soft-update parameter $\tau = 0.005$ fixed. We also instantiated the Replay Memory to have a buffer capacity of $1,000,000$.

- learning rate $\in [10^{-3}, 3 \cdot 10^{-3}]$
- learn entropy? $\in [\text{Yes}, \text{No}]$
- $\gamma \in [0.99, 0.995]$
- nodes per hidden layer $\in [128, 256]$
- number of initial episodes to sample for replay memory $\in [3, 5]$
- target model update frequency $\in [25, 50]$
- batch size $\in [64, 256]$

SAC was evaluated for 100 epochs.

5.0.3 DDQN

For the DDQN, we chose to vary the following parameters:

- learning rate $\in [10^{-3}, 5 \cdot 10^{-4}, 2.5 \cdot 10^{-4}, 10^{-5}]$
- steps until max learning rate decay $\in [50000, 100000, 200000]$

We fixed the initial epsilon, used for the GLIE policy, to be 1.0 with us linearly decaying to 0.1 over the steps specified above. We fixed $\gamma = 0.99$, the network architecture to have 64-nodes per hidden layer. The replay memory, like in SAC, had a maximum capacity of $1,000,000$. We

performed a hard model update every 10 episodes. The DDQN was evaluated for a maximum 300 episodes, or until it achieved a performance of −12300; note, this level of performance beat all the baselines.

5.0.4 Ablation Study

Finally, we chose to perform an ablation study on our novel contribution, using DDQN for data center cooling. In this study, we chose four well-performing models from the DDQN subsection, and decided to see whether including a three-day weather forecast improved model performance. We chose to study the inclusion of the weather forecasting information on the DDQN in particular, partially as a continuation of the work done for PPO, but also because the DDQN models are drastically simpler than PPO or SAC, outperformed PPO and SAC, and deserve more coverage since they're our novel contribution.

