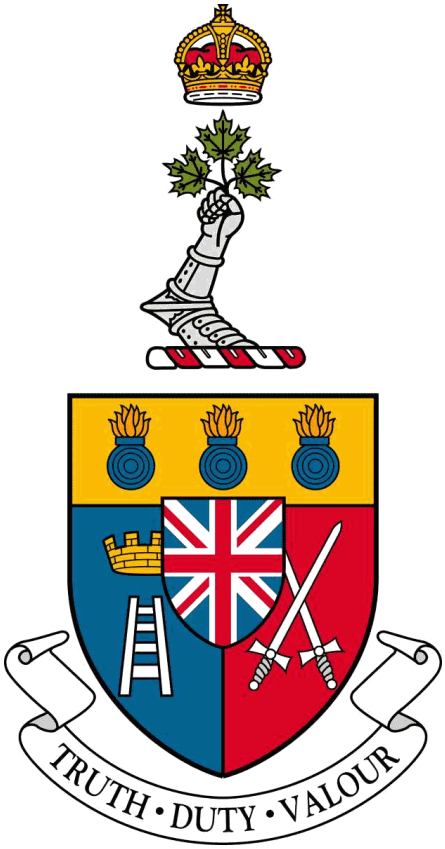
The ART of CityLearn: Observation Perturbation Attacks and Mitigation for Reinforcement Learning Agents in a Cyber Physical Power System



A Thesis Submitted to
the Department of Electrical and Computer Engineering
by

Kiernan Broda-Milian

In Partial Fulfillment of the Requirement for the Degree of
Master of Applied Science in Computer Engineering

31 May 2024

Acknowledgements

I'd like to express my gratitude to my supervisor, Dr Ranwa Al Mallah, for her mentorship throughout this project, and my co-supervisor, Dr Hanane Dagdougui, and Vincent Taboga for guiding my choice of environment. I'd also like to thank Karla Gonzalas for successfully training the continuous PPO agent. I'm especially gratefully to my wife, Janet, for supporting me through all the time I dedicated to this work.

Abstract

Components of Cyber Physical Systems (CPS), which affect real-world processes, are often exposed to the internet. Replacing conventional control methods with Deep Reinforcement Learning (DRL) in energy systems is an active area of research, as these systems become increasingly complex with the advent of distributed energy resources integration and the desire to improve the energy efficiency. Artificial Neural Networks (ANN) are vulnerable to specific perturbations of their inputs or features called adversarial examples. These perturbations are difficult to detect when properly regularized, but have significant effects on the ANN's output. Since DRL relies on artificial neural networks to link optimal actions with observations, they are also susceptible to adversarial examples in a similar manner. While Adversarial RL (ARL) has been explored in energy distribution, research is lacking in Demand Response (DR). Furthermore, the ARL literature lacks research into the stealth of adversarial attacks. This work proposes a novel attack technique for continuous control using Group Difference Logits (GDL) loss with a bifurcation layer. By combining aspects of targeted and untargeted attacks, it significantly increases the impact compared to an untargeted attack, with drastically smaller distortions than an optimally targeted attack. This thesis demonstrates the impacts of powerful gradient-based attacks in a realistic smart energy environment, measures how the impacts change with different DRL agents and training procedures, and uses statistical and time series analysis to evaluate attacks' stealth. It finds that adversarial attacks can have significant impacts on DRL controllers in DR, and constraining an attack's perturbations makes it difficult to detect. However, certain DRL architectures are far more robust, and robust training methods can further reduce the impact.

Index Terms **- Deep Reinforcement Learning, Adversarial Examples, Smart Energy, Cyber Physical Systems, Load Altering Attacks**

Résumé

Les composants des systèmes cyberphysiques qui affectent les processus du monde réel sont souvent exposés à l'Internet. Le remplacement des méthodes de contrôle conventionnelles par l'apprentissage par renforcement profond (aussi connu sous le nom de Deep Reinforcement Learning, DRL) dans les systèmes énergétiques est un domaine de recherche actif, car ces systèmes deviennent de plus en plus complexes et connectés avec l'avènement des ressources énergétiques distribuées et la volonté d'améliorer l'efficacité énergétique. Les réseaux de neurones artificiels (aussi connus sous le nom de Artificial Neural Networks, ANN) sont vulnérables à des perturbations spécifiques de leurs entrées. Ces perturbations aux entrées permettent de créer des exemples antagonistes. Ces derniers sont difficiles à détecter lorsqu'ils sont correctement régularisés, en plus d'avoir des effets significatifs sur la sortie de l'ANN. Étant donné que le DRL utilise les ANN pour mapper les actions optimales aux observations, il est également vulnérable aux exemples antagonistes. Bien que l'Adversarial RL (ARL) ait été exploré dans le domaine de la distribution d'énergie, il existe peu de recherche sur le sujet dans le domaine de la Réponse à la Demande (RD). De plus, la littérature de l'ARL manque de recherches sur la furtivité des attaques adverses. Ce travail propose une nouvelle technique d'attaque sur un système de contrôle en continu utilisant la différence de groupe de logits dans la fonction de coût suivi d'une couche de bifurcation. En combinant les aspects des attaques ciblées et non ciblées, cette technique augmente considérablement l'impact par rapport à une attaque non ciblée, avec des distorsions considérablement plus faibles qu'une attaque optimale ciblée. Cette thèse démontre les impacts d'attaques puissantes basées sur le gradient dans un environnement énergétique intelligent réaliste. On mesure comment les impacts changent avec différents agents DRL et procédures d'entraînement. On utilise des analyses statistiques et des analyses de données temporelles pour évaluer la furtivité des attaques. Il a été également démontré que les attaques contradictoires peuvent avoir des répercussions significatives sur les contrôleurs DRL en RD, et restreindre les perturbations d'une attaque les rend difficiles à détecter. De plus, il s'avère que certaines architectures DRL sont bien plus robustes que d'autres, et que des méthodes d'entraînement robustes peuvent réduire encore davantage l'impact d'une attaque.

***Index Terms* - Apprentissage par Renforcement Multi-Agent, Exemples Antagonistes, Systèmes Cyberphysique, Attaques Altérant la Charge, Systèmes Énergétiques Intelligents**

Contents

Glossary

ACG - Auto-Conjugate Gradient. A maximum confidence attack using the conjugate gradient first order optimization method, which automatically adjusts the size of its search step.

Adversarial Budget - The proportion of perturbed observations, the features perturbed, and the perturbation size ϵ (if applicable).

Adversarial Regret - The effects of the adversary measured either as the percentage of time in adversarial state during an attack or the reduction of total rewards in testing, depending on whether the adversarial goal is state- or reward-based. Adversarial regret measures how much an attack degrades performance.

ART - Adversarial Robustness Toolbox. A Python library of adversarial attacks.

AMI - Advanced Metering Infrastructure.

ANN - Artificial Neural Network.

ASR - Adversarial Success Rate. The proportion of successful adversarial examples, i.e., where an observation perturbation results in the victim deviating from its policy.

ATLA - Alternating Training with Learned Adversary. A robust training method where an agent is trained with an adversary perturbing its observations. In alternations the agent is trained to adapt to the adversary, then the adversary trains to adapt to the agent.

CI - Critical Infrastructure.

Classifier - A machine learning algorithm which maps samples to a discrete set of labels.

CPPS - Cyber Physical Power System.

CPS - Cyber Physical Systems. Networks of OT.

DER - Distributed Energy Resources.

DL Loss - Difference Logit loss.

DoS - Denial of Service.

DR - Demand Response. A smart grid technique where a grid operator modifies the load or demand on the grid in response to the power available.

DRL - Deep-Reinforcement Learning.

Evasion Attack - An adversarial attack on a trained ANN, which aims to find small changes to its input which changes the output.

Features - The inputs to a machine learning algorithm such as an ANN. In RL, observations are a set of features.

FDI - False Data Injection.

FGM/FGSM - Fast Gradient Method/Fast Gradient Sign Method. Both are used interchangeably. This is the original adversarial attack.

IBR - Inverter Based Resources.

KPI - Key Performance Indicator. One of several cost functions for evaluating performance in the CityLearn gym environment. The KPIs are normalized by the cost functions for an environment with no smart controller. So, the KPI is the relative performance compared to a scenario with no controller. This means lower KPIs are better, and KPIs less than one mean that the controller is better than the baseline.

LA - Learned Adversary. A DRL agent which is trained to degrade the performance of a victim agent by adding perturbations to its observations.

LAA - Load Altering Attack.

Label - The classification or value assigned to a sample by a machine learning algorithm such as an ANN.

Logit - The non-normalized output of an ANN.

Maximum-Confidence Attack - Method for maximizing the loss caused by an ANN's output by changing its input within a specified boundary. The loss function can

be the same as that used to train the ANN, or a specialized function chosen by the attacker.

MDP - Markov Decision Process.

Minimum-Norm Attack - An adversarial attack which aims to find the smallest perturbation for an ANN's input which changes output.

MLP - Multi-Layer Perceptron, synonymous terms include feed forward or fully connect network.

MMD - Maximum Mean Discrepancy.

OT - Operational Technology. Technology used to control or monitor a physical process.

Policy Induction - Manipulating the observations of a DRL agent so its actions follow an adversarial policy.

PGD - Projected Gradient Descent. A maximum confidence attack which iteratively follows the gradient of a loss function.

Regressor - Machine learning algorithm, such as an ANN, which maps input features to label from a continuous range of values.

RES - Renewable Energy Systems. Electrical generation from sources such as wind turbines, solar panels, and hydro-electric dams.

SB3 - Stable Baselines 3. A Python library for DRL.

SCADA - Supervisory Control and Data Acquisition.

Snooping Attack - Black box attack which uses historical data to train a proxy model of the victim RL agent, to enable gradient-based adversarial attacks.

SoC - State of Charge. This is shorthand for the CityLearn feature electrical storage SoC, which is the charge state of a battery.

ST Attack - Strategically-Timed attack. An observation perturbation attack in DRL which aims to only perturb the victim at key moments, and not to cause any perturbations when their effect is insignificant.

UFLS - Under Frequency Load Shedding. Contingency for maintaining the frequency on a power grid, where non-essential loads are disconnected when the frequency drops.

List of Figures

List of Tables

1 Introduction

Cyber Physical Systems (CPS) are networked Operational Technology (OT), which control processes in the real world [9]. Thus, actions in cyberspace can have physical consequences. There are many precedents for cyber attacks on Critical Infrastructure (CI), using a variety of vectors [1]. Smart grids are a prominent example of critical CPS. They leverage Distributed Energy Resources (DER), which provide a significant attack surface for bad actors to disrupt a power grid.

Reinforcement Learning (RL) including Deep-Reinforcement Learning (DRL) is pervasive in CPS control research, including smart energy systems [10, 11]. These systems require fine grained management to instantaneously match power generation with the load, by controlling several variables and safety systems. As energy systems are a component of CI, efficiency and reliability is broadly consequential. RL addresses the increasing complexity of integrating multiple energy sources, rising demand, and non-linear behaviour. Unlike conventional control methods, RL agents can learn optimal actions without knowing the environmental dynamics, through interactions with historical data, simulations, or live systems. In contrast, conventional control methods require a known model before the optimal actions can be determined. RL applications are successfully studied in many power systems applications, from the operational control of energy generation and distribution, to the efficient use on the load side, and determining market prices. More research is required to ensure that RL agents are sufficiently robust to enable these applications.

Although OT is used for control, it is rife with security concerns given its safety-first focus, which prioritizes uninterrupted communication and continuous operation over security[9, 1]. OT's exposure to the internet as a part of modern CPS creates additional vulnerabilities. In fact, a variety of threats exist, ranging from ransomware gangs to state-sponsored actors. Thus, protecting these systems and exploring their vulnerabilities is an important area of research. With the numerous applications for RL, adversarial examples, in concept, provide another exploit to adversaries [12]. This type of attack may be particularly difficult to detect using conventional Intrusion Detection Systems (IDS) given that the examples are designed to resemble normal traffic and IDS struggle to identify novel attacks.

As Artificial Neural Networks (ANN) are used in supervised learning to associate features with labels, they are used in RL to associate observations with optimal actions [13]. ANN allow DRL agents to store and update a number of weights far smaller than the number of possible observations. While this improves the algorithms, it also makes them vulnerable to adversarial examples. Specifically, an adversarial example is a specially designed perturbation that describes the smallest changes to the input values of the algorithm that changes the prediction to a predefined output. The same gradient-based methods used to optimize the ANN's parameters can be used to find the smallest perturbation to the inputs which changes the output. With an adversarial example, an attacker can lure a victim towards a state or action of their choosing with only access to the means of communication between the agent and its sensors [12]. This attack to the model's integrity is a type of evasion attack. Evasion attacks at test time are a breach of model input integrity where the end result is a compromise to the model's performance. Attackers can send perturbed data to the trained model. Even when the victim algorithm is a black box to the attacker, this threat still exists because of the transferability principle [14]. Moreover, attacks are becoming less computationally expensive and increasingly difficult to detect because several techniques try to optimize the size and number of perturbations used in the attacks [12, 15]. The techniques also have been demonstrated to reduce the time required to generate the perturbations [16, 5, 17, 18].

1.1 Motivation

Protecting CI as the technology evolves is a primary motivator of this research. As CI is an attractive target for hackers, all potential vulnerabilities should be explored and addressed [9]. The security vulnerabilities of every component in a CPS must be understood to protect it, which may soon include DRL agents. Renewable Energy Systems (RES) and rising demand are increasing the complexity of modern energy systems [19]. As the power outputs of renewable generation depend on environmental conditions, they cannot be controlled like conventional generators. Electricity demand increases the peak load generation, and distribution networks must be capable of supporting it. Both of these are requirements for grid operators, as the electrical grid must instantaneously match demand while maintaining tight tolerances for voltage, frequency, and phase. Thus, smart grids are becoming a popular concept for improving grid stability and reducing peak demand, and DRL controllers are expected to become integral to CPS in CI. Adopting DRL in control offers superior optimizations compared to conventional control techniques for increasingly complex energy systems. This is due to the fact that DRL effectively optimizes non-convex functions, doesn't require physical models, and handles larger action and state spaces [11]. DRL agents learn an optimal policy during training, which maps its observations to actions. Determining optimal control actions in smart grids is increasingly difficult, making DRL an attractive alternative.

In terms of attacks, targeted observation perturbations of a victim DRL agent are capable of controlling its actions and coercing it into following an adversarial policy, instead of the policy it learned during training. As in classification tasks wherein an adversarial example can be generated from a sample to correspond to a label of the attacker's choice, an observation can be perturbed towards a target action. In each case, an attacker can use gradient ascent to discover small changes to the ANNs' inputs which result in a different output. These changes can be constrained such that they are imperceptible to the human eye and impractical to discern from unperturbed observations [20]. These attacks are enabled by the gradient methods used to optimize the ANN, making this vulnerability an inherent property. Thus, DRL controllers present an exploit to attackers, which is difficult to detect.

This means that attackers only need to manipulate a DRL controller's observations to gain control of the entire system. Furthermore, the attacks are difficult to detect, concealing the attack vector from network defenders. Sensors observing the environment cannot always be physically co-located with the controller. This is the case for a large building or distribution network for example, which increases the attack surface [9]. While the controller might be stored in a secure and isolated location, it may not be practical to do the same for remote sensors. Furthermore, if communication between the controller and sensors is over a network, the traffic must be secured. However, secure traffic is not often a primary concern in OT. This is particularly true of smart grids managing energy distribution and demand, as these require dispersed sensor networks. These sensors can be in isolated locations on distribution networks or inside private user buildings where the smart grid operator cannot access or protect them. These factors complicate managing and securing sensors and their communications.

This research seeks to motivate the designers of smart energy systems to conduct robustness testing and build robust algorithms. In fact, unlike conventional control techniques, DRL is vulnerable to adversarial examples. An attacker using adversarial examples could effectively control the DRL victim and destabilize the CI [21, 22]. Demonstrating a successful attack in the area provides future researchers with a framework to assess their own DRL agents and shows why agents used in this area of control must be robust. Understanding the scope and limitations of such attacks enables risk assessments when DRL technologies are implemented in CI. Such research is necessary to demonstrate the importance of robust DRL agents, which are not easily exploited with

adversarial techniques.

The factors above motivate four primary research questions:

1. Can observation perturbations using adversarial examples significantly degrade the performance of a DRL agent in a smart energy system?
2. Can adversarial attacks be used to apply an adversarial policy for a DRL victim in a smart energy environment?
3. Are adversarial examples statistically distinguishable from normal samples?
4. Are there general training or architectural designs which make DRL agents more robust to observation perturbations?

1.2 Statement of Deficiency

Current research into evasion attacks on DRL smart energy systems is limited, as most attack techniques are demonstrated against agents in video game gyms. Some works have shown how smart energy distribution networks controlled with DRL are vulnerable to evasion attacks, but research is lacking in attacks on DRL used for energy demand management [12]. Demand management involves controlling the power consumed by a user to avoid stressing the power distribution and generation infrastructure, and includes smart buildings. DRL has shown superior performance to conventional control for maintaining a building's temperature while reducing power consumption; however the security and robustness of DRL to evasion attacks in this application remains to be evaluated [11].

The reviewed works which study the robustness of DRL control to adversarial observation perturbations do not evaluate the magnitude of the perturbations. While significant distortions to the inputs of any controller will change its output, adversarial examples are remarkable for the small size of their perturbations. To fully assess the threat posed to DRL controllers, both the degradation of the victim's performance and the size of the perturbations should be considered. The threat posed by these perturbations is greater if they are non-trivial to detect, and this must be assessed with domain specific samples.

Moreover, few tools to conduct robustness testing in DRL exist. No smart energy RL gym includes the capability to add observation perturbations during training or testing, which is necessary for testing model robustness or vulnerability to attacks. While libraries exist for crafting adversarial examples, they are designed for classifiers, meaning they are incompatible with continuous action spaces in DRL, and do not directly interface with DRL agents. Adversarial example crafting techniques must be interfaced with both the victim agent and the environment to conduct the security and robustness research.

1.3 Aim

This research aims to Demonstrate evasion attacks and defensive techniques on DRL algorithms in a simulated smart energy environment . We hypothesizes that statistically undetectable input perturbations can significantly degrade the performance of a DRL smart energy system. We test this hypothesis by asking these 4 sub-questions:

1. Can false data injection using adversarial examples significantly degrade the performance of a DRL agent? Testing the potential impact of adversarial samples as the payload of a cyber attack in a realistic application. This tells us adversarial examples are a significant threat to performance.
2. Can adversarial attacks be used to apply an adversarial policy for a DRL victim? Adding on to question one, can such an attack control a CPS, rather than just degrade its performance? Previous research has demonstrated targeted adversarial

attacks inducing agents into following an adversarial policy, but is it practical in this setting?

3. Can statistically indistinguishable adversarial examples be crafted? The most significant aspect of adversarial examples is that they only require small changes to the inputs, making them difficult to distinguish for normal data. To show that adversarial attacks are a threat worth mitigating, we must validate that detection is non-trivial.

4. Are there general training or architectural designs which make DRL agents more robust to observation perturbations? Once adversarial attacks are demonstrated as a significant threat, mitigations will be tested. Algorithm agnostic techniques are important, since they can be adopted in existing systems, making them easier to implement.

The attacker's goal in a demand management environment is to perturb the victim's observations to increase energy demand. Increased demands when a power grid is near its capacity can result in outages, with the impact dependant on the size of the spike. To achieve our aim, the attacker would have read and write access to the victim DRL agent's observations but only read access to the victim's model, simulating an intermediary attack between the victim and its sensors, akin to an Adversary-in-the-Middle scenario. This setup represents a white box attack.

A white box attack represents the worst-case scenario for a defender as it allows the attacker to craft the strongest attacks. Studying the worst case scenario is appropriate in this sparsely studied area, as exploring attacks with greater constraints is sensible once the threat and its impacts are shown to be credible. Knowing the maximum impact of an adversarial attack in this area will contextualize future research on more constrained attacks, and provide data for comparing the effect of constraints and defences on the attacker.

This research will be conducted in RL gym environments, which are distinguished from other simulations by their interface for RL agents. An RL gym's input at each timestep is a selection from predefined actions called the action space. The gym returns an observation and reward based on the action. Through this process, the agent receives feedback on its action from the reward and learns to choose the optimal action for a given observation. The same setting will be used to train an adversarial policy, which selects the target for adversarial observation perturbations used to change the victim agent's actions. The adversarial policy can be a trained DRL agent with a reward function specific to the attacker's goal, e.g. a negation of the victim's reward. Training, attacking, and defending a DRL agent in an existing smart building gym will demonstrate the threat of adversarial attacks in this application.

By simulating the effects of manipulating sensor observations to determine the severity of the threat of DRL algorithms controlling smart energy systems, this research hopes to inform the future design of algorithms under an adversarial setting. To that end it will demonstrate that adversarial examples are an attack vector for control systems and validate that they are difficult to distinguish from normal samples. For this demonstration, an environment with a realistic application and data has be selected, and the attack will be compared to clean data with statistical tests. Given the lack of research in this area, the goal is to show the design of robust and adversary-aware machine learning systems.

A SotA victim agent is not required to fulfill the aim, and developing one is out-of-scope. The agent must be sufficiently capable to demonstrate an attack. The victim is designed such that it is compatible with strong adversarial attacks, to test their impacts. This limits which ANN architectures may be used.

1.4 Research Activities

The following research activities will be conducted in order to demonstrate evasion attacks and defensive techniques on DRL algorithms controlling smart energy systems, as shown in Figure 1:

1. **Demand Response Environment**:
   (a) Selecting the tool set for the experiments requires compatible DRL algorithms, a Cyber Physical Power Systems gym environment, and adversarial attacks.
   (b) Training a victim agent for testing adversarial attacks.
2. **White Box Attacks**: White box attacks are the best case scenario for the attacker, which allows the most powerful attacks to be assessed. Various types of adversarial attacks are tested and the reduction in the victim agents' performance is measured. Initial attacks will aim to force the victim agent into taking any suboptimal action. Targeted attacks will then be used to force the victim into following an adversarial policy of the attacker's choosing. Finally, a novel attack called the bifurcation attack is proposed. The timing of attacks are explored to test if the attack frequency can be reduced without reducing its impact.
3. **Defences**: In this activity we experiment with two defence paradigms: Detection and correction.
   (a) Detection: To defend against attacks, one paradigm is to detect the adversarial example in order to potentially generate an alarm about this anomaly. In this activity, we experiment with two detection approaches not to defend against our proposed attacks but rather to determine if the adversarial observations produced by our attacks resemble typical observations and are thus plausible. The attacks will be modified as needed to test if they can be both stealthy and impactful.
      i. Analysis of the original and adversarial observations to test if the adversarial observations can be detected by comparing the changes between subsequent observations.
      ii. Model agnostic statistical tests to determine if the original and adversarial observations are drawn from the same distribution.
   (b) Correction: Defence methods that are based on the correction paradigm change how training or inference is done. A "protective layer" is added to make the system more robust by changing the input data. We experiment with adversarial training as a robust training method to reduce the impact of adversarial attacks.
4. **Black Box Attack**: The black box attack restricts the information known to the attacker, decreasing their impact. This type of attack is easier to execute. The efficacy of black box attacks is compared to the prior attacks, in terms of the attacks' impact and the amount of distortion required.

1.5 Summary of results

Agents with different architectures were trained in the CityLearn gym environment to reduce energy usage by controlling the charging and discharging of a battery. These agents were used to test three types of white box attacks: untargeted, optimally targeted using an adversarial agent, and using a novel attack called the bifurcation attack. All

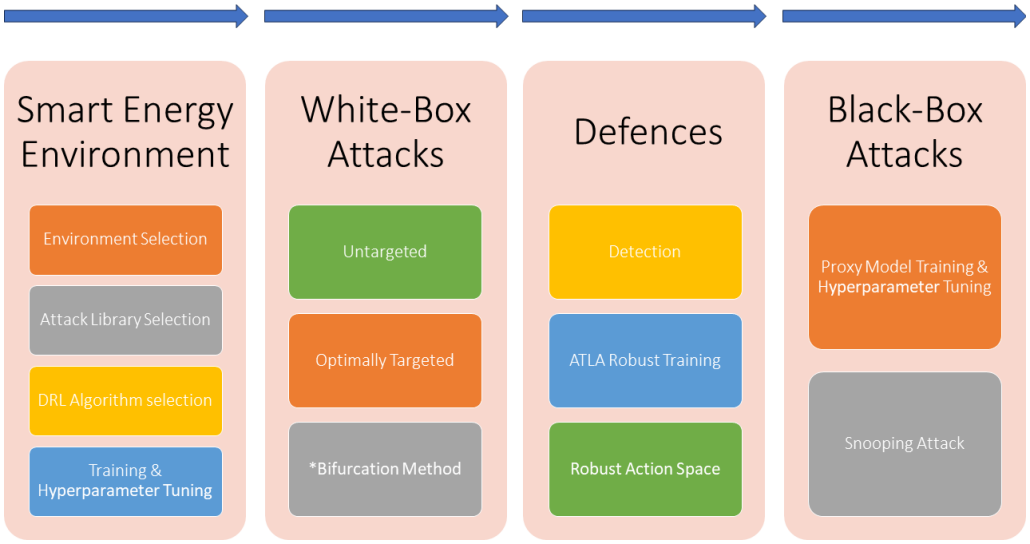


Figure 1: Figure showing, from left to right, how the aim was achieved. Starting with a realistic gym environment, victim agents are trained to establish a baseline for performance. Then conducting strong attacks with perfect information to determine the extent of the threat, using SotA techniques. The defense section tested algorithm agnostic mitigations for the previous section's strong attacks and validates that the attacks are stealthy by trying to detect them. Finally, the agents and mitigations are tested against black box attacks, where the attacker cannot access the victim's parameters or environment.

attacks used the agent's parameters to craft strong adversarial observations. The success of these attacks was measured by how much distortion the attack introduced to the observations, how successful the attack was in changing the agent's actions, and how much the power usage increased.

The untargeted attacks had relatively small effects on power consumption, but were successful in changing the agent's action for nearly every observation. The optimally targeted attack more than tripled the power consumption, but introduced obvious distortions. The novel attack more than doubled the effects of the untargeted attack with similar distortions to the observations.

The adversarial observations produced during these tests were analyzed during the detection phase. The attacker's budget was reduced until the adversarial observations appear indistinguishable from the originals. This constraint reduced the performance of the adversarial attack, making the effect insignificant for one of the two types of agents tested. These results indicated that stealthy adversarial attacks are possible and that simple changes can be made to the DRL agent to increase its robustness.

A DRL agent's action space significantly affects the impact of attacks. Comparing the same DRL algorithm with discrete and continuous action spaces showed that the former reduced the attacks effect by approximately three fourths.

The black box attack was successful in significantly increasing power usage without access to the agent's parameters. It involved training a proxy model using the agent's historical observations and actions, and using the proxy's parameters to enable a simple attack. This attack is more feasible in practice as this type of historical data would be available to an attacker with access to the agent's observations. The compromise is that larger distortions are required for the black box attack to achieve a similar effect as a white box attack.

1.6 Organization

Chapter 2 of this thesis will provide a background of DRL and cyber security in energy systems, and adversarial attacks and defences. Chapter 3 discusses related works in adversarial and robust RL. Chapter 4 provides the threat model. Chapter 5 is on the implementation of a demand response environment for this research. Chapter 6 shows the impact of adversarial attacks from the strongest adversary. Chapter 7 demonstrates how attacks can be detected or mitigated. Chapter 8 repeats the methodology of chapter 6 for a limited adversary. Chapter 9 discusses this thesis' contributions and future work.

2 Background

This chapter will introduce the core concepts of Deep-Reinforcement Learning (DRL) in energy systems. We then discuss the cyber security of energy systems. Finally, we introduce adversarial examples because evasion attacks on deep neural networks are done through adversarial examples.

2.1 RL in Energy Systems

Modern energy systems include a variety of Distributed Energy Resources (DER) and load devices, where generators must meet the load's energy demand in real-time with narrow margins. Supplying too much can damage the load and distribution systems and too little results in brown or blackouts. Thus, energy systems can be described by their reactive and active powers, current, phase, voltage, and frequency where:

1. Power is proportional to current and voltage.
2. Reactive power corresponds to ripples or feedback in an energy system caused by the difference in phase between the current and voltage.
3. The frequency in a power system affects the impedance of a system, which in turn affects the phase and reactive power.
4. The system components have frequency, voltage and current limits, and can physically fail if exceeded. For example, synchronous machines like generators and induction motors rotate at the frequency of the power grid and are damaged by large deviations, and the operations of power electronics like rectifiers and transformers change with the input frequency.

Keeping the system within tolerance limits is a control problem where grid operators need to take optimal actions like changing the frequency on generators or varying the number online for the safe and efficient operation of energy systems.

Grid operators make daily and hourly predictions based on historical usage and weather data to ensure that enough generation capacity is available, and that the system has sufficient inertia [7]. The operator computes the power flow in high voltage transmission lines, which carry electricity from generators to local distribution networks, in advance to ensure generation supply will match demand.

Conventional generation involves a mechanical input (e.g., hydro-dams or steam) powering generators' rotors, and the load on the grid is proportional to the energy required to turn connected rotors' (in the same way it's harder to peddle a bicycle up hill). Because the rotors rotational frequency is analogous to the frequency of Alternating Current (AC), changes in demands affect the operating frequency. Deviations on the order of Hz can damage generation equipment. Thus, operators' *primary response* to small frequency deviations is to change the mechanical inputs. Note that the primary response is limited for Renewable Energy Systems (RES), because the operator cannot control the energy input of solar and wind generation. As the *primary response* can exceed demand, a *secondary response* involves connecting additional generators, changing the active power target, and/or adding controllable loads, which requires reserve generation capacity, perhaps provided by another entity, a reserve of reactive power, and/or inactive controllable loads. If these responses fail to stabilize the grid, possibly from insufficient reserves, the operator must shed load and possibly disconnect generators to protect equipment. Such failures occur when the operator is unable to determine a feasible control action for the state of the power grid. Over current protection, which disconnects transmission lines to prevent exceeding their thermal limits, is one of the most common occurrences in

cascading failures [23]. Lines can be overloaded when generation exceeds demand, and once a line is disconnected, power must flow through the remaining lines, triggering their current protections.

Similarly, voltage protections involve disconnecting loads to protect generators from damage caused by large deviations of the voltage profile. Once a generator is disconnected, a cascading failure can follow from insufficient generation capacity. Thus, grid operators must select optimal primary and secondary responses based on system models to maintain power flow within safe margins. Protecting generation and transmission equipment in contingency situations involves localized blackouts from loads shedding or widespread power loss from cascading failures.

The complexity of modern energy systems is increasing in terms of operating states and control actions [10]. This makes determining the feasible power flows more difficult. The causes include the introduction of RES and rising demand [19]. It is increasingly difficult to optimally control energy systems, due to uncertainties in supply and demand. This leads to more disturbances and must be addressed with better methods of monitoring the dynamic security, which is defined as transient, voltage, and frequency stability [19]. While legacy power systems were designed for unidirectional power flow from centralized generators to remote loads, RES employ Distributed Energy Resources (DER) throughout the power grid [6]. Users operating RES will, at times, consume and generate power from the grid's perspective, making their power flow bidirectional. This requires additional monitoring and communication between DER and grid operators to maintain power flows.

Conventional control methods struggle to model or apply heuristics for complex and distributed power grids [22]. Such algorithms require robust models of their environments, which are ever more difficult to produce and maintain or update. Both supervised and DRL methods can learn to model complex and uncertain systems from historical data or environmental interactions, and model-free DRL algorithms can learn optimal control policies. These properties of DRL algorithms enable them to act optimally in environments where the global optimal policy is unknown [10]. Due to uncertainties in renewable energy generation and demand, DRL outperforms other techniques by addressing three major issues [11]:

1. Non-linear systems and non-convex optimization functions in modeling. DRL algorithms are able to operate without a world model and can learn to approximate their environment.

2. Nonexistence of physical models, using model-free learning as above. This is a common issue with building energy management systems which typically lack practical models.

3. Operating in large state and action spaces is less challenging for RL, for example, using weighted function approximations can learn a set of weights which is much smaller than the set of the state or action spaces.

Research on DRL for energy systems has steadily increased in the last decade, whereas works on model predictive control (which does not use reinforcement learning) have peaked. DRL is alluring to researchers because of its model-free approach to optimizing control, as opposed to model predictive control which requires a model of physical principles to relate control inputs to outputs. Model dependencies make extending the latter techniques to complex energy systems impractical. This is also true of fuzzy control systems as the decision space explodes with increased energy system complexity [11].

There are at least four major categories of energy systems management where DRL has been successfully applied in research as identified in [10, 11], these reviews cited 68 and 84 works in these categories respectively:

1. Smart energy distribution (dispatch): Optimal supply of electricity, heat, or cooling to match demand. Dispatch for power grids includes voltage regulation. It is influenced by demand, grid conditions, energy market, and performance of the system components.

2. Grid operational control: Operational controllers maintain stable operations for power systems, including transient stability and power flow. In this domain, the optimal control strategies for energy storage, dispatch sources, and demand fluctuations are found with RL. These strategies are necessary for the grid's secure and stable operation, particularly when using DER.

3. Demand Side Management (DSM): Managing the amount of power used by consumers. System stability is improved by reducing peak loads. This includes building energy management systems, which aims to increase comfort or energy efficiency, improve air quality, and/or decrease energy bills. The vast majority of publications concern optimal Heating Ventilation and Cooling (HVAC) control in buildings. Demand Response (DR) is a related technique where the system operator responds to grid conditions by modifying user loads [24]. DR can work through price incentives, communication between grid operators and large consumers, or automated and direct control over consumer loads[23]. The latter applies to both large commercial users and residential users alike.

4. Energy Markets: The review provided by [10] references 9 works which use DRL for: automatic bidding by generation companies, electricity market modeling, or dynamic pricing.

Research is particularly promising for applying DRL to HVAC systems, where modeling the system dynamics is particularly difficult due to the system's complexities, and uncertainties in environmental conditions like occupancy and weather. DRL can outperform conventional techniques by up to 20 %, which stands out compared to research in other applications [11]. A Multi-agent Actor-Critic (MAAC) MARL algorithm has shown 56.50% and 75.25% increases in energy efficiency compared to heuristic and rules-based set-point control respectively, for a large building with 30 zones, simulated in EnergyPlus with real world traces [25]. This demonstrates the scalability of DRL for applications where modeling an increasing number of uncertainties has diminishing returns. Such models are difficult to generalize and controllers using them are limited by the model's accuracy, and require near-exhaustive searches of the action space to act optimally [26].

As determined from the aforementioned works, DRL will be critical for managing future energy systems, both for increased efficiency and reliability. There is a wide variety of applications for RL, from top-down grid level control to user level energy management systems. The complexity for the system is proportional to the difficulties in modeling it for conventional control, and it is feasible that conventional methods will be impractical for selecting optimal actions. This problem is addressed with model-free RL.

2.2 Cyber Security in Energy Systems

The digitization of Operational Technology (OT) into Cyber Physical Systems (CPS) enables threat actors to disrupt OT systems because of increased connectivity, including Critical Infrastructure (CI) [9]. OT is hardware and software integrated with devices for monitoring and changing the physical world, and CPS are smart systems incorporating information, analytic, and OT infrastructure. OT in critical infrastructure experiences the same cyber threats as any large organization, in addition to state-sponsored actions during geopolitical tensions.

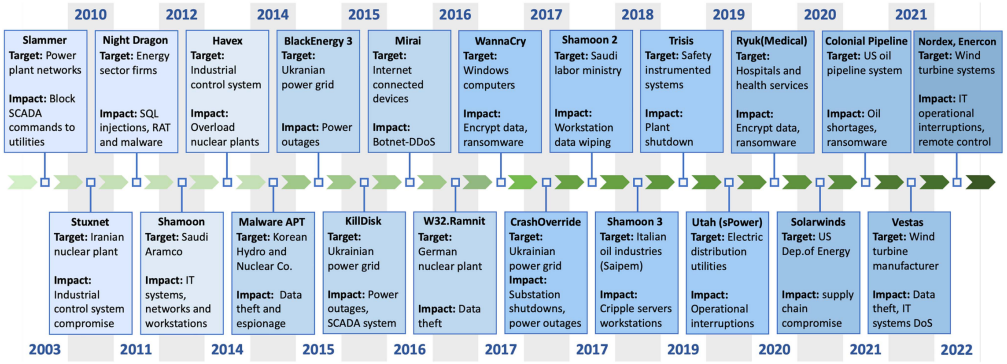


Figure 2: Figure 2 from [1]: "Timeline of cyberattacks targeting the energy sector and other CI sectors" ©2023 IEEE.

The lifespan of OT is measured in decades and, unlike in Information Technology (IT), the personnel maintaining OT tend not to have security backgrounds or prioritize security. These factors lead to OT devices remaining unpatched in the face of new threats. Because OT is designed to maintain safe operating parameters at all times, it often eschews secure communication protocols and Intrusion Detection Systems (IDS) which could delay or disrupt communications. Early systems instead relied on segregation or air-gaping, but modern CPS tend to be permanently connected and internet-facing [27][9]. In 2021, the Canadian Centre for Cyber Security identified 128 000 ports associated with OT and 13% were unpatched; this included 24 000 from utilities and 6000 for building control and automation [9].

OT is used to monitor four major threat categories for energy systems [19]:

1. Power Quality Disturbance (PQD)
2. Supervisory Control and Data Acquisition (SCADA) Network Vulnerabilities and Threats
3. Transient Stability Assessment (TSA)
4. Voltage Stability Assessment (VSA)

Transient instability causes disruptions like islanding and wide area blackouts. TSAs guide operator decisions in contingency scenarios, but conventional methods' computational requirements mean that assessments often fail to meet the demands of modern power systems [19]. Voltage stability is the system's capacity to maintain bus voltages after deviations, which can be caused by load dynamics. Voltage stability includes reactive power management. It requires continuous monitoring of loads and generator dynamics. Voltage instability causes blackouts and can also damage generators and induction motors with rapid deceleration. SCADA networks are not designed for security and tend to rely on IDS when security measures are present. Because of their inflexibility, they struggle to protect modern heterogeneous systems. Like IDS in other fields, they struggle with imbalanced and outdated training data. DRL is one solution to control problems like transient and voltage stability, however the scope of the cyber security challenge is greater.

In [28], the authors identify principal types of cyber attack on Cyber Physical Power Systems (CPPS), which can be categorized as service degradation or spoofing. On a networked system, spoofing can be conducted via an intermediary attack (also called Man-in-the-middle (MITM)) where a malicious node inserted between devices, allows the attacker to read, block, or modify traffic. The typical implementation of this attack

involves spoofing the hardware addresses of the devices the attacker inserts themselves between. This tricks devices into forwarding traffic to the attacker, who can choose what data to forward to its destination. This is accomplished by abusing the Address Resolution Protocol (ARP) used to map Ethernet Medium Access Control (MAC) addresses to Internet Protocol (IP) addresses. An intermediary attack can enable False Data Injection (FDI) to change the system operator's estimation of the CPPS' state, which impedes the operator's decision process. To cope with noise from interference and meter malfunctions, CPPS often employ detectors to reject measurement outliers. Thus, the false data injected must be within the detector's tolerance. A special case of FDI is a replay attack, where an attacker delivers a deliberate sequence of historical measurements to disrupt the target system. A service can be degraded using delay or Denial-of-Service (DoS) techniques. A delay attack is used to desynchonize the measurements and operational action of a system, and a DoS prevents any communications between devices. A typical DoS attack saturates the means of communication between nodes of a network, to the point no data can be transferred or processed. This denies the attacked service to users, which in the context of CPPS could prevent the flow of measurements to the system operator or commands from the operator to the system's components. Table 1 summarizes the different attack vectors for DER, and figure 2 shows a timeline of cyber attacks on CPPS.

Attack vector	**Description**	**Threat**
Interoperability of DER assets	Requirements crucial in implementation communications (e.g., security and control messages)	Denial of legitimate messages and control commands
Data integrity violations	Stored, transmitted, or received unauthorized access to control information	Malicious modification of control parameters
Implementation errors	Security flaws within DER systems and communication modules enabling the remote control	C2 of demand/load side devices
Supply-chain compromises	Hardware-based eavesdropping, worms, and oversights during manufacturing of components, devices, or systems	Sensitive information disclosure
Insecure firmware	Digital signatures of firmware updates are not always verified, granting malware (viruses, trojans, worms, etc) access, to otherwise, secure systems	DER systems privilege escalation

Table 1: Attack vector description and potential threats for DER assets [1].

More sustainable grid architectures with RES has also boosted the adoption of a class of OT called DER [1]. Global DER generation capacity is expected to reach 528.4 GW by 2026 with the US contributing 7.5 GW of Battery Energy Storage Systems (BESS). DER are interconnected, interoperable, and remotely controllable devices throughout the electrical distribution network. They can be categorized as:

1. storage: BESS, Inverter-Based Resources (IBR) and EV batteries.

2. generation: IBR, solar Panels, wind farms, local or backup generators.

3. controllable loads: HVAC, smart thermostats, smart appliances, EV charging, electric water heaters.

DER reduce both distribution costs and transmission losses as they provide power in proximity to its consumption [1]. They also buffer the grid from unforeseen demands and extreme weather events. The flexibility and autonomy of DER makes them invaluable to CPPS CI. Because most DER are derived from renewables rather than a mechanical input like turbines, their frequency cannot be modulated. Instead, the grid operator defines modes of operations and set points for each DER to integrate them with the power grid, which increases the cyber attack surface [6]. Because they are distributed by nature, DER must be remotely accessible over a network, thus providing attackers with a method of interacting with power flow. Attacks on CI are becoming increasingly popular. Energy, government IT communications, public utilities, and healthcare systems have recently been targeted by state and non-state actors [1]. The vulnerability of communications of a CPPS were demonstrated in Utah, US on 5 March 2019 where attackers conducted a DoS attack by exploiting vulnerable firewall devices [1].

Published works have shown that in CPPS, both SCADA networks and smart substations are vulnerable to FDI attack [28]. Where SCADA is used to monitor and control systems, a smart substation leverages SCADA for real-time control on a larger scale. Attacks on these system components allow an attacker to affect the wider grid without detailed or real-time knowledge of its state.

The following wired and wireless protocols enable monitoring and control between DER and management systems of utility aggregators [1]:

1. **IEEE 1815-DNP3**: Interoperable communication framework for secure information exchange in industrial systems (e.g., SCADA).

2. **Sunspec Modbus**: Modbus protocol extension for DER parameters (e.g., power and voltage) and ancillary services monitoring and control.

3. **OpenADR**: Energy market management standard regulating demand-response via signals to DER and other controllable devices.

4. **IEEE 2030.5**: Smart energy profile application standard and default protocol for DER management.

Embedded OT handles these communications using the above protocols for monitoring and executing remote commands, and of those protocols, the IEEE 2030.5 is the only protocol to be originally designed with strict cryptography requirements [1]. Table 2 highlights the vulnerabilities of DER communication protocols. However, it is still vulnerable to DoS and attackers can brute-force user-level credentials to reconfigure connected DER [6]. DNP3, which is used by 75% of utilities in the USA, and Modbus have data interruption, interception, modification, and fabrication vulnerabilities [1]. The latter two vulnerabilities could allow an attacker to reconfigure devices, issue commands, or present false measurements of the system's state. Both the Modbus and DNP3 protocols lack any encryption, authentication or error correction, allowing packet sniffing malware on the client or an intermediary to modify the contents of packets [29][27]. The open-source CERT Java-based implementation of OpenADR has documented data integrity vulnerabilities, which permit attackers to access sensitive user data [1].

While the latest standards for all these protocols functionally comply with IEEE 2030.5, limitations in the existing infrastructure inhibits the deployment of secure protocols and the entire system is only as strong as its weakest link. Operators are hesitant to replace otherwise functional legacy systems. Operation or device constraints may prevent

Protocol	SunSpec Modbus	IEEE 1815 (DNP3)	IEEE 2030.5 (SEP2)
Intermediary	X	X	
Replay	X	X	
DoS	X	X	X
Data Capture	X	X	
SSL Spoofing	X	X	
Data/Parameter Modification	X	X	X

Table 2: Vulnerabilities of DER communication protocols[6]

firmware updates to support encrypted communications, and vendor-specific protocol implementations don't necessarily include the latest security measures. These systems may be one or two decades old, and designed a decade before that [29]. Nor would replacing these devices address the issue, as the remaining vulnerable devices still provide footholds for an attacker. Encryption algorithms are designed to be irreversible, meaning the encryption computations are many orders of magnitude easier than brute force, however, quantum computers could remove this property for all but the most recent algorithms [1]. Even those legacy devices that could be upgraded to support encryption may not support the Quantum Key Distribution (QKD) schemes required for quantum-resistant encryption. Furthermore, false-positives may affect healthy DER, negatively impacting system operation [6]. Consequently, some utilities disable DER security functions for simplicity sake.

Additionally, DER devices are Commercial-Off-The-Shelf (COTS) devices, so multiple companies in different countries contribute to their intellectual property, design, or manufacture [1]. This makes DER devices vulnerable to supply chain attacks, because the supply chain is difficult to verify and other states may add security back doors. A recent precedent was 2020's SolarWinds attacks [1]. Some DER still rely on the plaintext User Datagram Protocol (UDP) traffic, which makes manipulating their data trivial. Not only can the attack read and replay the packet's contents, but also enables ARP poisoning and port stealing attacks. In these attacks the adversary inserts themselves as an *intermediary* between two CPPS devices, and modify, repeat, or drop packets sent between them. In microgrids, particularly during autonomous operation, disturbances caused by an intermediary attack could result in relays tripping, equipment damage, and load shedding events.

