UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

HARNESSING DEEP REINFORCEMENT LEARNING: STUDIES IN ROBOTIC MANIPULATION, ENHANCED SEMANTIC SEGMENTATION, AND SECURING IMAGE CLASSIFIERS

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

DONG HAN
Norman, Oklahoma
2024

HARNESSING DEEP REINFORCEMENT LEARNING: STUDIES IN ROBOTIC MANIPULATION, ENHANCED SEMANTIC SEGMENTATION, AND SECURING IMAGE CLASSIFIERS

A DISSERTATION APPROVED FOR THE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY THE COMMITTEE CONSISTING OF

Acknowledgements

I would like to express my deepest gratitude to all those who have supported me throughout my PhD journey.

First and foremost, I would like to extend my heartfelt thanks to my mentor, Dr. Samuel Cheng, for his unwavering guidance, support, and encouragement. His expertise and insights have been invaluable, and his patience and dedication have been a constant source of inspiration. I am truly grateful for the opportunity to work under his mentorship.

I would also like to thank the faculty and staff at the University of Oklahoma Tulsa Campus for their support and for providing a stimulating academic environment that has fostered my growth as a researcher.

My deepest appreciation goes to my father, whose constant support and belief in me have been the bedrock of my achievements. His encouragement has been a driving force throughout my academic journey. To my mother, I extend my heartfelt thanks for her endless love and sacrifices. Her support has been invaluable and has greatly contributed to my success.

I would also like to express my gratitude to my classmates Lu Zhang, Feng Chen, Lei Yang, Zhihao Zhao, Beni Mulyana, Haoliang Zhang, Pham Huong, and Reza Babaei. Their camaraderie, collaboration, and support have enriched my academic experience and made this journey memorable.

Lastly, I owe a special debt of gratitude to my wife. Her unwavering love, patience, and understanding have been my greatest source of strength. Her support has been pivotal in helping me navigate the challenges of this journey.

Thank you all for your invaluable contributions to this milestone in my life.

Table of Contents

List of Tables

List of Figures

Abstract

This dissertation investigates the transformative potential of Deep Reinforcement Learning (DRL) in three critical domains: robotic manipulation, enhanced semantic segmentation, and the security of image classifiers. Through comprehensive exploration and analysis, this research addresses the challenges and limitations inherent in current DRL methodologies, offering novel insights and practical solutions.

In the domain of robotic manipulation, the study provides an in-depth examination of various DRL algorithms, including value-based, policy-based, and actor-critic methods. The findings highlight the specific strengths and limitations of each algorithm, guiding the selection of appropriate methods for diverse robotic applications. Additionally, the research proposes new directions for integrating multiple learning paradigms to enhance robotic adaptability and performance in complex environments.

For enhanced semantic segmentation, the dissertation develops a robust framework utilizing reinforced active learning methodologies. By integrating advanced techniques such as Dueling Deep Q-Networks (Dueling DQN), Prioritized Experience Replay, Noisy Networks, and Emphasizing Recent Experience, the framework addresses imbalanced datasets and optimizes annotation processes. Experimental results demonstrate the framework's robustness and efficiency across various domains, particularly under constrained annotation budgets.

In securing image classifiers, the research focuses on developing surrogate models capable of replicating proprietary image classification models under stringent constraints. An open-source framework integrating popular DQN extensions is introduced, demonstrating their effectiveness in enhancing attack methodologies. The evaluation of synthetic data generation techniques identifies best practices for training robust adversarial models, advancing the understanding of effective attack strategies in AI security.

This dissertation underscores the importance of improving sample efficiency, stability, generalization, and robustness in DRL algorithms. Ethical and practical considerations are addressed, ensuring the minimization of risks associated with model extraction attacks. The practical implications extend to various fields, including autonomous vehicles, robotics, and AI security, providing actionable insights for deploying DRL technologies.

The research paves the way for future work in integrating multi-paradigm learning, expanding evaluation frameworks, developing robust defense mechanisms, and leveraging advanced data generation techniques. The findings reinforce the transformative potential of DRL, shaping its future applications and ensuring its role as a critical tool in tackling complex decision-making tasks across diverse fields.

Chapter 1

Introduction

1.1 Background and Motivation

Deep Reinforcement Learning (DRL) is a powerful subset of machine learning where algorithms learn to make a sequence of decisions by interacting with a complex environment to achieve a specific goal. This approach combines the insight from reinforcement learning (RL) with the representational abilities of deep neural networks, enabling the solution of problems that were previously intractable due to their high dimensionality and complexity [2]. Deep reinforcement learning is at the forefront of AI research due to its ability to solve problems that involve sequential decision-making, which is crucial for many applications ranging from autonomous vehicles to strategic game-playing [3]. The integration of deep reinforcement learning into fields such as semantic segmentation, model extraction attacks, and robotic manipulation demonstrates its versatility and powerful capability to improve decision-making processes.