5.1 Results and Analysis

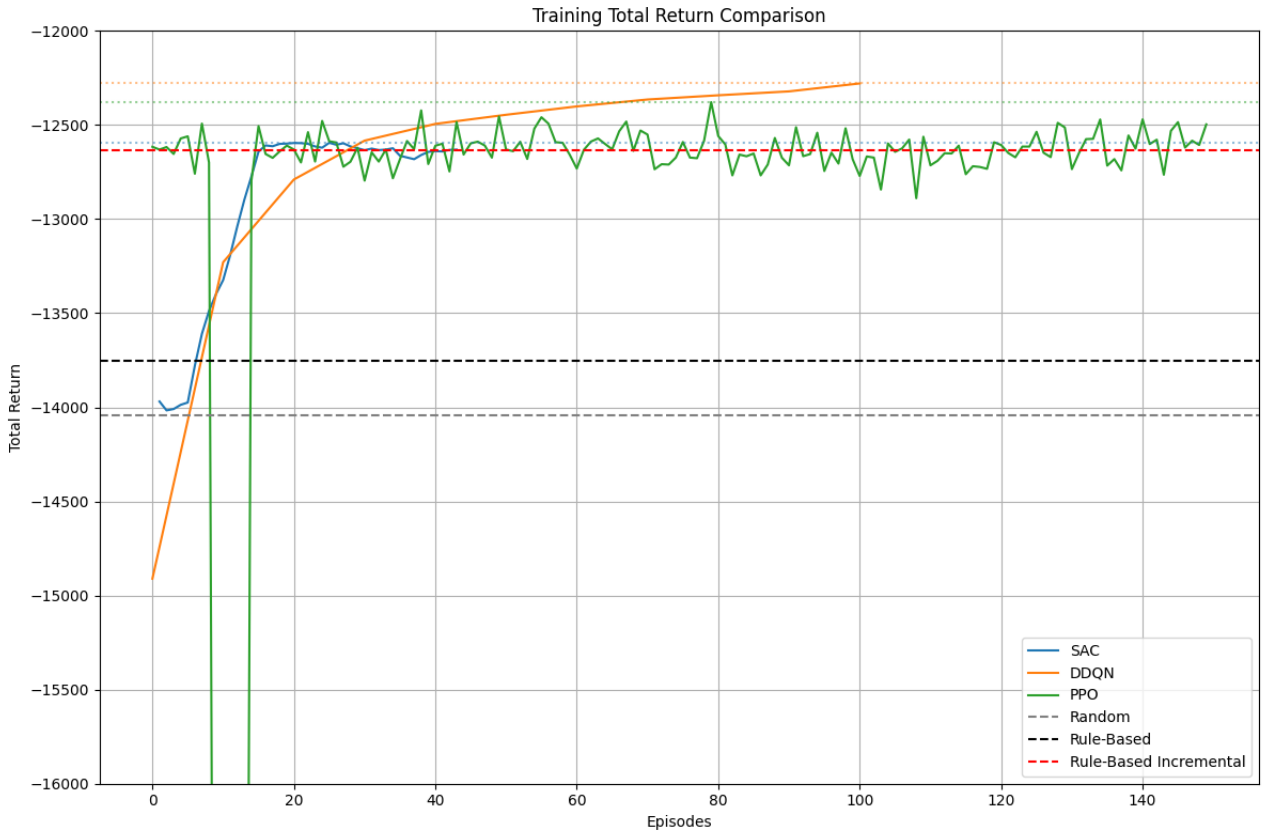

Figure 3: Models against baselines on small mixed continuous stochastic energyplus data center.

Figure 3 shows the training curves of our models, contextualized by the performance of the baseline models averaged across the episodes. Table 1 shows the cumulative final rewards across episodes and averaged by days for these models. The percentage column describes the percentage improvement our model had beyond the improvement that RBC-INC made over the random baseline. We expected our PPO and SAC to perform comparably and exceed our baselines. SAC surpassed RBC-INC by .284%, but PPO exceeded RBC-INC's improvement by 10.9%, contradicting our comparability hypothesis. This may be due to insufficient grid

Table 1: Final Performance by Model

Model Name	Total Reward	Average Reward	%
DDQN	-12130	-1.835	35.8
PPO	-12496	-1.893	10.9
SAC	-12630	-1.916	.284
RBC-INC	-12634	-1.907	0
RBC	-13753	-2.08	-
Random	-14042	-2.2	-

searching for SAC. Most notably and surprisingly, our DDQN adaptation far surpassed PPO, at a 35.8% improvement over RBC-INC's. We suspect this could be because of DDQN's relative simplicity compared to PPO's, but this suggests that model free DRL methods offer significant potential improvement over traditional rules based approaches for real datacenters. Furthermore, small to mid sized datacenters struggling with high setup costs may find DDQN to be cheaper and better than PPO and SAC at the cost of slightly lost granularity, but the returns offset this concern for simpler environments, like our simulator. With richer information found in real world settings such as server thermal data, PPO and SAC offer promise for real world applications in their ability to handle more complex state spaces than our relatively simple DDQN, and given the cost constraints smaller data centers face, especially PPO. Most importantly, the novel application of DDQN to datacenter cooling demonstrates great promise.

Actionable hyperparameter takeaways include clip ratios at most .1, 64x64 node network, linear learning rate scheduling, and a batch size of 64 for optimal PPO results in our method. We found that a learning rate of .0005 decayed over 50k timesteps with a gamma of .99 yielded optimal DDQN performance, and a gamma of .99, tau of .005, alpha of .2, learning rate of .0003, (256, 2) network architecture, learnable alpha, and automatic entropy tuning yielded the best SAC for our approach.

5.2 Ablation Study: Impact of Weather Forecasting

To assess the impact of weather forecasting on model performance, we conducted an ablation study where we trained the DRL agents with and without access to weather forecasts to assess the necessity of weather data for performance gain.

Analysis:

In our ablation study, group A and B, the red and blue lines, respectively feature DDQNs with and without 3 day weather forecast data. Group 1, 2, 3, and 4 use learning rates of .001, .00025, .00025, and .0001 with 50000, 200000, 50000, and 50000 step learning rate decays, respectively. Our results suggest that while incorporating 3 day weather forecasting usually introduces instability and restricts model performance and convergence, more work to tell the merits of weather forecasting data seems due. This doesn't necessarily imply that such forecasts are ineffective in real-world scenarios either. While weather forecast data may complicate cooling optimization in our specific DRL frameworks, it could still prove valuable for other related tasks, such as cost and workload forecasting.