Renewable DER generators, i.e., solar and wind generation, are remotely operated by Internet-of-Things (IoT) devices to coordinate frequency and voltage deviations after transient events, and have controllers to match local user demand [1]. These devices are often owned by end-users, connected to user networks, and remotely operated by third-party and/or consumer applications, which is an impediment to a grid-wide security policy and increases the attack surface of the network. IoT devices typically compete on cost, leading vendors to prioritize that over security. This leads to a lack of host-based security like firewalls and antivirus, encryption, and firmware updates to patch exploits and add security function. Researchers have discovered commercial devices which expose user and device data and credentials in plaintext, or allow adversaries to authenticate by brute force. Additionally, [7] note that Honeywell controllers have known authentication vulnerabilities, Nest thermostats lack hardware protections to prevent the installation of malicious software, and Ardunio Yun controllers are similarly vulnerable to malware. Furthermore, smart thermostats are common IoT DER devices in demand side management/demand response schemes, as they learn user habits and limit demand during peak loads. In [1], the authors have documented that an attacker can toggle the state of con-

nected heaters and coolers, which modifies instantaneous demand and can lead to black or brown outs. In large-scale demand-side attack scenarios, grid operators are forced to shed non-critical loads and will also affect demand-response schemes. Vulnerabilities in IoT CPPS connected devices give attackers a vector to attack the wider network.

Similarly, Advanced Metering Infrastructure (AMI) involves IoT smart meters, which can be used to provide demand data to the system operators or demand response [28]. As an AMI network can span a neighborhood, they are generally connected through fixed wireless networks making them vulnerable to intermediary attacks. This vulnerability can enable an attacker to control the demand response by conducting a Load Altering Attack (LAA). A LAA overwhelms grid infrastructure by changing remotely controlled loads and causes a PQD such as a large frequency or voltage deviation, and may trip safety systems [24]. Demand response and demand side management are used to reduce peak loads, by deactivating high power devices or providing locally stored energy during peak demands while activating or charging these devices during low demand: for example, only running air conditioning during off peak hours. If an attacker can control these systems to increase the grid load at peak times, this is a LAA. LAAs can be classified as static (SLAA) or dynamic (DLAA), where the former's effect is proportional to the volume of altered load while the latter also aims to steer the system on a particular trajectory. DLAAs can use open or closed loop control: where an open loop could involve a replay FDI attack corresponding to a predetermined trajectory; a closed loop DLAA monitors and adapts to current measurements. Any type of LAA is contingent on a critical mass of vulnerable loads for any effect, which may affect the local network or have a ripple effect across the wider grid. Vulnerable loads in the literature include consumer appliances, HVAC devices, server farms, and electric vehicles.

The effects of a LAA include: frequency instability, cascading failures, or increased operating costs [7]. A significant and sudden change in load can result in an abrupt change in the CPPS operating frequency as generation is unable to match the load. Increased loads decrease the operating frequency and vice versa. For sufficiently large frequency drops, the operator may shed nonessential loads, which is called Under Frequency Load Shedding (UFLS) [23]. While a smaller change in load may be stabilized using reserve generation capacity, this increase in power may lead to line overloads and even cascading failures [7]. If the altered load slowly increased, the result may simply be that the operator must purchase additional power from a reserve supplier at an increased price. There is a possibility that a supplier would use such an attack to create a business opportunity, but far more insidious would be the use of such an attack in hybrid warfare, wherein a nation state may seek to increase its influence of a neighbor, or a central government over a breakaway or independent region. Table 3 shows the relative proportion of loads an attack must harness to achieve different goals; note the relatively small proportion of load required to cause a cascading failure. While 1% of a large national power grid may still be a tall request, this may be far more feasible on a regional or isolated CPPS. Note that this data is based on simulations of the Polish power grid, which is the largest and most detailed real-world power grid available for academic study [7].

Further research in [23] determined that a cascading failure was unlikely on a well configured North American regional system with additional protections. It assumes that all grid operators satisfy the N-1 security criterion, while noting that blackouts have occurred on such systems without malicious interference, due to mis-configurations of protections. The criterion specifies that the loss of a single component (generator or transmission line) does not reduce operational capacity or safety. Another prominent protection was load shedding, which automatically drops predetermined customers, meaning the larger grid is protected at the cost of localized blackouts, so this approach is a trade off. This can be triggered by low frequencies or declining voltage profiles. An attacker can manipulate the UFLS response with a DLAA that deactivates the controlled load once the frequency

Goal	Action	Impact	Load
Frequency deviation	Synchronously toggle loads	Generation tripping	20-30%
Blackout recovery disruption	Synchronously toggle loads after restart	Generation tripping	10-20%
Cascading line failures	Synchronously toggle loads distributed throughout the grid	Overload or trip transmission lines	0.4-1.0%
Increased operating costs	Slowly activate loads during peak demand	Increase reserve generation utilization	3-5%

Table 3: Selection of LAAs with the percentage of loads the adversary must control, based on detailed simulations of the Polish power grid [7].

stabilizes to cause frequency overshoot and take a generator offline. If the attacker can avoid detection, this attack can be repeated to force UFLS several times until no more loads can be shed and the generator must be disconnected. Neither study considers a power grid with a high penetration of RES, which would limit the operator's primary response. These rigorously simulated LAAs[7] demonstrate that an attacker who controls a sufficient load can degrade the performance of a CPPS, and cause cascading failures if safety mechanisms are absent or mis-configured. These works simulated an attacker controlling IoT loads. Widespread automated DR schemes would increase the attack surface.

DRL like any ANN has an inherent exploit called adversarial examples, where small and specific perturbations to their inputs cause large and possible targeted changes to their outputs [30]. As discussed in the previous section, DRL provides many advantages over conventional control in CPPS and may see wider adoption in CI. An attacker could use this property of ANNs to perturb the victim towards a target state, such as an uncomfortable temperature for a building environment system or unsafe operating frequencies for a power grid node. Research has demonstrated attacks that: minimize a DRL voltage regulator's reward (reward-based attack) by tricking it into taking sub-optimal actions; and lure it to a target failure state [21, 22]. The attack budget was constrained by an L_2 distance of 5%, meaning no input was more than 5% different from the unperturbed input, making the attack stealthy in a field where IDS is already a challenge. In a grid operational control simulation, a reward-based attack was sufficient to cause a critical failure [22]. The vulnerability of systems such as AMI to intermediary attacks makes this type of FDI feasible in the real world. Thus, adversarial examples represent an open exploit in a potentially valuable model-free control technique for CPS in increasingly complex environments.

Given the safety requirements of energy systems, physical constraints should be built directly into the DRL model, rather than as penalties in the reward function [10]. Because real-world data is often incomplete due to jamming, malfunctions, or attacks, control systems must be robust under these conditions [19]. Systems would be more robust if their controllers are incapable of unsafe actions, particularly in the face of cyber threats and adversarial examples.

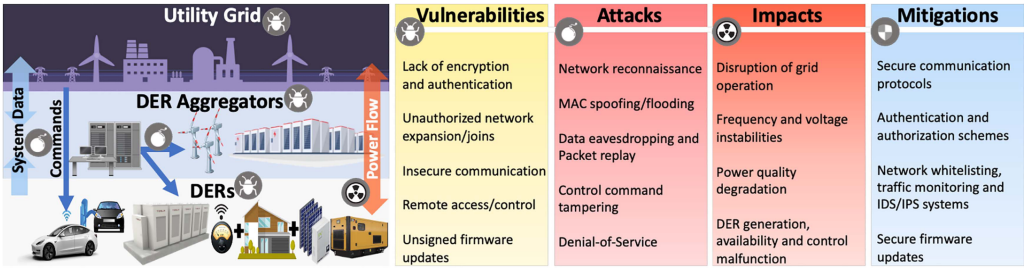


Figure 3: Figure 6 from [1]: "Overview of DER-integrated electric grid that illustrates how the layered architecture expands the threat surface" ©2023 IEEE.

2.2.1 Summary

OT is the foundation of CPPS and is not subject to the same security considerations as IT, causing it to lag in that domain. At present, CPPS rely on unsecure legacy devices or IoT devices which can provide a foothold for attackers or leave devices vulnerable to FDI through intermediary attacks. The capital investments often make upgrading these systems infeasible. There are numerous and well resourced threat actors interested in compromising CI, and the exposure of CPPS to the open internet provides ample opportunity. Finally, because ANNs are fundamentally exploitable, it's conceivable that adversarial samples present a stealthy payload for FDI.

2.3 Adversarial Examples

An evasion attack is used in this work instead of a poisoning attack that happens during training time. This is because there is no timing restriction for the attacker, i.e., the evasion attack could happen anytime the control system is operating. Furthermore, during inference, the victim must be exposed to the real world, while its training can be isolated. Because any algorithm must observe the real world to make decisions, observation perturbations are always a potential attack vector. For example, an attacker could manipulate the sensors' environment, compromise their firmware, or alter their transmissions through adversarial examples.

Adversarial examples are specially crafted samples which are misclassified by ANNs, where small perturbations are added to samples. An adversarial example closely resembles the sample used to create it, and may be indistinguishable. For example, an adversarial example of an object in an image would be recognizable to a human but not to an ANN classifier [31]. Since the technique was first discovered by the authors in [30], creating adversarial examples with higher Adversarial Success Rates (ASR) has been a topic of research for several years. While generating adversarial examples is possible for both classification and regression tasks, most techniques were developed for image classification so many implementations do not support regression. These two tasks use different loss functions, and regression networks have only one output per "head" of the network. This means implementations for generating adversarial examples for classifiers are not directly compatible with regressors.

ML classification predicts the class of samples of some distribution D. Supervised training uses samples $x \in R^d$ with known labels $y \in \{Y\}$ (where Y is the set of all possible labels) to find parameters θ which minimize the prediction loss $L(x, y, \theta)$ [31]. In this way, supervised training is formulated as a minimization:

$$\min_{\theta} E_{(x,y)\sim D} L(x, y, \theta) \tag{1}$$

which can be solved through iterative gradient descent, typically Stochastic Gradient

Descent (SGD). This problem can be reformulated as a maximization to find perturbation δ which maximize loss within some boundary $\epsilon \subseteq R^d$:

$$\max_{\delta \in \epsilon} E_{(x+\delta,y)\sim D} L(x, y, \theta) \tag{2}$$

As above, this problem can also be solved through gradient-based methods. These methods for producing perturbations are known as adversarial attacks, which generate samples called adversarial examples (or samples). Consequently, human imperceptible changes to a model's input can result in high confidence mis-classifications. The simplest method is the Fast Gradient or Fast Gradient Sign Method (FGM/FGSM), which adds a fixed perturbation ϵ with the sign of the loss function's (L) gradient to the original sample [32]:

$$x_{adv} = x + \epsilon \times sign(\nabla L(x, y, \theta)) \tag{3}$$

This method adds a perturbation in the direction of increased loss by taking only one step. More advanced attacks use multiple iterations to maximize the loss by taking multiple steps.

Multiple metrics exist to measure the success of attacks, precisely [33]:

1. Adversarial Success Rate (ASR): the proportion of successful adversarial examples, i.e., where an observation perturbation results in the victim deviating from its policy. ASR is best for discrete action spaces, whereas Mean Absolute Error (MAE) will be used for continuous action spaces. MAE measures how much a continuous action has changed, whereas ASR is used to measure if a sub-optimal discrete action was selected.

2. Adversarial regret: the effects of the adversary measured either as the percentage of time in adversarial state during an attack or the reduction of total rewards in testing, depending on whether the adversarial goal is state- or reward-based. Adversarial regret measures how much an attack degrades performance.

3. The ASR ratio between untargeted adversarial examples and random noise shows the relative efficacy of adversarial examples.

4. The statistical distance between training and perturbed observation distributions, which measures their resemblance. This metric demonstrates if the attack is trivial to detect.

5. Adversarial budget: the proportion of perturbed observations, the features perturbed, and the perturbation size ϵ (if applicable). Generally, smaller perturbations are harder to detect, as large perturbations can distort the original observation.

Attacks can be divided into black and white box attacks, where the former allows the attacker only the ability to query the victim ANN, while the latter provides read access to the ANN's weights and architecture. Most white box attacks are iterative methods which use a gradient of the loss function to guide their searches. White box attacks generally have higher ASRs, as access to the victim ANN allows them to follow the gradient of the loss function. However, this advantage over black-box attacks disappears when the gradient is masked [34].

Attacks are also categorized as maximum-confidence or minimum-norm [35]. A maximum-confidence attack finds adversarial examples which maximize loss within a given boundary, whereas a minimum-norm attack finds the smallest perturbation which causes mis-classification. Figure 4 visualizes this concept. A variety of techniques exist for crafting adversarial examples which are difficult to separate from the original data when properly constrained.

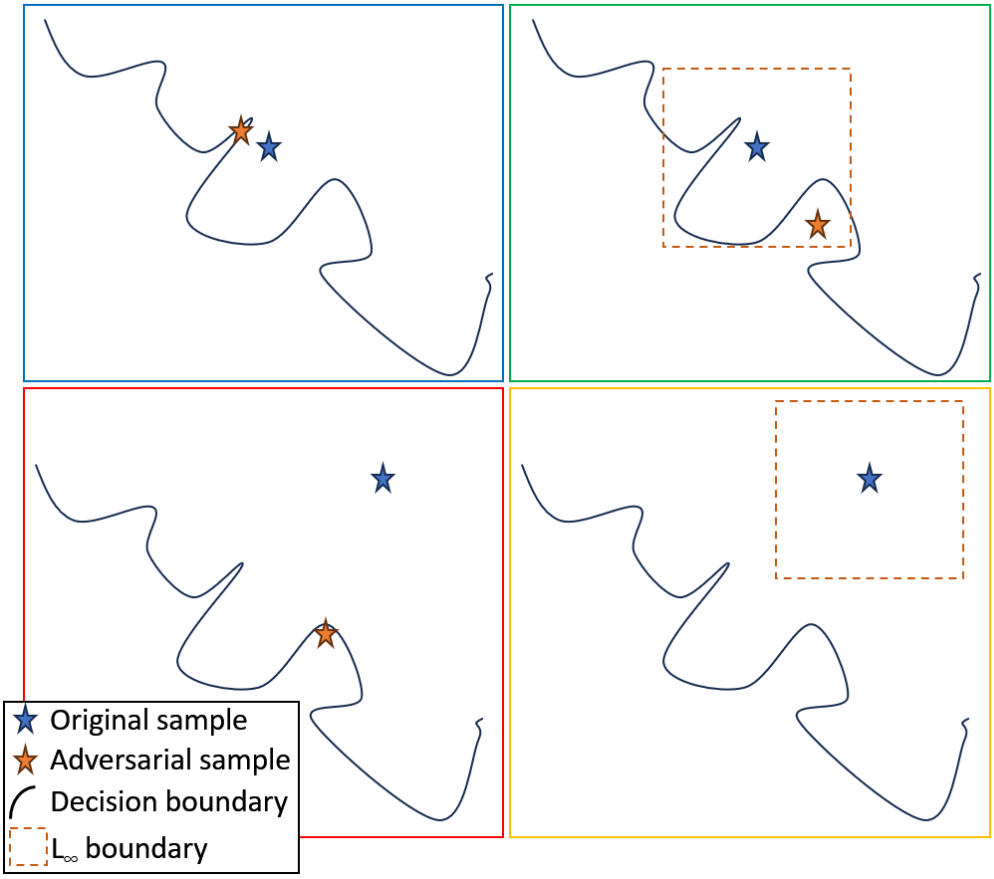


Figure 4: A visualization of minimum norm and maximum-confidence attack scenarios. The upper quadrants consider samples near the decision boundary, and far away in the lower ones. The minimum norm attacks on the left find the closest adversarial sample, which are just past the closest point on the decision boundary. The maximum-confidence attacks on the right attempt to find the furthest point on the opposite side of the decision boundary, within the attack budget. Because of this budget, the maximum-confidence attack is successful in the green quadrant, but not in yellow because the original sample is too far from the decision boundary. Both min norm attacks are successful, with the adversarial sample in the blue quadrant closer to the original than the maximum-confidence attack in yellow. The minimum norm attack in red is successful with an adversarial sample that is far from the original, demonstrating that minimum norm attacks sometimes result in adversarial samples that are *further* from the originals than a maximum-confidence attack.

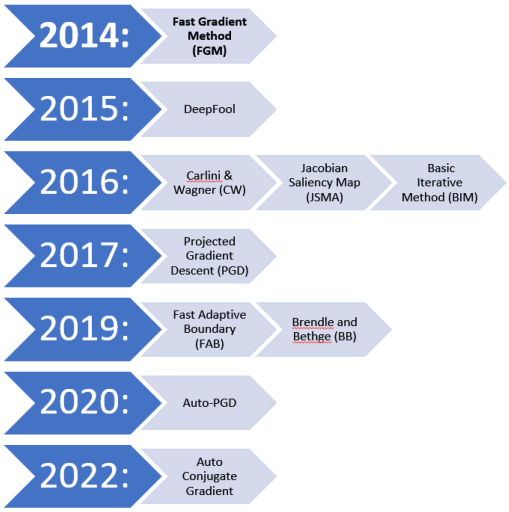


Figure 5: A brief timeline of open source adversarial attacks.

2.3.1 White Box Evasion Attacks

Several open source modules are available for generating adversarial examples, this review focused on tools which were compatible with PyTorch models:

1. Adversarial Robustness Toolbox (ART) [36]
2. AdverTorch[37]
3. Clever Hans [38]
4. Foolbox [39]
5. TorchAttacks [40]

ART, TorchAttacks, and Foolbox are actively maintained and implement a large number of attacks. ART was the only module to include adversarial defences. These tools limit the loss functions and types of models which can be attacked, meaning they are limited to attacking classifiers. Some AdverTorch attacks do allow arbitrary loss and prediction functions, but these are restricted to older attacks because the library has not been updated in years.

Different attack techniques are optimized for maximum-confidence or minimum-norm perturbations. See figure 5 for a brief chronology of adversarial attacks. Maximum-confidence white box attacks designed for the highest ASRs, such as Projected Gradient Descent (PGD), Auto-PGD (APGD), or ACG, are given perturbation budgets ϵ which represent the largest deviations for the original sample. These are constrained by the L_1, L_2, or L_∞ distances, which are the number of perturbed features, the distance in space between the original and adversarial example, and the maximum perturbation which can be added to any feature.

Minimum-norm white box attacks like Fast Adaptive Boundary (FAB) and the older Brendle and Bethge (BB) don't have this constraint as they are designed to minimize the size of successful perturbations. FAB is a notable omission from ART as it's the most recent minimum-norm attack implemented in any of the libraries listed above. However, it's limited to use with image data and therefore not relevant to this research. The BB attack is used instead.

2.3.2 Maximum-Confidence Attacks

Recent gradient-based attacks like PGD methods and ACG iteratively step away from the starting point (normal sample) to a point which maximizes the loss function (adversarial example), requiring a value for the step-size and a non-zero gradient to follow:

$$x^{(k+1)} = P_S(x^{(k)} + \eta^{(k)} s^{(k)}) \tag{4}$$

Where P_S is a projection onto the region S, $\eta^{(k)}$ is the step-size, which can be constant or vary by iteration, and $s^{(k)}$ is a vector which determines the direction of the step. $s^{(k)}$ incorporates the loss function's gradient at point $x^{(k)}$, with the exactly calculation varying by attack. The search space $S = \{x \in D, ||x_{init} - x_{adv}||_s < \epsilon\}$ is the input space constrained by a regularization function, typically the L_∞, L_2, or L_1 distance between the sample and the adversarial example. $s^{(k)}$ is calculated differently for different attacks, but each method iteratively calculates x over k iterations.

The simplest and original gradient-based attack, the Fast Gradient Sign Method, uses $s^{(k)} = sign(\nabla L(x^{(k)}))$, while the improved PGD attack uses $s^{(k)} = \nabla L(x^{(k)})$. These methods of gradient-based updates require that $s^{(k)} \neq 0$ and an appropriately selected $\eta^{(k)}$, either constant or variable by iteration. With the auto-stepsize η chosen automatically rather than manually tuned as in [34], accurate robustness assessments are simpler. The APGD attack improves upon the widely used PGD method for assessing model robustness, by automatically scaling the step size along the gradient of the loss function [34]. The strongest open source white box attack is the Auto Conjugate Gradient (ACG) method, only available through the ART [41].

In [34], the authors introduce the Difference Logits Ratio (DLR) loss function to replace Cross Entropy (CE) loss, widely used with PGD and for training models on multi-classification problems. The gradient of CE loss approaches 0 when the victim is nearly certain of its prediction for sample x with the correct class y:

$$\nabla_x CE(x, y) = (p_y - 1)\nabla_x z_y + \sum_{i \neq y} p_i \nabla_x z_i \tag{5}$$

$p_y \approx 1$ when the victim has learned to label x as y, meaning the probability of predicting another class $p_{i \neq y} \approx 0$ and both terms of the CE loss are near 0. Following the example above, if $s^{(k)} \approx 0$, then $x^{(k+1)} \approx 0$. DLR loss solves this issue because the difference between logits can always be measured, as each logits indicates the model's preference for the corresponding label and a prediction is made based on the largest logit [34].

$$DLR(x, y) = -\frac{z_y - \max_{i \neq y} z_i}{z_{\pi_1} - z_{\pi_3}} \tag{6}$$

Untargeted DLR loss is only positive when any logits greater than the logit of the original class z_y [34]. DLR Loss is regularized by the difference between the largest and third largest logits $z_{\pi_1} - z_{\pi_3}$, which pushes the original label's logit to be the second largest $z_y \approx z_{\pi_2}$.

$$Targeted - DLR_T(x, y) = -\frac{z_y - z_t}{z_{\pi_1} - \frac{z_{\pi_3} + z_{\pi_4}}{2}} \tag{7}$$

Targeted-DLR loss only uses the difference between the original and target class logits, rather than any logit but the target [34]. The regularization in the denominator is modified to prevent the loss from becoming constant.

The "Auto in APGD represents The automatically scaling the step-size η over a finite number of iterations, which allows the algorithm to explore, then exploit within the adversarial example space S [34]. The large initial step-size is effective for exploring S, and when one of two conditions is met, η is halved to explore a smaller space with

increasing resolution. These conditions are checked at predetermined checkpoints: if a predefined proportion of iterations between the current and previous checkpoints don't show improvement (I) or neither the step-size nor the best result has changed since the last checkpoint (II) $\eta = \frac{\eta}{2}$. Together these conditions will change η when little or no improvement is shown with the current value. This is valuable because a large η will efficiently find areas of high and low losses, but smaller values are needed to exploit this knowledge and avoid skipping of a maxima.

PGD calculates the search direction $s^{(k)}$ in which to take a step of size η using $s^{(k)} = \nabla f(x^{(k)})$, and APGD uses the same method in addition to a momentum term [34]. This is a projection of the loss function's gradient. ACG incorporates auto-step-size and DLR loss, while defining $s^{(k)}$ using the conjugate gradient method, as defined using the equations below. The Conjugate Gradient (CG) method is an iterative solution to optimization problems [41].

$$y^{(k-1)} = \nabla f(x^{(k-1)}) - \nabla f(x^{(k)}), \tag{8}$$

$$\beta^{HS} = \frac{\nabla f(x^{(k-1)}) \cdot (y^{(k-1)})}{(s^{(k-1)} \cdot y^{(k-1)})}, \tag{9}$$

$$s^{(k)} = \nabla f(x^{(k)}) + \beta^{HS} s^{(k-1)}, \tag{10}$$

$$x^{(k+1)} = P_S(x^{(k)} + \eta(k) \cdot \sigma(s^{(k)})) \tag{11}$$

The CG update parameter β incorporates past search information into the current step, and this parameter is the primary distinction between APGD and ACG [41]. Seven of the methods for computing the CG parameter in [42] were tested in [41]. The HS method was most performant in all test cases. Though no theoretical analysis was provided, it's notable that the 3 best methods share the same numerator. These methods are not susceptible to jamming when the distance between search points $x^{(k)} - x^{(k-1)}$ approaches 0, because the y also becomes 0 so the direction of the next update is $s^{(k)} = \nabla f(x^{(k)}) + 0$ [42]. Where other methods get jammed making insignificant search updates, methods like HS exhibit a restarting behaviour because the initial update is also $s^{(0)} = \nabla f(x^{(0)})$. ACG differs from typical CG methods by using the auto stepsize η from [34] instead of a linear search.

Algorithm 1 shows the ACG algorithm. Its parameters can be divided into three categories:

1. Computational Constraints:
   (a) The number of random restarts, with a default of 5.
   (b) Maximum iterations per restart, 100 by default.
2. Distortion Constraints:
   (a) ϵ: The maximum perturbation magnitude, which is the distance between the input observation and adversarial example.
   (b) Normalization method to constrain the distance between the sample and adversarial example. The options are:
      i. L_1 known as Manhattan or taxi-cab distance, is the most restrictive and least common in this work's review.
      ii. L_2 is the Euclidean distance.
      iii. L_∞ is often referred to as a box around the input observation, and allows each feature to change $\pm\epsilon$ and is the least restrictive and most common of these normalizations.

3. Search parameters:

   (a) The initial step-size $\eta^{(0)}$ determines how far the search explores, with lower values favoring exploitation: $\eta^{(0)} = \frac{\epsilon}{3}$ is ART's default, however both [34, 41] set $\eta^{(0)} = 2\epsilon$.

   (b) Loss function: The search for adversarial examples is formed as an optimization problem using the loss function. Changes are iteratively made to the input sample in the direction of the loss function's gradient. CE loss is used for training ML algorithms in multi-classification problems and is used in AML. However, Difference Logits Ratio (DLR) loss has been shown improved performance in producing adversarial examples for classifiers [34]. Both CE and DLR are compatible with ART's implementation of ACG, but all experiments will use DLR loss.

   (c) The search can be either targeted or untargeted. An untargeted attack search for an adversarial example leads to a prediction of any class but the original. This is analogous to forcing a DRL agent to choose any action which is not the learned action. A targeted attack seeks an adversarial example which leads to the prediction of a specific class. A targeted attack can be used to choose the victim's actions.

Where PGD calculates the step direction using only the current input's loss gradient, ACG adds the product of the previous step and the CG update parameter. Compared to APGD, ACG's search travels further in the early stages of the attack when the auto step size is large, which increases the diversification/explorations of its search [41]. This can be explained by projection loss from APGD's steps taking the search outside the boundaries. Wherein, projecting the search step back to the feasible region S reduces the update's magnitude and confines the APGD search close to the initial point. With this advantage, ACG outperformed APGD in terms of adversarial regret for SotA image classifiers, often using a fifth of the computation budget. ACG with a single random restart and 100 iterations outperformed APGD with 5 restarts and 100 iterations each. From these results, ACG is the most powerful white box attack available in an open source tool like ART. However, this review did not find any work using ACG or APGD for robustness testing of a DRL agent.

The perturbation boundaries make maximum-confidence attacks useful when the goal is ensuring that a model is robust to perturbations of a specific size. Because these attacks will fail if a sample of sufficient loss does not exist within the boundary, the boundary's size is a useful robustness metric. Conversely, minimum-norm attacks have no such constraints in searching for adversarial examples. Their goal is finding adversarial samples close to the decision boundary, minimizing the distance between the adversarial and clean samples, even when this distance is, in fact, relatively large. Thus, a minimum-norm attack can demonstrate higher ASRs than a maximum-confidence attack because their searches are not so constrained.

2.3.3 Minimum-Norm Attacks

The Brendle and Bethge attack is the most recent open source minimum-norm attack which supports tabular data [2]. As shown in Algorithm 2, BB begins with a sample $\tilde{x}$ of the adversarial class, and uses the gradient of an adversarial loss function $adv(.)$ to step towards a local estimate of the decision boundary b and minimize the distance to the original sample x. Figure 6 provides a visual explanation. In the ART implementation, $adv(.)$ can be DLR [34] or CE loss [36].

The authors of BB show that the algorithm is not sensitive to changes to its hyperparameters, suggesting the default values should be used in most cases [2]. These

Algorithm 1 ACG [41]

Input: f, feasible region S, initial sample $x^{(0)}$, initial stepsize $\eta^{(0)}$, N_{iter}, checkpoints $W = \{w_0, \ldots, w_n\}$
Output: x_{adv}
$x_{\text{adv}} \leftarrow x^{(0)}; \beta^{(0)} \leftarrow 0; s^{(0)} \leftarrow \nabla f(x^{(0)})$
$x_{\text{pre}} \leftarrow x^{(0)}; s_{\text{pre}} \leftarrow s^{(0)}$
$N_{x-update}, f_{max} \leftarrow 0$
for $k = 0$ to $N_{\text{iter}} - 1$ **do**
    Compute $x^{(k+1)}$ using equation (11)
    if $f(x^{(k+1)}) > f(x_{\text{adv}})$ **then**
        $x_{\text{adv}} \leftarrow x^{(k+1)}; x_{\text{pre}} \leftarrow x^{(k)}; s_{\text{pre}} \leftarrow s^{(k)}$
        $N_{x-update} \leftarrow N_{x-update} + 1$
    end if
    $\eta^{(k+1)} \leftarrow \eta^{(k)}$
    if $k \in W$ **then** //checkpoint reached
        if $(N_{x-update} < \rho)$ or $([\eta^{(k)} \equiv \eta^{(k-1)}]$ and $[f(x^{(k)}) \equiv f_{max}])$ **then**
            $\eta^{(k+1)} \leftarrow \eta^{(k)}/2;$
            $x^{(k+1)} \leftarrow x_{\text{adv}}; x^{(k)} \leftarrow x_{\text{pre}}; s^{(k)} \leftarrow s_{\text{pre}}$
        end if
        $N_{x-update} \leftarrow 0$ //reset count for next checkpoint
        $f_{max} \leftarrow f(x_{adv})$
    end if
    Compute $\beta^{(k+1)}$ using equation (9) and $s^{(k+1)}$ using equation (10)
end for

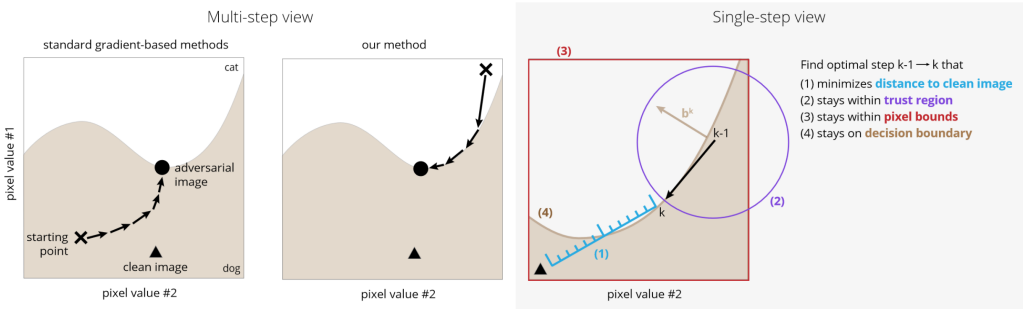


Figure 6: Fig 1 from [2]: "Schematic of our approach. Consider a two-pixel input which a model either interprets as a dog (shaded region) or as a cat (white region). Given a clean dog image (solid triangle), we search for the closest image classified as a cat. Standard gradient-based attacks start somewhere near the clean image and perform gradient descent towards the boundary (left). Our attacks start from an adversarial image far away from the clean image and walk along the boundary towards the closest adversarial (middle). In each step, we solve an optimization problem to find the optimal descent direction along the boundary that stays within the valid pixel bounds and the trust region (right)". b^k is the normal vector of the decision boundary at the current adversarial sample $\tilde{x}_k$.

Algorithm 2 BB Generate Algorithm [2].

Inputs: clean sample x, the min and max values for x, momentum M, learning rate lr, learning rate decay interval , learning rate decay $decay$, differentiable adversarial criterion $\texttt{adv}(\cdot)$ (the difference between the model logits for the clean and the target class, where the target is any but the clean class for untargeted attacks)
Optional Inputs: adversarial starting point $\tilde{x}^0$
Result: adversarial example $\tilde{x}$ such that the distance $d(x, \tilde{x}_k) = \|x - \tilde{x}_k\|_p$ is minimized
$k \leftarrow 0$
$b^0 \leftarrow 0$
If no $\tilde{x}^0$ is given: $\tilde{x}^0 \sim U(0,1)$ s.t. $\tilde{x}^0$ is adversarial (or sample from the target class)
for $k <$ maximum number of steps , $k \leftarrow k+1$ **do**
    $b^k := \nabla_{\tilde{x}^{k-1}} \text{adv}(\tilde{x}^{k-1})$ // estimate local geometry of adversarial boundary
    if $k > 1$ **then**
        $b^k \leftarrow (1-M) \times b^{k-1} + M \times b^k$ // update estimate with momentum
    end if
    if lr decay interval reached **then**
        $lr \leftarrow lr \times decay$
    end if
    $c^k = \text{adv}(\tilde{x}^{k-1})$ // estimate distance to local boundary, where $c^k = 0$
    $r^k \leftarrow lr \times (max - min)$ //the max step for each iteration
    $\delta^k \leftarrow L_p Optimizer(x, \tilde{x}^{k-1}, b^k, min, max, c^k, r^k)$
    $\tilde{x}^k \leftarrow \tilde{x}^{k-1} + \delta^k$
end for

hyperparameters can be divided into two categories:

1. Initialization:
    (a) Number of random samples: the number of times $\tilde{x}^0$ is sampled from $U(0,1)$ to produce a starting point of the adversarial class. The class is the target for a targeted attack, or any class but the predicted class for untargeted attacks.
    (b) Binary search steps: The number of search iterations used to locate the decision boundary between x the original sample and starting point $\tilde{x}^0$.
2. Optimization:
    (a) Trust Region (TR): like the learning rate, this value specifies how large of a step the search takes at each iteration.
    (b) Number of TR decays: the number of times the TR is reduced during a search.
    (c) TR decay rate: proportion the TR is reduced.
    (d) Momentum: Specifies the proportion of the current and previous search direction used to determine the next perturbation δ.
    (e) Number of iterations: number of steps taken to find the smallest distance between x and $\tilde{x}$.
    (f) Overshoot: specifies how far $\tilde{x}$ should be from the decision boundary, ensuring the class is adversarial.

2.4 Defences

In terms of defences, several methods for defending against adversarial examples have been proposed. In this thesis we focus on two defense paradigms: Detection and cor-

rection, knowing that the State of the Art (SotA) is to build models robust to attacks rather than detecting the attacks directly [20].

2.4.1 Detection

Detection techniques can roughly be categorized as statistical or built-in to a model. Statistical detection is model agnostic as it determines if the distribution of model inputs matches that of the training data. The underlying assumption is that adversarial examples are statistically distinct from the training data, as they would otherwise be correctly classified [43]. While this assumption only holds for relatively weak attacks, the two works cited above demonstrate Maximum Mean Discrepancies (MMD) for comparing inputs during training and inference. Thus, MMD is a measure of the attacker's stealth, showing that attacks can be indistinguishable from normal inputs.

MMD is used to determine if randomly drawn samples belong to the same underlying distribution. Consider the samples $X = \{x_0, ...x_n\}$ and $Y = \{y_0, ...y_n\}$, for a sample X drawn from a distribution p, $X \sim p$, and $Y \sim q$. The purpose of the MMD test is to determine if $p = q$. This test can be applied in high dimension spaces using the kernel trick, where the kernel function is denoted as $f(x)$. Thus the distance between two probabilities represented by X and Y is [43, 44]:

$$MMD[X, Y] = (\frac{1}{n}\sum_{i=0}^{n} f(x_i) - \frac{1}{m}\sum_{i=0}^{n} f(y_i)) \tag{12}$$

In addition to the distance, the p-value that $p = q$ can be calculated by repeating the test many times while sub sampling X and Y with replacement, is called bootstrapping.

An attacker can generally bypass methods which add preprocessing to the input pipeline by mimicking that step during the attack, and crafting new loss functions to ensure models with an adversarial example class are still fooled, although, some defences require an increased adversarial budget to be bypassed [20]. Semi-successful methods rely on the unique property of adversarial examples: small changes have large effect on the model's output. For example, using an ensemble of predictions from a model with dropout layers, or testing if additional noise changes the classification [20, 45].

Notably, these techniques do not rely on some underlying structure of the original training dataset. Because adversarial examples rely on small and specific perturbations to the victim's input, they themselves are fragile to perturbations which interrupt their delicate structure. The goal of model training is generalization. It should be robust to noise, but this does not hold for adversarial examples which are crafted using gradient's from the ANN's weights.

2.4.2 Correction

Robust training of an ANN can be formulated as a saddle or min-max problem, where the goal is to produce training samples which maximize loss then learn model parameters which minimize loss in the worst case scenario [3]:

$$\min_{\theta}[\max_{\delta \in \epsilon} E_{(x+\delta, y)\sim D} L(x, y, \theta)] \tag{13}$$

If the inner maximization is satisfied with an adversarial attack, i.e., no greater loss is possible in the boundary ϵ surrounding the input x, and the model parameters satisfy the outer minimization such that loss nearly vanishes, the model is perfectly robust [3]. In practice, it cannot be certain that the inner maximization is satisfied as future attacks may introduce greater losses. By combining minimum-norm FAB, maximum-confidence APGD, and black box Square attacks, the auto-attack demonstrated significant reductions in the robust accuracy of SotA image classifiers [34]. Square Attack is a black-box

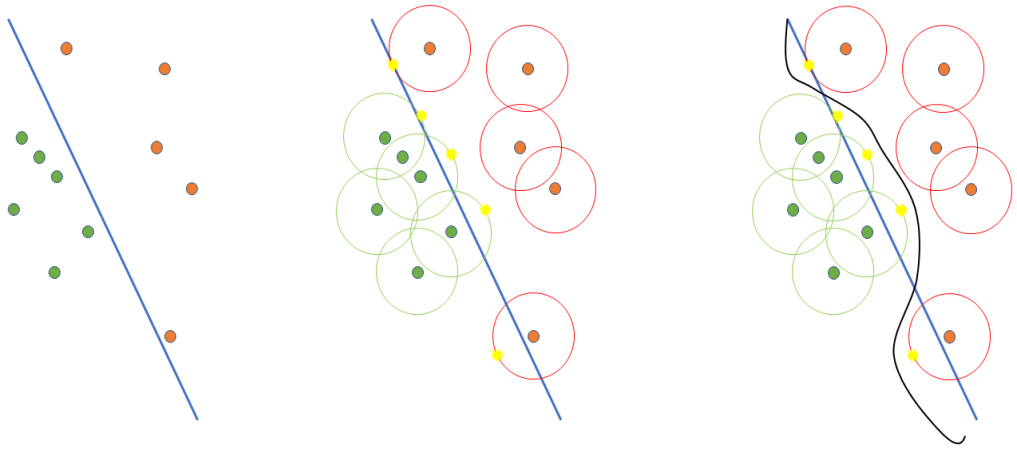


Figure 7: Illustration of standard and robust decision boundaries [3]. The leftmost figure shows a collection of points which can be separated by a linear decision boundary. The middle figure shows that the linear decision boundary does not separate points by the distance represented by the circles. The circles represent the ϵ boundary for a maximum-confidence adversarial attack. The yellow stars represent potential adversarial examples, which would be mis-classified. The right figure shows that separating the points outside the boundary for the attack requires a more complicated decision boundary. Using the decision boundary on the right, the classifier is robust to attacks within ϵ.

maximum-confidence attack for image classifiers, which uses a random search instead of gradient-based optimizations. Compared to other black-box attacks, it has a higher success rate using the same or a smaller numbers of queries, and approaches the performance of white box attacks [46]. FAB and Square attack increase the ASR for masked gradients, and APGD with DLR loss improves upon the previous SotA attack for robust training, PGD with CE loss. Furthermore, ACG has since surpassed the performance of APGD [41]. Hence, this method of adversarial training can guarantee that an ANN is robust to the attack model.

Model capacity is a factor for successful robust training, as the model must not only learn the boundary between different classes, but learn a boundary which is far enough from every sample that the attack model cannot find a perturbation $\delta \in S$ that results in misclassifcation as shown in Figure 7 [3].

2.5 Summary

None of the literature covered has tested the latest white box attacks on an DRL agent, with the most recent being PGD. This review finds no work on detecting adversarial inputs for DRL agents as most research concerns image classification. While the most successful detection techniques include an output for adversarial perturbation detection, it is unclear what action should be mapped to this output. Because statistical methods are algorithm agnostic, they could be employed with DRL agents. This makes statistical techniques a reasonable starting point for adversarial example detection for RL.

3 Related Work

Supervised learning models are trained to correctly label features in tasks like classification and regression. The former is categorizing input from a finite set of categories like pictures of cats and dogs for example, while the latter outputs a continuous value like the value of a house. DRL agents are trained to perform the optimal sequence of actions in a given environment. Where supervised models require labeled training data to associate a label with the given features, DRL agents learn with feedback provided from the environment. Thus, DRL agents do not need a dataset to train from but need an environment to train in. DRL problems are generally structured in terms of a Markov Decision Processes (MDP), which has four components [13]:

1. The state $s_t \in S$ is a description of the current environment at time t, where S is the set of all possible states. The present state can be denoted as s, and the following state s'. It contains all relevant information of previous states such that knowledge of the previous state is unnecessary.

2. The action $a_t \in A$, where A are the actions available to an agent and a_t is the chosen action at time t.

3. The transition function (also called dynamics) T (s, a) or $p(s', r|s, a)$ indicates the environment's next state s' and the agent's reward r:

$$\sum_{s' \in S} \sum_{r \in R} p(s', r|s, a) = 1 \tag{14}$$

4. The reward $r(s, a)$, $r \in R$ where R is the set of all rewards. The reward is a function of the current state and action, and normally a scalar:

$$r(s, a) = \sum_{s' \in S} r \sum_{r \in R} p(s', r|s, a) \tag{15}$$

   The environment is deterministic if there is a unique value for each state-action pair, or stochastic if it's a probability distribution function. Following from the transition function is the state-transition function:

$$p(s'|s, a) = \sum_{r \in R} p(s', r|s, a) \tag{16}$$

   The reward can also be understood in terms of the state-action pair s, a and the next state s':

$$r(s, a, s') = \sum_{r \in R} r \frac{p(s', r|s, a)}{p(s'|s, a)} \tag{17}$$

An DRL agent may observe s and choose an action, but T is often unknown. The strength of DRL is its efficacy when the state transitions are unknown and conventional control fails. DRL agents learn to act optimally by estimating the rewards for their given states and/or actions, and updating those estimates as training progresses using the following concepts:

1. An agent's policy $\pi(a|s)$ maps a probability for taking a given action for the given state. Like T, where the policy returns a single non-zero value corresponding to an action if it's deterministic, and otherwise a probability if it's stochastic. Since a deterministic policy corresponds to a single action for any state, it can be written as $\pi(s)$. The policy is how the agent behaves in a given situation.

2. The value function $v_\pi(s)$ is the expected value of all future rewards for a given state following π:

$$v_\pi(s) \doteq \sum_{a \in A} \pi(a|s) \sum_{s' \in S} \sum_{r \in R} p(s', r|s, a)[\gamma v_\pi(s') + r] \tag{18}$$

Where $\gamma \in [0, 1)$ is the discount factor, which dictates the importance of rewards of future states.

3. The action-value function $q_\pi(s, a)$ is the reward for taking action a in state s then following π for all subsequent states:

$$q_\pi(s, a) \doteq \sum_{s' \in S} \sum_{r \in R} p(s', r|s, a)[\gamma v_\pi(s') + r] \tag{19}$$

4. The optimal policy is denoted π^* which corresponds to the optimal value and action-value functions v_* and q_* defined as:

$$v^*(s) \doteq \max_\pi v_\pi(s) \tag{20}$$

$$q^*(s, a) \doteq \max_\pi q_\pi(s, a) \tag{21}$$

and π^* itself is:

$$\pi^*(a|s) \doteq \arg\max_a \ q^*(s, a) \tag{22}$$

such that there is no state in which another policy would obtain a higher value:

$$v^*(s) > v(s) \text{ for all } s \in S \tag{23}$$

It is impractical to store all the possible policy and values for every state and action of complex environments. Instead, $\pi(a|s)$, $q(s, a)$ and $v(s)$ can be parameterized as $\hat{\pi}(a|s, \theta)$, $\hat{q}(s, a, \theta)$ and $\hat{v}(s, \theta)$ where $|\theta| << |S|$ [13]. Note that the actual functions are $q(s, a)$ and $v(s)$, whereas $Q(s, a)$ and $V(s)$ are estimates and often used as shorthand for $\hat{q}(s, a, \theta)$ and $\hat{v}(s, \theta)$. Often, non-linear functions are approximated with neural networks, for example the $\hat{v}(s, \theta)$ would have features representing the state and approximate the state's expected value. The features used to represent s are called the agent's observation as it is impractical to perfectly capture and represent all aspects of the state.

There are two DRL algorithms which are particularly relevant to this work, the Proximal Policy Optimization (PPO) [47], and Soft Actor-Critic (SAC) [48]. Both are *actor-critic* algorithms, which use separate policy π_θ and critic $V(s)$ networks rather than an action-value network $Q(s, a)$.

PPO is an on-policy policy-gradient algorithm which can operate with continuous and discrete action spaces. As a policy-gradient algorithm, PPO improves its policy (the actor) π_θ using gradient ascent of an *advantage function*. The advantage function is loosely defined as the difference between the model's expected and actual rewards at a given timestep, as predicted by the *critic* $V(s)$. The PPO is unique for the surrogate loss function used to train its policy, which prevents large changes during each update.

An on-policy algorithm like PPO learns from experiences of actions sampled from its current policy alone, while an off-policy algorithm like SAC learns from the stored experiences of previous policies. The SAC is unique for its combination of off-policy actor-critic training, with a stochastic actor with an entropy maximization objective to encourage exploration. While the SAC outperforms the PPO on various continuous control tasks, it's limited to continuous action spaces [48].