In RL, an agent learns by interacting with an environment, receiving rewards or penalties based on its actions, and gradually refining its policy to maximize cumulative rewards. This learning process involves exploring different actions, evaluating their outcomes, and converging on an optimal strategy over time [4]. By integrating deep learning, DRL enhances this process by utilizing neural networks to represent complex policies, allowing agents to handle high-dimensional inputs and learn abstract features. This combination provides a powerful tool for tackling complex decision-making tasks in various domains. The motivation for this dissertation stems from the need to overcome the limitations of existing DRL techniques and extend their applications to new domains. Challenges such as sample efficiency, stability, and generalization hinder the broader adoption of DRL. This research aims to address these challenges, advancing the field and expanding its capabilities. The study focuses on improving the performance and robustness of DRL algorithms, ensuring they can handle diverse environments and tasks, from real-time strategic planning to adaptive problem-solving. By doing so, the dissertation aims to reinforce DRL's role in driving innovations in AI technologies, making it an indispensable tool for tackling complex decision-making tasks in various fields.

1.2 Problem Statements and Contributions

In this section, I present the above-mentioned issues as Problem Statements (PS) and present the outline of proposed solutions in this dissertation's contributions (C).

PS1: The primary problem addressed in this dissertation study is the development of surrogate models capable of replicating the functionality of proprietary image classification models under stringent constraints. Practical challenges identified in current methodologies include (1) Limited access to the victim model's outputs, hindering comprehensive training data acquisition. (2) The need to maximize model fidelity with minimal interaction with the victim model, minimizing both ethical and legal risks associated with excessive query volumes.

C1-A: We develop an open-source framework that integrates popular DQN extensions such as Prioritized Experience Replay, Double DQN, and Dueling DQN. We illustrate how these can be used to enhance attack methodologies against neural networks. To the best of our knowledge, this work presents the first attempt to leverage DQN extensions to model extraction attacks, which has not been explored in the existing literature.

C1-B: We conduct comparative experiments that benchmark the effectiveness of these methods against traditional approaches, providing a clear demonstration of their relative performance and advancing the understanding of effective attack strategies in the realm of AI security.

C1-C: By evaluating different techniques for synthetic data generation, such as FGSM and LinfPGD, and comparing their effectiveness in the context of model extraction, our work identifies best practices and guidelines for practitioners. This helps in refining the approach to generating synthetic data, which is crucial for training robust adversarial models.

PS2: Inspired by a burgeoning interest in active learning for semantic segmentation, considerable research has explored various methodologies to optimize annotation processes and address imbalanced datasets, with emerging techniques reporting promising efficiencies. However, the bulk of these studies fail to confront the pivotal challenge of training large models under constrained annotation budgets. Overcoming this barrier is essential for deploying enhanced semantic segmentation methods effectively, not only within academic realms but also across practical applications in diverse fields.

C2: We propose a comprehensive framework utilizing reinforced active learning methodologies for semantic segmentation. While existing solutions have demonstrated substantial progress in tackling dataset imbalances and improving annotation efficiency, our framework offers a robust performance across diverse domains, particularly excelling in scenarios with constrained annotation budgets. Our approach enhances the precision of semantic segmentation tasks.

PS3: Within the current landscape of robotics research, practical solutions for effectively applying deep reinforcement learning (DRL) to complex robotic manipulation tasks remain scarce. This shortfall is primarily due to the intricate nature of real-world environments where robots operate and the diverse array of manipulative actions required. Additionally, the challenge is compounded by the limited integration of multiple learning paradigms—supervised, unsupervised, and reinforcement learning—which are essential for enhancing robotic adaptability and performance across varied tasks.

C3: We meticulously catalogues and examines the extensive array of deep reinforcement learning algorithms that have been proposed to improve the adaptability and precision of robots in dynamic environments. This analysis delineates the evolving landscape of value-based methods, policy-based approaches, and actor–critic techniques, providing a granular understanding of each method's strengths and limitations in the context of robotic tasks. Moreover, we explore the frontier challenges and propose structured directions for future research, thus charting a course for forthcoming advancements in robotic manipulation.

1.3 Current Publications

Peer-Reviewed Journal Articles:

J1: Han D, Babaei R, Shangqing Z, Samuel C. Exploring the Efficacy of Learning Techniques in Model Extraction Attacks on Image Classifiers: A Comparative Study[J]. Appl. Sci. 2024, 14, 3785. https://doi.org/10.3390/app14093785.

J2: Han D, Huong P, Samuel C. Enhancing Semantic Segmentation through Reinforced Active Learning: Combatting Dataset Imbalances and Bolstering Annotation Efficiency[J]. Journal of Electronic and Information Systems, 2023, 5(2): 45-60.

J3: Han D, Mulyana B, Stankovic V, et al. A Survey on Deep Reinforcement Learning Algorithms for Robotic Manipulation[J]. Sensors, 2023, 23(7): 3762.