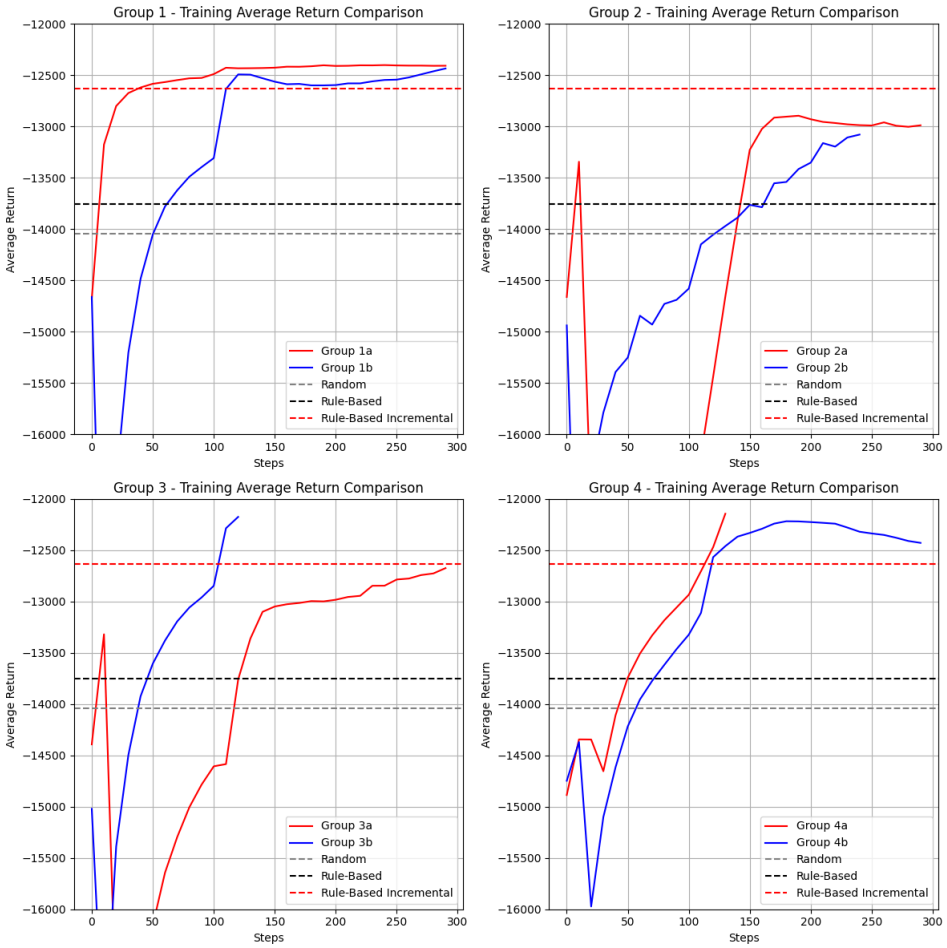

Figure 4: Models without weather data usually outperform those with

6 Conclusions

This paper sought to contextualize a new application of Dueling DQN's for data center cooling by offering a basic contrastive study between the a standard DDQN, for this, a GAE modification PPO implementation, and SAC with automatic entropy tuning. While the latter two algorithms are common and work well, DDQN's performance, slightly faster convergence time, and relative simplicity make it an excellent contender for real life data center cooling applications. DDQN's far exceed standard performance for little expert tuning and involvement, making them an excellent choice for data center cooling optimization amidst the high and increasing data center demand. If not DDQN's, PPO with GAE modifications remain a strong choice, but SAC may be more poorly fit for real world application due to steep training and tuning requirements a growing data center may not be able to fund. Weather forecasting data unlikely helps performance, but more research is needed for conclusive results.