3.1 Adversarial DRL

With the use of ANN heuristics, DRL agents are vulnerable to adversarial examples, as are ANNs. However, the optimal approach is different. Maximizing the loss function for a supervised network only guarantees that a victim DRL agent will mis-classify its observation of the current state. This will lead to a sub-optimal action but might be insignificant over the course of an episode. Other methods are required to select the worst possible actions [12]. In fact, the attacker must first choose a goal for optimization. The following goals have been used in adversarial DRL literature [8] :

1. Untargeted: Randomly altering the victim's behaviour.
2. Action: Altering the victim's behaviour towards specific actions.
3. Reward-based: Maximizing the adversary's or minimizing the victim's rewards.
4. State-based: Luring the victim to a target state.

For example, [49] shows that untargeted attacks can produce sub-optimal performance, demonstrating that the victim performs worse than a conventional control algorithm when attacked. Here, untargeted refers to perturbations of the observed state where there was no targeted classification, while the attacker's goal was reward-based.

On the other hand, to attain its goal, the attacker can use different attacks vectors. DRL attack space can be taxonomized as [12]:

1. Reward perturbations: An attacker can perturb or flip the rewards of the victim agent. These attacks are conducted during training (poisoning attack), where the victim uses the reward signal to learn its policy. Perturbations to the reward during training cause the victim to learn a sub-optimal policy. Because the reward function is integrated with the victim, poisoning generally requires more access to the victim than other methods.
2. Action perturbations: A rare attack type which changes the action executed by the victim. Altering the actions of an agent, leading it to an unexpected state. For example, physically perturbing an actuator.
3. State or observation perturbation: This is the perturbation of the victim's perception of its environment, by changing the environment itself or modifying the victim's observations. An attacker can perturb the environment of an autonomous vehicle, for example, by defacing traffic signs, or its observations through an intermediary attack between the victim and its sensors. These methods are effective during training and inference as both poisoning and evasion attacks. As the attacker is limited in their access to the victim, securing the environment occupied by the agent presents a greater challenge for the defender than preventing access to the victim's instance. Similarly, sensors observing the environment may be isolated from the victim or communicate by unsecured means.

Note that these vectors are not mutually exclusive, e.g., observation perturbations can target specific actions as action perturbations [50]. Vectors are compatible with multiple goals, wherein observation perturbations can have state-based goals as in [51] or reward-based goals as in [50]. This means that both attacks change how the victim perceives the environment. The victim is either lured into making poor decisions, which reduces its reward, or the victim is lured into interacting with the environment such that it transitions into a state chosen by the attacker. Table 4 provides an overview of the different frameworks classified as described above.

RL Framework Applications			
Framework	Agents	Definition Tuple	Modeled Attack Vectors
MDP[52]	Single	(S,A,T,R)	None
POMDP [52]	Single	(S,A,T,Ω,O,R)	None
DEC-POMDP [53]	Multi	(I,S,$\{A_i\}$,T,$\{\Omega_i\}$,O,R)	None
SG [54]	Multi	(I,S,$\{A_i\}$,T,$\{R_i\}$)	Malicious Communications, Natural Adversarial Examples
POSG [55]	Multi	(I,S,$\{A_i\}$,T,$\{\Omega_i\}$,O,$\{R_i\}$)	Malicious Communications, Natural Adversarial Examples
PR-MDP [56]	Single	(S,A,T,R,ν,α)	Action Perturbations
NR-MDP [56]	Single	(S,A,T,R,ν,Δ)	Action Perturbations
SA-MDP [57]	Single	(S,A,T,R,ν)	Observation Perturbations

S	Set of states
A	Set of actions
T	The state transition function that stochastically maps a state s' ∈ S and action a ∈ A to a next state s' ∈ S such that T (s, a) = s'
R	Set of rewards
Ω	Set of observations
O	The observation probability function that maps a state s ∈ S and action a ∈ A to an observation o ∈ Ω such that $O(s, a) = o$
I	The set of agents with $i \subset I$ and $\{.i\}$ representing joint values
ν	Probability Robust (PR), Noisy Robust (NR), or state-permutation (for State Adversarial (SA)) function
Δ	Perturbation magnitude
α	Probability of an adversarial action replacing the victim's

Table 4: Overview of DRL frameworks [8]. Here are the abbreviations which are not explained in the table: Markov Decision Process (MDP), Partially Observable (PO), Decentralized (DEC), and Stochastic Game (SG).

3.1.1 Evasion Attacks in DRL

Training in a controlled setting allows the defender to prevent poisoning attacks, however victims must be exposed to their environment during inference. This presents opportunities for evasion attacks on the state or observation space. A properly constrained attack is imperceptible to human observers, and threatens virtually any DRL agent much like a software exploit, where the defender must detect intrusions and deny the attacker access. While most attacks require knowledge of the victim's model and testing environment, this can be bypassed with the transferability principle [14]. The principle stipulates that with sufficient knowledge to train a surrogate victim model, the attacker can generate adversarial examples that are then transferable and effective on the intended victim. In a supervised setting, this entails either training with the victim's training data or generating training data through queries.

Transfer attacks are also possible in DRL, without training a surrogate agent in the

victim's training environment, using the snooping attack [58]. Prior works assumed that similarities in the victim and surrogate model's decision boundaries enabled transfer attacks, however, similarities in the input feature extractors between models can enable adversarial attacks. An ANN can be considered to consist of two parts: the earlier layers of the network which encodes the inputs as intermediate features, and the head which chooses an output based on the intermediate features. The authors in [58] propose three threat models for the snooping attack where the attacker uses limited information on the victim's performance to train a proxy victim using supervised learning:

1. SA: The attacker can use the victim's state (S) (or observations) and action (A) to train an imitator model, which follows the victim's policy $\pi(a|s)$. Typically this produces the best performance of any threat model, approaching that of the surrogate trained in the victim's environment.

2. S: The attacker can only access the victim's state. This can be used to train a psychic which models the environmental dynamics, predicting the following state from the present. The psychic attempts to infer the victim's policy from the state transitions, but learning this is much noisier as aspects of the state can change independent of the victim's actions. As the scenario provides the attacker with the least information, it's unsurprising that it's the least effective in terms of adversarial regret.

3. SR: With access to the state and reward (R), the attacker can train an assessor which approximates the value function for the victim's policy $V_\pi(s)$.

4. SRA: Using all the information provided by the environment, a psychic, assessor and imitator can be trained, enabling the attacker to predict possible next states, assess their expected value, and produce stronger adversarial examples. This provides the requirements for the strategically timed attack which is explained below.

These proxies vary from the victim in terms of their loss, decision boundary, and output shape, yet still enable an attacker to craft adversarial examples, even without knowing the victim's goal. All the attacker must know is how the victim behaves in its environment.

However, only simple attacks effectively transfer from the proxy to the victim, as highly optimized adversarial samples (maximum-confidence or minimum norm) are closely fit to the loss topology of the ANN used to craft it. Difference between the victim and proxy makes iterative methods less effective than the single step Fast Gradient Method (FGM) [32]. Because of differences between the proxy and victim parameters, the better an adversarial example is optimized for one, the less likely its to be effective on the other. Finally, because noise is several orders of magnitude less effective than adversarial examples, crafting adversarial examples increases the threat of adversarial DRL [16].

3.1.2 Attack Optimizations

In a relatively simple and early example of an optimized evasion attack on an DRL victim, it was demonstrated that an attacker with access to a similar training environment can coerce the victim to follow an adversarial policy with targeted observation perturbations [15]. Their attack involved training an agent in the victim's environment with inverted rewards, such that its policy would minimize the victim's score. This is an adversarial policy. Each of the victim's observations were subject to targeted perturbations, which induce the victim into taking the actions selected by the adversarial policy. The targeted perturbations allow the attacker to select each of the victim's actions, which were chosen by an agent trained to minimize the victim's reward. As with supervised learning, these perturbations are most effective when crafted using the victim ANN, though it was shown

that a second victim agent trained in the same environment was approximately 70% as susceptible to the same perturbations [15].

As noted above, it's imperative that observation perturbations are made in real time for the victim, which makes optimizing their generation significant for executing an attack. In [16], reusing the perturbation for multiple frames or only perturbing frames with $V(s, \theta) > c$, where c is a constant representing a threshold of the value function, for a victim trained on the Pong game had similar performance to perturbing every frame while generating a tenth of the perturbations [16]. This reduces the computational overhead for the attacker, making their threat more realizable. On five Atari games, the Strategically Timed (ST) attack achieves parity with uniform perturbations (every observation) with a quarter of the perturbations [5]. States are perturbed based on the difference between the maximum and minimum values of Q or π, so states where all actions have similar values are unperturbed. This means that the agent doesn't have a strong preference for any action, so no action in this state had a large effect on the reward during training. Note that the victim action-value or policy function were known, and the attack was not tested on a surrogate model in a black box setting.

The CopyCat attack addresses the issue of producing observation perturbations in real time with pre-computed universal masks using [17]:

$$\arg\max_{\delta_a} \underset{o_t^k \in D}{E}[log\pi_{victim}(a|f(o_t^k + \delta_a, o_{1:t-1}^k) + \alpha||\delta_a||_2] \; s.t. ||\delta_a||_\infty < \epsilon$$

where,

1. the subset of observations $D = (o_t^k)_{t\in(1,T_k)}^{k\in(1:k)}$ is collected from K episodes used for generating universal perturbations.

2. δ_a is the universal perturbation corresponding to the action $a \in A$.

3. $\alpha||\delta_a||_2$ is a regularization parameter of the L_2 distance between the original and perturbed observations weighted by the parameter $\alpha \in R^+$.

Equation (3.1.2) is solved for all $a \in A$ for each $o_t^k \in D$ so there is a universal mask for each observation. The masks are used to perturb the victim's observations and as above, allow the attacker to coerce the victim into following an adversarial policy, in both black and white box settings.

The Decoupled Adversarial Policy (DAP) attack uses elements of both CopyCat and ST attacks [18]. DAP uses an adversarial policy selecting universal masks to perturb the victim's observations and a second agent trained to select when to attack. DAP limits the number of perturbations allowed in an episode and learns the optimal timing to change the victim's state observations. In contrast with the ST attack, DAP learns the optimal threshold for the target environment, rather than treating it as a hyperparameter.

The number of perturbed observations can be optimized with increasingly complex modeling and additional computations. As the techniques above were tested in simple Atari gym environments, additional research on their feasibility in CPS is required. The same is true for calculating universal masks, as the computations to produce them and memory to store them may be prohibitive for CPS applications with larger state and action spaces.

Evaluating the trade-off between the optimization and simplicity of the attacks can be understood in terms of adversarial regret and budget, as the results are not presented identically in every study [33]. Adversarial regret is the difference in reward between the victim's ideal performance and in the presence of an attack, for a given period T. The adversarial budget considers the number of features and observations perturbed, and the probability of perturbing an observation. Comparing the adversarial regret and budget for different attacks provides a meaningful comparison of test-time performance.

The framework proposed by [33] does not include training and implementation resources, or computation time delays. Certain attacks may be infeasible based on the time or resources required for training, or only possible for specific threat actors. A model could be considered more robust if effective attacks are difficult to train, and modeling the threat actor is influenced by the resources required to build the attack. Additionally the resources required to run the attack affect how it can be deployed. Consider an attack launched from an infected host, if the adversarial process consumes a significant portion of the available memory or computation, the attack becomes easier to detect. Finally, the delay between when an observation is taken and a perturbation is generated must conform to the victim's sampling rate.

The enchanting attack determines the action sequence required to transition from s_t to a goal state s_g and applies observation perturbations to lure the victim into performing those actions [5]. This method allows the attack to target a specific fail state for the victim rather than minimizing its rewards as above. The target state could be the power factor of an electrical grid or the temperature in an environmental system which would lead to critical failures. The enchanting attack requires a state prediction model M to determine the end state of an action sequence, so the sequence which corresponds with the closest end state to s_g can be selected. M models the environment's transition function, which is independent of the victim, meaning the attacker only needs access to the environment. The computational difficulty increases with the length of the action sequence, and the outcomes are sensitive to the accuracy of the prediction model M.

Additional information on adversarial MARL is available in annex A.

3.1.3 Summary

RL agents are similarly vulnerable to evasion attacks as supervised ANNs and well crafted adversarial examples are far more effective than random noise. Targeted perturbations of the victim's perception of the environment allow an attacker to enforce an adversarial policy without changing the majority of samples. An adversarial policy can cause a greater reduction in rewards than naively perturbing every observation and allows the attacker to control the victim's behaviour. By modeling the environmental dynamics, an attacker can also lure the victim to a target state. Whenever the environment is exposed to the attacker, there exists a threat from observation perturbations.

3.2 Robust RL

In terms of defending against adversarial examples, while adversarial training is popular and effective for supervised learning [31], the results of experiments using adversarial training in DRL are at best mixed [59, 57]. Supervised learning directly trains an ANN as a function approximator or model, whereas DRL learns a policy optimized for a reward function, implemented with a function approximator like an ANN. Supervised algorithms learn by minimizing a loss function based on the difference between their predictions and the training label, whereas DRL agents train on a surrogate loss function based on their reward. Unlike supervised learning, the function approximator is not trained directly.

Adversarial training works by maximizing loss as per Equation (13), and gradient-based search methods provide reasonable solutions for maximum loss. Finding an analogous loss for DRL is less effective, as taking a sub-optimal action based on an adversarial example is not guaranteed to minimize the agent's reward. Simply adding adversarial perturbations during training can prevent the agent from learning or converging, while in many cases this training method does not provide significant protection [59]. In most cases, even the adversarial agents had scores near or below zero, depending if scores could be negative. A further limitation was that adversarial observations were crafted with the comparatively weak FGSM, which both caused training to collapse and was still effective

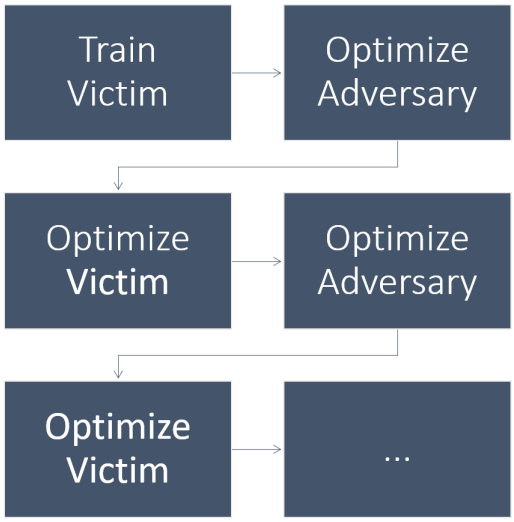


Figure 8: High level representation of ATLA training.

against adversarially trained agents. Stronger attacks could further degrade performance during training, and result in lower scores during testing. Further experiments in [57] showed adversarial training with 50% and 100% adversarial examples lead to natural rewards far lower than the naturally trained agent under any of the tested attacks.

Other approaches summarized in [57] showed similarly mixed results or required new DRL algorithms. Instead, [60] proposes the Alternating Training with Learned Adversary (ATLA) method for improving model robustness to a hostile environment. This involves training an DRL Learned Adversary (LA) which adds optimal perturbations to the agent's observations, in a reward-based attack. Figure 8 provides a basic depiction of ATLA training, and Algorithm 3 shows more detail. Taking turns training the adversary and agent ensures that the newly learned policy is not easily exploitable by some new policy, where neither adversary nor agent has any incentive to change their policy lest the other exploit it. ATLA involves training with multiple agents, which are not described by MDP or Partially Observable (PO)MDPs[52] used for single agents. Instead, frameworks like Decentralized (DEC)-POMDPs[53], Stochastic Games (SG)[54] and Partially Observable (PO)SGs[55] are typically used in multi-agent settings, though these do not model the adversary. Probability Robust (PR), Noisy Robust (NR), or State Adversarial (SA) MDPs can model the adversary in single agent settings. These frameworks are described in Table 4.

The framework for ATLA is the SA-MDP which adds the adversary's perturbation ν to the standard MDP [57]. Unlike a SG with two players whose actions affect the environment, the adversary ν is only able to change the perception of the state. Without any restrictions on ν, the adversary's task becomes trivial and the agent may fail to learn during training. Instead, a task specific boundary $B(s)$ is defined where $\nu(s) \in B(s)$. From the adversary's perspective, for a fixed policy π, the SA-MDP collapses to an MDP. Both $V(s)$ and $Q(s,a)$ must be redefined to account for both the state (s) and adversarial state ($\nu(s) = \tilde{s}$):

$$\tilde{V}_{\pi o \nu}(s) = \sum_{a \in A} \pi(a|\nu(s)) \sum_{s' \in S} p(s'|s,a)[R(s,a,s') + \gamma \tilde{V}_{\pi o \nu}(s')] \tag{24}$$

$$\tilde{Q}_{\pi o \nu}(s,a) = \sum_{s' \in S} p(s'|s,a)[R(s,a,s') + \gamma \sum_{a' \in A} \pi(a'|v'(s'))\tilde{Q}_{\pi o \nu}(s',a')] \tag{25}$$

Note that instead of inputting the state s into the policy π, it is replaced with the adversary's perturbation $\nu(s)$, making the optimal value function:

$$\tilde{V}^*_{\pi o \nu}(s) = \min_{\tilde{s} \in B(s)} \sum_{a \in A} \pi(a|\tilde{s}) \sum_{s' \in S} p(s'|s,a)[R(s,a,s') + \gamma \tilde{V}(s')] \tag{26}$$

The optimal adversary (ν^*) selects adversarial states which minimize the current and future rewards of the agent's policy (π), and such a ν is trained by setting the reward to $\hat{r} = -r$ where r is the agent's reward:

$$\nu^*(s) = \arg\min_{\tilde{s} \in B(s)} \sum_{a \in A} \pi(a|\tilde{s}) \sum_{s' \in S} p(s'|s,a)[R(s,a,s') + \gamma \tilde{V}(s')] \tag{27}$$

There is no corresponding definition of π^* with the presence of the adversary [57]. By (23) π^* must outperform every other policy in every other state, but the adversary's influence forces π to take an action which does not maximize $v(s)$ in some states. This is a trade-off between maximizing value and reducing the effects of adversarial perturbations, where the agent becomes less exploitable at the cost of decreased values. This approach lowers the ceiling of clean performance, but raises the floor of performance under attack. This is much like the trade-off between an all-terrain vehicle and a racecar; the racecar will be much faster under good conditions, but almost useless on a dirt road.

The definitions above hold only for a *fixed* policy π, so the adversary ν can be trained through an MDP. Alternating training between fixed policies allows the problem to be treated as a single agent POMDP, rather than a multi agent SA-MDP [60]. During training, either the LA or agent's policy is static, and effectively part of the training environment, making the optimization problem stationary for each policy during training. For the adversary ν as shown above, training is an MDP, however, it's a POMDP for the agent. The fixed $\nu(s)$ in the SA-MDP is equivalent to the observation probability function $O(s,a)$, and the set of observations Ω are akin to a combination of the set of states S and perturbation space surrounding those states $B(s)$:

$$\Omega = S \cup B(s), s \in S \tag{28}$$

Thus our SA-MDP (S,A,T,R,ν) becomes the POMDP (S,A,T,$S \cup B(S)$,$\nu(s)$,R) for a static adversary ν [60]. A POMDP can still be solved with conventional DRL algorithms, albeit with more difficulty. As function approximators (i.e., ANNs) are used to parameterize the optimal mapping of states to values and actions, they can similarly learn the mapping of observations to states [13]. However, there are no formal proofs to guarantee this property. Incorporating a fixed-length history of observations and actions can improve performance, such as using a Long Short-Term Memory (LSTM) or similar Recurrent Neural Network (RNN) in place of the Multi-Layer Perceptron (MLP) more common in control tasks [60].

To demonstrate the method's effect, ATLA was benchmarked against several gradient-based attacks which are unique to RL[57]:

1. The Critic attack: this attack uses the gradient of the agent's action-value function to generate perturbations of a fixed size, similar to the FGSM in supervised learning:

   $$\tilde{s} = s - \eta \nabla Q(s, \pi(s)) \tag{29}$$

   The method can be iterated for an arbitrary number of steps. However it requires that the victim algorithm uses an action value function, which excludes policy-gradient and actor-critic algorithms.

2. The Maximal Action Difference (MAD) attack: This is a similar attack to the Critic attack, except the loss gradient is generated from the Kullback-Leibler Divergence

Algorithm 3 Alternating Training with Learned Adversaries (ATLA) [60].

Inputs Transition function T(s,a) for the environment, number of alternations N_{alt}, number of pre-training iterations N_{pre}, agent training iterations N_π , adversary training iterations N_ν
Initialize θ_0 and ϕ_0 to parameterize π and ν
if $N_{pre} > 0$ **then** //optional agent pre-training
  for $i = 0$ to N_{pre} **do**
    $\theta_{i+1} \leftarrow train(T(\pi(s, \theta_i), s))$
  end for
  $\theta_0 \leftarrow \theta_{N_{pre}}$
end if
for $i = 0$ in N_{alt} **do**
  for $j = 0$ to N_ν **do**
    $\phi_{j+1} \leftarrow train(T((\pi(\nu(s, \phi_j), \theta_0)), s)$ //note that θ is constant
  end for
  $\phi_0 \leftarrow \phi_{N_\nu}$
  for $j = 0$ to N_π **do**
    $\theta_{j+1} \leftarrow train(T((\pi(\nu(s, \phi_0), \theta_j)), s)$
  end for
  $\theta_0 \leftarrow \theta_{N_\pi}$
end for

(D_{KL}), which measures the distance between probability distributions, between the policy's output for the original and adversarial state. Because it does not rely on the action-value function, it often outperforms the critic attack.

3. The Robust SARSA (RS) attack: Like a combination of the critic and snooping attacks, RS uses the agent's trajectory to train a State-Action-Reward-State-Action (SARSA) algorithm using the Temporal Difference (TD) error to generate an action-value function which can be used in a critic attack. The SARSA model can be augmented with a robust object to further increase the adversarial regret caused by the attack.

4. Hybrid RS-MAD attack: Combines the RS and MAD loss functions to conduct an MAD attack, where:
$$L = \alpha L_{MAD} + (1 - \alpha) L_{RS} \tag{30}$$
α is chosen to balance the losses in cases where they are normalized differently. It typically outperforms either attack individually.

5. Learned Adversary (LA) or Optimal attack: $\nu^*(s)$ is approximated using a model-free RL, in this case a PPO.

Of the five attacks, the Optimal/LA attack had the largest adversarial regret in four MuJoCo [51] continuous control environments. ATLA was not tested against adversarial observations generated from the function approximator, such as those discussed in the adversarial DRL section.

The greatest advantage of ATLA is that it does not require a unique or modified DRL algorithm to ensure robustness, because it's purely a training method. This makes it particularly interesting to study, as the cost of adding it to an existing agent or training system is relatively small compared to using a different robust algorithm. Essentially, ATLA can be applied to any off-the-shelf algorithm from existing libraries. This is important because while robust algorithms like the State Adversarial (SA)-PPO, State Adversarial Deep Deterministic Policy Gradient (SA-DDPG), and State Adversarial Deep Q

Network (SA-DQN) [57], and the Robust Student Deep Q Network (RS-DQN) exist [61], their adoption is not widespread and they are not implemented in popular libraries. This approach aims to train a robust policy (π) rather than a robust function approximator or ANN.

4 Threat Model

Vulnerabilities in Artificial Neural Networks (ANN)s are akin to unpatched exploits in software. This research is a proof-of-concept for exploiting these vulnerabilities in Deep-Reinforcement Learning (DRL) specific techniques to conduct Load Altering Attack (LAA)s. The primary threat model for the attack is a DRL agent that is a white-box, meaning that the attacker can read its parameters, and is able to modify sensor readings sent to the agent. The white-box model demonstrates the strongest attacks available, which establishes the risk ceiling. Knowing the ceiling informs smart energy designers the level of risk they are accepting by using vulnerable models. Furthermore, analyzing such strong attacks can suggest mitigation techniques and quantifying risk is an important step towards mitigation. A well resourced attacker could obtain the victim agent parameters without the privileges required to compromise the entire DR system. Possible methods would be compromising a back-up server, compromising an employee to exfiltrate the model, or obtaining a low level of access on the host or server running the controller DRL which is able to read the model's parameters. Enabling a black-box snooping attack [58] is even simpler, the attacker only needs historical data to train a proxy model. There are precedents for poor security in CI, and security DR infrastructure would be even more fraught when it is maintained by the user, who may lack the resources of a utility to secure it.

CPPS rely on distributed sensors to transmit measurements to a controller. Injecting observation perturbations can be accomplished using multiple vectors depending on the exact application. Both the Sunspec Modbus and IEEE 1815 (DNP3) protocols are vulnerable to intermediary attacks, which allow an attacker with a LAN presence to capture and modify data in transit [6]. Many consumer smart energy devices have known software vulnerabilities [7], some of which have already been exploited to form bot nets [1]. The diverse manufacturing stages for smart devices also make them vulnerable to supply chain attacks. Attacks might physically access sensors or the means of transmission in insecure public or isolated locations. These vulnerabilities can give attackers a foothold to modify traffic, or compromise sensor firmware to add perturbations. Given the vulnerabilities and precedents discussed above, an attacker could gain the access to conduct a LAA using adversarial observations. This level of access is less than what would be needed to compromise the DRL controller itself. Furthermore, adversarial observations have the potential to conceal the cause of the attack. For this research, the attacker is assumed to have write access to the observations of a DRL agent, and in some cases access to the agent ANN's parameters.

5 Demand Response (DR) Environment

A trained Deep-Reinforcement Learning (DRL) victim is required in order to eventually measure the adversarial regret that is going to be caused by observation perturbations. This means that each observation returned by the environment will be modified using a gradient-based attack before the victim receives it. This simulates an intermediary attack between the victim and the smart building sensors. The observation perturbations will cause the victim to deviate from its learned policy. In Figure 10, the trained DRL agent should have selected the optimal action a^*, but under attack, will select another action. The proportion of deviations will be measured as the ASR, and the reduction in score is the adversarial regret. The distortions added to the original observations by the adversarial attack can be measured from the observation data generated from such attacks. The size of the distortions and adversarial regrets illuminate the threat of such attacks.

There are three tools required for this experiment: a DR gym environment, a compatible DRL library, and an adversarial attack toolkit. The rational for the toolset selected is discussed below.

5.1 DR Gym Environment - DRL Library - Attack Toolkit

CityLearn has been selected as it is a well documented and actively maintained smart energy DR gym, following the OpenAI Gym standards [4]. Several DRL algorithm libraries compatible with OpenAI Gym are available, including Stable-Baselines3 (SB3) [62], Tianshou, and Ray-RLLib. CityLearn's own examples use SB3. Its environments feature districts with customizable quantities and types of buildings who's energy demands are set using historical data. Figure 9 visualizes CityLearn.

The DRL agents are tasked with charging and discharging energy storage units, given rules-based restrictions such as preventing discharging when there is no demand. Agents' performance is normalized against this simple rules-based controller. Each agent's action space is the proportion of each storage device the agent controls that can be charged or discharged. Multiple buildings are supplied by the same energy storage, and the number of such units determines if the environment is single-agent or multi-agent. User-defined reward functions allow agents to train towards a variety of goals. The observation space consists of up to 42 features characterized as date, weather, building, or district observations, though the exact number varies with the number of buildings. One timestep is one hour within the simulation.

A smart energy gym adds value as both the ASR and detection of adversarial examples varies with the application, thus threats must be assessed in a variety of applications where DRL is applicable. The greatest assets of CityLearn are its use of real-world data, compatibility with RL libraries, and low computation requirements, making it a perfect choice for preliminary research. It however still has some limitations, mainly that is does not dynamically calculate energy pricing nor does it simulate the wider power grid. These limitations are imposed by its reliance on recorded data rather than dynamic calculations, which reduces its computational cost. Despite its inability to simulate the effects of an Load Altering Attack (LAA) on a large scale power grid, the realism in the data enables realistic analysis of the significance of the distortions introduced by adversarial observations. On the balance it is the best environment for this research.

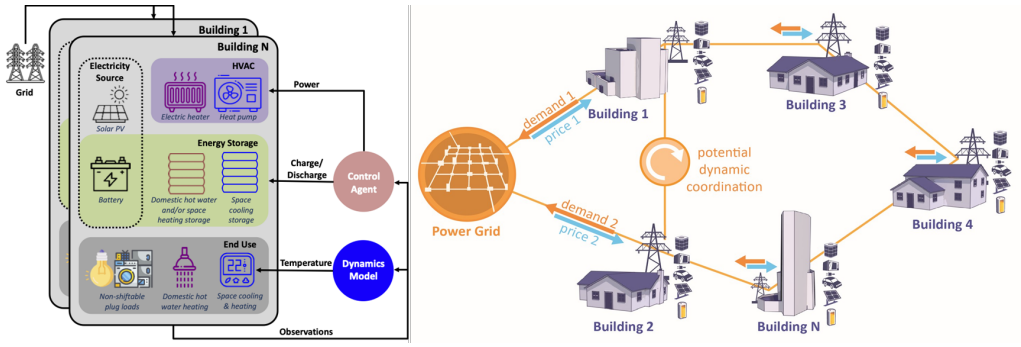


Figure 9: Visual depiction of CityLearn [4]. The left side shows the loads and storage devices which can be controlled by the agent, and the features dynamically calculated by the environment. The right shows a network of buildings, with their respective loads and storage devices, that could be controlled by a single or multi agent algorithms.

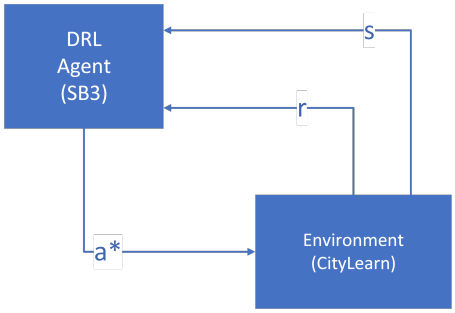


Figure 10: DRL agent acting in CityLearn, where a^* is the learned optimal action and s and r are the state and reward returned by the environment.

Because there is no publicly available library for adversarial attacks on DRL agents, a supervised learning library was used as an interface. This means that the policy network of the victim algorithm must be treated as a supervised learning network, which maps states to actions, to generate adversarial examples. This requirement constrains the selection of DRL libraries as the agents' neural networks must be accessible through its API. The action and value networks of SB3 agents are accessible in this fashion, so they can be copied and adapted for generating adversarial examples.

The Adversarial Robustness Toolbox (ART) is a library of adversarial techniques for robustness testing, compatible with Tensorflow and Torch, actively maintained, and implements the largest attack variety of any available option [36]. The Auto-Conjugate Gradient (ACG) attack is the most powerful maximum-confidence white box attack available in any open source library; it is remarkable for its ASR on image datasets and small number of parameters [41].

The use of any recent adversarial attack imposes a significant restriction on the type of agent which can be attacked, and by extension its environment. Because robustness testing began as attacks on image classifiers, the open source attacks reviewed expect a single label which corresponds to a one dimension action space for a DRL agent. The loss functions used, Cross-Entropy (CE) or Difference Logits Ratio (DLR), suppose that there is only one correct label and corresponding logit, whereas a DRL agent with a multidimensional action space would have groups of outputs corresponding to each action or action space dimension. The adaptations for classification problems also restricts the use of these attacks on regression problems, which are analogous to DRL agents with

continuous action spaces. While a single SB3 DRL agent could control multiple buildings with multiple energy storage devices, the use of ACG requires a one-dimensional action space, thus the victim can only control a single building with a single storage device. Multiple victims of this type are akin to component agents in Multi-Agent Reinforcement Learning (MARL) environment, where each agent is assigned a different building. To meet these constraints this work uses building 6 of CityLearn's (version 2.0.0) 2022 challenge phase 2 environment. Compatibility with this gym environment requires SB3 version 1.8.0, which is stable with the deep learning library torch version 1.12.1. Finally, the latest version of ART when this project began, 1.16.0, was used.

5.2 Victim Agent Training

Here the goal is to train an agent with performance significantly better than CityLearn's baseline, and who's architecture is compatible with ART. The performance must be sufficient to make the adversarial regret from an attack evident, but this does not require State of the ART (SotA) performance. RL in DR is a significant sub-field for research, so perfecting the victim is out of scope. This research will use a Proximal Policy Optimization (PPO) algorithm [47], which is among the most advanced algorithms for discrete action spaces in the SB3 library.

In the 2022 challenge environment, one or more agents learn to charge and discharge building electrical storage to reduce and smooth demand from the electrical grid. It includes up to 5 residential buildings each with their own battery. The observations are generated using real-world data from a California study on residential photo-voltaic integration [63, 64]. Because the victim's observations are generated from recorded data, the attack's observation perturbations will be generated is a realistic setting. This realism enables testing of the hypothesis that adversarial examples would be a difficult payload to detect during a real attack. Due to the restrictions of the ACG attack , the victim agent controls the battery's State of Charge (SoC) of a single building so that the action space remains one-dimensional. This frames the RL task as a classification problem, with Figure 11 representing the agent's policy network.

With the CityLearn environment selected, there are three major factors for the agent's performance: action space, reward function, and RL algorithm-specific hyperparameters. These factors are evaluated using CityLearn's eight cost functions called Key Performance Indicators (KPIs) [63]:

1. The district-level KPIs use the aggregated district-level hourly net electricity consumption (kWh), E_h^{district} to calculate:
   (a) average monthly/annual peak,
   (b) ramping,
   (c) daily/monthly average (1 - load factor).
2. The building-level KPIs that are calculated use the building-level hourly net electricity consumption (kWh), E_h^{building}, and are reported at the grid level as the average of the building-level values:
   (a) electricity consumption,
   (b) cost,
   (c) carbon emissions.
3. Electricity consumption is the sum of electricity consumed from the grid E_h^{building}:

$$\text{electricity consumption} = \sum_{h=0}^{n-1} \max\left(0, E_h^{\text{building}}\right)$$

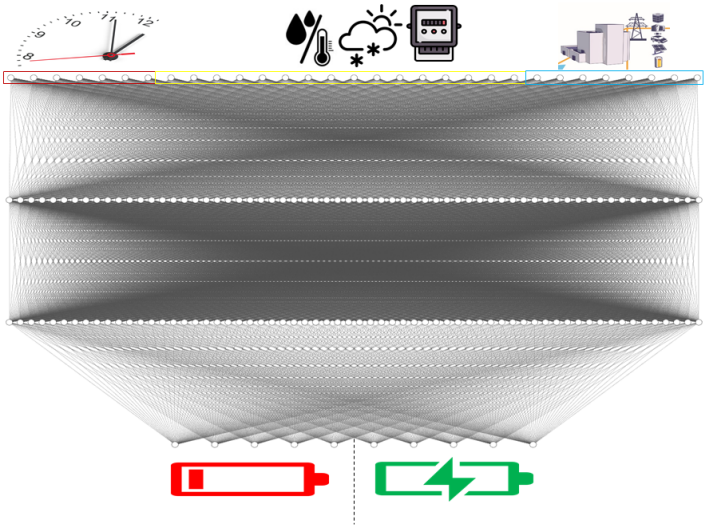


Figure 11: Example of the Multi-Layer Perceptron (MLP) actor network for a CityLearn agent. It has 31 inputs, 2 hidden 256 layers, and 10 outputs each corresponding to an action. The images above the network each correspond to a category of features. There are 6 sinusoidally encoded temporal features, 17 weather features, and 8 building specific features. The 10 outputs correspond to the desired (dis)charge level of the building's electrical storage. The logits on the left indicate the minimum level to which the battery can be discharged (0-100%), while those on the right indicate the max level of charge. At any timestep the agent will choose a single charge or discharge action.

4. Cost is the sum of building-level electricity cost, $E_h^{\text{building}} \times T_h$ ($), where T_h is the electricity rate at hour h:

$$\text{cost} = \sum_{h=0}^{n-1} \max\left(0, E_h^{\text{building}} \times T_h\right)$$

5. Carbon emissions are the sum of building-level carbon emissions (kg_{CO_2e}), $E_h^{\text{building}} \times O_h$, where O_h is the carbon intensity (kg_{CO_2e}/kWh) at hour h:

$$\text{carbon emissions} = \sum_{h=0}^{n-1} \max\left(0, E_h^{\text{building}} \times O_h\right)$$

6. Average daily peak is the mean of the daily E_h^{district} peak where d is the day index and n is the total number of days:

$$\text{average daily peak} = \frac{\sum_{d=0}^{n-1}\sum_{h=0}^{23}\max\left(E_{24d+h}^{\text{district}}, \ldots, E_{24d+23}^{\text{district}}\right)}{n}$$

7. Ramping is defined as the absolute difference of consecutive E_h^{district} measurements, which is the district's load profile's rate of change. High ramping indicates abrupt changes in grid load that increase strain on the power grid, and blackouts may result from supply deficit:

$$\text{ramping} = \sum_{h=0}^{n-1} |E_h^{\text{district}} - E_{h-1}^{\text{district}}|$$

8. Load factor is defined as the average ratio average and peak E_h^{district} for a period of a day or month, where m is the month index, d is the number of days in a month and n is the number of months. Load factor represents the efficiency of electricity consumption with a value between 0 and 1. 1-load factor is a cost to be minimized, while the load factor alone should be maximized:

$$\text{1 - load factor} = \Big(\sum_{m=0}^{n-1} 1 - \frac{\left(\sum_{h=0}^{d-1} E_{d \cdot m+h}^{\text{district}} \right) \div d}{\max \left(E_{d \cdot m}^{\text{district}}, \ldots, E_{d \cdot m+d-1}^{\text{district}} \right)} \Big) \div n$$

CityLearn reports agents' KPIs normalized to the outcome of buildings not equipped with batteries or a controller.

$$\text{KPI} = \frac{\text{KPI}_{controlled}}{\text{KPI}_{uncontrolled}}$$

The most important KPI for assessing impact of an attack on grid stability is electricity consumption, as blackouts result when power consumption exceeds grid capacity. This would require a large-scale attack, either on many victims or a high consumption victim like an industrial park. Peak consumption is the second most important KPI, as a significant peak during peak grid demand could also cause a blackout. Ramping is an important KPI as the grid must instantaneously match demand, so large fluctuations therein are challenging for the grid operator. However, if the ramping is not accompanied by significant energy consumption or high peaks it is unlikely to affect grid stability. Cost is also considered a useful KPI in this work, as a consideration for cost-based LAAs. If the electricity price was dynamically calculated based on grid demand this would be a valuable signal for the attacker, as it would be a surrogate for the total load on the grid. However, CityLearn uses historical prices and the power grid has infinite capacity.

5.2.1 Action Space

Because ACG has not been implemented for regression tasks, the victim must choose from a discrete number of actions which resemble a classification problem. CityLearn's action space can be continuous or divided into a series of bins, which is a parameter that can be tuned. Generally, more actions give the agent finer control, but also increase the action space it must explore. Thus, too many actions increases the challenge in training, while too few results in imprecision actions. Given the necessary training times, an exhaustive search of action spaces is superfluous.

5.2.2 Reward Function

With the action space defined, the reward function is the next victim parameter explored. To frame CityLearn as an optimization problem, reward functions are crafted to penalize energy use, cost, carbon emissions etc. These measurements are negated to penalize the agent for larger amounts, so it learns to reduce those values. Custom rewards in CityLearn v2 must be calculated from the current observation, so rewards cannot use information from past states, nor include ramping or peak values as found in the KPIs. Five rewards were considered from defaults in CityLearn and those found in the literature, which penalize energy usage.

The default reward in CityLearn is the energy consumption penalty:

$$\min(-E_h^{\text{building}}, 0) \tag{31}$$

The solar penalty is built into CityLearn and is designed to reward the agent for changing its battery with solar energy:

$$\sum_{i=0}^{n} -((1 + \frac{E_h^{\text{building}}}{|E_h^{\text{building}}|}\text{storage}_i^{\text{SoC}}) \times |E_h^{\text{building}}|) \tag{32}$$

Cost penalty:

$$C = \min(0, -E_h^{\text{building}} \times T_h) \tag{33}$$

Cost-SoC penalty is designed to minimize electricity cost, C [63]. It encourages net-zero energy use by penalizing grid energy consumption when there is energy in the battery, and net export when the battery is not fully charged. The agent is penalized in proportion to the energy stored when power is consumed from the grid, encouraging charging when grid demand is low and discharging when it is high :

$$-\left(1 + \text{sign}(C) \times \text{storage}_i^{\text{SoC}}\right) \times |C| \tag{34}$$

The energy-SoC penalty is adapted from the cost-SoC penalty by removing the price signal:

$$-\left(1 + \text{sign}(E_h^{\text{building}}) \times \text{storage}_i^{\text{SoC}}\right) \times |E_h^{\text{building}}| \tag{35}$$

5.2.3 Hyperparameters

With the reward function selected the victim agent can be tuned, as the optimal hyperparameters are correlated with environmental factors: action space and reward function. Hyperparameter tuning for the victim was conducted using Optuna version 3.5.0, a Python library which uses heuristic searches of the user defined hyperparameter search space to select the best values based on the performance of previous trials [65]. Other automated popular hyperparameter tuning methods are grid and random searches. A grid search exhaustively tests every combination of the candidate hyperparameters, which requires more time and computation than other methods. A random search randomly samples the candidate hyperparameters for a set number of trials.

Optuna's heuristic search, like the random search, does not require sampling the entire candidate space. However, the heuristic search will only trial the hyperparameter combinations which demonstrate the best performance, improving the probability of finding an optimal combination. Optuna will also stop trials early, when evaluations during a trial perform significantly worse than the best trial. Hyperparameter tuning used 50 episode Optuna trials With the Tree-structured Parzen Estimator (TPE) sampling algorithm, which is recommended when parallelization resources are limited.

5.3 Results

5.3.1 Action Space Tuning

A coarse search of possible action spaces was conducted. Figure 12 shows that performance is significantly reduced below the default of 10 bins while performance peaks near 20 bins. Agents were trained for up to 300 episodes with early stopping enabled, meaning that training would end when there was no improvement for four evaluations. The other hyperparameters were identical for all agents, and they were trained using the default reward function. Both the 20 and 24 bin models performed similarly, indicating that increasing the number of bins further would not meaningfully improve the results. The 20-bin model was chosen based on its slightly better performance with the daily peak average, and the difference in power consumption was considered negligible. Given that the goal of this research is to demonstrate the performance reduction achieved with an

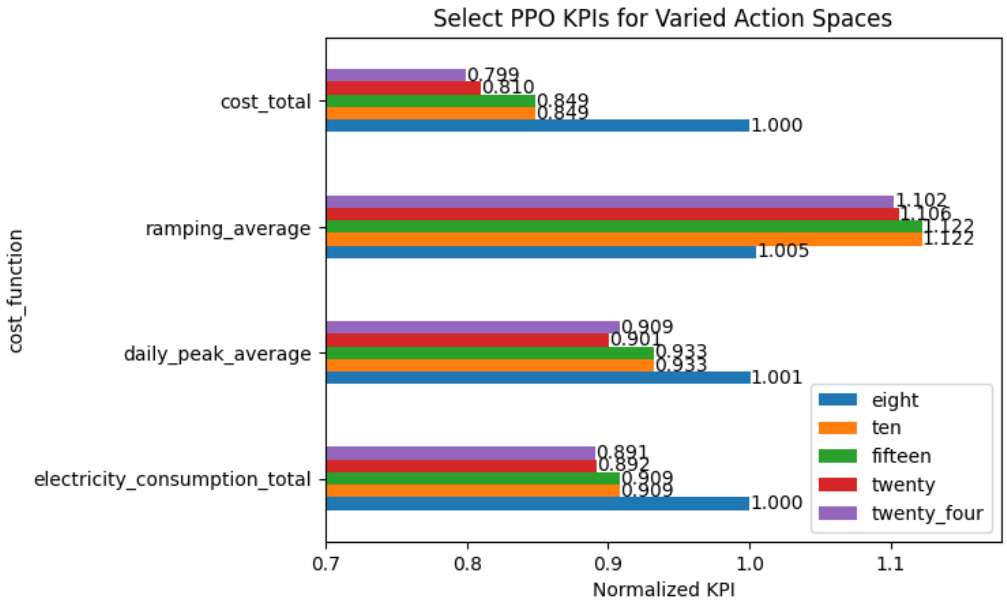


Figure 12: KPIs compared between agents with varied discrete action spaces. This data was used to select the optimal number of bins for discretizing CityLearn's continuous action space.

adversarial attack, merely a competent rather than an optimal victim is required. Figure 13 shows how the agent's action changes the SoC. The agent's action corresponds to the change in SoC at the following time step. Thus an action of 0.25 would charge 25% of the storage capacity for the following hour, while -0.10 would discharge 10% of the total capacity. Charging at full capacity or discharging when empty has no effect.

5.3.2 Reward Tuning

The five rewards introduced in the previous section were used to train PPO victims, with 20 discrete actions for 300 episodes. The results are found in Figure 14. Several KPIs were identical for all the agents when rounded to 3 digits, so the differences were considered insignificant. As discussed above, the four most important KPIs are energy consumption, average peak, ramping for grid stability, and cost for cost-based attack. Where significant differences existed between the agents, the agent trained with the energy consumption penalty performed best in terms of electricity consumption and average daily peaks, though it exhibited the worst ramping by a small margin. In contrast, the victim trained with the solar penalty was worst in the first two metrics but best for ramping, and the other two were in the middle for all three. Because the energy consumption performed best in terms of the two most important metrics, it was selected as the victim for the rest of this research. It was also the second best for cost, which is important for evaluating the effects of potential cost-based LAAs, and makes the system more economically viable.

5.3.3 Hyperparameter Tuning

The Optuna search found that a 2-by-256 dense network outperformed narrower networks and that additional depth did not improve performance. Also, the Tanh activation function outperformed Relu. The energy consumption during the final evaluation episode, in the same CityLearn environment used for training with a different random seed, was the metric used for comparison. Changing other hyperparameters from the default values

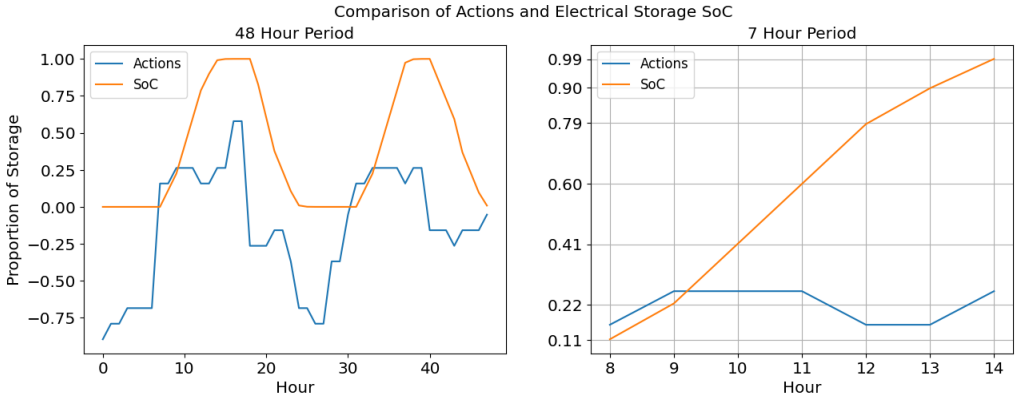


Figure 13: Line plots comparing an agent's actions with it is electrical storage SoC observation at each timestep. The agent had 20 discrete actions. The left plot shows 2 day period, while the right plot zooms in to illustrate the relationship between actions and the SoC. The latter shows that the agent's action corresponds to the slope of the SoC at the following time step (hour). These values are not perfectly aligned, this is likely an artifact of translating discrete actions numbered 1-19 to [-1, 1].

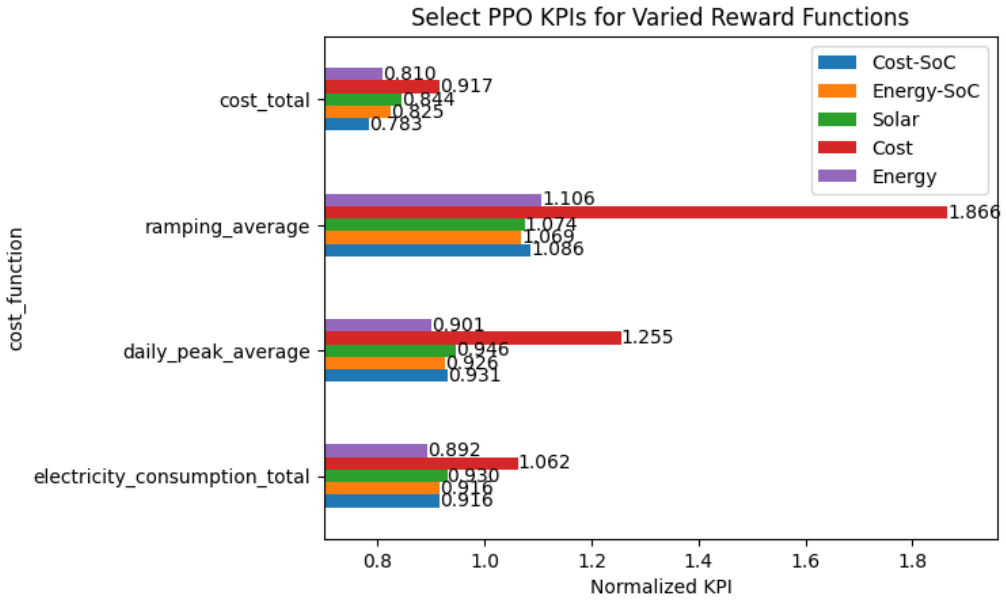


Figure 14: These select cost functions are the most relevant to LAA and are used to select the victim. Cost is relevant for cost-based attacks, and the other three are the most relevant to grid stability. The annual peak average was omitted due to the limited variation between agents.

did not improve performance on agents trained for 300 or more episodes. While agents converged after training for 300 to 500 episodes, this would have been time prohibitive when varying 13 different hyperparameters, so 50 episodes were used.

5.4 Summary

In this section the toolset for this research was selected, and the decisive factors were realism and compatibility. CityLearn's use of recorded weather and electrical consumption data enables the study of detection later in this work. Its compatibility with OpenAI gym allows the use of SB3 agents, which provides two major advantages: the variety of algorithms implemented and the API access to agent parameters. The latter minimizes the amount of glue logic required to interface SB3 with ART, which uses the model parameters for gradient-based attacks. The second major accomplishment of this section was training a victim agent, which enables all future experiments. Because ART is designed to attack classifiers, this victim needed a discrete and one-dimensional action space. So, the victim is only given control of a single building, and different action spaces were tested, along with reward functions, neural network architectures, and PPO specific hyperparameters. The trained victim reduced power use by over 10% compared to a scenario with no controller, which is large enough to show if an attack has a significant effect on the victim agent's performance.

6 White-Box Attacks

This phase demonstrates the effects of a State of the ART (SotA) observation-perturbation attack on a single DRL agent in a demand response gym using real world data. This environment simulates a Cyber Physical Power System (CPPS). In this white box setting, the attacker has access to the victim's training environment and model parameters, enabling the most powerful attacks. The goal of this phase is to maximize the attacker's impact, and learn the greatest threat posed by adversarial attacks.

6.1 Untargeted Attacks

6.1.1 Methodology

We start by conducting SotA untargeted gradient-based attacks on the agents trained in the previous section. The attack's success is measured in terms of its Adversarial Success Rate (ASR), and adversarial regret for the electricity consumption, ramping, daily peak, and cost Key Performance Indicator (KPI)s. The attacks employed are the maximum-confidence Auto-Conjugate Gradient (ACG) and minimum-norm Brendle and Bethge (BB). All attacks are conducted using L_∞ regularization, where both attack demonstrate the best performance. For a maximum-confidence attack, L_∞ regularization clips the perturbation for each feature in the range $[-\epsilon, \epsilon]$ and is the least restrictive regularization method.

ACG will be the first attack tested. The boundary ϵ for a maximum-confidence attack may exceed the value required for a successful adversarial example, meaning the same ASR could be achieved with a smaller boundary, but this can vary by sample. As proposed in [50], a maximum-confidence attack can be restarted with different ϵ to find the minimal distortion boundary which results in a successful attack. While [50] tested a list of ϵ starting with the smallest, this work attempts to improve the algorithm using a binary search rather than a simple for loop to minimize the number of restarts. First the middle ϵ is tested, and depending on its success either a larger or smaller value is subsequently tested as shown in Algorithm 4. The approach improves computation time for samples which require larger ϵ and is only less efficient where the lowest ϵ are most effective. The dynamic distortion algorithm is implemented for the ACG attack to minimize the size of its perturbations.

The inputs for this algorithm are a list of initialized Adversarial Robustness Toolbox (ART) attack objects in ascending order of ϵ (atk_list), the victim's policy network (π), and the Keyword Arguments (kwargs) for generating the adversarial sample. The kwargs include the input sample to attack (x), a mask specifying which feature not to change, and a target if applicable.

Different computational constraints will be tested for a default number of total iterations of 500. This involves changing the number of random restarts while scaling the number of iterations per restart accordingly. The literature review covers several value-based timing attacks, with the simplest being the threshold based Strategically-Timed (ST) attack [16, 5]. A threshold c is set based on the distribution of $V(s)$ during a clean episode, with the percentile of the threshold corresponding to the desired perturbation frequency. The ST attack is combined with an adversarial attack (ACG in this case), the threshold determines which obersvations will be attacked, then the attack generates the adversarial observation. To validate the efficacy of the ST attack, the adversarial regret will be compared to an attack with random timing at a similar frequency. If the ST attack proves successful, an agent could be trained to learn the optimal time to attack as in [18].

The results of the maximum-confidence attack ACG will be compared with the minimum-norm BB attack. As a minimum-norm attack, it does not use a perturba-

Algorithm 4 Dynamic Distortion Algorithm.

$Inputs : atk\_list, \pi, kwargs$
$idx \leftarrow \lfloor \frac{\text{length}(atk\_list)}{2} \rfloor$
$best\_candidate\_idx \leftarrow$ list of atk_list indices
$atk\_idx \leftarrow 0$
$eps\_idx \leftarrow 1$
$min\_eps \leftarrow$ None
$min\_eps\_sample \leftarrow sample$
$a \leftarrow a\_min\_eps \leftarrow \pi(sample)$
while $atk\_list \neq empty$ **do**
    $adv\_sample \leftarrow atk\_candidates[idx][atk\_idx].generate(**kwargs)$
    $a\_adv \leftarrow \pi(adv\_sample)$
    if $a \neq a\_adv$ **then** ▷ adversarial example successful
        $min\_eps \leftarrow \epsilon$ for the successful attack
        $atk\_list \leftarrow$ list of attacks with smaller ϵ
        $min\_eps\_sample \leftarrow adv\_sample$
        $a\_min\_eps \leftarrow a\_adv$
    else ▷ no adversarial sample found
        $atk\_candidates \leftarrow$ list of attacks with larger ϵ
        $best\_candidate\_idx \leftarrow$ next highest index
    end if
    $idx \leftarrow \lfloor \frac{\text{length}(atk\_list)}{2} \rfloor$
end while
return $min\_eps\_sample, a\_min\_eps, min\_eps$

tion boundary ϵ and instead finds the minimum L_∞ distance for a successful attack. This distance must be measured directly, so it, along with the ASR and adversarial regret, can be compared to the results with ACG.

6.1.2 Untargeted ACG Attack

The goal in this section was to generate a significant reduction in the victim's performance using ACG with the smallest possible distortion ϵ. It demonstrates that ACG can degrade a victim's performance with minuscule perturbations.

Attack Parameters Different combinations of random restarts and iterations per restart were tested, while fixing the product of both (the total number of iterations per attack) to the default value of 500. Meaning that as one increased the other decreased and the maximum number of iterations does not increase, so the computational budget was constant. Between tests the ASR remained consistent, however the individual samples where the attack failed changed. This implies that some of the successes and failures can be attributed to random factors between runs. Given that the CityLearn environment is deterministic and most features are generated from recorded data, ACG's random restarts is the most significant source of randomness. This reasoning was supported by the fact that increasing the number of restarts while number of search steps per restart decreased, increased the ASR.

The optimal number of random restarts was correlated with the initial step size, but ranged from approximately 30 to 40 restarts, with 16 to 12 iterations each. With ART's default initial step-size of $\frac{\epsilon}{2}$ the ASR increased from 97% to 98.7% by increasing the number of random restarts from 5 to 40. An initial step-size of 2ϵ increases the ASR further to 99.3%. Both random restarts and larger initial steps diversify the search,

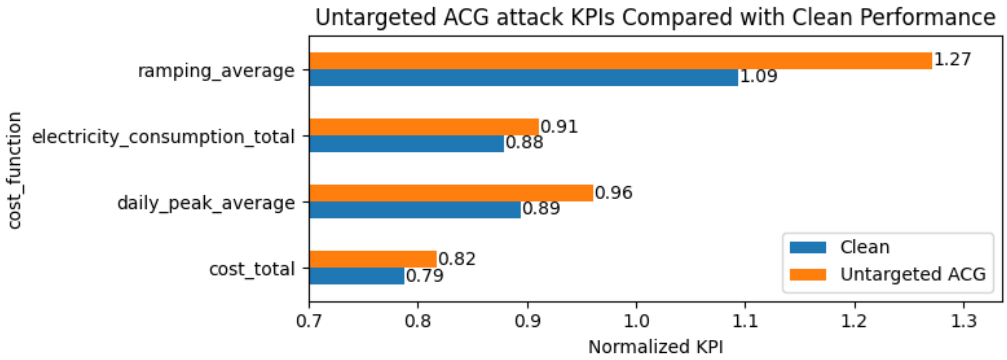


Figure 15: The average KPIs over 5 runs with the untargeted ACG attack are compared to those with no attack. The x-axis represents normalized performance without a smart controller, and larger values indicate stronger attacks. Values less than one indicate the improvement provided by the DRL agent for the particular cost function.

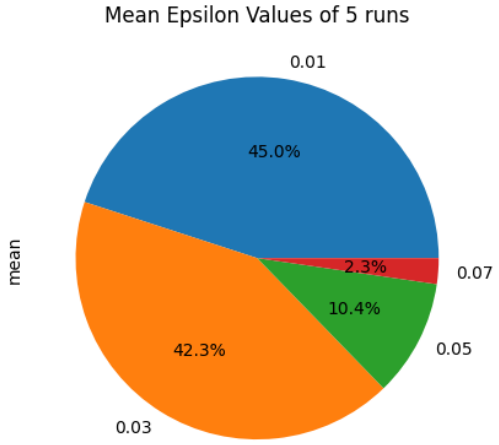


Figure 16: Mean ϵ values of 5 runs, with the the untargeted ACG attack using the dynamic distortion.

ensuring the attack explores a larger space, while decreasing the number of iterations per restart reduces the amount of exploitation or intensification. However, no benefit was found for increasing the number of restarts past 50. Due to the computation time required, many restart values were tested for 100 timesteps; the best were tested with 1000, and finally 1-2 were tested on full episodes.

ACG with Dynamic Distortion Because the ACG is a maximum-confidence attack it will always maximize loss within the given budget ϵ. However, the minimum budget for a successful adversarial sample varies with the input. While $\epsilon = 0.07$ achieved a 99% ASR it is also unnecessarily large in 97.7% of cases as seen in Figure 16. By conducting the ACG attack multiple times with different ϵ, the successful adversarial observation with the smallest perturbation can be selected. A multi-threaded implementation was tested, with each thread assigned an attack with a different value of ϵ. However, this did not reduce the computation time, as the results from each thread were not returned until all had finished. Attacks with smaller ϵ have lower ASRs, and will expend their entire

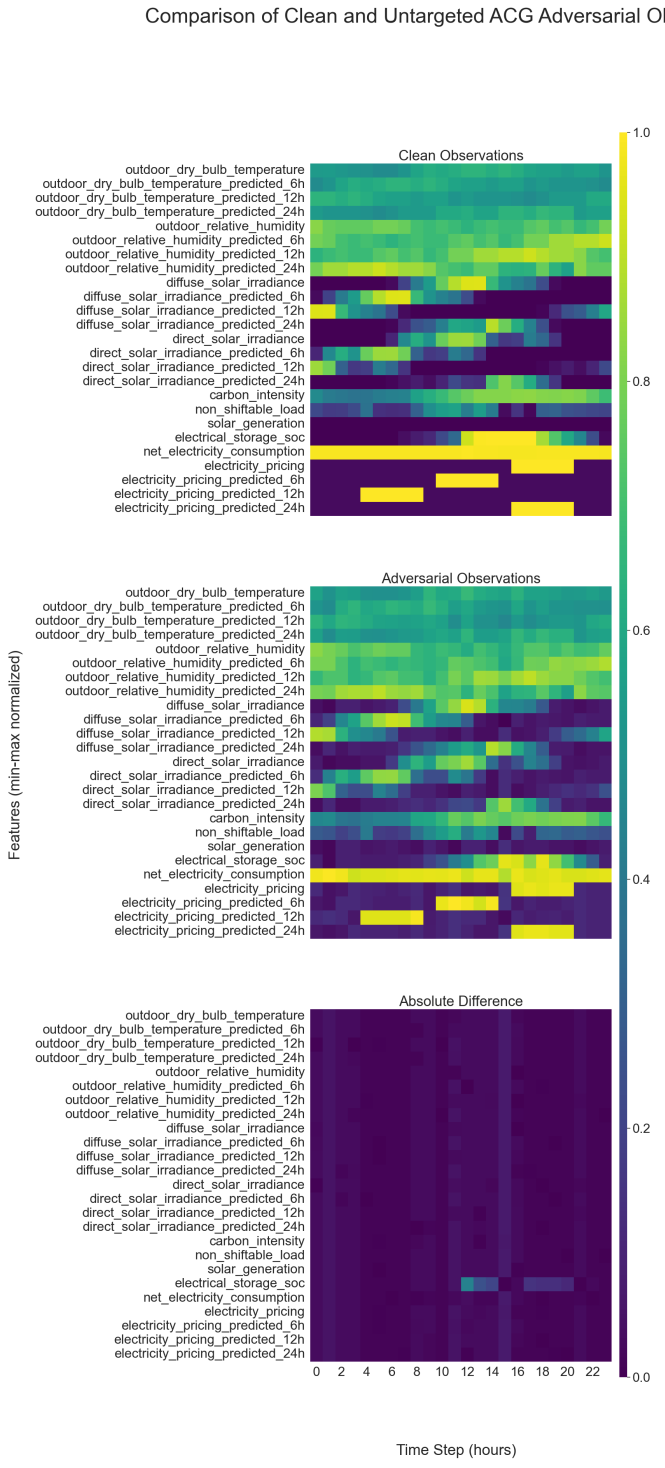


Figure 17: Heatmaps comparing the clean and adversarial observations over the course of one day in CityLearn. All features are min-max normalized, making them unit-less values between 0 and 1. Note from the bottom (absolute difference) sub-figure one observation had a large change to the electrical storage SoC, but the changes to other features appear minimal.

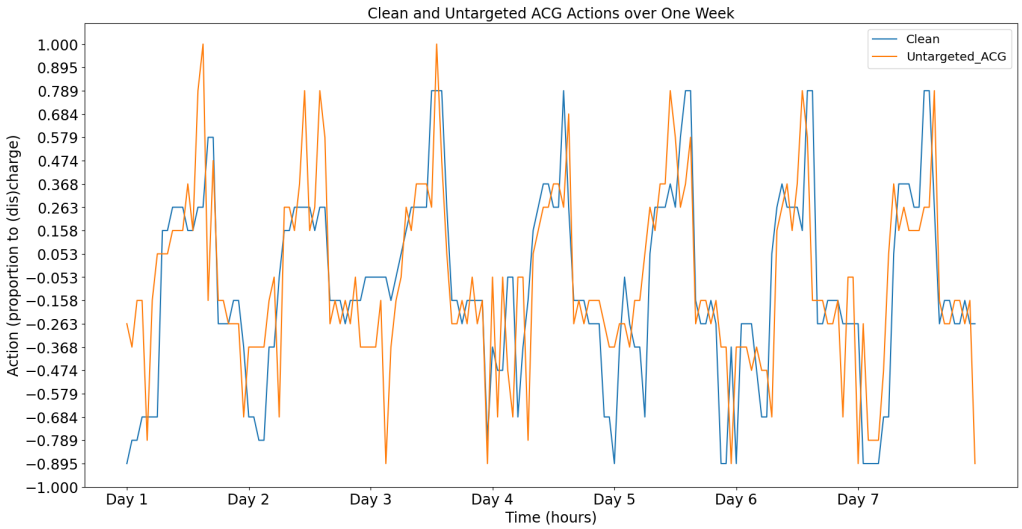


Figure 18: Line plot of the agent's actions over one week in CityLearn, during normal operation and subject to an untargeted ACG attack. The y-axis denotes the proportion of electrical storage to charge or discharge. An new action is selected every hour.

computational budget in failed searches. The multi-threaded approach could not return an adversarial sample in less time than a full search, which made it on average slower than the single threaded sequential binary search.

Figure 17 visualizes the size of the perturbations introduced by the attack. discerning the original and adversarial observations seems infeasible without a direct comparison. The absolute difference shows the size of the difference between the two, and it is apparent that ACG can craft adversarial examples by both making small changes to several features, or making a larger change to one or two. Note the large changes to the electrical storage SoC without obvious changes to other features.

Figure 18 shows the agent's actions under this attack compared to its normal actions. The attack is inducing the agent into taking sub-optimal actions, which are on average 2-3 actions away from the optimal. Note that the two lines have similar shapes, and that the sub-optimal actions rarely cause the battery to be discharged when the original action was to charge (and vice versa). This only happened for 6.36% of time steps in the episode depicted below. The adversarial actions typically oscillate around or translate the optimal actions, which only has a small effect on power consumption. Despite a 99% ASR, this is the reason the adversarial regret shown in Figure 15 is relatively small.

6.1.3 Strategically-Timed Attack

The strategically timed attack uses the estimated value of a state $V(s)$ to reduce the frequency of perturbations without reducing performance, by only perturbing high value states where $V(s) > c$ [5]. The purpose of this technique is making the perturbations generated with ACG more difficult to detect, as they are diluted with unperturbed samples or states.

To assess the ST attack in CityLearn, several thresholds c were compared to randomly timed perturbations. Values of c were chosen to include the first, second, and third quantiles of $V(s)$ for a clean evaluation, and compared to random perturbations with equal probability. For an ST attack with a second quantile $c = -70$, the KPIs were similar to randomly perturbing the observation 50% of the time as per Figure 19. The same results were obtained for the 25% and 75% quantiles. There was no significant

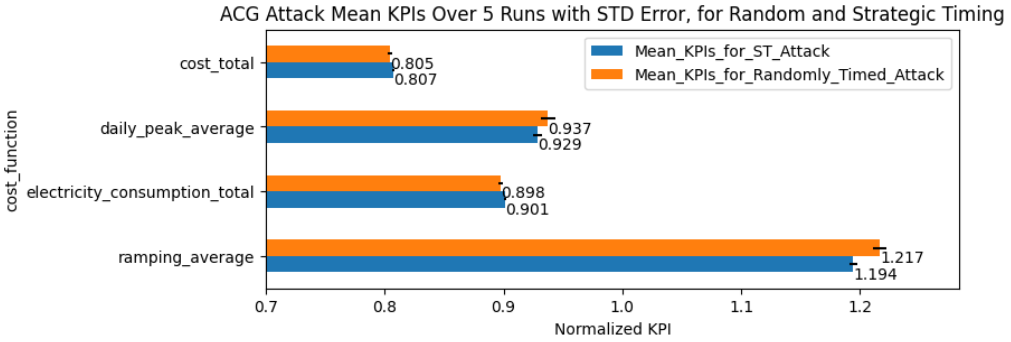


Figure 19: These are the resulting average KPIs after subjecting an agent to intermittent adversarial attacks generated from untargeted ACG. This test was performed with $c = -70$, which is the 50% quantile of the $V(s)$, compared to random perturbations 50% of the time. The ST attack [5] fails to outperform the randomly timed attacks in terms of adversarial regret by a notable margin.

	ST	Random Timing
ASR	$98.8\% \pm 0.2\%$	$98.0\% \pm 1\%$
Perturbation Rate	$49.8\% \pm 0.1\%$	$49.6\% \pm 0.6\%$

Table 5: Mean ASR and perturbation rates over 5 runs each of ST and randomly timed attacks. The ASRs and number of perturbations per episode between ST and random timing untargeted ACG attacks were nearly identical.

difference in either the ASRs or number of perturbations per episode between attacks as shown in Table 5. This indicates that the ST attack does not provide a significant advantage in this environment. CityLearn did not present critical moments when any but the optimal action results in a significant failure. In the Atari Pong environment where the ST attack was first implemented, a well timed sub-optimal action causes the paddle to miss the ball. However, there is no analogous situation in CityLearn, where the worst actions result in less stored power when it is needed or charging during periods of high demand, because the power grid has infinite capacity. This would be different if the environment considered the load on the entire power grid, where a well timed LAA could cause localized or cascading blackouts. Thus, future works in environments which include the load on the power grid can find the ST attack useful in their threat models.

6.1.4 Untargeted BB Attack

As a minimal-norm attack, BB typically achieves a higher ASR than ACG. As the feature space is the only boundary on its solutions, unlike maximum-confidence attacks which are bounded by ϵ, BB achieves an ASR of 100%. The goal of the BB algorithm is to find the closest adversarial sample to the input sample, but this can still result in large distortions. The L_∞ distance between the clean and adversarial observations for this attack had a mean of 2.7×10^{-2} and a standard deviation of 3.4×10^{-2}, but the maximum was 0.41. The adversarial regrets are shown in Figure 22.

BB generally takes longer to generate adversarial examples for an equal number of iterations. Each iteration in BB involves iteratively solving a dual Lagrangian optimization for each update of x_{adv}, and the complexity of this optimization can affect computation times. Unlike ACG, BB lacks an early stopping mechanism, so it always takes the maximum allowed number of steps when finding an adversarial sample.

The clean and adversarial actions in Figure 20 show adversarial actions which share

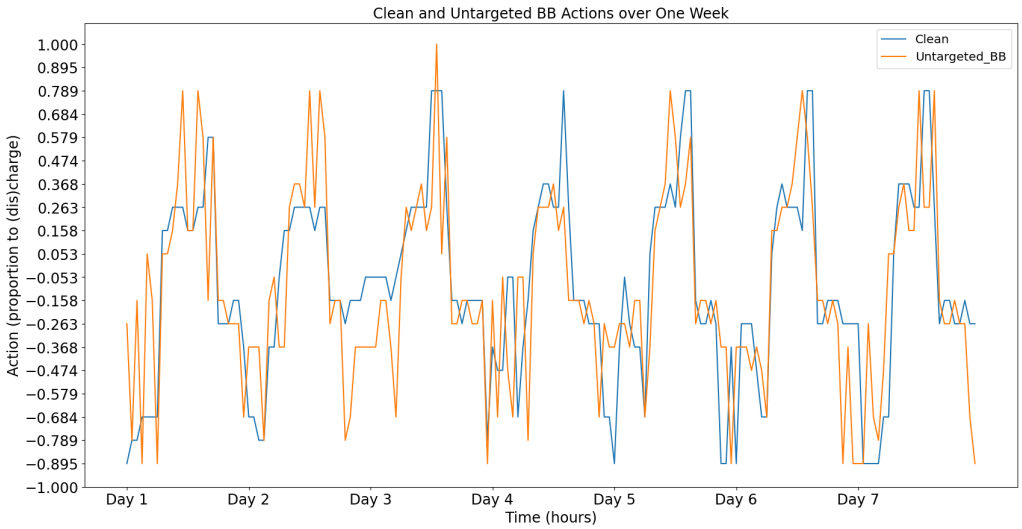


Figure 20: Line plot of the agent's actions over one week in CityLearn, during normal operation and subject to an untargeted BB attack. The y-axis denotes the proportion of electrical storage to charge or discharge. A new action is selected every hour.

the basic form of the clean actions, but oscillate or appear spikier. While this attack changed the victim's action at every time-step, it only reverses the victim agent's (dis)charge decision in 7.51% of cases. The adversarial actions are again on average 2-3 away from optimal.

Figure 21 shows the adversarial distortions introduced by the untargeted BB attack. While the absolute differences generally appear smaller than those for ACG in Figure 17, the adversarial observations at hours 6 and 15 stand out. The BB attack is not constrained to any distance from the original observation, it found a sub-optimal solution outside the ACG attack's boundary. Both attacks were successful at this time step, as both Figures 18 and 20 show that the original and adversarial actions don't match.

No hyperparameter tuning was required for the BB attack. Its creators note that the attack is insensitive to the changes in its hyperparameters, and it achieved an ASR of 100% with the default hyperparameters.

6.1.5 Comparison of ACG and BB results

Now we can compare the results of the ACG and BB adversarial attacks. BB had a higher ASR, adversarial regret, and smaller perturbations than ACG, even with the dynamic distortion algorithm minimizing the ϵ used. Table 6 shows the differences in mean perturbation size between the two attacks. Given the differences in attack methods, where ACG maximizes loss given a budget while BB searches for adversarial samples close to the decision boundary, the smaller norms for BB are predictable. Figure 23 shows the adversarial regrets for both attacks, which is the performance reduction cause by the attacks. It shows BB performed better for all relevant KPIs. The performance can be explained by comparing the adversarial actions for both attacks, as shown in Figure 24. The BB actions tend to oscillate more, appearing spikier than those for ACG. BB also changed the sign of the agents action, reversing its discharge decision, 1.18 as many times as ACG. These observations help to explain the increased electrical consumption under the BB attack.

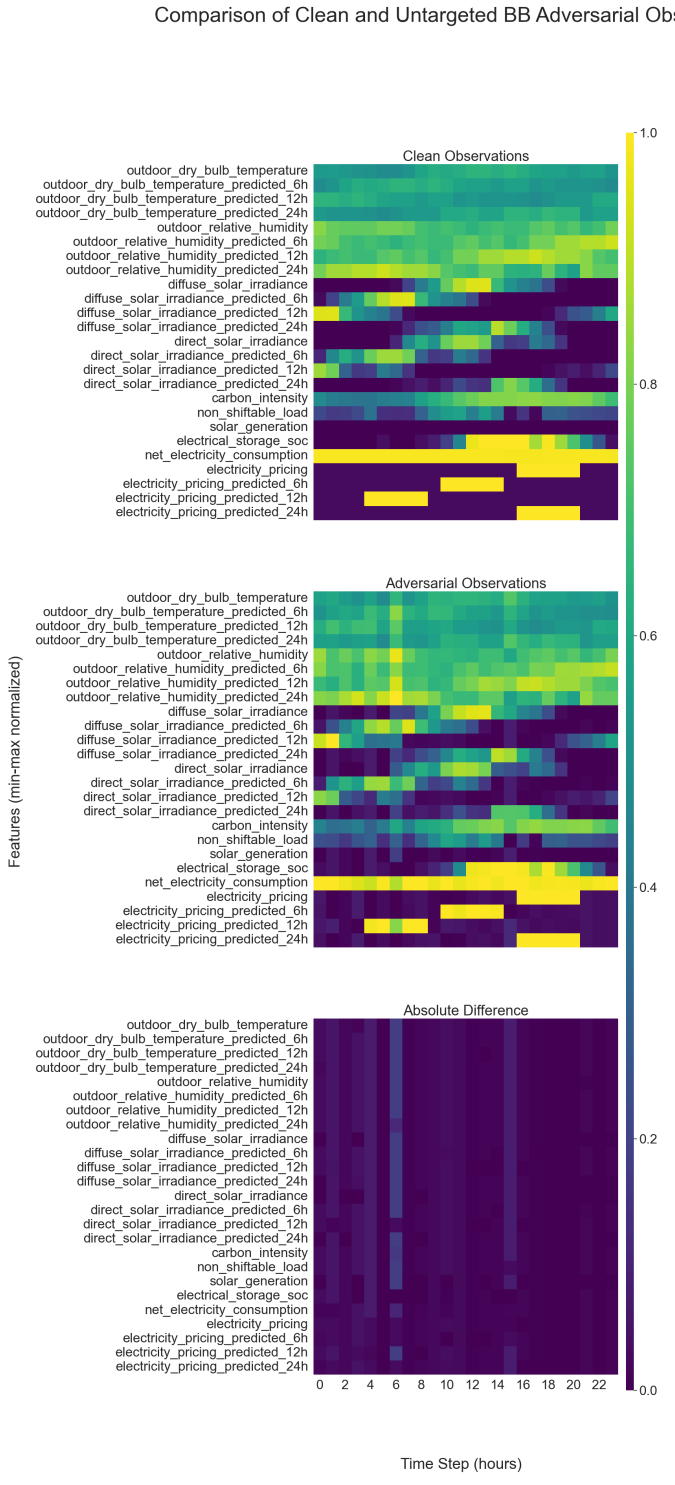


Figure 21: Heatmaps comparing the clean and adversarial observations over the course of one day in CityLearn. All features are min-max normalized, making them unitless values between 0 and 1. From the absolute difference plot, it appears that the measured L_∞ norm exceeds the maximum $\epsilon = 0.07$ for the ACG attack for hours 6 and 15.

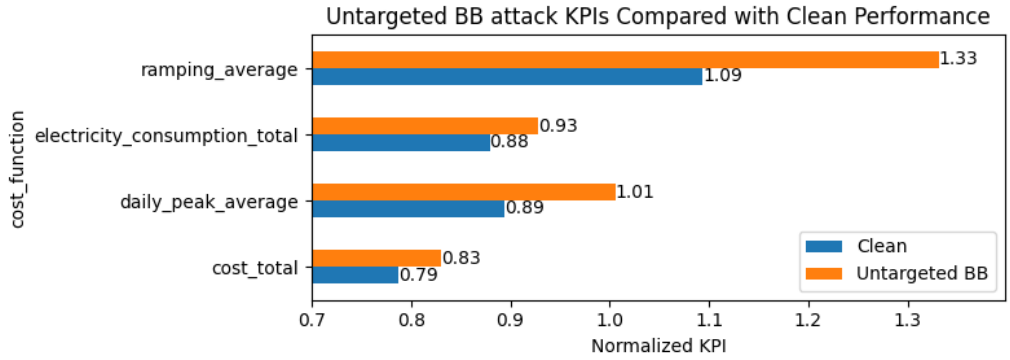


Figure 22: The KPIs with the untargeted BB attack are compared to those with no attack. Because smaller KPIs indicate better performance in CityLearn, the larger KPIs associated with the attack indicate that it reduced the victim agent's performance. The x-axis represents normalized performance without a smart controller

L_∞ Perturbation Distances		
Attack	Mean	Standard Deviation (STD)
Dynamic Distortion ACG	5.2×10^{-2}	8.2×10^{-2}
BB	2.7×10^{-2}	3.4×10^{-2}

Table 6: Statistics on the measured perturbation sizes for different adversarial attacks, smaller distances indicate a better attack. Note that the STD for the ACG attack exceeds the maximum $\epsilon = 0.07$, which indicates that it does not correspond to the maximum L_∞ perturbation.

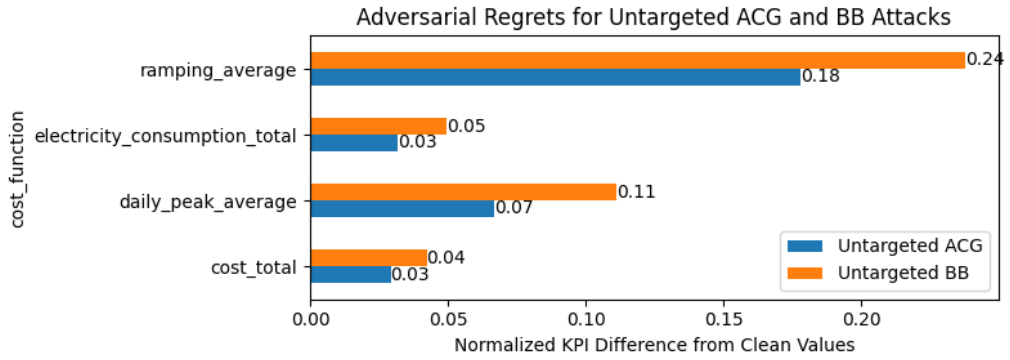


Figure 23: Adversarial regrets for the BB and ACG attacks, which is the difference in KPIs with and without the attacks. Larger regrets indicate a stronger attack. BB produces a larger regret for all the relevant cost functions.

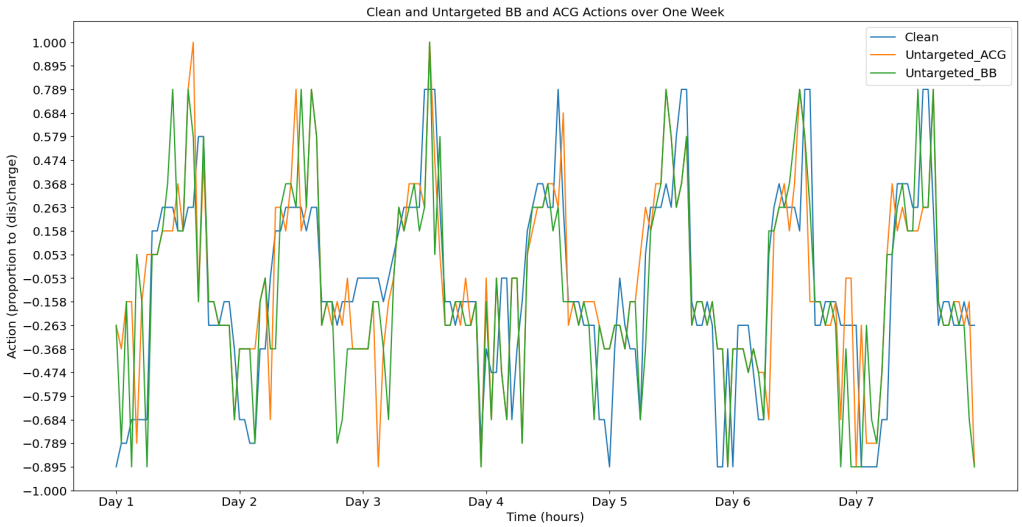


Figure 24: Line plot of the agent's actions over one week in CityLearn, clean and subject to untargeted ACG and BB attacks. The y-axis denotes the proportion of electrical storage to charge or discharge. A new action is selected every hour.

6.1.6 Untargeted Attack Summary

In this section, untargeted adversarial attacks were applied to the victim agent and the adversarial regret was measured. The attacks used were the maximum-confidence attack ACG and the minimum-norm attack BB. The dynamic distortion attack was implemented for ACG in order to improve its ASR while reducing the size of its perturbations. An ST attack was attempted, but its results were worse than random timing. This result indicates a lack of critical state where a sub-optimal action has a significant effect on an agent's performance in CityLearn. Because sub-optimal actions do not result in a failure state for CityLearn, such as a power outage, this type of attack requires an environment which simulates the wider power grid for future research. This continuous control problem is like steering a car, in that randomly changing the steering input by a few degrees has an insignificant effect on its driving performance. Unlike driving a car, there is no way to crash into something in Citylearn. The L_∞ norms of these adversarial attacks suggest they would be difficult to detect, but their effect may not be significant. The adversarial regrets are around 5% and the ST attack did not identify any key moments where the victim would be totally disrupted. This leads to the conclusion that attempting to detect these attacks may not be cost effective. Although, even a 5% increase in demand across a large percentage of users could cause cascading outages under a high grid load [7], but this is not modeled in CityLearn.

6.2 Optimally Targeted Attacks

6.2.1 Methodology

The previous section studied untargeted adversarial attacks whose goal is making the victim choose any sub-optimal action. While this requires a small adversarial budget, particularly in terms of distortion size, the adversarial regret was proportionally small. By instead inducing the victim to take the worst possible action at any timestep, the adversarial regret will significantly increase. To select the optimal worst action, a DRL agent was trained to choose the action which minimized rewards in CityLearn. This

represents a targeted attack. The adversarial policy was another A Proximal Policy Optimization (PPO) trained independently of the victim agent, for a reward based attack. It's hyperparameter's were identical to the victim's with the exception of the reward, which was the negative of the victim's reward. In this case the adversarial policy maximizes power usage, rather than minimizes it. For each observation, the adversarial policy selects the optimally worst action, and this is the target for the adversarial attack. This is called *policy induction*. Thus a successful attack induces the victim into taking the optimally worst action, which increases the adversarial regret compared to an untargeted attack.

6.2.2 Results

Even with $\epsilon = 0.7$, the targeted ACG demonstrated an ASR of $\tilde{2}5\%$, which produced a smaller adversarial regret than the untargeted attack. There were not enough changes to the victim's actions to meaningfully increase power consumption. Since ACG is the strongest open source maximum-confidence attack and increasing the adversarial budget was unsuccessful, a minimum-norm attack was used instead (recall that minimum-norm attacks are not confined to a boundary). The BB attack starts from a sample of the target class and steps towards the decision boundary, minimizing the norm. The BB attack had an ASR of 100% over an evaluation episode, however it required 4-16 seconds for each timestep compared to less than 3 seconds for ACG.

Though the iterative optimizations performed by BB are inherently slower than ACG, the majority of computation time was spent in a random search for samples of the target class. ART's documentation suggests initializing targeted attacks with a starting sample to avoid random searches, but the results were inconsistent and resulted in lower ASRs. The initializations were selected from clean samples of the target class or action, either for the minimal L_2 distance from the input sample or for the highest softmax confidence. L_2 distance was used to mimic a binary search step BB performs on random samples to begin them closer to the decision boundary. Upon finding an adversarial starting point, BB uses a binary search to minimize the L_2 distance between the stating point and the decision boundary. Softmax confidence was calculated using the logits of the victim agent's actor network. Given the reduced ASR observed with the min L_2 samples, it was hoped that selecting max confidence samples would remedy the issue, but was no more successful.

Since initialiations were less successful than random search, optimizing the search was the best method for reducing the computation time. BB as implemented in ART, uses an inefficient for loop which generates and predicts single samples. The search is successful when the prediction matches the target (or is different from the original for untargeted attacks). This implementation, shown in algorithm 5, is entirely sequential; it does not employ vectorized Numpy operations despite using Numpy, and uses batches of a single sample for ANN predictions. The Numpy library is a fast C library for Python, which can perform sequential operations much faster than native Python, using a technique called vectorization. Thus, a large batch of random samples can be produced faster in a single Python instruction than by generating the same number in a Python for loop. For ML libraries like PyTorch, larger batch sizes allow more samples to be loaded onto the computation device (CPU or GPU) for faster predictions than passing them individually. This is particularly important when the prediction is generated on a separate device (GPU) as performing singular operations in a for loop results in more latency from communication between devices. Through iterative testing, generating 10 000 random samples, and predictions in batches of 10 000 was up to 4 times faster than the original implementation, and never slower. This improvement is described by algorithm 6. Unfortunately this was the only simple optimization to be made. The optimizers in BB are sequential searches which do not significantly benefit from parallelism. They are

ran using the Python Just in Time (JiT) compiler Numba.

Algorithm 5 BB Random Initialization Algorithm.

```
for Init_size steps do
    Generate random sample
    Predict sample class
    if Prediction matches target class then
        return sample
    end if
end for
```

Algorithm 6 Improved BB Random Initialization Algorithm.

```
for Init_size steps do
    Generate 10 000 random samples
    Predict classes for all samples
    if One or more prediction match target class then
        choose sample of target class with minimal L2 distance to the original
        return sample
    end if
end for
```

Having reduced the computation time for the BB attack, the results of the attack were collected. The impact of the optimally targeted BB attack was enormous, as shown in Figure 26. The electricity consumption and cost, and the size of the daily peaks, were approximately tripled. Figure 25 shows how the optimally targeted adversarial actions differ from the originals. The adversarial policy is to alternate between large charge and discharge actions to drastically increase power consumption. These results show the potential threat posed by a white box adversarial attack on a DR system, as a LAA which triples power consumption is significant.

From Figure 27, note that the distortions from the optimally targeted BB attack are clearly visible in the data. While the attack succeeds in quickly crafting adversarial observations which effectively control the victim and impose an arbitrary policy, they could be discerned from normal data. The mean L_∞ norm for this optimal attack was 0.29, with a standard deviation of 0.09, and a maximum of 0.8. This means that for most observations, each feature was changed by approximately one third of its spread. Perturbations of this size cannot be considered stealthy.

6.2.3 Perturbation Reduction Methodology

Given the excessively large perturbations from the optimally targeted BB attack, three experiments will be run to see if the perturbations can be reduced:

1. BB parameter tuning,
2. Attacking models with less training time, and
3. Using an adversarial policy more similar to the victim's.

The parameter space is searched first so the results may be used in the subsequent experiments. An Optuna study is used to search the BB parameter space as shown in Table 7, for values which reduced the L_∞ for adversarial samples without decreasing the ASR. To reduce the computation time, the unperturbed observations and targets

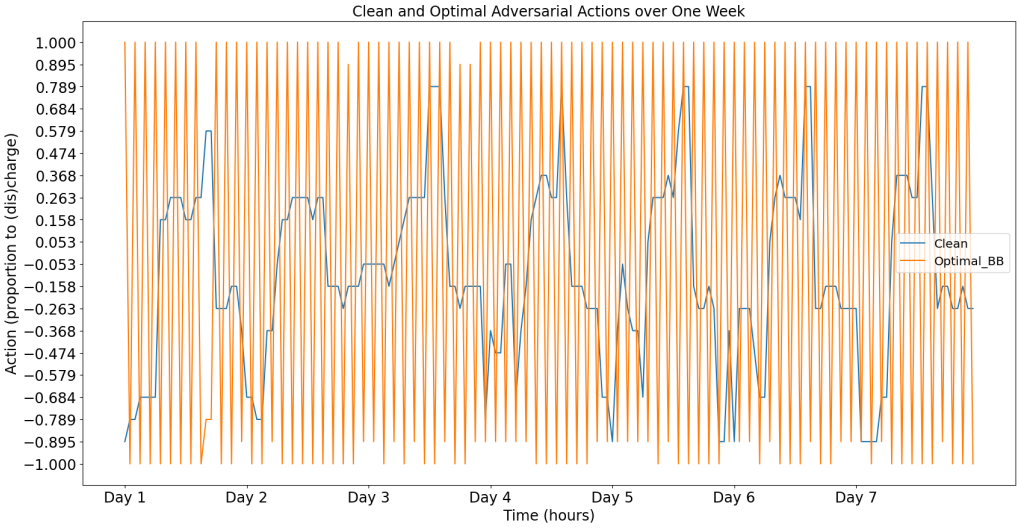


Figure 25: Actions induced by optimal adversary using the BB attack, plotted with the original actions over one week in CityLearn.