7 Limitations & Future Works

Our project focused on model-free methods in datacenter cooling optimization. Aforementioned complementary model based methods show promise beyond model free methods and deserve further exploration. Our PPO implementation remained true to theory, but further tailoring to the problem could offer comparable performance to the DDQN. Finally, adding gradient

flow based entropy management techniques to our SAC algorithm could improve and hasten convergence in SAC. Future work in these directions would sizably support data center owners and achievement of decarbonization targets down the line.

References

[1] Data center infrastructure market analysis. https://www.gminsights.com/industry-analysis/data-center-infrastructure-market. Accessed: December 2024.

[2] Data center stats. https://brightlio.com/data-center-stats/. Accessed: December 2024.

[3] Energyplus: Building energy simulation software. https://energyplus.net/. Accessed: December 2024.

[4] Zhiwei Cao, Ruihang Wang, Xin Zhou, and Yonggang Wen. Toward model-assisted safe reinforcement learning for data center cooling control: A lyapunov-based approach. In *Proceedings of the 14th ACM International Conference on Future Energy Systems*, e-Energy '23, page 333–346, New York, NY, USA, 2023. Association for Computing Machinery.

[5] Dominik Franjo Dominković, Claire Marie Bergaentzlé, Gerald Englmair, Simon Furbo, Henrik Madsen, Rune Grønborg Junker, Niclas Brabrand Brok, Rikke Stefansen, and Hanne Kokkegård. Cool-data: Flexible cooling of data centers. https://orbit.dtu.dk/en/projects/flexible-cooling-of-data-centers. Accessed: December 2024.

[6] Richard Evans and Jim Gao. Deepmind ai reduces google data centre cooling bill by 40%. https://deepmind.google/discover/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-by-40/, 2016. Accessed: December 2024.

[7] Antonio Manjavacas, Alejandro Campoy-Nieves, Javier Jiménez-Raboso, Miguel Molina-Solana, and Juan Gómez-Romero. An experimental evaluation of deep reinforcement learning algorithms for hvac control. https://arxiv.org/pdf/2401.05737, 2024. Accessed: December 2024.

[8] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)*, 37(4):143:1–143:14, 2018.

[9] qatshana. Cool.ai: A reinforcement learning framework for data center cooling optimization. In *Proceedings of the IEEE Conference*, 2023.

[10] Yongyi Ran, Han Hu, Xin Zhou, and Yonggang Wen. Deepee: Joint optimization of job scheduling and cooling control for data center energy efficiency using deep reinforcement learning. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 645–655, 2019.

[11] John Schulman, Philipp Moritz, Sergey Levine, Michael I Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[12] H. Francis Song, Abbas Abdolmaleki, Jost Tobias Springenberg, Aidan Clark, Hubert Soyer, Jack W. Rae, Seb Noury, Arun Ahuja, Siqi Liu, Dhruva Tirumala, Nicolas Heess, Dan Belov, Martin Riedmiller, and Matthew M. Botvinick. Comparison of deep reinforcement learning algorithms in data center cooling management: A case study. In *Proceedings of the IEEE Conference*, 2021.

[13] UGR-SAIL. Sinergym: A reinforcement learning framework for energyplus. https://github.com/ugr-sail/sinergym. Accessed: December 2024.

[14] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2016.

[15] Chi Zhang, Sanmukh R. Kuppannagari, Rajgopal Kannan, and Viktor K. Prasanna. Building hvac scheduling using reinforcement learning via neural network based model approximation. *arXiv preprint arXiv:1910.05313*, 2019.

8 Appendix

8.1 State Space

- Site Outdoor Air Drybulb Temperature
- Site Outdoor Air Relative Humidity
- Site Wind Speed
- Site Wind Direction
- Site Diffuse Solar Radiation Rate per Area
- Site Direct Solar Radiation Rate per Area
- Zone Thermostat Heating Setpoint Temperature
- Zone Thermostat Cooling Setpoint Temperature
- Zone Air Temperature
- Zone Air Relative Humidity
- Zone Thermal Comfort Mean Radiant Temperature
- Zone Thermal Comfort Clothing Value
- Zone Thermal Comfort Fanger Model PPD
- Zone People Occupant Count
- People Air Temperature
- Facility Total HVAC Electricity Demand Rate