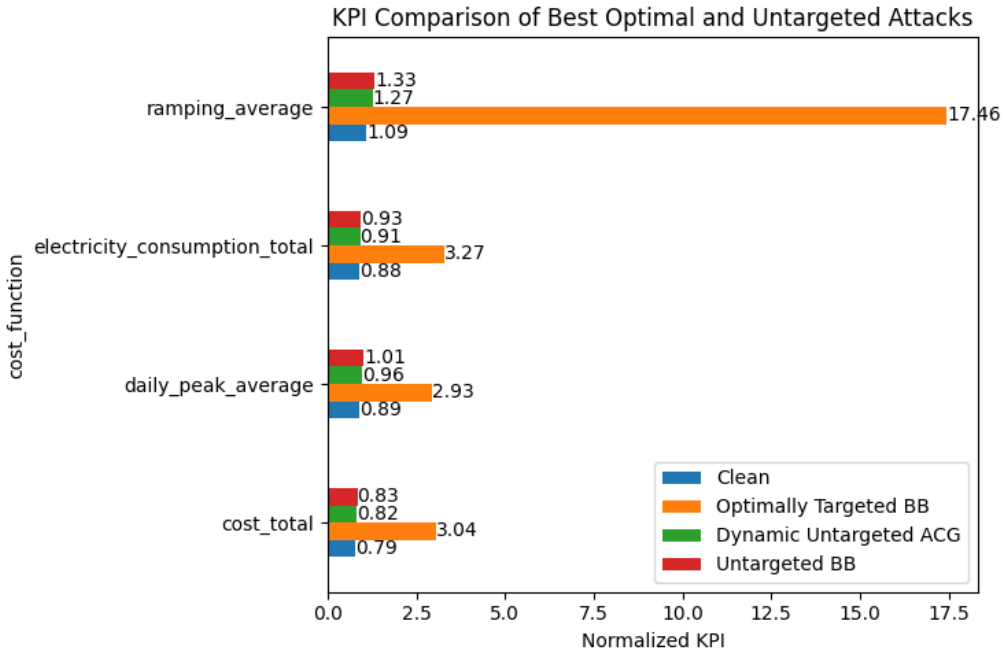


Figure 26: Comparison of the KPIs for the optimally targeted BB attack, with previous untargeted attacks, and clean performance. While the other attacks have a small but noticeable effect, the optimally attack drastically increases all KPIs. The targeted ACG attack was not included because it was not a successful attack, the difference in KPIs from clean performance was negligible compared to the clean performance.

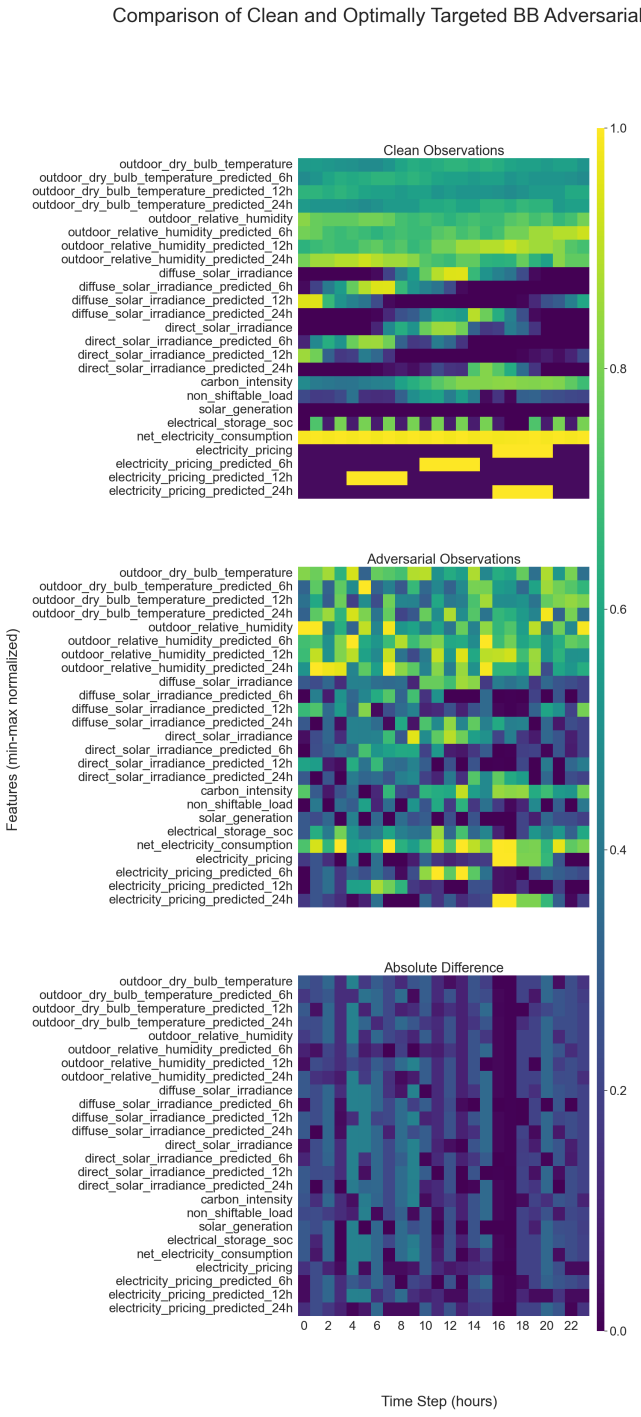


Figure 27: Heatmaps comparing the clean and adversarial observations in a targeted BB attack over the course of one day in CityLearn. All features are min-max normalized, making them unit-less values between 0 and 1.

Parameter	Learning Rate (LR)	LR Decay	Number of LR Decays	Momentum	Binary Search Steps
Search Range	$[1 \times 10^{-4}, 1 \times 10^{-2}]$	0.3-0.7	20-50	0.5-1.1	10-30

Table 7: Optuna search space for optimal BB attack parameters.

stored from an episode of CityLearn under the optimal BB attack will be fed to the BB attack with varied parameters. These stored observations for one episodes are stored as a dataset, so that BB can produce and adversarial observation for each using the stored targets. This allows BB to produce batches of adversarial samples instead of doing so sequentially in CityLearn. This also removes the overhead of running the environment, and the predictions of both the victim agent and adversary. Because BB had an ASR of 100% for targeted attacks, the next action in the sequence of observations is known, so nothing is gained from running the environment for the BB parameter search. The best search parameters alone would best tested in the CityLearn environment. Computation time is a significant factor since the BB attack may be run for all observations during hundreds of trials. A trial in CityLearn with optimally targeted BB takes approximately 24 hours, while using the method described for the parameter search takes less than one hour per trial.

The study also uses initialized adversarial starting points, which removes the need for the random search. While this did slightly reduce the ASR during the untargeted attack, the trade-off is necessary to enable a larger search of the parameter space. Designing the study objective was nuanced, as the goal is to maximize ASR while minimizing the norm. This is further complicated because BB does return the original sample when it does not succeed, making its norm from the original 0. Thus, the objective function replaced the norms of unsuccessful samples with a multiple of the maximum norm among all adversarial samples, and returns the mean norm as the value to minimize over the course of the study. The objective algorithm is written out in Algorithm 7 with a penalty P scaling the penalty for unsuccessful attacks. For a successful attack the score is equal to its L_∞ norm, and $P \times L_{max}$ otherwise. This scoring method combines both objectives into a minimization problem.

Algorithm 7 Optimized BB Attack Hyperparameters Objective Function.

Inputs: original samples: x, targets y_t, initialization samples: x_{init}, parameters: p, agent actor network: A, penalty multiplier: P
$\tilde{x} \leftarrow BB(x, y_t, x_{init}, p)$
$\tilde{y} \leftarrow A(\tilde{x})$ //Actor networks predictions for the adversarial samples
$L \leftarrow ||x - \tilde{x}||_\infty$
for each index i in L **do**
  if $\tilde{y} \neq y_t$ **then**
    $L_i \leftarrow \max(L) \times P$
  end if
end for
Return mean(L)

Next, targeted BB attacks are tested against a variety of victim agents, trained between 50-500 episodes using optimal adversary (used for the optimally targeted attack) and helpful (described in the following paragraph) policies . Attacking agents with shorter training times will test if a stealthy attack is possible for a victim which has not

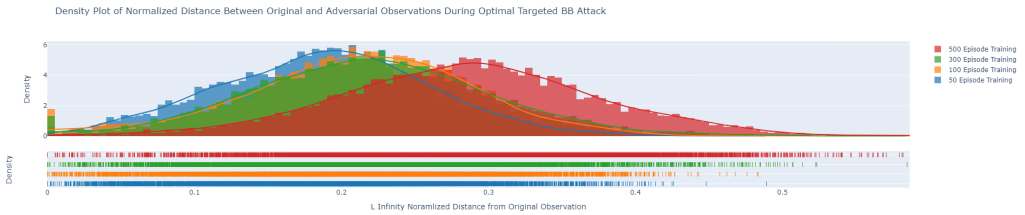


Figure 28: Density plot of the L_∞ distances between the original and adversarial observations during a targeted BB attack using the optimal adversary. Note that the distance is proportional to the duration of the victim agent's training, however these distances are significantly larger than for the untargeted attack.

yet learned the best action. The untargeted attacks showed that smaller perturbations can change the victim's action, but the adversarial actions were 2-3 bins from the originals. The optimal adversarial actions for the targeted attacks were typically at either extreme of the action space, and thus further from the original, while also requiring much larger perturbations. This suggests that the perturbation required is loosely proportional to the distance between the original and adversarial action.

Finally, using the helpful policy, that of a PPO trained for 500 episodes in CityLearn, this experiment will test if an adversarial policy similar to the victim's requires smaller perturbations. The helpful policy is used to improve the scores of agents trained for 50 and 100 episodes, instead of reducing them. If inducing a similar policy to the victim's using a target BB attack can be stealthy, this will at least demonstrate that stealthy policy induction is possible. Conversely, if stealthy policy induction is not poosible with such a similar policy, then it is unlikely to be possible with an adversarial policy.

6.2.4 Perturbation Reduction Results

The outcome of the Optuna parameter space search is sensitive to penalty P, as too large a values will skew towards high ASRs at the expense of the L_∞ norm. $P = 1.1$ was used for the final trial. The outcome of these trials was that the parameters have a limited effect on the ASR as the authors note in [2]. Though poor choices lead to larger norms, no parameters improved upon the default values. Both decreasing the overshoot parameter to 1.01 and double the number of steps to 2000 had an insignificant effect on the distortion size. Thus, modified hyperparameters did not improve the distortion size during the optimal targeted BB attack. The performance of BB was not improved for optimally targeted attack through a hyperparameter search.

The results of optimally targeted attacks on victims with less training do show a significant but insufficient reduction in the perturbation size as can be seen in Figure 28. These results show that agents with less training are easier to induce into adversarial actions. without the benefit of additional training, these agents act sub-optimally. With less training, the agent's actions are closer to the worst action in the feature space, so smaller distortions are required. The mean L_∞ norm for 50 training episodes is two thirds of the norm for 500. However, even with this reduction the data is still significantly distorted.

Figure 29 shows the results of the targeted BB attack using the helpful policy for a victim trained for 100 episodes. Again, there is a significant but insufficient reduction in the L_∞ norm. The victim trained for 100 episodes was selected because agents with less training were more susceptible to policy induction in the previous experiment, but the victim trained for 50 episodes had a policy too dissimilar from the helpful policy. This is effectively the best case for the adversary. Because this test was unsuccessful, no

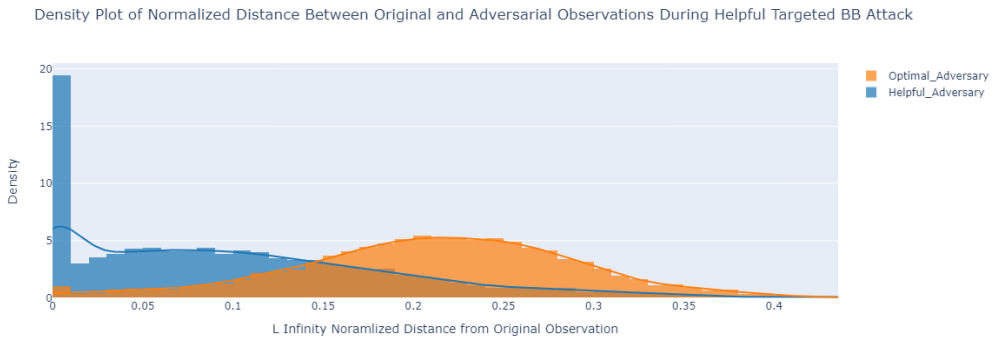


Figure 29: Density plot of the L_∞ distances between the original and adversarial observations during a targeted BB attack using helpful and optimal adversarial policies. The helpful policy is that of the PPO trained for 500 episodes, with the victim agent in both cases trained for 100 episodes.

optimally target BB attack is likely to be stealthy.

6.2.5 Optimal Attack Summary

In this section an adversarial policy was trained for a reward-based attack designed to maximize the victim agent's power usage. The attack was successfully conducted using the BB attack after ACG's ASR was too low to produce a significant adversarial regret, and the victim's power usage was more than tripled. However, the L_∞ distance between the clean and adversarial observations reached 0.8 in the worst cases, making the adversarial distortions evident. Hyperparameter tuning of the BB attack was attempted, to test if different parameters resulted in a more efficient search for adversarial perturbations closer to the original inputs. This was unsuccessful which indicates the large distortions are not the result of an inefficient search for the minimum adversarial perturbation. While attacks on victims with less training time and using adversarial policies closer to the victim's reduced the L_∞ distance, the change was insufficient to make the attack stealthy.

These results show that an attacker can induce a victim DRL controller to follow an arbitrary adversarial policy, while minimizing the perturbation size through the use of the minimum-norm BB attack. The fact that an adversary can quickly and effectively generate false data which allows them to control a victim model makes investing in detecting this false data worthwhile. An LAA which can increase the power consumption of a large industrial user by three times could have a significant effect on grid stability. However, given the size of perturbations required, detecting and flagging such perturbations would be an effective means of mitigating SotA attack algorithms. Given the service lives of energy infrastructure, considerations should be given to future attacks which have yet to be developed and could prove more difficult to detect.

6.3 Bifurcation and Target Group Methodology

The optimally targeted attack had an extraordinarily high adversarial regret, with unacceptably large distortions. Training it also requires access to the victim's training environment, which is not guaranteed for black box settings, like the snooping attack. The optimal targeting policy oscillates between maximum charging and discharging, while the much less effective (in terms of adversarial regret) untargeted attacks only change

the action by an average of three positions and rarely changed the direction of the action (∼7%). This effect is described by Figure 30. Could a similar effect be achieved by targeting the furthest action in the opposite direction with a maximum confidence attack, to force the agent to charge when it would discharge and vice versa?

Algorithm 8 Furthest Action Algorithm.

Inputs agent parameters θ, environment, attack function $atk()$
$a_{mid} \leftarrow \frac{||A||}{2}$ //boundary for (dis)charge decision
for Each observation o in episode for environment **do**
  $a \leftarrow \pi(o, \theta)$
  if $a < a_{mid}$ **then**
    $search_{min} \leftarrow a_{mid}$
    $search_{max} \leftarrow \max(A)$
  else
    $search_{min} \leftarrow \min(A)$
    $search_{max} \leftarrow a_{mid}$
  end if
  while do$search_{min} < search_{max}$ //Binary search
    $y_t \leftarrow search_{min} + \lfloor \frac{search_{max} - search_{min}}{2} \rfloor$
    $\tilde{o} \leftarrow atk(o, y_t, \theta)$
    $\tilde{a} \leftarrow \pi(\tilde{o}, \theta)$
    if $a \equiv \tilde{a}$ **then**
      $\tilde{o}_{best} \leftarrow \tilde{o}$
      $search_{min} \leftarrow y_t$
    else
      $search_{max} \leftarrow y_t$
    end if
  end while
end for

Using Algorithm 8, iteratively targeting the closest opposite action was entirely unsuccessful with $\epsilon \leq 0.1$. This approach was not able to flip the agent's decision, meaning that ACG was unable to solve the optimization for a single target. While this method could be improved by duplicating the observation to form a batch of observation-target pairs for different targets, the lack of success for the iterative method made further testing unattractive. Furthermore, making each observation into a batch linearly increases the necessary computation. The problem is a mix of a targeted and untargeted attack, where a subset of labels are acceptable, which will be referred to as the *target group*.

6.3.1 Grouped Difference Logit Loss

Consider this simple function, Difference Logit (DL) loss, which is similar to the loss used in [2]:

$$DL(x, y) = z_y - \max_{i \neq y} z_i \tag{36}$$

An attacker can use this loss function to craft an adversarial sample corresponding to any label excluding the original (y). But this is not helpful for the problem at hand, as the untargeted attacks demonstrated that small changes to the agent's actions do not significantly affect its performance. But as above, nor is a targeted attack helpful, as there are multiple acceptable adversarial labels or actions. Consider the group G which is a subset of the agent's actions, a loss function which maximized any label in G over the original would be applicable to the problem at hand. This Grouped DL (GDL) loss

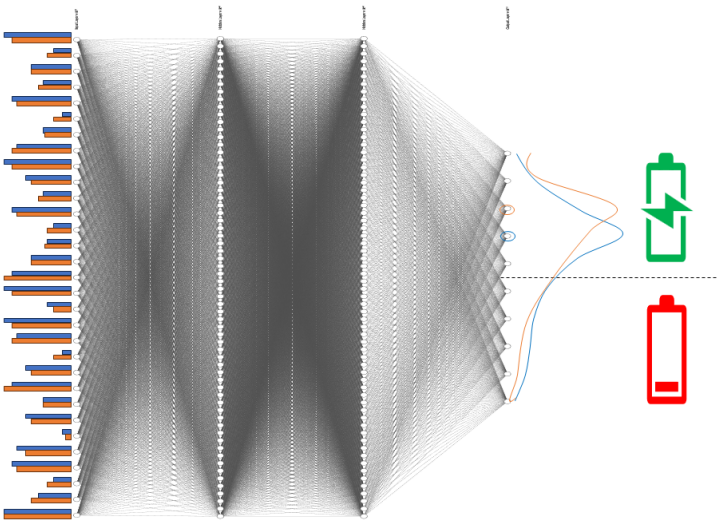


Figure 30: Example of an adversarial attack on a discrete actor network trained in CityLearn. The original observations and actions are represented by elements in blue, and adversarial in orange. The bars on the left represent features in an observation, and the curves on the right represent the value of each logit (which correspond to actions). The upper five logits represent different levels of charge actions, and the bottom five discharge. In this example, small changes to the original observation result in an adversarial action different from the original. But, the result is only slightly more charging than is optimal, so the impact on power consumption is limited.

can be written as:

$$GDL(x, y) = z_y - \max_{i \in G} z_i \tag{37}$$

Like the target, the group G changes over the duration of an attack on a DRL agent. Specifically, when the agent would choose one of the charge actions, all discharge attacks are added to the target group. ART[36] does not include the proposed GDL loss (nor does any other library reviewed), so the most direct way of implementing GDL in ART's framework is by modifying the ANN provided to it.

Using DL or GDL instead of Difference Logit Ratio (DLR) also removes the regularizing effect of the denominator, which aims to keep the original class as the second most likely. This may contribute to the proximity of the original and adversarial samples. This approach is ideal for control tasks where an adversarial action's distance from the optimal action is proportional to the adversarial regret it causes. Consider a steering input for a vehicle, the further the input is from optimal, the further it will stray off course. In classification tasks generally, any label but the correct one has an equal adversarial regret in terms of accuracy. However, there are circumstances where certain groups of mis-classifications result in a larger adversarial regret than others. Consider an image classifier used by an autonomous vehicle: causing a Sedan to be misclassified as a truck or bus might have some effect on the vehicle's behaviour, but mis-classifying it as any non-vehicle could have an even greater effect.

6.3.2 Bifurcation Layer

The actor network for a discrete agent in CityLearn will have symmetrically graduated actions for various level of (dis)charge, e.g. charge to 75% capacity, or discharge until you

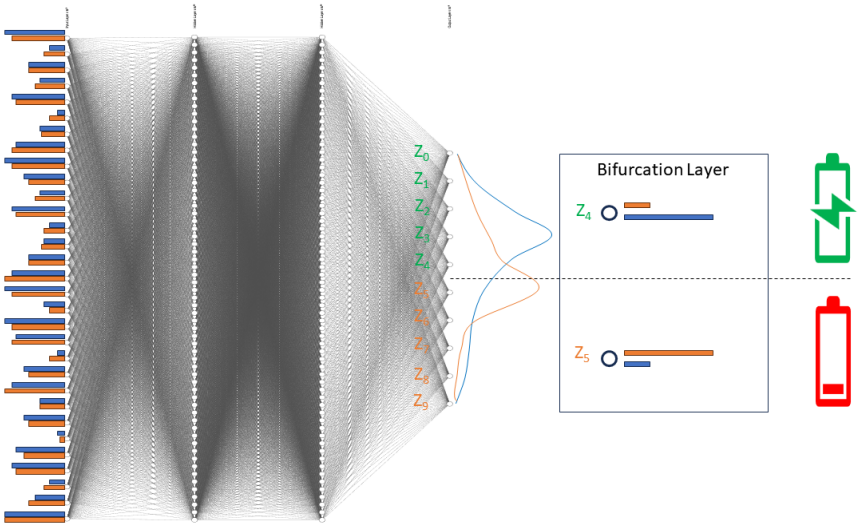


Figure 31: Example of an adversarial attack on a discrete actor network trained in CityLearn, using the bifurcation method. The original observations and actions are represented by elements in blue, and adversarial in orange. The bars on the left represent features in an observation, and the curves on the right represent the value of each logit (which correspond to actions). The upper five logits represent different levels of charge actions, and the bottom five discharge. The output of the bifurcation layer is the maximum logit value for each of these groups of logits. In this example, small changes to the original observation result in a discharge action instead of the original charge action. Inducing the victim agent to reverse its (dis)charge decisions increases electricity consumption.

have 8% charge left. These can be combined into target groups for GDL loss by modifying the network used by ART to return only a pair logits, corresponding to the maximal charge and discharge logits. This binary maximum layer makes DL loss equivalent to GDL loss. We call this layer the bifurcation layer, which is described by Figure 31. Unlike GDL, DL is easily implemented in ART using PyTorch. By implementing GDL through the input network, the method is practical for most adversarial frameworks and methods, rather than restricting it to a single library or attack. Note that DL Loss must be used in place of DLR as shown in equation (6), since the latter requires more than 2 logits to compute the loss. Cross-Entropy (CE) loss could also be used, but it is both more difficult to illustrate the concepts above with it, and the issues of vanishing gradients (see Section 2.3.2) lower its ASR compared to other loss functions. Compared to DL, CE Loss resulted in 8 fewer percentage points in the ASR for otherwise identical ACG attacks.

6.3.3 Continuous Action Spaces

Using the bifurcation method an adversary changes the victim model's outputs to suit their objective. Above, logits are grouped to surrogate targets, but the same can be done to add logits to a regressor with only a single output. The majority of adversarial attacks are designed for classifiers, for which one logit corresponds to a label and the largest logit value is the model's decision. A regressor has a single output with a continuous range of values. These correspond to discrete and continuous action spaces in DRL, and continuous action spaces are typically used for control tasks such as the DR simulated in CityLearn. This limitation in adversarial example research significantly limits their

application to DRL controllers. However, a logit layer can be added to a regressor as glue logic between the ANN and attack algorithm. The simplest bifurcation layer tested consists of two outputs, one is the same as regressor's original output, and the other its negative. Instead of returning a single prediction value, the network is made to return a tuple of two values. The output y becomes the vector $(y, -y)$. Thus the greater of the two logits is treated as the ANN's prediction. The loss gradient will be used to reduce the original logit while increasing the the negative logit. It does not matter if the greatest logit changes during the attack, as the attacker's goal is to find inputs which significantly changes the regressor's or agent's prediction. Other bifurcation layer configurations were explored, using various exponential and linear functions, but none were found to outperform this method in preliminary experiments.

Minimum-norm attacks are inappropriate for this task because they rely on a classifier's decision boundary, the boundary is crossed when the logit with the largest value changes. There is no analogous concept for a regressor with a single output. Maximum-confidence attacks simply maximize an arbitrary loss function with a boundary, so they are easily adapted for regression. Instead of maximizing the value of a logit other than the original, they maximize the difference between the original and adversarial outputs within their budget. The attacker can change the adversarial budget to balance adversarial regret and stealth.

To compare the bifurcation method to a direct attack on an agent with a continuous action space, an attack is required which is compatible both with the continuous output of the agent and the bifurcated output. ACG can only do the latter as it amounts to a classification. To this end, a simple Projected Gradient Descent (PGD) attack for arbitrary loss functions is implemented, which is shown in Algorithm 9. This same algorithm is an improvement of the Fast Gradient Method (FGM) (3), and is a simplified version of ACG (auto-stepsize selection and a CG coefficient could be added to improve performance, though this is outside the scope). Furthermore, the simplicity of this implementation reduces the computation time, which enables more test iterations. This PGD implementation allows attacks using loss functions such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and Huber loss, which are compatible with the regression networks used by continuous action agents. This attack is used to compare the adversarial regrets for the Soft Actor Critic (SAC) agent with and without the bifurcation layer, and between the SAC and discrete PPO. In addition to adversarial regret, there are several metrics which are useful in comparing the effects of these attacks:

1. Because there is no discrete boundary between the decisions of a regressor or continuous agent, MAE will be used instead. The discrete agent's actions are mapped onto a continuous action space, so the MAE can be calculated for both. MAE is linear and does not alter the unit of the inputs, so it can be intuitively understood as the *distance* between the original and adversarial actions.

2. The (dis)charge reversal is proportional to the adversarial regret, and the goal of the bifurcation method. The early untargeted attacks had a limited effect on this metric, and similarly limited adversarial regrets. By comparing the signs of the original and adversarial actions, the proportion of timesteps where the victim agent's decision is reversed can be counted.

6.4 Bifurcation and Target Group Results

To test the bifurcation layer on an agent with a continuous action space, a SAC was trained for 500 episodes using the energy consumption reward. Its training environment was identical to the discrete PPO, except the action space was not discretized, so $A \in [0, 1]$. The μ network, analogous to the policy net for the PPO, has a single output with a

Algorithm 9 PGD with Decaying Stepsize [66].

Inputs: model θ, sample x, model prediction y, loss function L, ϵ, stepsize η, iterations N_{iter}, decay rate α, number of decays N_α
$\delta^0 \leftarrow 0$
$k_\alpha \leftarrow \lfloor \frac{N_{iter}}{N_\alpha} \rfloor$
for $k = 1$ to N_{iter} **do**
  $s^k \leftarrow sign(\nabla L(x + \delta^{(k-1)}, y, \theta))$
  $\delta^k \leftarrow P_\epsilon(\delta^{(k-1)} + \eta s^k)$
  if $k \mod k_\alpha = 0$ **then**
    $\eta \leftarrow \eta\alpha$
  end if
end for
return δ

ϵ	Initial Stepsize	Iterations	Number of Stepsize Decays	Decay Rate
0.05	0.01	100	4	0.5

Table 8: Parameters for the PGD attack used in this section, ϵ varies in the following section.

linear activation function. This function does not constrain the network's output, instead it is mapped to the action space by the SB3 SAC algorithm. So the μ network extracted to craft adversarial observations has a continuous and unconstrained output, despite the agent as a whole having a constrained action space. Using this simple PGD algorithm, both the robustness of the SAC and Discrete PPO agents, and the performance of direct and bifurcation attacks, were compared. All tests are conducted using PGD with the same hyper parameters shown in Table 8, except the loss function for compatibility reasons. There was no variation in adversarial regret or MAE between the MAE, MSE, and Huber loss functions for the PGD attack on the SAC. DLL was used for PGD on the discrete PPO. All these loss functions are continuously differentiable, and do not exhibit the gradient masking behaviour of CE loss. The discrete PPO and SAC are best compared using the direct PGD attack because it minimizes the variation of test conditions between them. In contrast, using the bifurcation involves adding different layers to each agent's actor output.

6.4.1 Comparison of Direct and Bifurcation Attack performance

Figure 32 demonstrates a significantly larger adversarial regret from the bifurcation method compared to attacking the SAC directly. This shows that using the bifurcation method not only interfaces adversarial classification attacks to regression tasks, but also improves their performance in continuous control settings, which is consistent with

Attack	**SAC**		**Discrete PPO**	
	MAE	(Dis)Charge reversal	MAE	(Dis)Charge reversal
PGD	0.583	27.4%	0.126	4.2%
Bifurcated PGD	0.957	95.7%	0.226	26.1%

Table 9: Metrics for comparing direct and bifurcated PGD on discrete and continuous agents. MAE can be applied to both the discrete and continuous agents when their actions are transformed to the same action space $A \in [-1, 1]$, making them directly comparable. The (dis)charge decision measures the proportion of adversarial observations which changed the sign of the agent's action by reversing the decision to (dis)charge the battery.

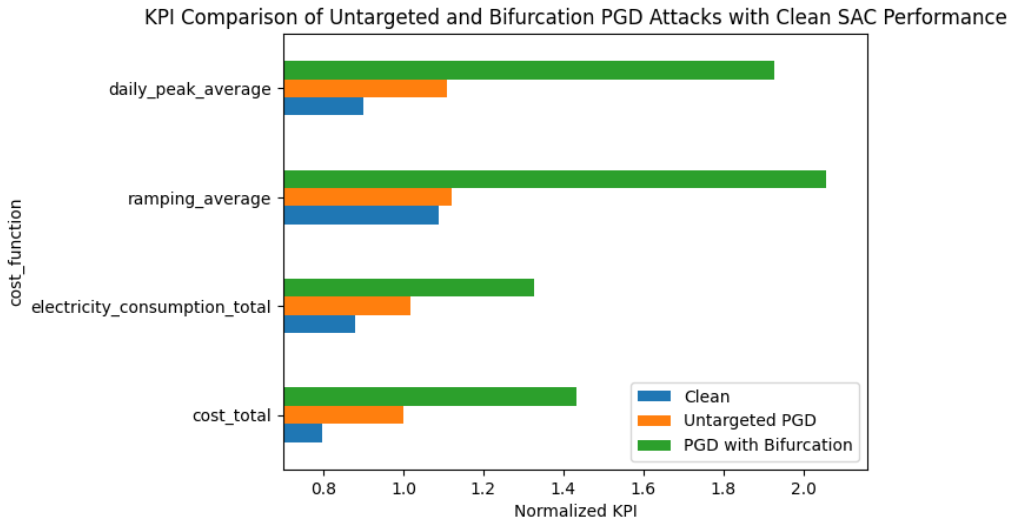


Figure 32: Graphs comparing KPIs of episodes with no attack, PGD, and PGD using the bifurcation method. Because the PGD attack had identical parameters (besides the loss function), and the figure shows the increased adversarial regret of the bifurcation attack over PGD alone, it demonstrates the superiority of the bifurcation method.

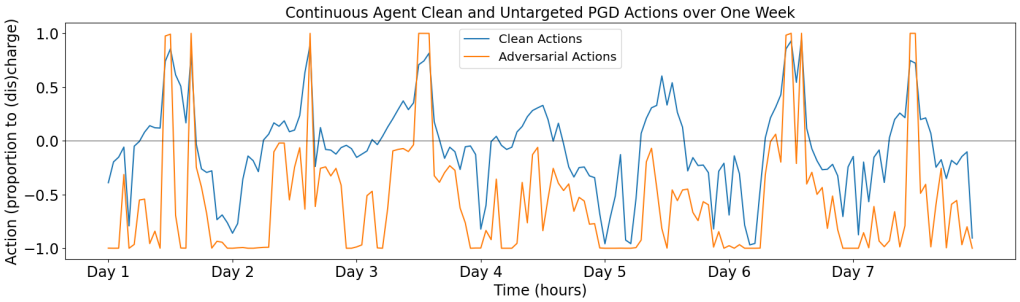


Figure 33: Plot of the clean and adversarial actions from a PGD attack for a SAC, over one week. Note how frequently the sign changes during this period, showing that the attack reversed the agent's (dis)charge decision.

the findings comparing the bifurcation method to direct attacks on the discrete PPO. The PGD bifurcation attack reversed the SAC's (dis)charge decision 3.5 times more than PGD alone, and the MAE was 1.8 times larger. The differences in the adversarial actions are visualized in Figure 33 for the direct PGD attack, and Figure 34 for bifurcated PGD. Note that for both figures the clean and adversarial actions rarely overlap. While the PGD actions share a similar plot to the clean actions, those corresponding to the bifurcated attack appear somewhat mirrored as this attack was designed to reverse the sign of the agent's action. Success here explains the increased adversarial regret.

6.4.2 Relative Robustness Between SAC and Discrete PPO

There are several metrics which explain the discrepancy in adversarial regret between the SAC and discrete PPO as shown in Figure 35, which are summarized in Table 9. When transformed to the same action space $A \in [-1, 1]$, the MAE between the unperturbed and adversarial actions can be computed as a measure of the change imposed by the adversary. The SAC's MAE is 4 times larger than the discrete PPO's, demonstrating

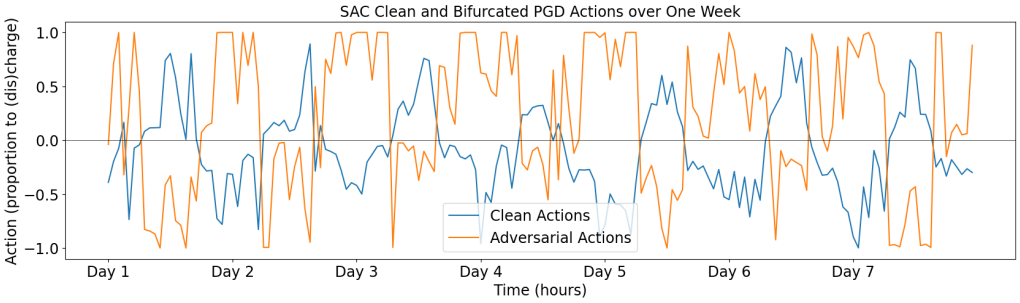


Figure 34: Plot of the clean and adversarial actions from a bifurcated PGD attack for a SAC, over one week. Note how the sign changes more frequently for this attack than the direct PGD.

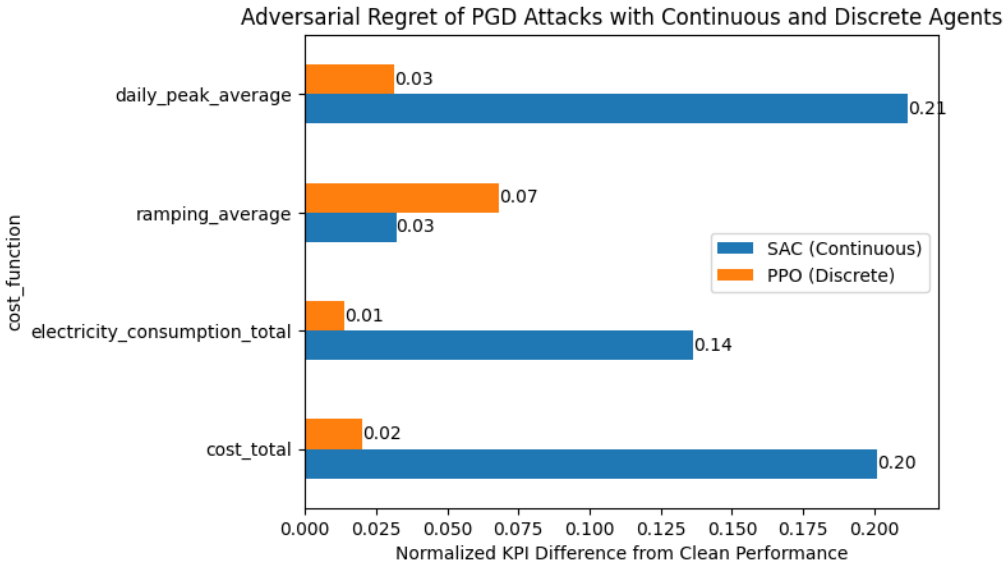


Figure 35: Adversarial regret of select KPIs for the PGD attack on a SAC and discrete PPO. The adversarial regret is calculated as the difference between the KPIs during the attack and with no attack, and lower is better. The figure shows that the Discrete PPO is more robust to this attack than the SAC.

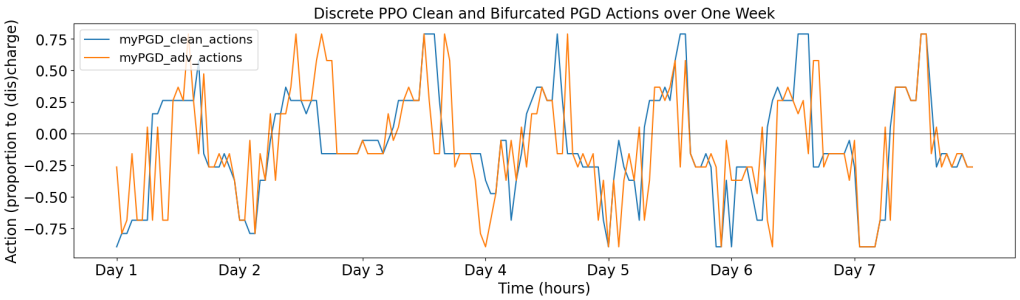


Figure 36: Clean and adversarial actions from a PGD attack for a discrete PPO, over one week. Note how they resemble each other, which is consistent with their small MAE. In this period the signs of both actions match, meaning the attack hasn't reversed the (dis)charge decision.

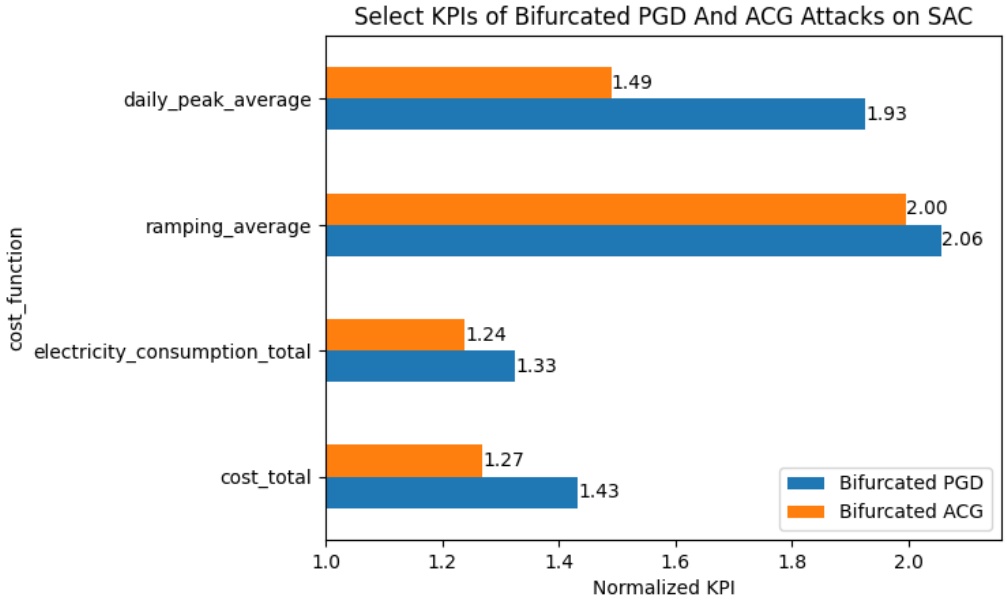


Figure 37: KPIs for select cost functions for the SAC agent under bifurcation attacks using ACG and PGD. The adversarial regret from PGD has outperformed ACG.

that the adversarial observations had a larger effect on its actions. This resulted in the SAC reversing its (dis)charge decision 6.5 times more than the discrete PPO under the same attack. The difference between the SAC and discrete PPO's performance are apparent by comparing Figure 36 and Figure 33, which illustrate the metrics discussed above. There is significantly more overlap between the clean and adversarial actions for the discrete PPO, which corresponds to a smaller adversarial regret.

Figure 38 shows the different distributions of the adversarial actions from bifurcated ACG and PGD attacks. The ACG distribution vaguely resembles the clean distribution and is clustered near 0, with less density near the extremes. This means the adversarial ACG actions in this scenario cause a nearly even amount of charging and discharging, and rarely fulling charge or discharge the battery. PGD exhibits a much less even distribution clustered near the maximum of 1, meaning these adversarial observations do cause the agent to fully charge, which consumes more energy.

Curiously, the ASR of 36.6% for bifurcated ACG on the discrete PPO was barely more than half the ASR of 62.8% from bifurcated PGD. These were identical for the

Attack	SAC		Discrete PPO	
	MAE	(Dis)Charge reversal	MAE	(Dis)Charge reversal
Bifurcated ACG	0.568	92.0%	0.178	21.2%
Bifurcated PGD	0.957	95.7%	0.266	26.1%

Table 10: Comparing bifurcated ACG and PGD on discrete and continuous agents. MAE can be applied to both the discrete and continuous agents when their actions are transformed to the same action space $A \in [-1, 1]$, making them directly comparable. The (dis)charge decision measures the proportion of adversarial observations which changed the sign of the agent's action by reversing the decision to (dis)charge the battery.

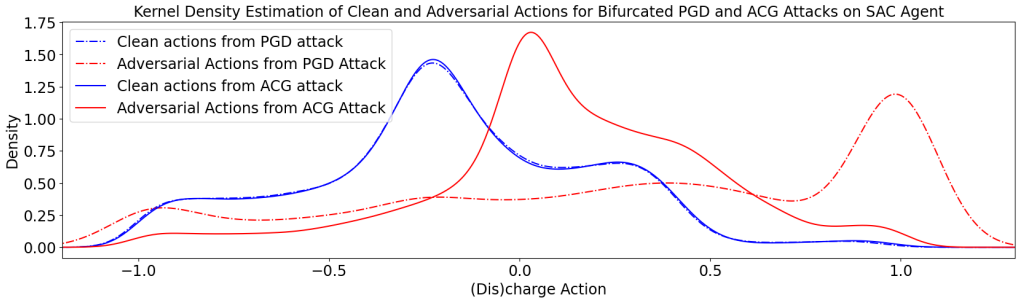


Figure 38: Kernel density estimation comparing the clean and adversarial actions of a SAC under bifurcated PGD and ACG attacks. In contrast to the PGD actions, those for ACG are concentrated near 0, and its distribution is not entirely dissimilar to the clean actions.

attacks on the SAC, and higher for ACG for direct attacks on the discrete PPO (ART's implementation of ACG is incompatible with the SAC, as discussed above). Table 10 shows that also PGD performed better that ACG, and that once again the discrete PPO is more robust then the SAC. Figure 39 shows that the metrics in Table 10 correspond to a higher adversarial regret for PGD. As with the SAC, the KDE of the actions taken during attack from Figure 41 shows that the action distribution of the ACG attack resembles the clean distribution more than the actions from PGD. Comparing the action distributions for these attacks between the discrete PPO and SAC shows how much more robust the PPO is compared to the SAC.

Experimentation with ART's ACG and Auto-PGD hyperparameters did not have a significant effect on their ASRs. Removing the feature mask, which is not implemented for the PGD attack, only improved ACG's ASR by ∼2%. An effect similar to ART's feature mask was achieved using a feature-specific ϵ and using $\epsilon = 0$ for temporal features, which had a negligible effect on adversarial regret. ACG was found to perform best in the dynamic distortion attack with 50 restarts and 20 iterations, each while the PGD attack's values were 1 and 100. Both ACG and APGD performed similarly with the former hyperparameters, but using only 1 restart with 100 iterations, ACG's ASR decreased by ∼2% and APGD's by ∼8%. The purpose of using APGD with the same hyperparameters as ACG was to test if the ACG algorithm was ill-suited to this task. Because ACG and APGD initially performed similarly, the ACG algorithm alone is not responsible for the difference in adversarial regret from this work's PGD implementation. The auto step-size updates and early stopping are also not present in the PGD attack, which always runs for the allotted number of iterations and the step-size is reduced at a fixed number of times. These differences in the two algorithms could explain the ASR discrepancy, but testing this hypothesis requires significantly modifying the ART library for these attacks. ART is otherwise more complex and feature-rich than the simple PGD implementation used is this work, but exhaustively testing how the ART implementation affects the performance

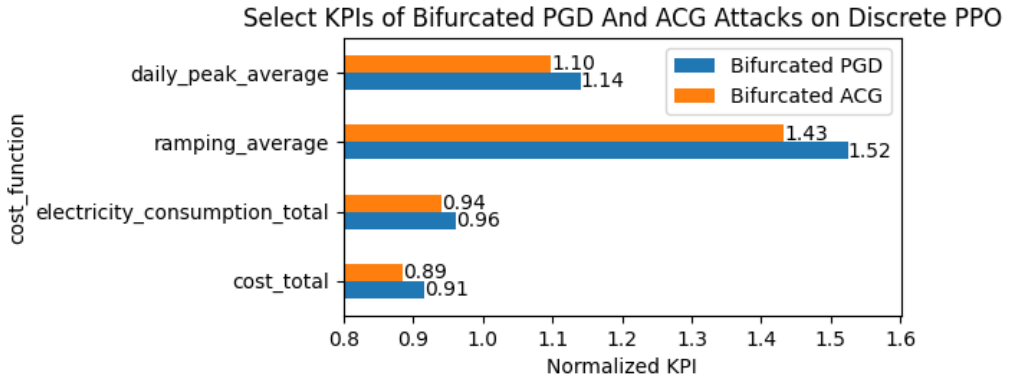


Figure 39: KPIs for select cost functions for the discrete PPO agent under bifurcation attacks using ACG and PGD. The KPIs are significantly lower than for the SAC, demonstrating the discrete PPO's robustness. As with the SAC, the adversarial regret from PGD has outperformed ACG.

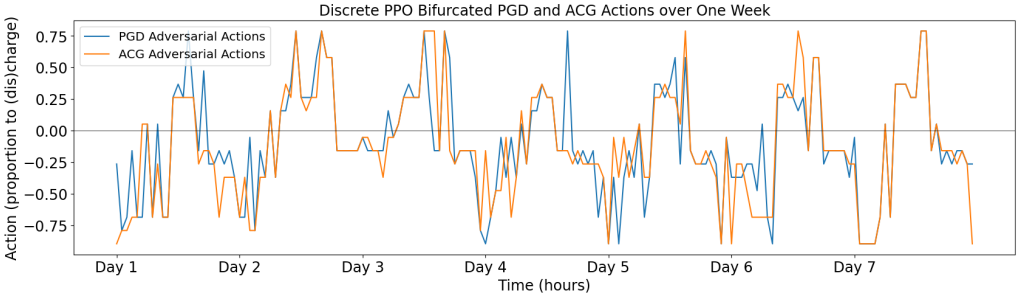


Figure 40: Adversarial actions from bifurcated ACG and PGD attacks on the discrete PPO. The PGD actions appear spikier, which visualizes the significantly larger ASR for that attack.

of bifurcated attacks is outside the scope of this work.

The PGD attack achieved a higher adversarial regret than ACG with a smaller computational budget for the SAC. This implies that the GDL function is not optimal, as maximizing it is not entirely proportional to adversarial regret, which is demonstrated by the inferior performance of the superior adversarial attack algorithm. Given that direct ACG outperforms PGD and that the work in [41] shows that ACG outperforms PGD, it seems that the GDL does not provide the optimal loss function to maximize power consumption and suggests a different bifurcation layer would perform better. In this case the ACG attack was allowed 10 times more steps and benefited from the auto stepsize adjustment proposed in [34], so it could be concluded that the ACG attack's inferior adversarial regret did not result from an inability to maximize the loss function. This suggests that another attack could be more effective with the same budget.

6.4.3 Toggle-Bifurcation Attack

Because the results for the untargeted bifurcation attack suggests that it is not the optimal approach, the following experiment attempts to improve it. The bifurcation method results in a network with two outputs, and using it with an adversarial attack attempts to reverse the victim's (dis)charge decision. Figure 34 shows how this results in alternating periods of charge and discharge, lasting several timesteps. Figure 25 shows the adversarial actions under the optimally targeted attack, where the (dis)charge action

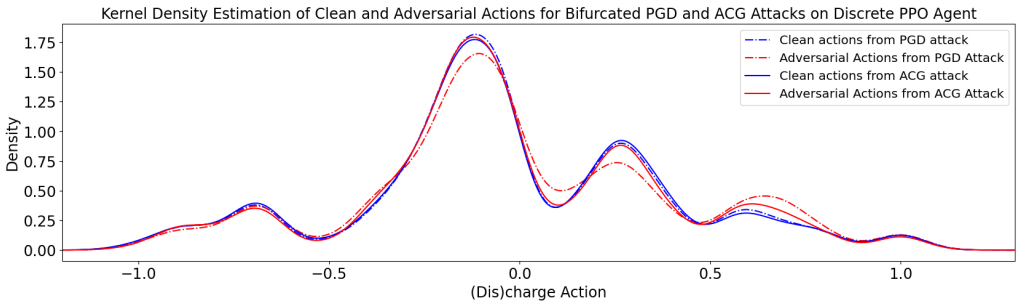


Figure 41: Kernel density estimation comparing the clean and adversarial actions of the discrete PPO under bifurcated PGD and ACG attacks. The deviations between the clean and adversarial distributions are relatively small, with ACG almost overlapping them. The larger deviation with PGD corresponds to its larger adversarial regret.

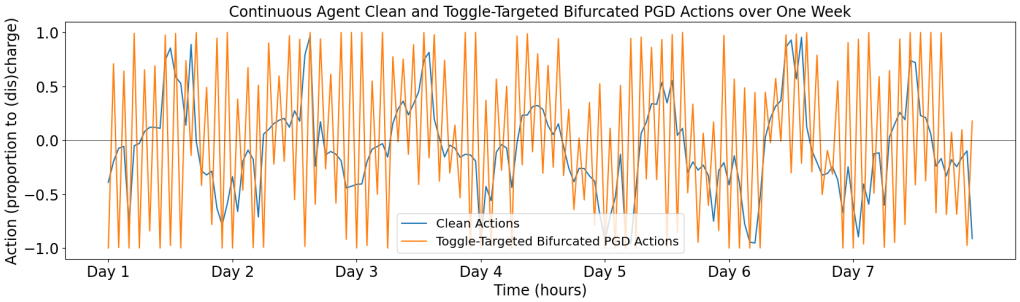


Figure 42: Clean and adversarial actions from a toggle-targeted bifurcated PGD attack for a SAC, over one week. Note how the action's sign changes more frequently for this attack than bifurcation alone in Figure 34.

reverses every timestep. Because the optimal attack targeted actions the victim agent was unlikely to take, at the extremes of its action space, it required an impractical budget. The bifurcation method addresses the issue of budget by targeting a group of actions, but does not reverse the action at every timestep, which would use more energy. So, instead of merely reversing the victim agent's action, leading to similar actions for multiple timesteps, the toggle attack alternates between targeting the first and second output of the bifurcated network. The goal is for the agent to take a charge action on one timestep and a discharge on the next e.g. charging on even timesteps and discharging on odd. Figure 42 show how the actions look during a successful attack.

The toggle bifurcation method is effective within a much smaller L_∞ constraint than the optimally targeted attack; it does not require training an adversarial policy so it does not require access to the victim agent's environment for training. However, as a targeted attack, it requires a larger adversarial budget than an untargeted bifurcated attack. When the budget is too small the latter attack will provide a larger adversarial regret as shown in Figure 43. This effect is further demonstrated for the SAC in Figure 48 as the adversarial budget is reduced in section 6: Detection.

6.4.4 Bifurcation Method Summary

The bifurcation method is a novel technique which enables a gradient-based attack to target a group of outputs in a single attack. It can be applied regardless of the number of outputs on the victim neural net, meaning it can be used on continuous and discrete action spaces in DRL, and regression or classification in deep learning. With this prop-

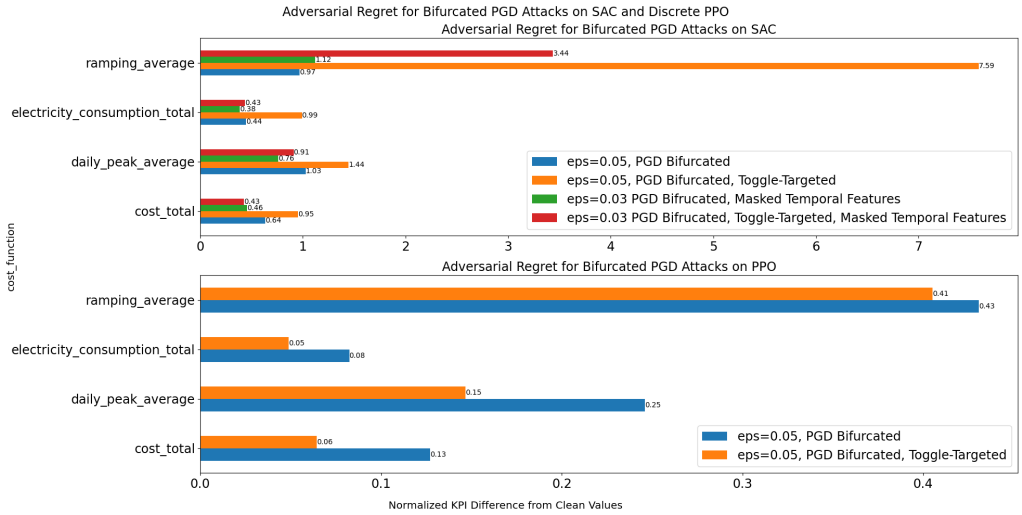


Figure 43: Adversarial regrets for bifurcated PGD attacks on Discrete PPO and SAC agents, which shows the effects of toggle-targeting. higher regrets indicate a better attack. The latter method is more effective for the SAC with the adversarial budgets tested. Because the ASR for the toggled attack on the PPO is only 36%, while it is 62% for bifurcation alone, the former performs worse. While the toggled attack can outperform a bifurcated attack, it requires a larger adversarial budget.

erty, a bifurcation layer can interface maximum-confidence classification attacks with regressors without altering the attack, which is valuable as this work did not find any adversarial attack libraries which explicitly supported attacks on regressors. The bifurcation method is also attack agnostic, and it will work with any attack using DL or CE loss. Though the former is more effective, the latter was included in every library found in this work's literature review.

The bifurcation method requires a smaller adversarial budget than conventional targeted attacks, and provides higher adversarial regrets than conventional untargeted attacks. Adversarial regrets are higher because the attack restricts which adversarial class or actions are chosen by the victim, which increases the effect of the attack. For smaller budgets, the attack exhibits higher ASRs than conventionally targeted attacks because it is successful for more than one adversarial output, and it is computationally more efficient than sequentially targeting each desired output or class. This new method sits between conventional untargeted and targeted attacks in terms of adversarial budget and regret, and represents a simple method for robustness testing for continuous control agents in DRL. The latter was entirely absent from the studies reviewed in this work, no study in the review used adversarial examples for a continuous action space.

The bifurcation method was validated for the SAC by comparing the results of bifurcated attacks to direct attacks with ACG and PGD. A custom PGD implementation was written to allow direct PGD attacks on an agent with a continuous action space. A SAC was trained in CityLearn and the attack using bifurcated PGD showed that the bifurcation method produces larger adversarial regrets.

This section shows that a discrete PPO is far more robust to adversarial attacks than a SAC, which had significantly higher adversarial regrets for all attacks. This was shown with bifurcated attacks, where the only difference was the bifurcation layer, and direct PGD attacks where the only difference was the loss function. These results suggest that DRL in critical infrastructure, including CPPS, should test the robustness of multiple algorithms and action spaces before choosing one for implementation. In this case, the

discrete PPO enhanced robustness with no loss of performance.

7 Defences

The goal of this phase is to demonstrate simple and model agnostic methods in which the threat of gradient-based adversarial attacks could be mitigated. The rationale is that defensive techniques are most useful when they can be applied to existing algorithms and pipelines and use open sources tools. Theoretical solutions are not useful if they are never implemented. This work will explore two types of defences: detection and robust training.

7.1 Detection Methodology

The purpose of detection in this work is to determine if the set of adversarial examples during one episode of an attack are plausible in aggregate. Determining if an individual sample or observation is adversarial is outside the scope of this work. Detection has dual purposes, both in defending against adversarial observation and proving if an attack is feasible. While the literature reviewed in this work suggests several novel methods for attacking DRL agents, none tested if their attacks were in fact stealthy. The aim of this section is to determine if gradient-based adversarial attacks can generate plausible adversarial observations.

To do so, we take a two pronged approach to ensure the plausibility of adversarial observations. First, the Maximum Mean Discrepancy (MMD) Gaussian kernel detector [67] is used to evaluate if the set of adversarial observations are plausible. Then, aggregated time-series analysis is used to analyze the variations in features over time. In this way both the construction of individual samples, and their relationship with the preceding sample is analyzed. We use model agnostic statistical techniques to identify outlying features of adversarial observations, and statistical testing to determine if the original and adversarial observations form distinct distributions.

This analysis is enabled by the real world measurements used to generate CityLearn's observations, because detection is tested in a realistic control setting. This section will determine if the adversarial attacks developed in previous sections are feasible. An attack is only considered feasible when the adversarial observations it generates are plausible, and the attack causes a significant adversarial regret. Unlike the image classification setting where adversarial attacks were first developed and are typically studied, observations in a Deep-Reinforcement Learning (DRL) environment have the following characteristics which enable detection:

1. Correlated features: the weather conditions are dependant on the time or day and are loosely periodic. e.g., temperatures and solar irradiance peak at mid day and drop at night. Adversarial observations which fail to follow these patterns could be detected simply by plotting them.

2. Observations in a DRL environment are a time series, which means there is a relationship between subsequent observations. So, adversarial observations must not only be individually plausible, but also be plausible given the previous observation. e.g., Weather features for a particular climate will vary within a particular range over time, so larger temperature variations between measurements could indicate FDI. The mean rate and range of inter observation changes for clean observations will be compared to those for adversarial observations, to test if they can be separated by some threshold value.

3. CityLearn has several modules which predict weather and electricity prices 6, 12, and 24 hours ahead. So, the change in prediction accuracy due to the perturbations and the final values of the features can be measured.

Based on this analysis of adversarial observations, the attack's adversarial budget will be reduced until the observations it produces appear plausible. Attacks in this work use an L_∞ boundary, which restricts the maximum perturbation for individual features. This size of ϵ directly affects the amount of distortion between original and adversarial observations, so reducing it can improve an attack's stealth.

MMD is a statistical method of identifying if two sets of samples are drawn from the same distribution. The MMD test as used in [43] will determine is the distribution of adversarial observations is plausible compared to the originals. Because it is made for sets rather than individual samples, it does not detect individual adversarial samples. However, it can be used as a metric to assess the significance of the adversary's distortions by comparing the clean and adversary observations generated during an episode. This represents the best case scenario for detecting the adversary's distortions, because the "correct answer" is used in the form of original observations. This would not be the case in a production system, as present observations can only be compared to historical data, and changing patterns in weather and electricity demand can also cause drift which increases MMD.

The purpose of this test is to suggest a threshold of distortion which is likely to be detectable with model agnostic statistical methods. If an attack exhibits a low MMD in this ideal setting, then it is unlikely to be detected statistically. In [20], the authors demonstrate that MMD will not detect strong minimum norm attacks, while [43] showed it is effective for weak attacks like FGM. These results make MMD valuable for determining if the SotA attacks are indeed plausible, because the test can be evaded by a strong adversary.

Using the same methodology as [43], the MMD is calculated using 10 000 bootstraps with a Gaussian kernel. The test provides the MMD value which quantifies the difference between two distributions, and a probability that both distributions are the same. Due to the correlated nature of CityLearn's time series observations, finding a clean MMD threshold is more complicated than simply randomly sampling a clean distribution as done in previous work. Through experimentation, the observations of a clean episode are divided into two representative samples, and the results of this test will be used as a baseline for clean data. If the MMD test results between the original and adversarial observations in an episode are outside the baseline range, the adversarial observations are considered implausible.

7.2 Detection Results

7.2.1 Time Series Feature Variation Analysis

While the purpose of MMD is determining if each individual adversarial observation is plausible, i.e. from the same distribution as clean observations, this does not suffice. Unlike Independent and Identically Distributed (IID) images where MMD was previously used, CityLearn produces time series data, so other methods are required to learn the effects of adversarial attacks on the relationships between samples. Even with insignificant MMDs, the variations between observations are significantly greater during adversarial attacks, and current attacks are not constrained based on the distance from the previous sample. Furthermore, adversarial samples do not conform to periodic observations, e.g., will show solar generation occurring at night. This specific behaviour could be avoided by only perturbing the solar generation feature during the day. Since many CityLearn features predict the values of others, perturbing them will affect the accuracy of their predictions, especially when future attacks don't consider prior perturbations, these changes to prediction features are self-inconsistent.

The attack has a delicate balance with the adversarial budget; a large budget leads to more successful attacks, but also larger distortions. The possible range of distortions must

Feature	**Maximum Perturbation for L_∞ ϵ**			
	$\epsilon = 0.07$	$\epsilon = 0.05$	$\epsilon = 0.03$	$\epsilon = 0.01$
Outdoor Dry Bulb Temperature (°C)	1.86	1.33	0.8	0.27
Outdoor Relative Humidity (%)	6.3	4.5	2.7	0.9
Diffuse Solar Irradiance (W/m^2)	71.2	50.9	30.5	10.2
Direct Solar Irradiance (W/m^2)	66.7	47.7	28.6	9.53
Carbon Intensity (kg_{CO_2}/kWh)	1.5×10^{-2}	1.1×10^{-2}	6.3×10^{-3}	2.1×10^{-3}
Non-Shiftable Load (kWh)	0.48	0.34	0.2	6.8×10^{-2}
Solar Generation (kWh)	61.6	44.0	26.4	8.8
Electrical Storage SoC (%)	7	5	3	1
Net Electricity Consumption (kWh)	62.7	44.8	26.9	9.0
Electricity Pricing ($/kWh)	2.3×10^{-2}	1.6×10^{-2}	1.0×10^{-2}	3.3×10^{-3}

Table 11: List of CityLearn features, and the variation introduced by different ϵ perturbation boundaries. Each feature is min-max normalized to [0,1], so the maximum perturbation is the product of ϵ and the feature's spread. This table omits prediction features, since their values are not significantly different from those listed.

be chosen such that distribution of adversarial observations is plausible, which is detected by MMD, and that the observations don't change drastically between two timesteps as they are also simple to detect. Table 11 lists the maximum distortion caused to each feature for a given ϵ value. Ideally the attacker can use an ϵ which will be lost in the noise of normal readings, or at least would not be apparent to a human. To that end, ϵ must be similar to the natural variation between samples, as listed in Table 12. From this data, it is apparent that $\epsilon = 0.03$ is less than or at least similar to the inter-sample variation for most features. Solar generation and Net Electricity consumption are outliers with variations, which is caused by improper min-max normalization in CityLearn. The maximum observed value for solar generation is 0.004, while this should be 1 if it is min-max normalized. In practice $\epsilon = 0.03$ is a 75% of it is spread, making a value like $\epsilon = 0.004 \times 0.03$ more appropriate. This normalization also skews net electricity consumption, which combine solar generation with the non-shiftable load and energy charged or discharged from storage.

Through reducing the adversarial budget and observing the adversarial regret, the parameters for a stealthy attack were selected. The adversarial budget was reduced to decrease the relative proportion of the adversarial mean absolute feature variation compared to the original. The absolute feature variations were analyzed because the changes in mean feature variations and values are negligible with this attack and unlikely to be detected. The absolute feature variation is the absolute difference between a feature and the values from the previous observation. Negative and positive changes can negate each other's effect on the mean, so absolute values are used to measure the magnitude of the variation.

An attack is used to assess the detectibility of adversarial observations. An attack on the Soft Actor Critic (SAC) was chosen as it quickly became apparent that the adversarial regret dropped significantly faster for the discrete Proximal Policy Optimization (PPO) for similar budgets. Curiously, masking a feature with Auto-Conjugate Gradient (ACG) in Adversarial Robustness Toolbox (ART) using their API as documented did not reduce the absolute variation in the associated feature. This is likely a bug, so instead, the Projected Gradient Descent (PGD) attack was used as it allows an ϵ to be assigned individually for each feature. ϵ was reduced for the solar generation and net electricity

Mean Normalized Inter-Observation Feature Variation		
Feature	**Mean Normalized Variation**	**Standard Error**
Outdoor Dry Bulb Temperature (°C)	2.6×10^{-2}	3.0×10^{-4}
Outdoor Relative Humidity (%)	4.9×10^{-2}	6.8×10^{-4}
Diffuse Solar Irradiance (W/m^2)	6.3×10^{-2}	8.9×10^{-4}
Direct Solar Irradiance (W/m^2)	0.07	1.3×10^{-3}
Carbon Intensity (kg_{CO_2}/kWh)	2.6×10^{-2}	2.7×10^{-4}
Non-Shiftable Load (kWh)	0.06	8.4×10^{-4}
Solar Generation (kWh)	2.6×10^{-4}	4.0×10^{-6}
Electrical Storage SoC (%)	8.2×10^{-2}	9.3×10^{-4}
Net Electricity Consumption (kWh)	6.4×10^{-4}	7.0×10^{-6}
Electricity Pricing ($/kWh)	7.2×10^{-2}	2.6×10^{-4}

Table 12: Mean Normalized Inter-Observation Feature Variation. This is the mean variation of normalized features between time-steps of a clean episode of CityLearn, and the Standard Error of the Mean (SEM). When ϵ is significantly larger than the mean variation for an adversarial attack, the perturbations may manifest with increased variation.

consumption features because they were not normalized in the same way as other features. This resulted in perturbations many times larger then the feature's value when a uniform ϵ was used. Using ART, features were also found outside the boundaries of [0,1], so the PGD attack was modified to keep features in this range. Thus, the stealthy PGD attack had the following characteristics, and its perturbations are visualized in Figure 44:

1. PGD was used with the bifurcation method on a SAC victim agent
2. $\epsilon = 0.03$ for all features except:
   (a) Solar generation and all temporal features had $\epsilon = 0$, and
   (b) Net electricity consumption has $\epsilon = 4.8 \times 10^{-4}$, which is the product of its spread and the ϵ for all other features
3. Each feature was constrained between [0,1]

This bifurcation method improves the adversarial regret by 50% compared to a direct PGD attack with the same parameters. Constraining the adversarial features to [0,1] reduced the adversarial regret by over one third, which was larger than the reduction for masking or scaling the solar generation and net electricity consumption features. However, the lack of such constraints makes adversarial observations obvious, particularly when a value like solar irradiance or generation becomes negative. Figure 48 shows the relationship between adversarial regrets and budget.

Table 13 shows how a stealthy PGD attack affects the absolute variation of each feature categories between timesteps. Despite the adversarial observations being very close to the originals used to craft them, the difference between two adversarial observations is greater than the originals. The stealthy PGD attack increases the adversarial regret for power consumption compared to a direct attack by 50%, but is limited by the adversarial

Relative Absolute Feature Variation for the Stealthy PGD Attack		
Normalized Feature	**Difference in Mean Absolute Variation**	**Proportion of Clean Variation**
Outdoor Dry Bulb Temperature (°C)	0.34	1.49
Outdoor Relative Humidity (%)	0.88	1.2
Diffuse Solar Irradiance (W/m^2)	4.67	1.09
Direct Solar Irradiance (W/m^2)	6.83	1.09
Carbon Intensity (kg_{CO_2}/kWh)	2.6×10^{-3}	1.47
Non-Shiftable Load (kWh)	3.7×10^{-2}	1.09
Solar Generation (kWh)	0	1
Electrical Storage SoC (%)	4.5×10^{-3}	1.07
Net Electricity Consumption (kWh)	0.29	1.5
Electricity Pricing ($/kWh)	1.5×10^{-2}	1.2

Table 13: Feature Variations during bifurcated PGD attack on a SAC, where neither the temporal or solar generation features were perturbed, the ϵ for net electricity consumption was reduced, and all observations were projected to [0,1]. This attack was chosen for comparison because of issues with ART's feature mask. Also, the SAC was chosen because an attack with these restrictions has a negligible adversarial regret for the discrete PPO. This is the most powerful attack given the perturbation restraints. The variations in SoC, carbon intensity, and net electricity consumption are also affected by the agent's suboptimal actions, not the perturbations alone. Absolute variation was used because the mean variation is at most four orders of magnitude smaller, making it negligible.

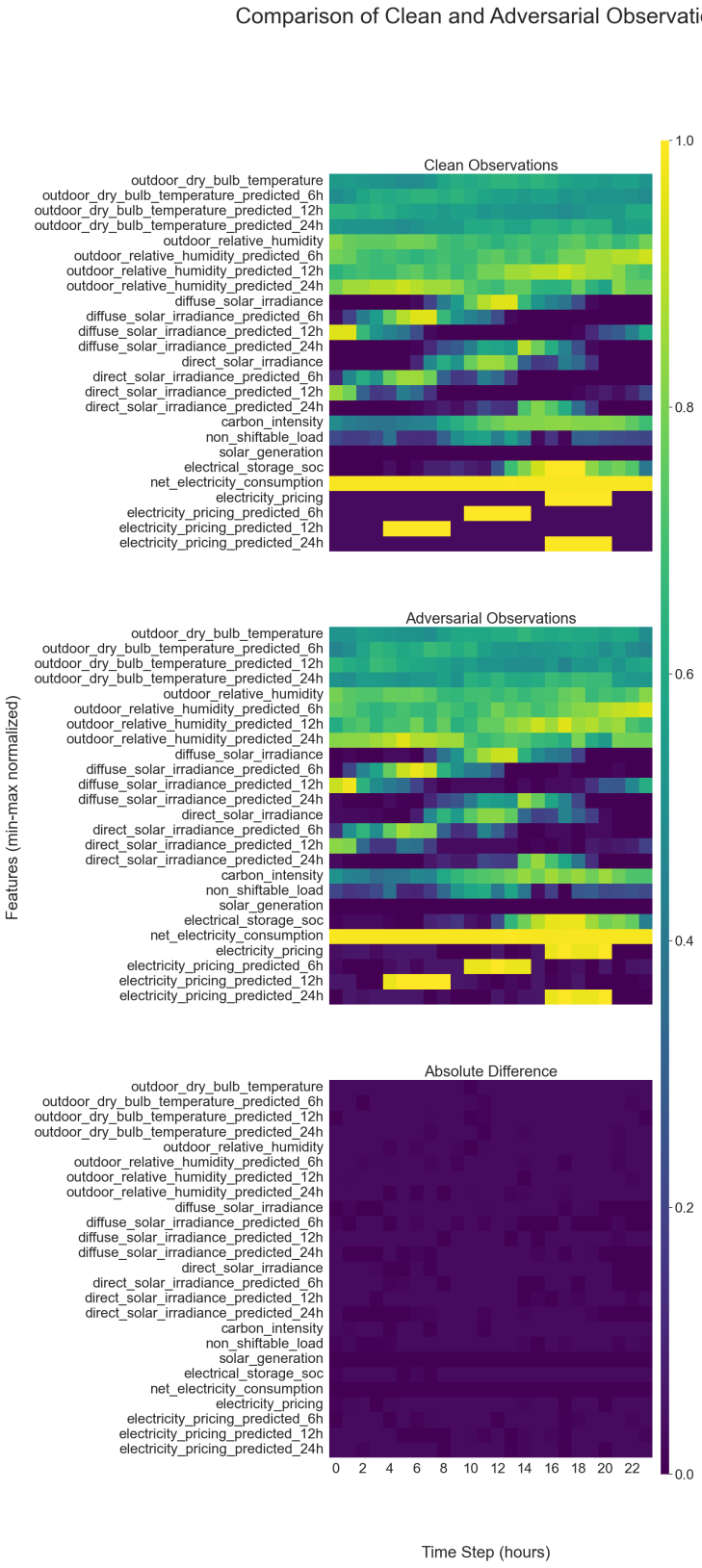


Figure 44: Observation heatmap for Bifurcation PGD Attack, with $\epsilon = 0.03$, Masked Temporal and Solar Generation Features, and scaled ϵ for Net Electricity Consumption. From top to bottom are the clean then adversarial observations, followed by the difference between them.

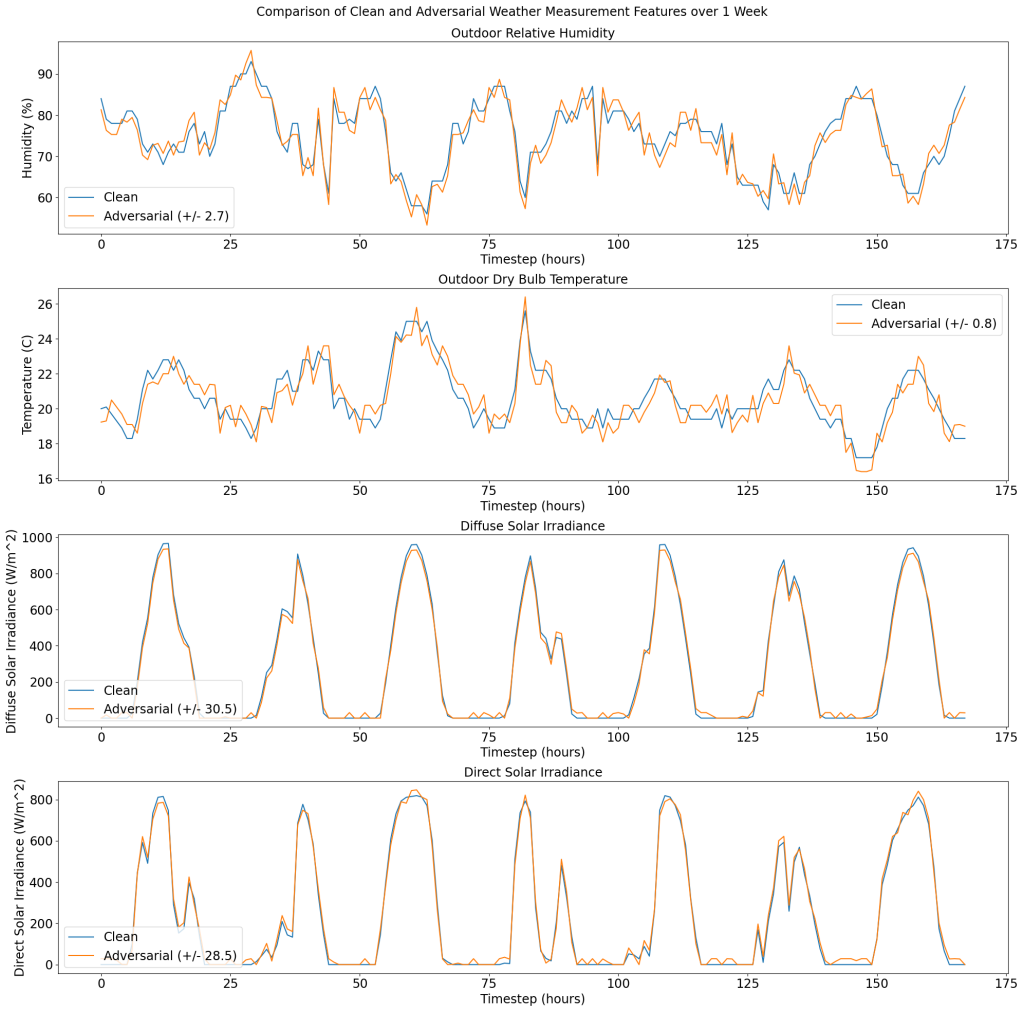


Figure 45: Comparison of clean and adversarial periodic weather features over one week. Adversarial features were generated with the stealthy PGD attack. Because these features depend on the time, perturbations may be evident, e.g., temperature or solar irradiance increasing during the night. Unlike carbon intensity or net electrical consumption, the plotted features are not affected by the agent's actions.

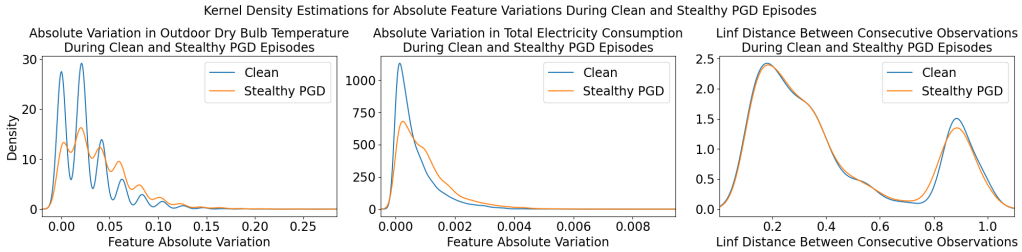


Figure 46: KDE of high variation features, and the L_∞ norm between observations. Both temperature and total electricity consumption had mean absolute variations 1.5 times more for the stealthy PGD attack than clean episodes. The left two plot show that while the distribution of variations is different for clean and adversarial episodes, they overlap and cannot be separated in this dimension. The L_∞ norm plot shows that in terms of distances between observations, those with adversarial observations are not outliers.

budget as the regret scale with ϵ. Further decreasing the budget would nearly eliminate the advantages from the bifurcation method. Because the budget cannot be decreased further, this section will analyze if the stealthy PGD attack is easily identified.

Figure 45 plots periodic features, which the adversarial observations manage to follow with a reasonable fidelity. These are weather related features which vary throughout the day and are correlated with the hour. Of these, the adversarial temperature readings had the highest variation. While the adversarial readings appear noisier, the clean ones also exhibit a similar shape at times. Without knowing how the clean plot looks it would be difficult to identify the adversarial perturbations. The adversarial measurements for solar irradiance closely follow their clean counterparts, though spikes are visible at night. To avoid these the attack should assign $\epsilon = 0$ to such features when their values are 0. To understand if features with relatively high variations can distinguish adversarial observations from originals, Figure 46 plots their distributions. While it is clear that each plot shows two distinct distributions, they entirely overlap. The adversarial features have single digit outliers. 97.6% of absolute inter-adversarial observation difference are within the original range, and only 15.3% have more than one outlying feature. This indicates that the distributions are difficult to separate, though it may be possible in higher dimensional space using Machine Learning (ML) with each features' absolute variation. Variations of the sizes introduced by the stealthy PGD attack would be non-trivial to detect, but the changes in observations are distributed differently from the originals. Further analysis on the difference between the original and adversarial distributions will be conducted in a later section.

CityLearn contains features which predict the outdoor temperature and humidity, direct and diffuse solar irradiance, and electricity price, 6, 12, and 24 hours/timesteps into the future. The accuracy of these predictions is affected both by the perturbation applied to the prediction but also the future measurement. Since adversarial attacks do not consider previous perturbations when generating adversarial examples, the distortion can be magnified. After analyzing the change in prediction accuracy over the course of attack episodes compared to a clean episode, the maximum reduction in accuracy is equal to 2ϵ. This is intuitive because ϵ could be subtracted from the prediction then added to the measurement several timesteps later. The mean reduction in accuracy tends to be slightly less than ϵ, because iterative attacks may change a feature by less than ϵ and for every time an ϵ is added to a prediction and subtraction from the measurement (or vice versa) it is equally likely to be added or subtracted from both. Table 14 shows the changes in prediction error for the stealthy PGD attack. While prediction accuracy can be used to detect adversarial attacks, an appropriately selected ϵ can make it imperceptible. A significant decrease in prediction accuracy accompanied by a decrease in performance could indicate adversarial observation perturbations.

7.2.2 MMD Gaussian Kernel Detector

The baseline for normal MMD was determined using the data from a clean evaluation episode. Due to the seasonality in CityLearn observations, where both weather and usage feature vary by season, MMD will indicate that different fractions of observations from a clean episode are not drawn from the same distribution, e.g., winter observations follow a different distribution than summer. Because the MMD is a comparison of two distributions, this clean data needed to be split into two representative samples. Multiple episodes could not be used because the majority of CityLearn's features are deterministic, and the others are influenced by the deterministic policy of the agent, meaning there is no significant variation between episodes. Due to the fact that an episode spans the course of a year, there are seasonal changes in the weather features, electricity use on weekdays is distinct from weekends, and both solar and temperatures vary periodically by hour. For these reasons, random separation is unlikely to generate two representative

Prediction Error for Stealthy PGD Attack		
Normalized Feature	**Mean Error Increase**	**Max Error Increase**
Outdoor Dry Bulb Temperature (°C)	0.75	1.6
Outdoor Relative Humidity (%)	2.6	5.4
Diffuse Solar Irradiance (W/m^2)	19.9	61.0
Direct Solar Irradiance (W/m^2)	18.0	57.2
Electricity Pricing ($/kWh)	6.5×10^{-3}	0.02

Table 14: Mean prediction error grouped by feature category.

distributions. Comparing consecutive months, weeks, or days often results in excessively large MMDs with low probability values, and so did splitting the data into even and odd time steps (hours). The MMD test indicated that the samples were from two distinct distributions, with p values below 0.05. This means that the previous methods of splitting the data were not producing representative distributions. These separation methods biased the data. Instead, randomly splitting the samples for each day equally produced two representative sets, which MMD indicates are of the same distribution with large p values. This entails assigning 12 random indices from each day to one distribution, and those remaining to another distribution. This ensures that each distribution has an identical number of samples from each day, and randomly sampling each day reduced bias in the selection. Repeating this for 100 pairs of distributions provides a distribution of MMDs and probability values P for comparing clean and adversarial observations. For an attack to pass the test, the MMD must be within or below this baseline distribution of MMDs, and within the distribution of probability values. For the latter, we can calculate the percentile of a probability value in the baseline distribution. The results of this analysis are shown in Table 15.

The results of the MMD test for the SAC are shown in Figure 47, and the associated adversarial regrets in Figure 48. With the stealthy PGD attack exhibiting non-negligible power consumption increase of nearly 10%, while the MMD is lower than any baseline value and the p-value is high, the SAC is not robust to stealthy attacks. Note how the adversarial regret decreases significantly when the adversarial budget is restricted. This effect is even more pronounced in the results of the discrete PPO, with Figure 49 showing the MMD results and Figure 50 displaying the adversarial regrets. The only attack which both have an MMD lower than the baseline and high p-values only increased power consumption by 3%. However, the ACG attack using ART was detectable for having features improperly masked and outside the range of [0,1] in the previous section. These results show that the discrete PPO is robust to stealthy attacks.

Of the attacks which pass the MMD, only the stealthy PGD attack had both a significant adversarial regret and had mean absolute feature variations less than twice the original. The ACG attack as implemented with ART incorrectly masked features and does not allow ϵ to be set for individual features, so these attacks were obvious from their absolute feature variations. The analysis so far suggests that the stealthy PGD attack is unlikely to be detected. The MMD test assigned a value of 3.5×10^{-3} and a probability value of 0% when comparing the adversarial and original absolute feature variations. This suggests that attacks with such limited budgets produced convincing adversarial observations, but could be identified by comparing them to earlier observations in a time series. SotA adversarial attacks are not designed to be consistent between samples.

MMD Detector results for different attacks on different victims			
Attack	**MMD**	**p-Value**	**p-Value Percentile**
Discrete PPO with Masked Temporal Features			
ACG with Dynamic Distortion	1.5×10^{-4}	1.0	1.0
Untargeted BB	6.0×10^{-4}	0.0	0.0
Untargeted BB (excluding Electrical Storage SoC)	1.0×10^{-4}	1.0	1.0
Bifurcated ACG $\epsilon = 0.05$	9.1×10^{-4}	0.02	2.0
Bifurcated ACG $\epsilon = 0.1$	1.1×10^{-3}	0.0	0.0
Discrete PPO with Masked Temporal Features, Solar Generation, and Net Electricity Consumption			
Bifurcated PGD $\epsilon = 0.03$	1.7×10^{-4}	0.98	88.0
Bifurcated ACG $\epsilon = 0.03$	7.8×10^{-4}	0.98	88.0
SAC			
Bifurcated PGD $\epsilon = 0.05$	5.4×10^{-4}	1.0×10^{-5}	0.0
SAC with Masked Temporal Features			
Bifurcated PGD $\epsilon = 0.05$	3.8×10^{-4}	4.7×10^{-3}	0.0
Bifurcated PGD $\epsilon = 0.03$	1.7×10^{-4}	1.0	88.3
Bifurcated Toggle PGD $\epsilon = 0.03$	1.7×10^{-4}	1.0	87.4
SAC with Masked Temporal Features and Solar Generation, and Scaled ϵ for Net Electricity Consumption			
Bifurcated PGD $\epsilon = 0.03$	2.3×10^{-4}	0.39	18.4
Bifurcated Toggle PGD $\epsilon = 0.03$	2.1×10^{-4}	0.57	25.2

Table 15: Results of the MMD Gaussian Kernel Detector, which provides the MMD as a measure of the similarity between two sets of samples, and the probability that both are drawn from the same distribution. The Percentile of the p-values were calculated from the baseline distribution for the observations of a clean episode. Note that the MMD and probability values are not directly correlated.

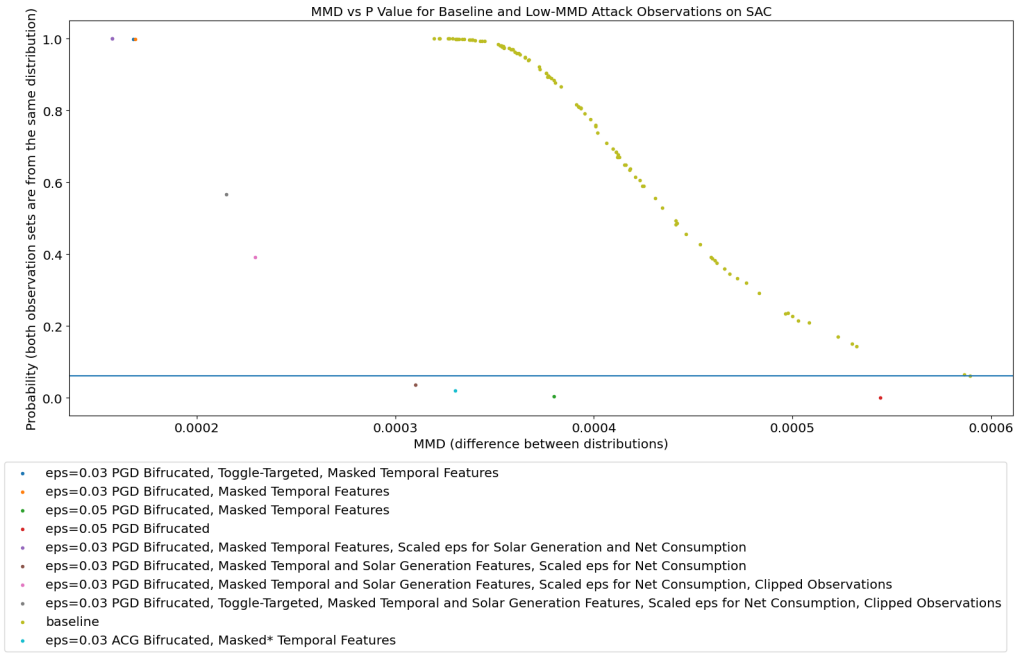


Figure 47: Scatter plot of MMD and Probability values from the MMD test for observations from clean and various attack episodes. Attacks which had relatively small MMDs and adversarial regrets were selected. The baseline points were generated from 100 random splits of clean observations for comparison. Sets of observations with MMDs smaller than this distribution and comparable p-values are considered plausible and difficult to detect. Note that attacks with higher budgets tend to be easier to detect. The blue line is the minimum p-value for the baseline distribution of 6%.

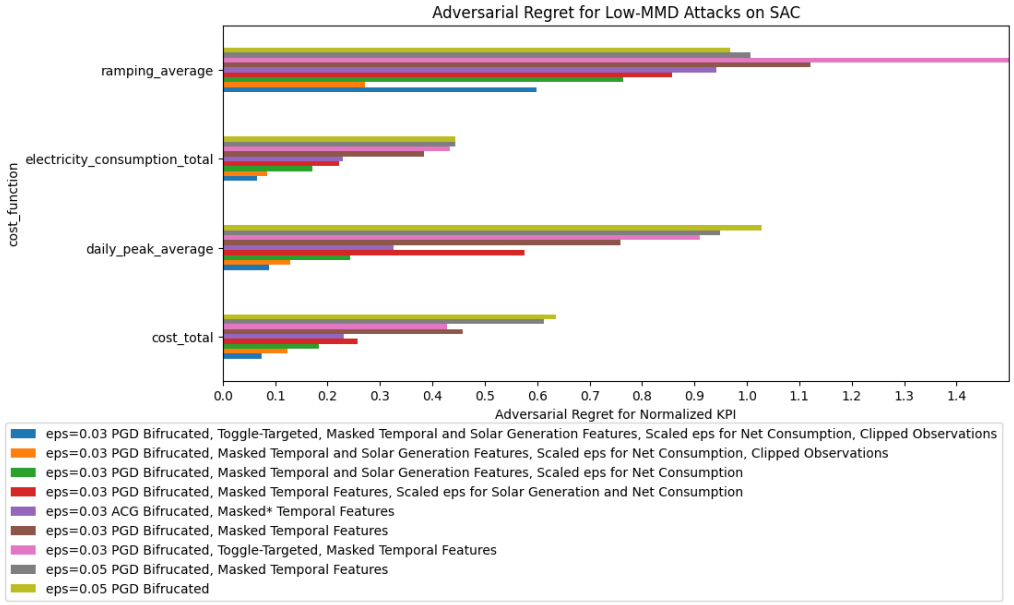


Figure 48: Plot of adversarial regrets for the low MMD attacks from Figure 47. This is the difference between the clean KPIs and those for the attack. Note that as the adversarial budget is decreased in terms of the perturbation size and available features, so does the adversarial regret. The ramping value for the attack in pink is 3.5, but was truncated so that the rest of the data becomes more visible.

However, there are caveats:

1. The MMD test was run under perfect conditions, where the original data was known. A defender could only compare the current distribution to historical data, and the current and past distributions can change for benign reasons. E.g., changing power consumption habits from the proliferation of EVs, and shifts in local weather patterns.

2. The MMD test only compares two distributions, and Figure 46 shows that this distribution for high variation adversarial features overlap with the originals. This makes identifying individual or small set of adversarial observations a promising subject for future research. Furthermore, a Load Altering Attack (LAA) may not require many altered observations to cause a power consumption spike which affects the grid. Because CityLearn does not model loads on the wider power grid, exploring the number of adversarial observations required for such an LAA is out of scope. Thus, the defender must detect a small number of adversarial observations for detection to be a viable defense.

These results show that adversarial observations can be detected in aggregate from a time series. There are clear statistical differences in the absolute feature variations between adversarial observations. Exploiting this for an effective defence requires detecting a small number of individual observations, and a high confidence model of how clean observations should behave. High confidence is critical because a detection system with a large false positive rate risks of being ignored. As a white box attack requires a well resourced adversary, the defender must be similarly well resourced to detect the adversarial perturbations.

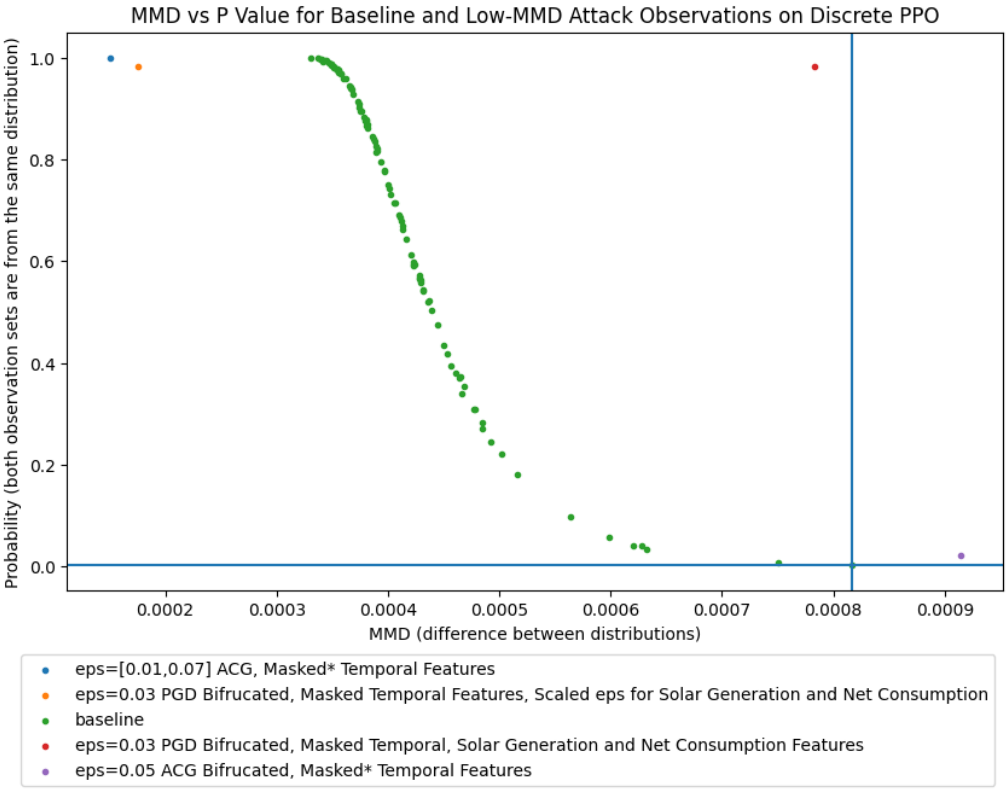


Figure 49: Scatter plot of MMD and Probability values from the MMD test for observations from clean and various attack episodes. Attacks which had relatively small MMDs and adversarial regrets were selected. The baseline points were generated from 100 random splits of clean observations for comparison. Sets of observations with MMDs smaller than this distribution and comparable p-values are considered plausible and difficult to detect. The blue lines indicate the maximum MMD 8.2×10^{-4}, and minimum p-value 3.0×10^{-3} for the baseline distribution.

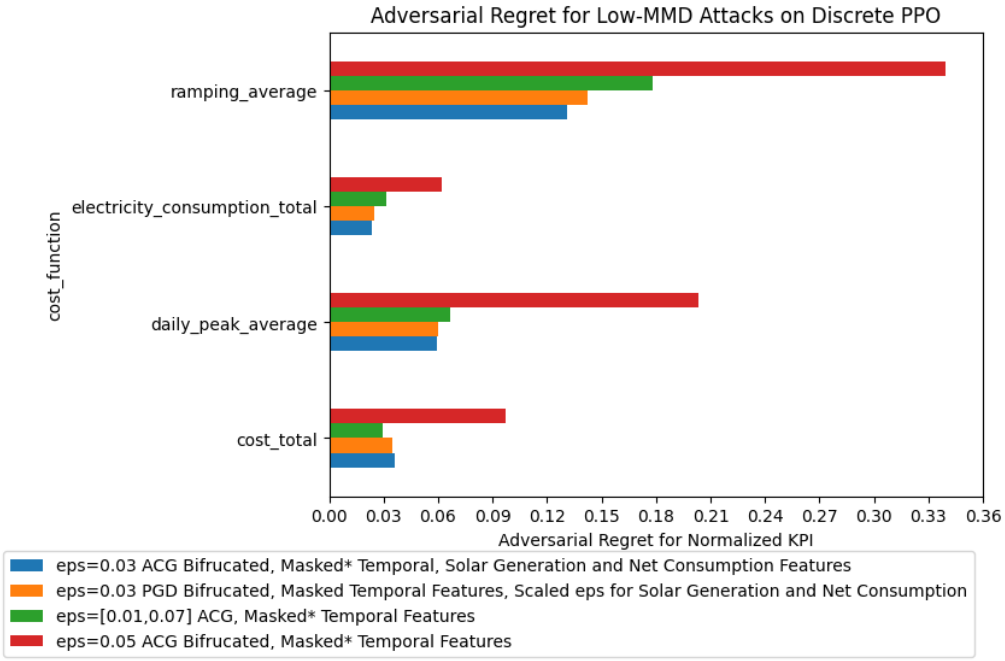


Figure 50: Plot of adversarial regrets for the low MMD attacks from Figure 49. This is the difference between the clean KPIs and those for the attack. Note that as the adversarial budget is decreased in terms of the perturbation size and available features, so does the adversarial regret.

7.3 Robust Training Methodology

Two methods of agent robustness are explored tested in this section: Alternating Training with Learned Adversary (ATLA) [60], and binning continuous action spaces.

7.3.1 Alternating Training with Learned Adversary

To test the defence of the ATLA, which is a training method which teaches an agent a robust policy (π). It was chosen for testing this work because it can be applied to any DRL algorithm, potentially making it an accessible method for defending against adversarial attacks. This work's literature review found no evaluation of resistance to adversarial attacks targeting a DRL agent's function approximator, so that is the goal of this experiment. Preparations for ATLA requires that CityLearn is modified so that the observations returned by the environment can be modified by the adversary during training, unlike the previous evasion attacks that happen during inference.

The environments for the adversary and victim use identical CityLearn parameters but have different action spaces and rewards. The adversary receives a clean observation from CityLearn and its action is adding a perturbation, then the victim's static policy in the adversary's environment selects an action based on the adversarial observation. Conventionally, the adversary's action space is constrained by a function $A_{adv} \in B(s)$, where s is the current state. To avoid violating the Markov property and be consistent with the L_∞ constraint from previous attacks, $B(s)$ is a static boundary for each feature. This boundary will be generated using typical variations between observations during a clean episode. $B(s)$ is a training hyperparamenter, and must be selected such that the adversary is capable of a significant adversarial regret without preventing the agent from learning.

The Learned Adversary (LA) is the first step in implementing an ATLA training environment. A successful attack validates both the adversary's environment and later the success of ATLA training. Training a learned adversary requires modifying the CityLearn environment. Rather than providing observations for an agent to choose an action, the adversary is an intermediary adding perturbations to the observations. Training the LA involves adding the victim agent to the environment, so the LA can add a perturbation to every observation and receive a reward based on the agent's action. Because the agent's policy is static, it is effectively part of the environment. Its job is to map the perturbed observation provided by the adversary to an action which is understood by CityLearn. In this way the adversary learned which observations lead to the worst outcome. As discussed in the background, this is an SA-MDP because the agent's policy is static during training. The action space for the LA is a subset of the observation space, which changes the original observation from the CityLearn environment. This means the LA has 31 features and up to 31 continuous actions, with each of the LA's actions corresponding to one of the agent's features. The boundary for each action could match each feature or be restricted to limit the perturbation size, and the number of actions can be reduced so some features are not perturbed.

With the adversarial environment implemented, a LA can be trained for an LA attack. This attack can be used as a black box attack, as the adversary is trained without access to the victim's parameters [60]. These are the objectives of studying the LA attack before proceeding to ATLA:

1. Demonstrating that it can produce a significant adversarial regret. The reduction of regret following ATLA is the metric for its success.
2. Assessing the $B(s)$ required for a successful attack, so it can be used in ATLA.
3. Comparing the LA as a standalone attack to gradient-based attacks, in terms of adversarial budget and regret. The LA would be preferable to white box attacks if it can increase the adversarial regret for the same perturbations size, measured as the L_∞ distance between the adversarial and original observations.

To assess the minimum adversarial budget or $B(s)$ required in order to achieve objectives 2 and 3, the LA has the dual objectives of minimizing its perturbations and the victim agent's rewards, given the maximum budget $B(s) = \Omega_{agent}$ using the notation for a Partially Observable Markov Decision Process (POMDP) from Table 4. Because the adversarial observation is the LA's action, its task is to produce the smallest reward with the smallest action. This requires a reward function affected by both the victim's reward and the size of the perturbations. The norm-scaled reward (38) calculates the adversarial reward as the negative of the agent's reward with a penalty for the distance between the original and adversarial observations. The smaller the distance, the larger the proportion of the reward received by the LA. The exponent b adjusts the importance of this distance penalty.

$$\tilde{r} = -r(1 - \frac{||o - \tilde{o}||_\infty}{||\max(\Omega) - \min(\Omega)||_\infty})^b \tag{38}$$

Both the rewards and L_∞ distances are measured during training, and this experiment is successful when the LA reward and distances converge. With the adversary's environment validated and action space determined, the last step is implementing the victim agent's environment. It must be configured to perturb every CityLearn observation according to the adversary's policy before it is provided to the agent. After these preparations ATLA can be conducted. The major hyperparameters for ATLA are:

1. Number of pre-training episodes for the agent. The authors in [68] found that pre-training improve the outcomes for ATLA in CityLearn. Since earlier experiments showed reward curves flattening after 50 epsiodes, this will be the initial value.

2. Number of episodes per alternation. The starting value will be 20 as per [68].

3. Number of alternations. The initial value will be 10, for 200 episodes of ATLA with 50 for pre-training. This will provide enough training to demonstrate if the agent is learning.

7.3.2 Robust Action Space

In addition to robust training with ATLA, the discrepancy in robustness between the SAC and discrete PPO will be investigated. The discrete action space seemed to make the discrete PPO more robust than the soft actor critic in Chapter 6. Because there are two variables between these agents, the learning algorithm and action space, they will be isolated by evaluating the robustness of a continuous action space PPO. The adversarial regret of all three agents for the stealthy attack defined in the detection section will be compared.

7.4 ATLA Results

This subsection presents the results for training agents using ATLA, including developing an environment for ATLA, followed by a comparison of robustness between a conventionally trained PPO and a PPO using ATLA.

7.4.1 Preparations and Training

Training a SAC LA with the norm-scale reward was ineffective for choosing $B(s)$ and showed no potential as a stealthy black box attack. Because the perturbation space is continuous and following the methodology in [60], a SAC was selected as the algorithm for the LA, using a Multi-Layer Perceptron (MLP) ANN with two hidden layers of 256 perceptrons. The LA had 31 actions with values between 0 and 1, which each replaced a feature in the original observation. The victim was the discrete PPO. Training was conducted for up to 600 episodes with the exponent b up to 3. No LA converged to a small distortion size as shown in Figure 52, though, they continuously were able to improve their rewards as Figure 51 shows. The perturbation sizes were an order of magnitude larger than those required for the bifurcation attack. This suggested that while a SAC is capable of producing distortions that minimize the victim's reward, learning to minimize these distortions is a more difficult task to grasp. Even the smallest distortions were twice the size of the optimal adversarial BB distortions, which were far too large for stealth.

Instead the focus became developing an attack that could be used to train a robust agent using the ATLA framework. Without success in training the adversary agent to constrain its perturbation size, a plausible $B(s)$ was selected based on the difference between sequential observations in a clean evaluation episode in Table 16.

Conducting ATLA is not directly possible with Stable Baselines 3 (SB3) agents, because training is not complete after each alternation. An SB3 agent will train until its budget of timesteps is exhausted, and further training will be unsuccessful unless the agent is reset. SB3 algorithms are not designed to learn once their training is complete, so if training continues without resetting training variables the agent may behave randomly and will not update its policy. While this can be done when loading an agent from storage, SB3's API does not provide another means. Instead, the most effective method was providing agents the maximum amount of timesteps they would train over the course of ATLA, and using a custom callback function to end training after each alternation but before the maximum timestep was reached. In this manner, ATLA training becomes uninterrupted from the agents perspective, rather than being a series of training and resets, which allows SB3 agents to train as intended.

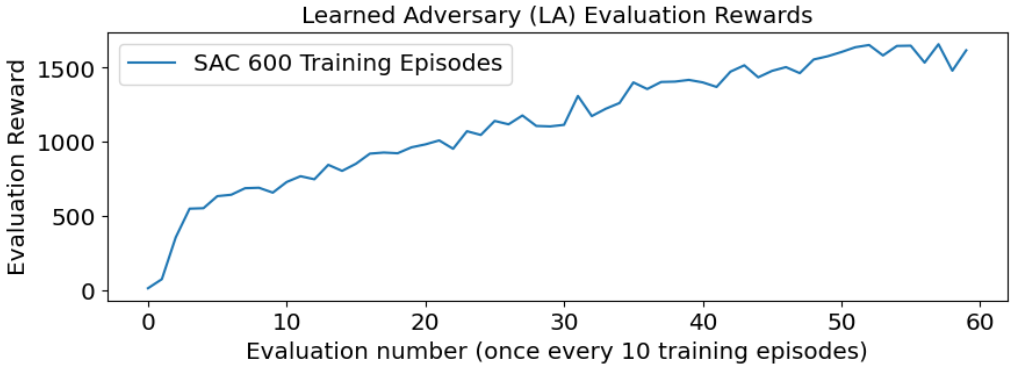


Figure 51: Evaluation rewards during LA training with the norm-scale reward and $b = 3$. An evaluation occurred every 10 episodes, and the scores appear to flatten by 600 episodes.

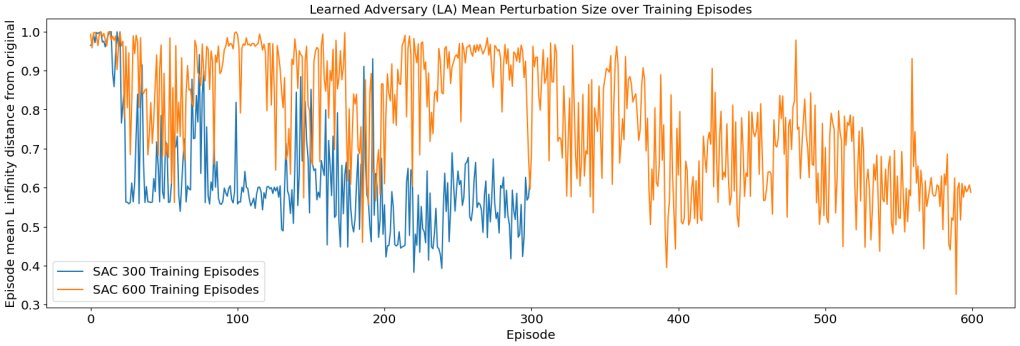


Figure 52: Mean L_∞ between original and adversarial observation for each training episode $\frac{1}{T}\sum_{t=0}^{T}||o - \tilde{o}||_\infty$, for the LA with the norm-scale reward and the exponent $b = 3$. The perturbation size did not converge over 300 or 600 episodes, and is 10 times higher than the ϵ used for untargeted and bifurcated attacks.

ATLA Adversarial budget $B(s)$ by Feature Category		
Feature	Max-Mean Intersample Variation	Reduced
Outdoor Dry Bulb Temperature	0.26	0.16
Outdoor Relative Humidity	0.62	0.36
Diffuse Solar Irradiance	0.53	0.33
Direct Solar Irradiance	0.75	0.45
Carbon Intensity	0.28	0.17
Non-Shiftable Load	0.61	0.37
Solar Generation	2.5×10^{-3}	1.5×10^{-3}
Electrical Storage SoC	0.47	0.32
Net Electricity Consumption	5.2×10^{-3}	3.3×10^{-3}
Electricity Pricing	0.90	0.52

Table 16: The perturbation space for each category of features in ATLA. This is the absolute amount the adversary can change each feature or the adversarial budget. Agents were most successful with the reduced $B(s)$, with the other listed being too large for the agent to learn a useful policy. Agents trained with a larger $B(s)$ were stuck in a local minima, and learned to never charge or discharge the battery. While the adversary cannot make this policy any worse, it also removes any benefit of the agent.

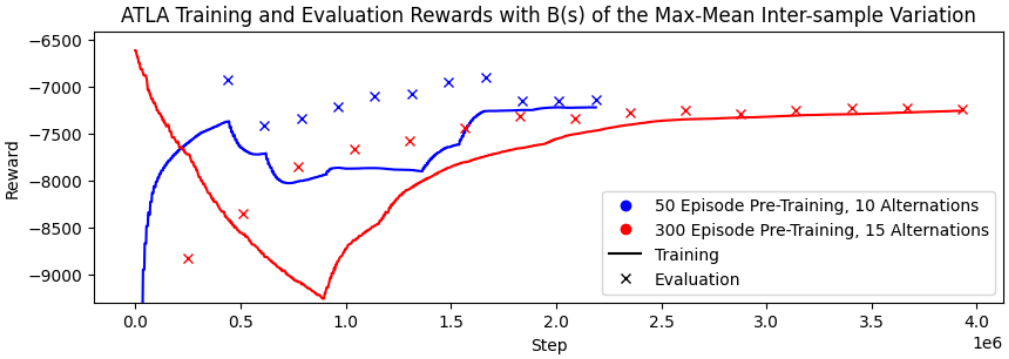


Figure 53: Line plot of early training and evaluations rewards, using the max-mean variation $B(s)$. The LA is a SAC and the agent a discrete PPO. The agent is evaluated after its turn training (denoted with an 'x'). With only 20 pre-training episodes, the agent is able to adapt to the adversary and regain its initial performance, but stops improving from there. The adversary has a much larger effect on the saved agent which had trained for 300 episodes, the minimum reward of ∼-9200 is significantly lower than for the first agent at ∼-8100. Its policy converged to a local minima, which does not charge and discharge stored electricity, and does not improve over 4 alternations.

ATLA was successfully conducted with a discrete PPO agent and SAC LA. The first ATLA agent was trained with the LA having a $B(s)$ equal to the difference between the mean and maximum change between clean observations over a clean episode (max-mean intersample variation from Table 16). Both 50 and 300 pre-training episodes were tested, since it was not certain if longer pre-training slowed learning with the adversary. For 10 alternations of 20 episodes, it seemed that increasing the pre-training time reduced performance, since the latter agent did not converge. Instead its reward curve formed a U-shape. With the number of alternations increased for 10 to 15, the agent converged to a policy of inaction. This is minimally exploitable since a powerful adversary can achieve much worse outcomes, but also useless during normal operations. Figure 53 shows the training curves which demonstrate these effects. This suggests that the adversary's action space should be restricted, and roughly (but not exactly) halving it improved on the first ATLA agent. The reduced $B(s)$ is also shown in Table 16. Halving the LA action space again produced too weak an attack to reduce the agent's reward, and the training and evaluation curves were flat. The training curves for a reduced adversarial budget are shown in Figure 54.

While [68] found alternating every 20 episodes to be effective, that study used a SAC for both the agent and LA. Furthermore the PPO agent did not seem to converge during each alternation in Figure 54, with training reward curves so steep they were nearly linear. With 100 episodes for the PPO agent, these curves began to flatten before each alternation, showing that the PPO needs more time to converge then the SAC LA. To ensure that both the agent and LA had a trained opponent, ATLA began with the LA so it would train against the pre-trained agent. Pre-training was always used as previous work found it most effective [68]. However, beginning ATLA with the agent training against a random LA policy actually improved performance and lead to the best ATLA agent. Random perturbations helped the agent adapt to optimal perturbations later on. These effects are shown with the training curves in figure 55. This suggests that augmenting the environment with random noise could improve training performance in CityLearn. CityLearn is deterministic for an agent with a deterministic policy, which suggests future work could improve agent training with random data augmentation.

Figure 56 compiles the KPIs from post training evaluations for each agent, with a

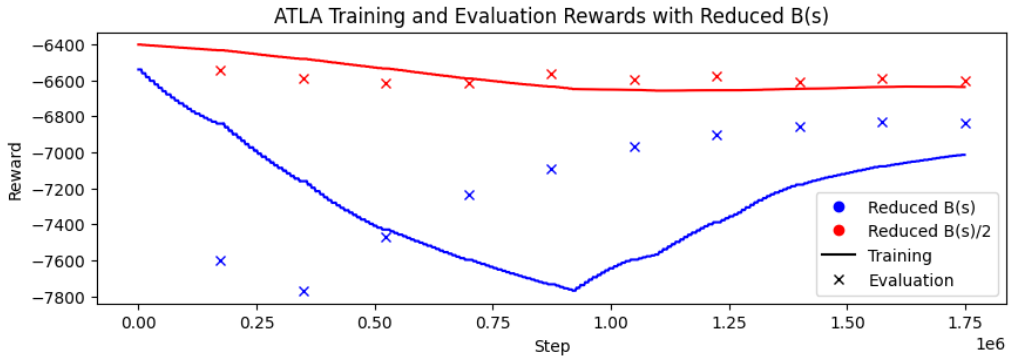


Figure 54: Line plot comparing the training curves for different adversarial budgets $B(s)$. The LA is a SAC and the agent a discrete PPO. The final evaluation rewards indicate that both agents reduce power consumption, however the attack using $B(s)/2$ does not have a significant effect on the agent. Reducing the adversarial budget produces more capable agents than those plotted in figure 53.

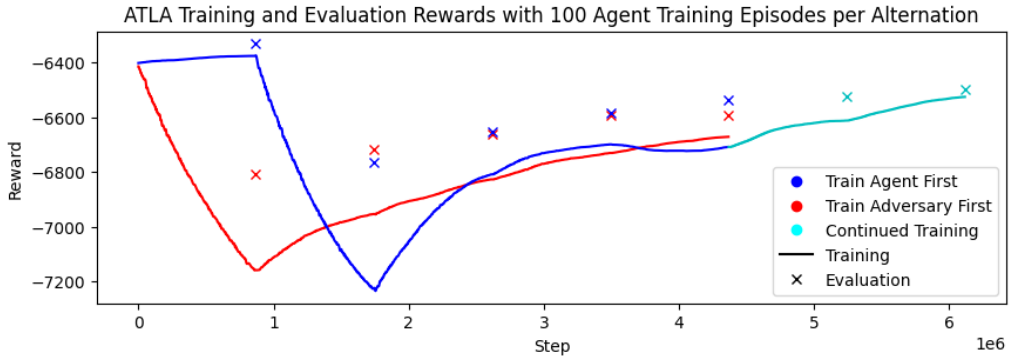


Figure 55: Line plot comparing the training curves for agents with longer training alternations. The LA is a SAC and the agent a discrete PPO. The increase from 20 to 100 episodes allows the agent to converge before the adversary is updated. This change increases agent performance for the same adversarial budget. Performance further improved by allowing the agent to train in the ATLA environment before the adversary, meaning the agent begins training with the perturbations of an untrained adversary. This further improved performance, and training was restarted for an additional 2 alternations following the first 5. This produced the best training and evaluation scores for an ATLA agent, though the improvement in evaluation scores was slight.

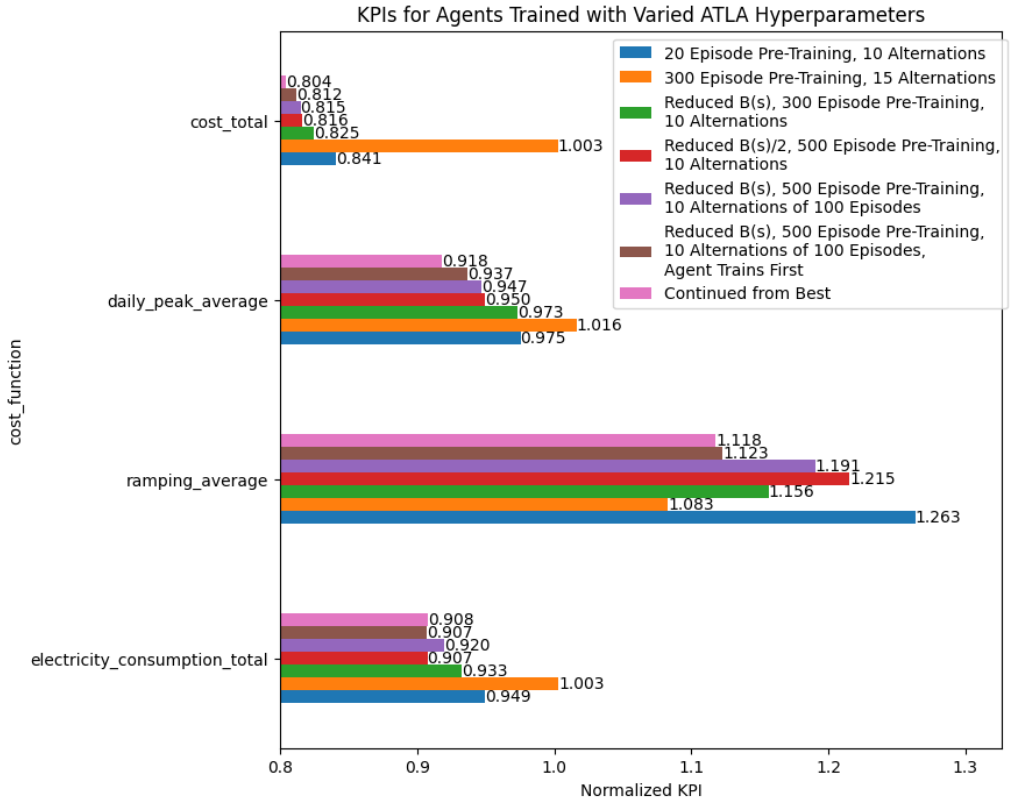


Figure 56: KPIs for the ATLA agents shown in Figures 53-54-55. The LA is a SAC and the agent a discrete PPO. Note that KPIs near 1 indicate that the agent has a negligible effect on the environment. Continued training shown in Figure 55 only slightly increased the performance of the agents, so further training was not attempted.

training curve above. This is roughly a third of the agents trained in the course of this work; those displayed illustrated decisions on ATLA parameters for the best agent. Tuning hyperparameters of the DRL algorithm used in ATLA could further improve their performance, but the scope of this work is simply training an ATLA agent with comparable performance to the conventional agent. Training discrete PPOs conventionally and with ATLA whose performance is similar, enables the comparison of their robustness to gradient-based attacks.

To understand the differences in the conventional and ATLA agent policies, Shapley Value Sampling (SVS) [69] was used for both agents. Figure 57 shows the mean of the absolute SVS values for the observations and actions of a clean evaluation episode. Because the goal is to understand feature importance, only the magnitude of the value is required. Otherwise, the mean SVS value for features which increase and decrease the output for different observations will be smaller, as the positive and negative values cancel each other out. A limitation of SVS, like any explainability method which permutes different feature values, is that the permuted samples are implausible when features are correlated. CityLearn's weather features are correlated both with each-other and the time, as the temperature depends on sunlight and sunlight varies by the time of day. This is detrimental to explaining either model individually, but does not prevent comparing the two, and the purpose of SVS in this work is simply understanding how the feature importances change with ATLA.

Feature Permutation (FP), which randomly substitutes a feature value and measures

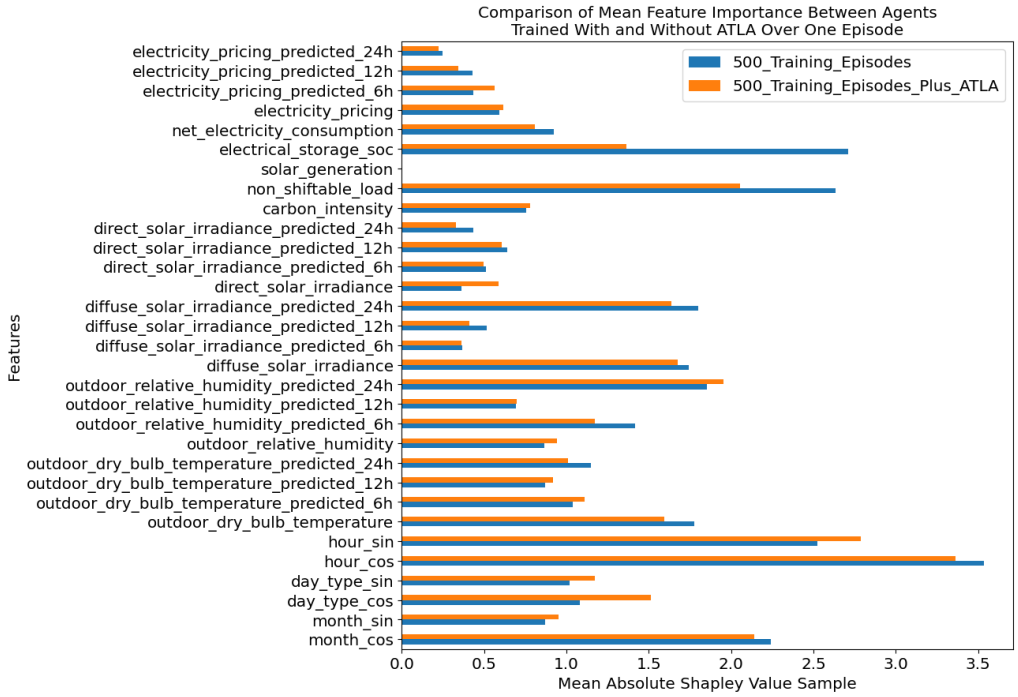


Figure 57: Comparison of feature importance between conventionally and ATLA trained agents, using the mean absolute Shapley Value Sample (SVS) for each action in one evaluation episode. The ATLA agent generally increases its reliance on temporal features, which were not perturbed, and significantly decreases its reliance on the SoC and non-shiftable loads. Given that loads are loosely correlated with the time and that the time was more reliable, the ATLA agent relies less on the current load which might be perturbed. Because solar generation is improperly normalized, changing it has very little effect on the agent's decision.

the change in output, was tested as an alternative to SVS. However, its results were not useful as all the features for ATLA were proportionally lower than those for the conventional agent. It turns out training an agent to be robust to perturbations to its inputs means perturbing its input makes the output change less, and this is what FP measures. Figure 57 shows that the ATLA agent relies more on temporal features and less on the SoC and non-shiftable load. The latter two are among the top three most important features for the conventional agent, which is understandable because it is hard to decide if you should charge your battery if you don't know how charged it is or how much power is being used. Because the temporal features were not perturbed during ATLA and they are loosely correlated with power usage, it makes sense that the ATLA agent is able to trade some performance for making decisions based on more reliable features.

7.4.2 ATLA Agent Robustness

The KPIs shown in Figure 58 demonstrate that ATLA nearly eliminated the effects of an LA attack, indicating that ATLA was successful in training an agent with a robust policy. The question of interest was then if the robust policy decreased the adversarial regret for gradient-based attacks, and if ATLA is a viable defense against such attacks. Figure 59 shows the adversarial regrets for both the conventional and ATLA agents for various attacks:

	Conventionally Trained	ATLA
Dynamic ACG	99.1%	99.0%
Stealthy Bifurcated PGD	39.0%	44.7%
Optimally Targeted BB	100%	93.4%

Table 17: ASRs for ATLA and conventionally trained agents for various attacks.

1. The stealthy bifurcated PGD attack will expose the performance of both agents under an attack that is difficult to detect in terms of both feature variations and MMD.
2. Optimally targeted BB represents the most powerful, though detectable, attack.
3. The dynamic distortion ACG attack is included as a direct (non-bifurcated) attack, which also has a low MMD.
4. LA attack, which ATLA is specifically designed to resist [60]. It outperforms methods which do not attack the neural network function approximator, like those above, and instead exploit weaknesses in the victim agent's policy.

This plot also shows the difference in clean performance between the ATLA and conventional agents, to contextualize the adversarial regrets. While in all cases the ATLA agent loses less performance from attacks, in the case of power consumption it is negated by the ATLA agent's inferior clean performance. Despite this, the ATLA agent performs better in terms of the other KPIs while attacked. Table 17 shows that the differences in adversarial regret are not caused by a reduced ASR for the untargeted attacks. Because ATLA provides a robust policy rather than a robust function approximator, it makes sense that the ATLA agent makes better decisions while subject to attacks but does not make the attacks less successful.

The optimally targeted attack induces the victim to follow an adversarial policy, the agent's original policy is less relevant. For this attack there is a reduction in both ASR and adversarial regret, though the effects are not significant. Here the reduced ASR can explain the ATLA agent's improved performance. The ATLA agent seems less likely to take the extreme actions imposed by the adversarial policy, making it slightly more resistant to the attack. it is harder to find a perturbation which makes the ATLA agent fully charge or discharge in certain states.

These results show that ATLA's usefulness for defending against attacks is nuanced, and depends on the threat environment in which the system operates. Using ATLA on a single agent in CityLearn does reduce the costs, ramping, and peaks during attacks, however these KPIs are larger during normal operation. Based on these results, ATLA is not recommended in general to defend against these white box gradient-based attacks in CityLearn. Because these results show some improvements to robustness, future work may find ATLA useful in different continuous control environments.

ATLA was not similarly successful with a SAC agent. As with the gradient-based attacks, the SAC was more sensitive to observation perturbations, and training with the reduced $B(s)$ made the learning collapse. While the SAC could adapt to attacks half this size, it would collapse after the next alternation once the LA had updated. Given the mixed results from ATLA with the discrete PPO and the large hyperparameter search space for both the SAC and ATLA, this effort was abandoned.

7.5 Robust Action Space Results

Here the results of comparing the robustness of continuous and discrete action PPOs, and the SAC are presented. Training a continuous action space PPO proved far more difficult

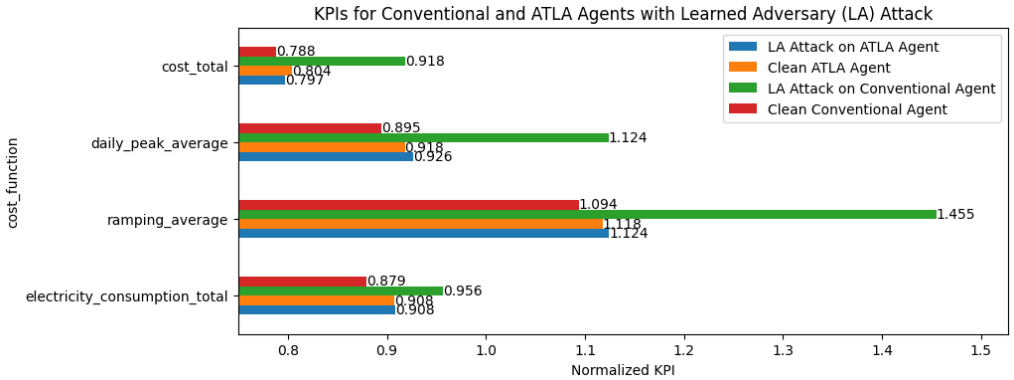


Figure 58: Adversarial regret of a SAC Learned Adversary (LA) attack on both conventionally and ATLA trained discrete PPO agents. This specific adversary was not used during ATLA training, so neither agents were trained against its policy. While the LA induces a significantly larger adversarial regret than untargeted attacks in previous sections (albeit with over twice the adversarial budget), the adversarial regret for the ATLA agent is insignificant.

than either the discrete action space PPO and SAC, and success required collaboration with another student. The continuous PPO tended to get stuck in a local minima where it learned never to fully charge than discharge the battery, so it had no effect on the environment. Achieving a similar electricity consumption KPI required 80 times as many training episodes and a different reward function. The solar penalty reward penalizes the agent for storing energy when consumption is high [63]. Figure 60 demonstrates that the action space makes the discrete PPO robust to adversarial attacks, as the continuous PPO and SAC exhibit significantly higher adversarial regrets. Furthermore, the attack's effect on the discrete PPO's power consumption was relatively small. This is particularly useful for defenders because by testing for the optimal number of bins, the discrete PPO's performance was nearly identical to the SAC for identical training times. There was no compromise in performance for the robustness gained from the discrete action space.

7.6 Defences Summary

Stealth is a significant advantage from adversarial examples, but evading detection significantly constrains the adversarial budget. Limits on budget similarly limit adversarial regret. This section shows that a well constrained attack produces adversarial observations which are statistically indistinguishable from the originals, according to the MMD test. The perturbation boundary ϵ can be selected for each feature such that: observations appear plausible, the mean variation between observations does not significantly increase, and the maximum mean absolute variation does not exceed the original maximum value. Detecting such an attack is non-trivial, but MMD test results on the distribution of the absolute differences between sequential observations do not exclude the possibility of detection. While the the adversarial observations themselves are match the distribution of original samples, the distributions of the absolute differences between the original observations form a distinct distribution compared to the absolute differences between the adversarial observations.

Overall, the stealthy attack only reduced the benefit of the DRL controller but failed to remove it, as the stealthy attack did not cause the electricity consumption, cost, or daily peak KPIs to exceed one. However, the PGD attack could be improved by using momentum, Adam, or conjugate gradients to calculate the δ, and the auto stepsize proposed by [34]. Re-engineering SotA gradient-based attacks is outside the scope of

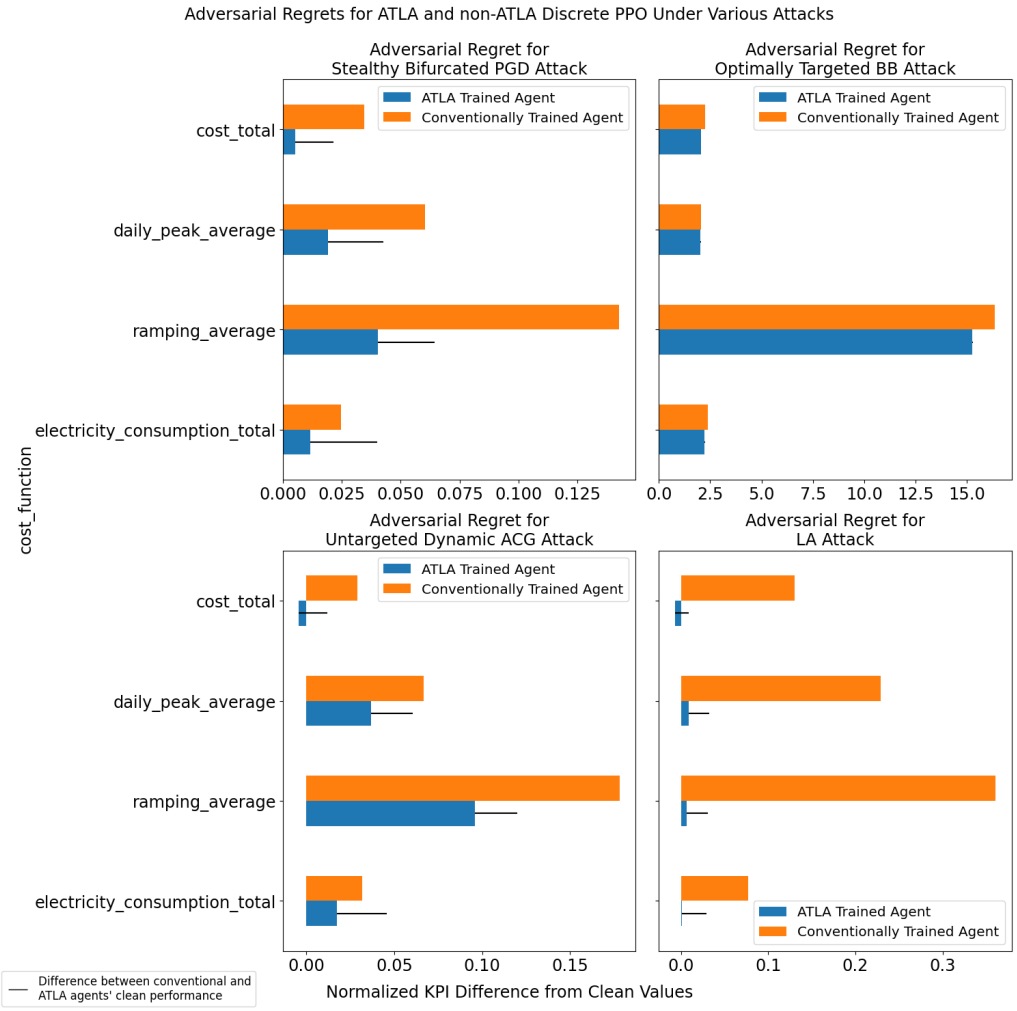


Figure 59: Histogram comparing the adversarial regrets of agents trained conventionally and with ATLA under various attacks. The black bar represents the difference in clean performance between the ATLA and non-ATLA agent, which indicates instances where the regret is smaller for the ATLA agent, but its reduced clean performance means the non-ATLA agent still performs better for that KPI. While the ATLA agent's adversarial regret is smallest in all cases, it still consumes more energy than the conventionally trained agent when attacked with untargeted adversarial examples. The stealthy attack is the bifurcated PGD attack, with $\epsilon = 0.03$ masked temporal and solar generation features, and scaled ϵ for net electricity consumption.

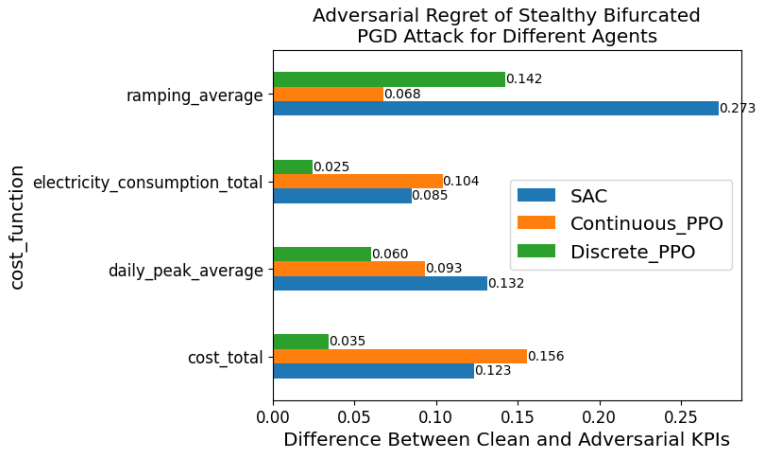


Figure 60: Adversarial Regret of Stealthy Bifurcated PGD Attack for Different Agents. These results how that regardless of the RL algorithm, a continuous action space is less robust than discrete.

this work. Implementing a simple attack demonstrated that regressors can be directly attacked and that the bifurcation method is more effective in continuous control than a direct attack. Due to limitations with ART, this custom PGD implementation was also used for attacks with limited features which could be perturbed. Future attacks could have greater adversarial regrets with smaller adversarial budgets.

ATLA does reduce the adversarial regret experienced by a victim discrete PPO agent, but the trade-off in clean performance did not justify the additional robustness for this task. Thus the decision to use ATLA is application specific and should reflect the model in which the agent would operate. While ATLA may not be beneficial in a domestic setting, it could be sensible for a hydro-electric or mass energy storage facility who's energy consumption could significantly affect the power grid. ATLA was unsuccessful for a SAC agent.

Discretizing or binning a the action space increased the agent's robustness without reducing performance. This is a simple technique which could be widely applied for DRL in continuous control. These findings suggest is should be adopted for any applicable DRL agent in a CPS.

8 Black-Box Attack

This section tests the impact of the black-box snooping attack on agents trained in Chapter 6, to show that adversarial attacks can decrease the performance of a DRL controller without access to its ANN parameters. It begins with the methodology, followed by the results.

8.1 Methodology

The snooping attack [58] allows an attacker to leverage adversarial attacks with *no knowledge* of the victim agent's algorithm or architecture. Unlike previous white box attacks which assume the attacker was able to copy the Artificial Neural Network (ANN) parameters. This is also distinct from training a surrogate model, which is done using the victim agent's environment. Instead, the attacker is only able to observe the victim agent acting in its environment, which is a requisite level of access for conducting an observation perturbation attack. If the attacker can change the victim's observations, it is probable that they can also observe them. Like the snooping attack, a replay attack can also be conducted with access to historical data and the ability to modify the victim's observations. There are several advantages to the snooping attack:

1. Adversarial examples can be crafted when some features cannot be changed. This is an advantage when the attacker can perturb most but not all features e.g. is able to change sensor readings but not the internal time on the controller.

2. The attacker must collect observations which correspond to the desired actions to conduct a replay attack. This is an issue when an agent seldom takes the desired action, or only takes it in a state very different from the present. The snooping attack can craft adversarial observations on the fly.

3. Using the snooping attack, perturbations can be constrained to suit the target environment. Despite being statistically distinguishable from normal samples, Fast Gradient Method (FGM) perturbations can still be relatively small and could feature smaller inter-sample changes compared to disjointed replay samples.

Using the SA threat model, a ANN proxy imitator was trained which enabled FGM attacks. Like previous white-box attacks, the bifurcation method is compatible with a proxy and the FGM attack. A bifurcation layer was appended to the trained proxy, in the same fashion it was added to a trained agent's policy network. The proxy used no prior architectural knowledge of the agent, instead hyperparameters were chosen during an Optuna trial which maximized the mean time series split validation over one episode of data. This means the training starts with the first $\frac{1}{k}$ of data, and is validated with the second, then trains on the first $\frac{2}{k}$ to validate on the third etc. The study chose hyperparameters to optimize accuracy over $k = 10$ splits. The search space and results are shown in Table 18, which were selected using a Tree-structured Parzen Estimator (TPE) sampler paired with a hyperband pruner [70] over 400 trials. The imitator was trained using the best parameters on the entire dataset. Given the seasonality in one episode representing a year of data, reserving any data for testing would prevent the imitator from seeing a portion of a season conducted with no more knowledge of the victim than how it behaves in its environment.

8.2 Results

Snooping attacks were conducted using ART's FGM attack, which is similar to a single step of Projected Gradient Descent (PGD). Iterative methods like PGD and Auto-Conjugate Gradient (ACG) which search for the best local maxima performed worse

Hyper-parameter	**Search Space**	**Discrete Best Value**	**Continuous Best Value**
Optimizer	Adam, RMSProp, or SGD	RMSProp	SGD
Learning Rate	$[1 \times 10^{-5}, 1 \times 10^{-1}]$	3.3×10^{-4}	6.2×10^{-2}
Batch size	$2^n, n \in Z,\ 3 \leq n \leq 10$	64	8
Layers	1,2, or 3	2	2
Activation Function	Relu or Tanh	Tanh	Tanh
Number of Units	64-512	300, 493	464, 192
Dropout	$[0, 0.5]$	0.388, 0.145	0.25, 0.45
Output	Linear or Softmax (discrete only)	Linear	Linear
Loss (Continuous)	MAE, MSE, Huber	-	Huber

Table 18: Optuna study search space and optimal hyperparameters for the imitator used in the snooping attack. The abbreviations are: Root Mean Squared Propagation (RMSprop), Stochastic Gradient Descent (SGD), Mean Absolute Error (MAE), and Mean Square Error (MSE). Only CE loss was used for the discrete case, and only linear outputs for continuous.

than FGM, as per the results of [58], because the imitator and victim do not share identical decision boundaries or loss topology. Because FGM is a weaker attack compared to ACG, it requires a larger ϵ for a similar adversarial regret. The relationship between the adversarial budget and regrets for the snooping attack on the discrete Proximal Policy Optimization (PPO) are shown in Figure 62. The Adversarial Success Rate (ASR)s of this attack are compared to random noise in Figure 61, which shows that a random attack requires almost 10 times the adversarial budget for a similar ASR. The Random attack did not have a significant adversarial regret for any perturbation size tested. With $\epsilon = 0.05$ the power consumption for bifurcated FGM is comparable to the white-box ACG dynamic distortion attack. The discrete PPO is robust in that it requires $\epsilon \approx 0.13$ before power consumption is equivalent to no controller, and is increased by over 10%. By comparing the slope of the direct and bifurcated attacks, the impact of the former becomes apparent. The bifurcation method significantly increases the cost and power consumption for the conventionally trained agent, though the Alternating Training with Learned Adversary (ATLA) trained agent sees a much smaller difference. Unlike the white-box attacks of the previous section, ATLA provides a significant resistance to the most powerful technique used in this section. The bifurcated snooping attack produced a significantly higher adversarial regret for the conventional agent compared to the ATLA agent. This is the same to a smaller extent for the direct snooping attack, and this was partially counteracted by the ATLA agent's lower clean performance. Because black box attacks require fewer prerequisites for the attacker, they are more likely to be encountered. These attacks are also easier to detect, as previous works have shown that adversarial samples crafted with FGM are detectable through statistical techniques [43]. Furthermore, this work found that attacks with $\epsilon > 0.03$ are statistically detectable in CityLearn, while snooping attacks require 4 times that to undo the benefit of the DRL controller for a discrete agent.

The performance of the discrete and continuous PPO, and Soft Actor Critic (SAC) under a bifurcated snooping attack are compared in Figure 63. While $\epsilon \approx 0.13$ is required to remove the benefit of the discrete PPO in terms of power consumption, the same happens for the SAC for $\epsilon \approx 0.06$ and $\epsilon \approx 0.05$ for the discrete PPO. At this smaller value of ϵ the adversarial regret for the discrete PPO is only 0.03. Even without ATLA

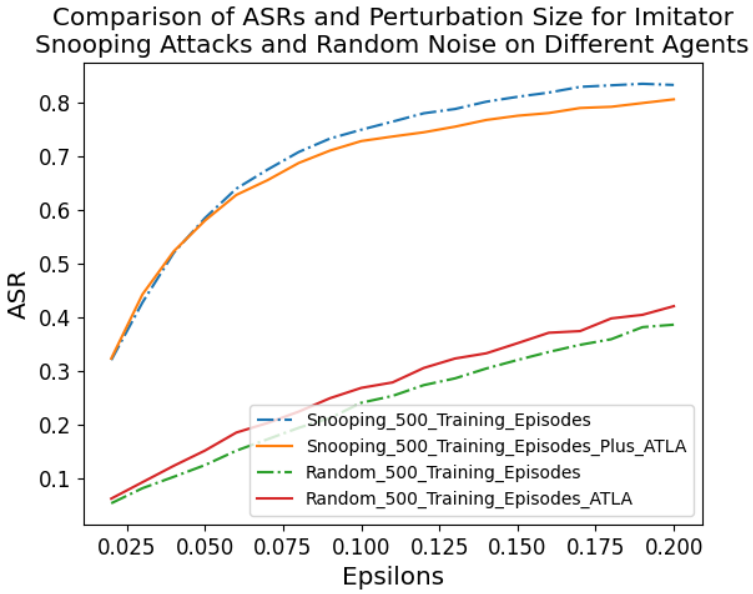


Figure 61: Line plot comparing the ASRs for random and imitator snooping attacks on conventionally and ATLA trained agents. It shows that the ASR is roughly doubled for the snooping attack compared to random noise, demonstrating the effectiveness of the snooping attack. The random noise requires approximately 10 times the perturbation size to match the ASR of the snooping attack. Note that the ASR is slightly lower for the ATLA agent for the snooping attack.

training, the discrete action space significantly improves the robustness of the PPO to the black-box snooping attack.

8.3 Summary

An adversary can achieve a significant reduction in performance, while only observing how the victim behaves in their environment and exploiting vulnerable sensor systems. Unlike the finding in [60] where the LA outperformed the snooping attack, combining the snooping attack with the bifurcation method outperforms the LA attack while using a smaller adversarial budget. The snooping attack is gradient-based like a white box attack, which typically outperform black box attacks when a gradient is available. The bifurcation method enhances the loss function for gradient-based attacks, improving the ratio of the adversarial regret to budget. The bifurcated snooping attack had comparable power consumption to the LA attack with $\epsilon = 0.07$, which is far smaller than the reduced $B(s)$, shown in Table 16, for all but solar generation and net electricity consumption features. Designers should invest in detection systems to counter such methods, and ATLA could be useful depending on the trade-off between performance and robustness.

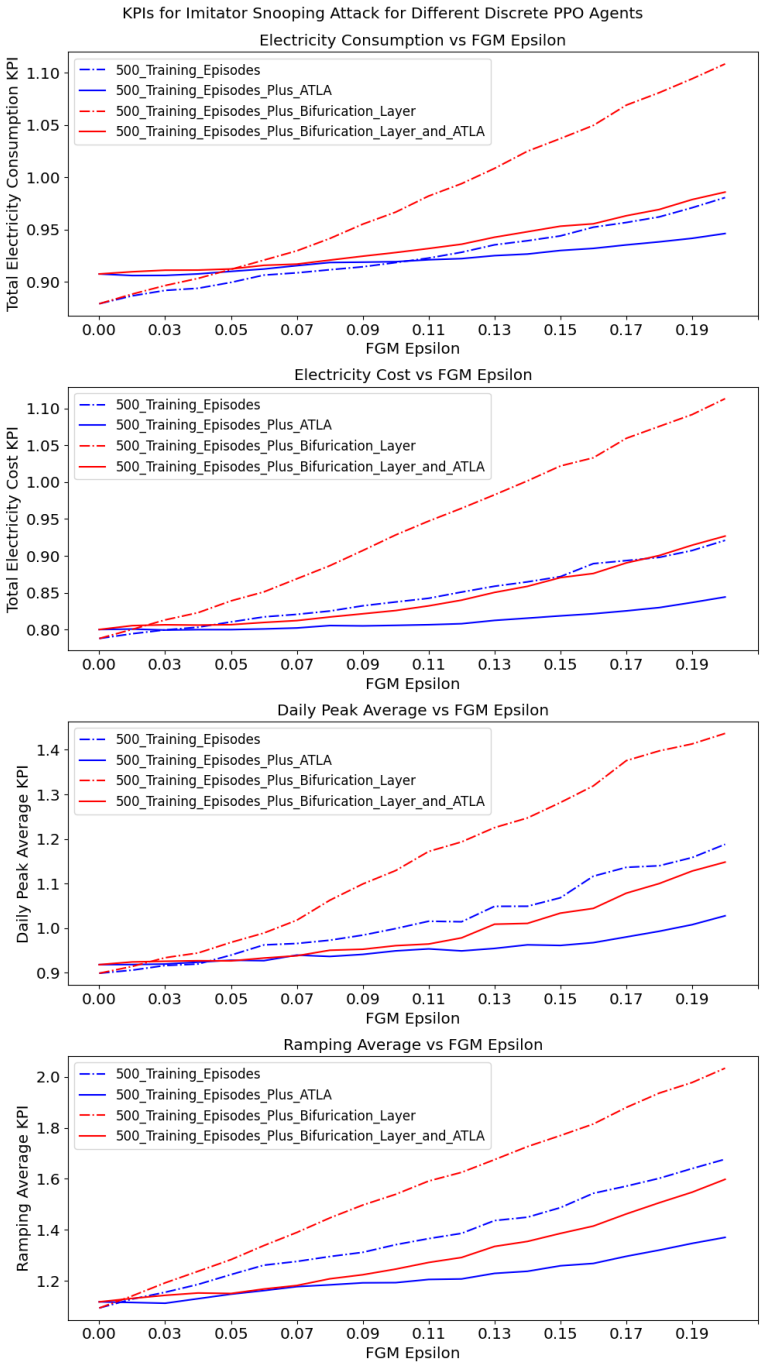


Figure 62: Comparison of electricity consumption, ramping, daily peak, and cost KPIs for direct and bifurcated FGM snooping attacks with a range of ϵ, for discrete PPO agents trained conventionally and with the ALTA method. The line symbols indicate the agent, while the colour indicates the attack. These plots show the trend between adversarial budget and regret. KPIs were chosen as the former is relevant to LAAs affecting grid stability, and the cost for cost based attacks. Solid lines represent ATLA training. While the robustness offered by ATLA is insignificant compared to its reduction in clean performance for energy consumption, the corresponding adversarial regret for bifurcated attacks is significantly reduced. For all other KPIs the ATLA agent performs best under attack.

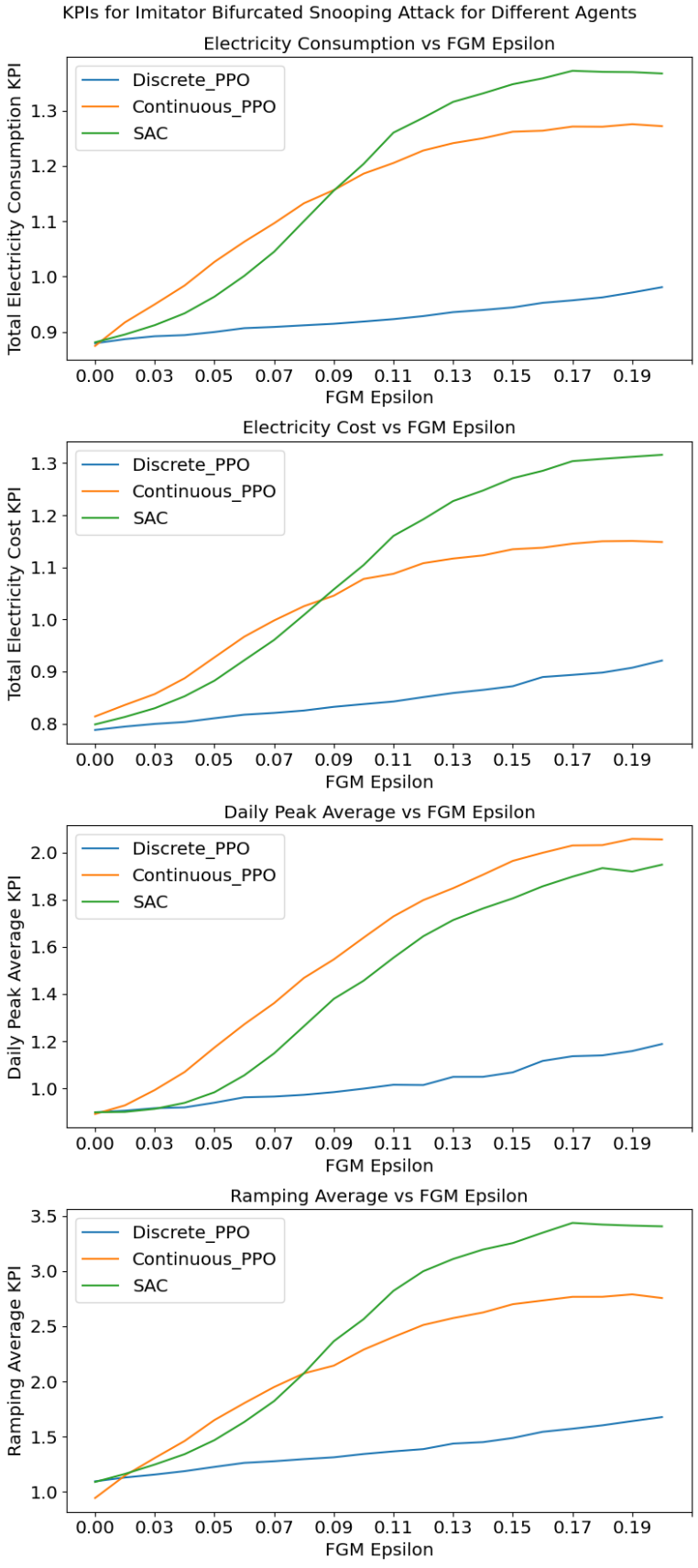


Figure 63: Comparison of KPIs for bifurcated FGM snooping attacks with a range of ϵ, for discrete and continuous PPO, and SAC agents. This figure compares the trend of adversarial budget and regret between various DRL algorithms and action spaces. This figure demonstrates that the discrete PPO is significantly more robust than either agent with a continuous action space, even without ATLA training.

9 Conclusion

This chapter will conclude the thesis by emphasizing what this research contributes to smart energy systems. As well, it will highlight some open research questions that were encountered during this process and could be the aim of future work in the domain.

9.1 Contributions

1. In Chapter 6 this work proposes a novel adversarial attack technique for continuous control. By implementing the Grouped Difference Logit (GDL) loss with a bifurcation layer, the adversarial regret for SotA attacks was doubled for continuous control agent in CityLearn. The bifurcation method can be used for continuous and discrete outputs, even for attacks which only support Artificial Neural Networks (ANN) with multiple outputs. This technique was validated using identical attacks without the bifurcation layer for both classification/discrete actions and regression/continuous actions. To the best of the author's knowledge, no other work has used this or a similar technique. To validate the bifurcation method for continuous action spaces, this work implemented Projected Gradient Descent (PGD) for a regressor, which no reviewed work had done.

2. Chapter 7 shows that with a carefully selected budget, distributions of adversarial observations, which are not significantly different from the originals, cause a large adversarial regret. These adversarial observations were produced using PGD with the bifurcation method. Conversely, adversarial observations form a distinct distribution when considering the absolute difference between subsequent observations in a time-series. The latter result could be leveraged in the detection of adversarial observations. To the best of the author's knowledge, no other work has studied the detection of adversarial observations (adversarial examples) in a Cyber Physical System (CPS) gym which uses real world data.

3. In Chapter 7 Alternating Training with Learned Adversary (ATLA) is an effective defence against strong black box attacks targeting the ANN function approximator. Combining the bifurcation method with the snooping attack significantly increased the adversarial regret for the conventionally-trained PPO, but the increase was less than half for the PPO trained using ATLA.

4. In Chapter 7 Using novel robustness testing techniques, this work found that the choice of off-the-shelf Deep-Reinforcement Learning (DRL) algorithm and action space can significantly affect the agent's robustness. When the adversarial budget was decreased to the point that the perturbations were considered stealthy, the attack had a minimal effect on the discrete Proximal Policy Optimization (PPO) but still caused a significant adversarial regret for the Soft Actor Critic (SAC) and continuous PPO. The effects of adversarial observations, even in the best case white box scenario, are significantly reduced using a discrete action space with a PPO. There wasn't a significant difference in clean performance between the PPO and SAC, meaning there was no concession for the additional robustness.

9.2 Future Work

The greatest limitation of CityLearn for studying Load Altering Attack (LAA)s was the absence of the wider power grid, limiting this work to reward based attack goals. The only effect of the adversarial observation perturbations is to increase power usage and other KPIs. An environment which simulates frequency instability and line overloads in a power grid would enable an end-to-end LAA with adversarial observation perturbations. Such

an environment could simulate if adversarial observations could cause Under Frequency Load Shedding (UFLS), localized, or wide area blackouts.

Though this work explored attacks and detection for single agent environments, CityLearn also supports MARL. The MARLISA [71] algorithm is an independent SAC algorithm with inter-agent communications, integrated with CityLearn. The attacks for continuous action space RL developed and demonstrated in this work could be used on the independent SAC agents. MARLISA sequentially estimates the power consumption for each agent's action using decision trees , and the estimate is communicated to the next agent in the district. These decision trees are also vulnerable to observation perturbations, which could be used to enable a communication attack. An adversary might instead modify the communications between agents.

The decayed PGD attack implemented in this work for attack on regressors could be improved with the following recent developments to build a SotA gradient-based attack which is compatible with regressors/continuous action spaces:

1. Auto-stepsize [34] for Auto-PGD, which reduces the number of parameters requiring tuning.
2. Using the conjugate gradient [41] instead of the projected gradient to maximize loss.
3. Early stopping, so computation ceases when the loss function stops increasing.
4. Random restarts, instead of only starting from the original sample.

A PPO using Long Short-Term Memory (LSTM), instead of MLP networks, was more robust to the LA attack in [60]. Because an LSTM considers previous observations in its current decision, it may also increase the robustness to gradient-based adversarial samples. Additionally, SB3 has an LSTM PPO agent.

This work demonstrated that a discrete action space significantly reduced the adversarial regret for an agent in CityLearn. This experiment should be reproduced in a variety of continuous control environments.

Increased differences between observations was the strongest indicator of adversarial observations. Future attacks for time-series data could account for the previous observation in the boundary for current perturbation.

SotA adversarial attacks are designed for ANNs with a single label, or a one-dimensional action space in DRL. However, agents in CityLearn can control multiple energy storage devices, resulting in multi-dimensional action spaces. Future research could devise new loss functions or attacks which are effective when the victim has multiple outputs.

9.3 Summary and Closing Remarks

Adversarial attacks are a fascinating aspect of ANNs because they exploit the fundamental methods used to train them, to fool them. DRL is poised to solve many future control problems in Cyber Physical Power System (CPPS), but the research in how DRL controllers could be affected by adversarial attacks is limited. This work addresses the gap.

The first research question was if adversarial attacks are a significant threat to DRL in Demand Response (DR). In Chapter 6 initially the answer seemed to be no. In fact, unlike other types of DRL applications in CPPS, like dispatch and operations control, one wrong move from a CityLearn agent won't cause frequency instability, disrupt power quality, or overload a relay. A LAA requires a significant increase in power draw, exceeding what the power grid can provide. CityLearn's continuous action space makes agents more robust to attack, even when it is discretized into bins as for the discrete PPO. Small

changes to the action, like (dis)charging slightly more or less, do not significantly affect overall power consumption. Thus, high Adversarial Success Rate (ASR)s do not lead to large adversarial regrets. The ASR would directly correspond to power consumption or adversarial regret in DR environments with more discrete or binary actions, such as controlling a load (air conditioners, water heaters, EV chargers, etc.) which is either on or off, rather than energy storage. Using CityLearn made the attacks harder, but has the advantage of realistic observations.

Early experiments showed that untargeted attacks had small effects on electricity consumption, and the State of the Art (SotA) maximum-confidence attack ACG was ineffective for an optimally targeted attack. Broadening the search for attacks lead to success with a minimum-norm attack, at the cost of large distortions to the victim's observations, which demonstrate the victim's robustness. This optimally targeted attack induced the victim into following an arbitrary policy. The novel bifurcation method was devised, which not only enabled adversarial attacks with larger impact than the untargeted attacks with much smaller distortions that the optimally targeted attack, but also interfaced SotA maximum-confidence attacks with ANNs they were never designed to work with.

Initially, this work only considered DRL agents with discrete action spaces, because that made their decisions akin to the classification problems adversarial attacks are designed for. The bifurcation method allowed some of these same attacks to work on agents with continuous action spaces. By implementing a simple adversarial attack which could attack both continuous and discrete agents, this work proved that the bifurcation method significantly increases adversarial regrets for both types of agent, compared to direct attacks.

In Chapter 7, having shown the impacts of adversarial attacks, this work studied if the adversarial observations were obviously fake. This was enabled by CityLearn's use of recorded environmental and power consumption data for many of its features. It provided realistic observations which could be perturbed then analyzed. This analysis took three approaches: do the changes between observations make sense, how is the accuracy of weather predictions affected, and do the adversarial and original observations belong to the same statistical distribution? As the attacker was given every advantage in the attacks, the original clean observations were used during the statistical analysis. In actual usage, the detector would only have access to historical data. If the distributions appear identical under these conditions, then they are unlikely to be separated in practice. This analysis found that an attack could be devised for which: the distribution of adversarial observations was identical to the originals, the increase in prediction error was less that 5%, and the variations between observations were within the original spread.

However, the distribution of *changes* between adversarial observations was distinct from the original. These results show that adversarial attacks are capable of crafting adversarial observations which are nearly indistinguishable from the originals, but do not account for the time-series of sequential observations. Because these adversarial observations closely resemble the originals, DRL in Critical Infrastructure (CI) must be robustly designed and validate the integrity of observations during observation. The results suggest that such validation will be more successful by analyzing the changes between observations.

In Chapter 7 ATLA was chosen as a model-agnostic robust training method, which could be integrated with the training of virtually any DRL algorithm. Results comparing agents trained conventionally and with ATLA showed that the ATLA agent had less adversarial regret for all adversarial attacks. However, there is a trade-off between clean performance and robustness, and the reduction in clean performance meant that the ATLA-trained agent still consumed more energy than its counterpart, even though the effect of the attack was lessened. These results indicate that ATLA alone does not address

the threats posed by adversarial attacks, given the performance compromise. Because the exact trade-off and risk tolerance varies by application and environment, ATLA is worth testing when training DRL agents for CI.

In Chapter 7 simple architectural choices can make a DRL controller more robust. Both a PPO with discrete actions and a continuous action SAC were trained to the same level of performance with the same training budget. However, the SAC was far more vulnerable to attacks. A stealthy attack was demonstrated on the SAC, but a similarly restrictive adversarial budget had a one third of the effect on power consumption for the discrete PPO. By comparing the adversarial regret between continuous and discrete PPOs, this work shows that the discrete action space is responsible for the increased robustness. This was discovered through robustness testing with the newly proposed attack techniques in this work. Robustness testing with adversarial attacks in continuous control was absent in the literature reviewed. Similar testing should be conducted for DRL agents in CI among different algorithms and architectures to select the most robust agents. The additional robustness provided by the discrete action spaces suggests it should be the default for DRL agents in CPS.

In Chapter 8 the snooping attack, enhanced with the novel bifurcation technique, demonstrated that an attacker with the means of injecting observation perturbations is also capable of a gradient-based adversarial attack. While the previous white box attacks required access to the victim agent parameters, the snooping attack only needed historical observations and actions. The assumption is that if an adversary is capable of modifying the victim's observations, the adversary could also collect them. The adversarial budget for these attacks was proportionally larger than that required of white-box attacks which achieved the same adversarial regrets, so the snooping attack could be detected from distortions in the adversarial observations. These aspects make the snooping attack comparable to a replay attack, where both have similar requirements and each with advantages and disadvantages. Additionally, the agent trained with ATLA was approximately twice as robust to the most powerful black box attack, bifurcated snooping. This is significant because a snooping attack is far simpler for an attacker to execute than a white box attack.

Overall, adversarial attacks must make a significant compromise between stealth and their impact. Robustness to sub-optimal actions caused by adversarial attacks is an attribute of the CityLearn's implementation of a DR environment. These actions will reduce efficiency but do not easily cause the system to fail in this environment. Stealthy attacks are possible, but with a high set-up cost and only for some victims. However, as the development of more powerful attacks progresses, there's no reason for this conclusion to hold. The risk-potential of adversarial attacks should be mitigated with robust designs. To this end, the product of this research is the importance of conducting robustness testing, and novel techniques for testing with stronger attacks. Nearly undetectable adversarial attacks are possible, and this work contributes tools to mitigate them.

References

[1] I. Zografopoulos, N. D. Hatziargyriou, and C. Konstantinou, "Distributed energy resources cybersecurity outlook: Vulnerabilities, attacks, impacts, and mitigations," *IEEE Systems Journal*, p. 1–15, 2023.

[2] W. Brendel, J. Rauber, M. Kümmerer, I. Ustyuzhaninov, and M. Bethge, "Accurate, reliable and fast robustness evaluation," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/hash/885fe656777008c335ac96072a45be15-Abstract.html

[3] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," no. arXiv:1706.06083, Sep. 2019, arXiv:1706.06083 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1706.06083

[4] J. R. Vazquez-Canteli, S. Dey, G. Henze, and Z. Nagy, "Citylearn: Standardizing research in multi-agent reinforcement learning for demand response and urban energy management," no. arXiv:2012.10504, Dec 2020, arXiv:2012.10504 [cs]. [Online]. Available: http://arxiv.org/abs/2012.10504

[5] Y.-C. Lin, Z.-W. Hong, Y.-H. Liao, M.-L. Shih, M.-Y. Liu, and M. Sun, "Tactics of adversarial attack on deep reinforcement learning agents," no. arXiv:1703.06748, Nov 2019, arXiv:1703.06748 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1703.06748

[6] R. S. de Carvalho and D. Saleem, "Recommended functionalities for improving cybersecurity of distributed energy resources," in *2019 Resilience Week (RWS)*, vol. 1, Nov. 2019, p. 226–231. [Online]. Available: https://ieeexplore.ieee.org/document/8972000

[7] S. Soltan, P. Mittal, and H. V. Poor, "BlackIoT: IoT botnet of high wattage devices can disrupt the power grid," 2018, p. 15–32. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/soltan

[8] M. Standen, J. Kim, and C. Szabo, "Sok: Adversarial machine learning attacks and defences in multi-agent reinforcement learning," no. arXiv:2301.04299, Jan 2023, arXiv:2301.04299 [cs]. [Online]. Available: http://arxiv.org/abs/2301.04299

[9] C. C. f. C. Security, "Cyber threat bulletin: Cyber threat to operational technology," Dec 2021, last Modified: 2021-12-16. [Online]. Available: https://www.cyber.gc.ca/en/guidance/cyber-threat-bulletin-cyber-threat-operational-technology

[10] D. Cao, W. Hu, J. Zhao, G. Zhang, B. Zhang, Z. Liu, Z. Chen, and F. Blaabjerg, "Reinforcement learning and its applications in modern power and energy systems: A review," *Journal of Modern Power Systems and Clean Energy*, vol. 8, no. 6, p. 1029–1042, Nov 2020.

[11] A. T. D. Perera and P. Kamalaruban, "Applications of reinforcement learning in energy systems," *Renewable and Sustainable Energy Reviews*, vol. 137, p. 110618, Mar 2021.

[12] I. Ilahi, M. Usama, J. Qadir, M. U. Janjua, A. Al-Fuqaha, D. T. Hoang, and D. Niyato, "Challenges and countermeasures for adversarial attacks on deep reinforcement learning," *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 2, p. 90–109, Apr 2022.

[13] R. S. Sutton and A. Barto, *Reinforcement learning: An introduction*. The MIT Press, 2020.

[14] F. Tramèr, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "The space of transferable adversarial examples," no. arXiv:1704.03453, May 2017, arXiv:1704.03453 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1704.03453

[15] V. Behzadan and A. Munir, "Vulnerability of deep reinforcement learning to policy induction attacks," no. arXiv:1701.04143, Jan 2017, arXiv:1701.04143 [cs]. [Online]. Available: http://arxiv.org/abs/1701.04143

[16] J. Kos and D. Song, "Delving into adversarial attacks on deep policies," no. arXiv:1705.06452, May 2017, arXiv:1705.06452 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1705.06452

[17] L. Hussenot, M. Geist, and O. Pietquin, "Copycat: Taking control of neural policies with constant attacks," no. arXiv:1905.12282, Jan 2020, arXiv:1905.12282 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1905.12282

[18] K. Mo, W. Tang, J. Li, and X. Yuan, "Attacking deep reinforcement learning with decoupled adversarial policy," *IEEE Transactions on Dependable and Secure Computing*, p. 1–1, 2022.

[19] O. A. Alimi, K. Ouahada, and A. M. Abu-Mahfouz, "A review of machine learning approaches to power system security and stability," *IEEE Access*, vol. 8, p. 113512–113531, 2020.

[20] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, ser. AISec '17. New York, NY, USA: Association for Computing Machinery, Nov 2017, p. 3–14. [Online]. Available: https://dl.acm.org/doi/10.1145/3128572.3140444

[21] A. Kelly, A. O'Sullivan, P. de Mars, and A. Marot, "Reinforcement learning for electricity network operation," no. arXiv:2003.07339, Mar 2020, arXiv:2003.07339 [cs, eess, stat]. [Online]. Available: http://arxiv.org/abs/2003.07339

[22] Y. Chen, D. Arnold, Y. Shi, and S. Peisert, "Understanding the safety requirements for learning-based power systems operations," no. arXiv:2110.04983, Oct 2021, arXiv:2110.04983 [cs, eess]. [Online]. Available: http://arxiv.org/abs/2110.04983

[23] B. Huang, A. A. Cardenas, and R. Baldick, "Not everything is dark and gloomy: Power grid protections against IoT demand attacks," 2019, p. 1115–1132. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/huang

[24] S. Amini, F. Pasqualetti, and H. Mohsenian-Rad, "Dynamic load altering attacks against power system stability: Attack models and protection schemes," *IEEE Transactions on Smart Grid*, vol. 9, no. 4, p. 2862–2872, Jul. 2018.

[25] D. Crawley, C. Pedersen, L. Lawrie, and F. Winkelmann, "Energyplus: Energy simulation program," *Ashrae Journal*, vol. 42, p. 49–56, Apr 2000.

[26] L. Yu, Y. Sun, Z. Xu, C. Shen, D. Yue, T. Jiang, and X. Guan, "Multi-agent deep reinforcement learning for hvac control in commercial buildings," *IEEE Transactions on Smart Grid*, vol. 12, no. 1, p. 407–419, Jan 2021.

[27] I. Darwish, O. Igbe, and T. Saadawi, "Vulnerability assessment and experimentation of smart grid dnp3," *Journal of Cyber Security and Mobility*, p. 23–54, Jun. 2016.

[28] F. Li, X. Yan, Y. Xie, Z. Sang, and X. Yuan, "A review of cyber-attack methods in cyber-physical power system," in *2019 IEEE 8th International Conference on Advanced Power System Automation and Protection (APAP)*, Oct. 2019, p. 1335–1339. [Online]. Available: https://ieeexplore.ieee.org/document/9225126

[29] C. Parian, T. Guldimann, and S. Bhatia, "Fooling the master: Exploiting weaknesses in the modbus protocol," *Procedia Computer Science*, vol. 171, p. 2453–2458, Jan. 2020.

[30] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," no. arXiv:1312.6199, Feb 2014, arXiv:1312.6199 [cs]. [Online]. Available: http://arxiv.org/abs/1312.6199

[31] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv*, 2018, accepted: 2021-11-05T14:56:05Z. [Online]. Available: https://dspace.mit.edu/handle/1721.1/137496

[32] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," no. arXiv:1412.6572, Mar. 2015, arXiv:1412.6572 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1412.6572

[33] V. Behzadan and W. Hsu, "Rl-based method for benchmarking the adversarial resilience and robustness of deep reinforcement learning policies," in *Computer Safety, Reliability, and Security*, ser. Lecture Notes in Computer Science, A. Romanovsky, E. Troubitsyna, I. Gashi, E. Schoitsch, and F. Bitsch, Eds. Cham: Springer International Publishing, 2019, p. 314–325.

[34] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," no. arXiv:2003.01690, Aug 2020, arXiv:2003.01690 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2003.01690

[35] ——, "Minimally distorted adversarial examples with a fast adaptive boundary attack," in *Proceedings of the 37th International Conference on Machine Learning*, ser. ICML'20, vol. 119. JMLR.org, Jul. 2020, p. 2196–2205.

[36] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, I. Molloy, and B. Edwards, "Adversarial robustness toolbox v1.2.0," *CoRR*, vol. 1807.01069, 2018. [Online]. Available: https://arxiv.org/pdf/1807.01069

[37] G. W. Ding, L. Wang, and X. Jin, "Advertorch v0.1: An adversarial robustness toolbox based on pytorch," *arXiv preprint arXiv:1902.07623*, 2019.

[38] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy, A. Matyasko, V. Behzadan, K. Hambardzumyan, Z. Zhang, Y.-L. Juang, Z. Li, R. Sheatsley, A. Garg, J. Uesato, W. Gierke, Y. Dong, D. Berthelot, P. Hendricks, J. Rauber, and R. Long, "Technical report on the cleverhans v2.1.0 adversarial examples library," *arXiv preprint arXiv:1610.00768*, 2018.

[39] J. Rauber, R. Zimmermann, M. Bethge, and W. Brendel, "Foolbox native: Fast adversarial attacks to benchmark the robustness of machine learning models in pytorch, tensorflow, and jax," *Journal of Open Source Software*, vol. 5, no. 53, p. 2607, 2020. [Online]. Available: https://doi.org/10.21105/joss.02607

[40] H. Kim, "Torchattacks: A pytorch repository for adversarial attacks," *arXiv preprint arXiv:2010.01950*, 2020.

[41] K. Yamamura, H. Sato, N. Tateiwa, N. Hata, T. Mitsutake, I. Oe, H. Ishikura, and K. Fujisawa, "Diversified adversarial attacks based on conjugate gradient method," no. arXiv:2206.09628, Jul 2022, arXiv:2206.09628 [cs]. [Online]. Available: http://arxiv.org/abs/2206.09628

[42] W. W. Hager and H. Zhang, "A survey of nonlinear conjugate gradient methods," *Pacific Journal of Optimization*, vol. 2, no. 1, p. 35–58, 2006.

[43] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel, "On the (statistical) detection of adversarial examples," no. arXiv:1702.06280, Oct 2017, arXiv:1702.06280 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1702.06280

[44] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *Journal of Machine Learning Research*, vol. 13, no. 25, p. 723–773, 2012.

[45] J. Martin and C. Elster, "Inspecting adversarial examples using the fisher information," no. arXiv:1909.05527, Sep 2019, arXiv:1909.05527 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1909.05527

[46] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square attack: A query-efficient black-box adversarial attack via random search," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, p. 484–501.

[47] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[48] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," no. arXiv:1801.01290, Aug 2018, arXiv:1801.01290 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1801.01290

[49] J. Guo, Y. Chen, Y. Hao, Z. Yin, Y. Yu, and S. Li, "Towards comprehensive testing on the robustness of cooperative multi-agent reinforcement learning," 2022, p. 115–122.

[50] J. Lin, K. Dzeparoska, S. Q. Zhang, A. Leon-Garcia, and N. Papernot, "On the robustness of cooperative multi-agent reinforcement learning," in *2020 IEEE Security and Privacy Workshops (SPW)*, May 2020, p. 62–68.

[51] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, p. 5026–5033.

[52] K. Åström, "Optimal control of markov processes with incomplete state information," *Journal of Mathematical Analysis and Applications*, vol. 10, no. 1, p. 174–205, Feb 1965.

[53] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The complexity of decentralized control of markov decision processes," *Mathematics of Operations Research*, vol. 27, no. 4, p. 819–840, Nov 2002.

[54] L. S. Shapley, "Stochastic games*," *Proceedings of the National Academy of Sciences*, vol. 39, no. 10, p. 1095–1100, Oct 1953.

[55] E. A. Hansen, D. S. Bernstein, and S. Zilberstein, "Dynamic programming for partially observable stochastic games."

[56] C. Tessler, Y. Efroni, and S. Mannor, "Action robust reinforcement learning and applications in continuous control," in *Proceedings of the 36th International Conference on Machine Learning*. PMLR, May 2019, p. 6215–6224. [Online]. Available: https://proceedings.mlr.press/v97/tessler19a.html

[57] H. Zhang, H. Chen, C. Xiao, B. Li, M. Liu, D. Boning, and C.-J. Hsieh, "Robust deep reinforcement learning against adversarial perturbations on state observations," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, p. 21024–21037. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/f0eb6568ea114ba6e293f903c34d7488-Abstract.html

[58] M. Inkawhich, Y. Chen, and H. Li, "Snooping attacks on deep reinforcement learning," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '20. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, May 2020, p. 557–565.

[59] V. Behzadan and A. Munir, "Whatever does not kill deep reinforcement learning, makes it stronger," no. arXiv:1712.09344, Dec. 2017, arXiv:1712.09344 [cs]. [Online]. Available: http://arxiv.org/abs/1712.09344

[60] H. Zhang, H. Chen, D. Boning, and C.-J. Hsieh, "Robust reinforcement learning on state observations with learned optimal adversary," *International Conference on Learning Representation (ICLR)*, Jan. 2021. [Online]. Available: https://par.nsf.gov/biblio/10318889-robust-reinforcement-learning-state-observations-learned-optimal-adversary

[61] M. Fischer, M. Mirman, S. Stalder, and M. Vechev, "Online robustness training for deep reinforcement learning," no. arXiv:1911.00887, Nov. 2019, arXiv:1911.00887 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1911.00887

[62] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[63] K. E. Nweye, A. Wu, H. Park, Y. Almilaify, and Z. Nagy, "Citylearn: A tutorial on reinforcement learning control for grid-interactive efficient buildings and communities," in *ICLR 2023 Workshop on Tackling Climate Change with Machine Learning*, 2023. [Online]. Available: https://www.climatechange.ai/papers/iclr2023/2

[64] R. Morell, "Grid integration of zero net energy communities," *California Public Utilities Commission*, Jan. 2017.

[65] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," no. arXiv:1907.10902, Jul. 2019, arXiv:1907.10902 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1907.10902

[66] Z. K. Madry and Aleksander, "Adversarial robustness - theory and practice." [Online]. Available: http://adversarial-ml-tutorial.org/

[67] T. Viehmann, L. Antiga, D. Cortinovis, and L. Lozza, "Torchdrift," Jan. 2024. [Online]. Available: https://github.com/TorchDrift/TorchDrift

[68] L. Zeng, D. Qiu, and M. Sun, "Resilience enhancement of multi-agent reinforcement learning-based demand response against adversarial attacks," *Applied Energy*, vol. 324, p. 119688, Oct 2022.

[69] N. Kokhlikyan, V. Miglani, M. Martin, E. Wang, B. Alsallakh, J. Reynolds, A. Melnikov, N. Kliushkina, C. Araya, S. Yan, and O. Reblitz-Richardson, "Captum: A unified and generic model interpretability library for pytorch," 2020.

[70] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: a novel bandit-based approach to hyperparameter optimization," *The Journal of Machine Learning Research*, vol. 18, no. 1, p. 6765–6816, Jan. 2017.

[71] J. R. Vazquez-Canteli, G. Henze, and Z. Nagy, "Marlisa: Multi-agent reinforcement learning with iterative sequential action selection for load shaping of grid-interactive connected buildings," in *Proceedings of the 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, ser. BuildSys '20. New York, NY, USA: Association for Computing Machinery, Nov 2020, p. 170–179. [Online]. Available: https://dl.acm.org/doi/10.1145/3408308.3427604

[72] J. Tu, T. Wang, J. Wang, S. Manivasagam, M. Ren, and R. Urtasun, "Adversarial attacks on multi-agent communication," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct 2021, p. 7748–7757.

[73] N. H. Pham, L. M. Nguyen, J. Chen, H. T. Lam, S. Das, and L. Weng, "c-mba: Adversarial attack for cooperative marl using learned dynamics model," Dec 2022. [Online]. Available: https://openreview.net/forum?id=AFfKSfcF6Sv

[74] Y. Hu and Z. Zhang, "Sparse adversarial attack in multi-agent reinforcement learning," no. arXiv:2205.09362, Aug 2022, arXiv:2205.09362 [cs]. [Online]. Available: http://arxiv.org/abs/2205.09362

[75] M. Samvelyan, T. Rashid, C. S. de Witt, G. Farquhar, N. Nardelli, T. G. J. Rudner, C.-M. Hung, P. H. S. Torr, J. Foerster, and S. Whiteson, "The starcraft multi-agent challenge," no. arXiv:1902.04043, Dec 2019, arXiv:1902.04043 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1902.04043

[76] M. Figura, K. C. Kosaraju, and V. Gupta, "Adversarial attacks in consensus-based multi-agent reinforcement learning," in *2021 American Control Conference (ACC)*, May 2021, p. 3050–3055.

A Adversarial MARL

This section contains additional background information on adversarial MARL. MARL was removed from this work's scope, but the information below is still valuable.

A.1 RL vs MARL Frameworks

This research found few studies has explored adversarial attacks on Cooperative (c)-MARL algorithms compared to RL, where multiple RL agents cooperate to achieve a common goal. Fewer still target a specific fail state or cause the victim to follow an adversarial policy, rather than simply degrading the algorithm's performance. This distinction is important as some c-MARL algorithms present unique challenges when compared to single agents [8]. MARL algorithms described by MDP or Partially Observable (PO)MDPs[52] as in the single agent case, complicate adversarial tasks. Instead, frameworks like Decentralized (DEC)-POMDPS[53], Stochastic Games (SG)[54] and Partially Observable (PO)SGs[55] are used, though these do not model the adversary.

Probability Robust (PR), Noisy Robust (NR), or State Adversarial (SA) MDPs model the adversary only in single agent settings. Interactions between cooperative agents increases complexity in predicting team rewards and can reduce the impacts of a single agent's sub-optimal action [50]. This makes the attacker's job more difficult as simple misinterpretations of the observed state may be insufficient to significantly reduce the victim algorithm's performance. There is no demonstrated and accepted method for optimizing adversarial MARL, but the Observations Adversarial (OA) and Action Robust (AR) POSG frameworks have been proposed [8].

In [8], the attacker must chose a goal for optimization, and the following have been used in adversarial RL literature [8]:

1. Untargeted: Randomly altering the victim's behaviour.
2. Action: Altering the victim's behaviour towards specific actions.
3. Reward-based: Maximizing the adversary's or minimizing the victim's rewards.
4. State-based: Luring the victim to a target state.

For example, [49] shows that untargeted attacks can produce sub-optimal performance, demonstrating that the victim performs worse than a conventional control algorithm when attacked. Here, untargeted refers to perturbations of the observed state where there was no targeted classification, while the attacker's goal was reward-based.

Attacks are also classified by their vector [8]:

1. Action Perturbations: Altering the actions of an agent, leading it to an unexpected state. For example, physically perturbing an actuator.
2. Observation Perturbations: Changes to the environment or the perceptions of the agent's sensors, to alter the victim's behaviour. They are one way of executing an action perturbation and could be called an evasion attack.
3. Communication Perturbations alter the messages between agents and can be considered observation perturbations if the agents share observations or intermediate features. This can be enacted with an intermediary attack, where the attacker intercepts and alters all messages to and from an agent.
4. Malicious communications: New messages can be sent to the victim, rather than altered messages as above.

5. Natural Adversarial Examples: States and observations which have an adversarial effect on victims, that naturally occur in the environment.

In the example above, the vector is an observation perturbation because the victim's perception of the state is changed [49]. Note that these vectors are not mutually exclusive, e.g. communication perturbations could also perturb observations as in [72] or observation perturbations can target specific actions as action perturbations [50]. Vectors are compatible with multiple goals, wherein observation perturbations can have state-based goals as in [51] or reward-based goals as in [50]. This means that both attacks change how the victim perceives the environment. The victim is either lured into making poor decisions, which reduce its reward or the victim is lured into interacting with the environment such that it transitions into a state chosen by the attacker. The following Table provides an overview of the different frameworks classified as described above.

A.2 Selection of Adversarial MARL Literature

Some works have shown that a victim subset of agents can be coerced with perturbations of state observations into following an adversarial policy[68][50][73]. One approach is training an adversarial policy to minimize the team reward, then producing adversarial perturbations which lead one victim agent in a QMIX algorithm to take the actions specified by the adversarial policy [50]. The authors demonstrate that their approach results in lower scores than naively perturbing the victim's observations. In sum, this attack uses an adversarial policy which minimizes score. On the other hand, in [73], the authors propose modeling the state transition dynamics to optimize perturbations based on the distance between the current and target state. Together, these papers demonstrate that targeted adversarial examples can be optimized with either a trained policy or state transition model for a c-MARL victim. Training either is possible given a white-box setting (where both the environment and victim models are visible). Furthermore, the attacks are more effective than naive attacks without optimization. Both attacks can be optimized to influence fewer of the victim's actions using a trained agent with a modified reward function which penalizes adversarial actions [74]. The purpose of reducing the number of perturbed samples is evading detection, though it was not attempted in this work.

While these results are useful, they are applied in RL gyms (StarCraft Multi-Agent Challenge (SMAC)) [75] and MuJuCo respectively [51]) which do not resemble any CPS, leaving room for future research in CPS simulations or settings. Additionally, all these examples required access to the victim agents and environment, which is not guaranteed in real world applications. These techniques would be more threatening if they prove effective in black-box settings, such as using surrogate models or querying the victim agents and environment to duplicate the models. These attacks used the DEC-POMDP framework which does not model an adversary because it lacks a competitive reward signal, meaning the potential for optimization exists using different frameworks [8].

Communication between agents is a distinction between RL and MARL algorithms, as the former typically concerns agents in isolation. These communications enable an attack vector not present for independent agents where a malicious agent can influence others. One such attack involves a malicious agent in a networked actor-critic algorithm, wherein agents share the parameters of their value and policy functions [76]. Sharing parameters allows agents to maximize the collective score without disclosing sensitive information. However when the malicious agent fails to update its functions with parameters from the group, the group will instead maximize the reward of the malicious agent. The use of a pre-trained agent leaves room for researching if the same effect could be achieved with adversarial perturbations towards an adversarial policy which ignores updates from other agents. Because the attack was demonstrated in a simple grid world, a more complex

testing environment would both be useful to explore CPS applications and provide more complex observations for the generation of adversarial examples. A limited feature space presents an additional challenge to the attacker as perturbations are confined to a smaller state space [50]. As of this writing, no framework models adversarial perturbations in a networked MDP.

Instead of parameters, [72] shows how ANNs sharing intermediate features are still vulnerable to adversarial attacks. While the victims are networked image classifiers on drones and self driving vehicles rather than RL agents specifically, this exemplifies how an agent's observations are vulnerable to targeted perturbations. Since a malicious agent modifies and transmits adversarial features of its observations which alter the victim's classification decisions, the attacker modifies observed features without changing the physical environment seen by the cameras. While c-MARL algorithms can be more robust to adversarial examples, inter-agent communications present another attack vector. This attack is best classified as an untargeted observation-communication perturbation as the attacker only aimed to change the observation of the state. Using these perturbations with action, state, or reward-based goals is promising for future work given previous successes with observation perturbations in MARL. However this attack does not induce an adversarial policy on the victim, making it similar to an untargeted attack.

A.3 Summary

There is much space for future research in the adversarial MARL field. The literature on adversarial RL includes many methods of reducing the number of samples generated and reducing the number of perturbed observations which have yet to be explored in adversarial MARL. Fewer studies still consider adversarial MARL for a CPS, and as found in this review, only consider adversarial communications as the attack vector.