J4: Han D, Wang S, Jiang C, et al. Trends in biomedical informatics: automated topic analysis of JAMIA articles[J]. Journal of the American Medical Informatics Association, 2015, 22(6): 1153-1163.

1.4 Organization

The rest of the dissertation is organized as follows: Chapter 2 delves into the theoretical underpinnings of DRL, discussing its fundamentals, the nuances of value-based, policy-based, and actor-critic methods, and the inherent challenges in the field. The focus shifts in Chapter 3 to the application of DRL in robotic manipulation, exploring different DRL algorithms and the specific challenges and future directions in this area. Chapter 4 examines how DRL can enhance semantic segmentation, addressing issues like dataset imbalances and annotation efficiency through reinforced active learning. In Chapter 5, the dissertation explores the security aspects of DRL by analyzing how to secure image classifiers against model extraction attacks, evaluating various DQN extensions and synthetic data generation methods for building robust defense mechanisms. Finally, Chapter 6 concludes the dissertation with a summary of the contributions, insights into future research directions, and final reflections on the study. Each chapter concludes with a summary, ensuring clarity and reinforcing the key points discussed, thereby weaving a coherent narrative throughout the dissertation.

Chapter 2

Theoretical Background on Deep Reinforcement Learning

2.1 Fundamentals of Reinforcement Learning

Reinforcement Learning: Semi-Supervised Learning is a subset and application of Machine Learning. Reinforcement Learning: Reinforcement learning is another name for semi-supervised learning, as it is a way to compel an agent to take steps and interact with an environment to receive the maximum summation of rewards. In this way, a machine solves the problem by trial and error. Therefore, the reward or the penalty corresponds respectively to the artificial intelligence of a machine that is meant to do the same things as what is in the mind of the programmer. It focuses on the total amount of reward to be obtained in the long run in a particular state.

The said problems are aimed to be solved by the mathematical structure of RL. While the designer sets the reward policy, that is, the rules of the game, he gives no hints or advice to the model for solving the game. It is up to the model to work out how to execute the challenge to optimize the reward, starting with completely random failed-to-solve games and ending with advanced techniques and superhuman abilities. Reinforcement learning is actually the most powerful way to express imagination of the system by leveraging the strength of research and multiple trial attempts. Unlike humans, artificial intelligence will gain experience from thousands of parallel gameplays if a reinforcement learning algorithm is run on a sufficiently efficient computing infrastructure [5].

Reinforcement learning may sound like a very complicated word at first, but it is essentially goal-driven nomenclature for the set of algorithms that an agent needs to learn in order to optimize some complex dimension of interest over the course of a run of time. For example, maximizing the amount of points collected in a game over several moves. These kinds of algorithms can start from a blank slate and do superhumanly well, given the right learning settings. This is how they engage the principle of reinforcement, just like a pet is incentivized by scolding and treating: the algorithms are penalized for wrong choices and rewarded for right choices.

Reinforcement learning can be described in terms of an agent, its environment, state, actions, and rewards. The definitions of these terms are listed below. The general convention will be to use uppercase letters for sets of things and lowercase letters for one specific instance of that thing, so for example A will be all the possible actions that can be taken, and a will be a specific action that is in the set.

Agent (the player): who moves around doing stuff; for example, a drone making a delivery, or Super Mario navigating a video game. The algorithm is the agent. It may be helpful to consider that in life, the agent is you.

Action (A): A is the set of all possible moves the agent takes. Action is what an agent can do in each state. Some environments, like Atari and Go, have discrete action spaces, where only a finite number of moves are available to the agent. Other environments, like where the agent controls a robot in a physical world, have continuous action spaces. In continuous spaces, actions are real-valued vectors.

Discount factor: The discount factor is multiplied by future rewards as discovered by the agent in order to dampen the rewards' effect on the agent's choice of action. It is designed to make future rewards worth less than immediate rewards; i.e. it enforces a kind of short-term hedonism in the agent. On an intuitive level: cash now is better than cash later. We often use discount factors in estimating value functions.

Environment: The environment is the world that the agent lives in and interacts with. The environment takes the agent's current state and action as input, and returns as output the agent's reward and its next state. At every step of interaction, the agent sees a (possibly partial) observation of the state of the world, and then decides on an action to take. The environment changes when the agent acts on it, but may also change on its own.

State (S): A state is a concrete and immediate situation in which the agent finds itself. The states are the locations. The location in which a particular robot is present in a particular instance of time will denote its state. It can be the current situation returned by the environment or any future situation. Literature that teaches the basics of RL tends to use very simple environments so that all states can be enumerated. This simplifies value estimates into basic rolling averages in a table, which are easier to understand and implement.

Reward (R): A reward is a feedback by which we measure the success or failure of an agent's actions in a given state. The goal of the agent is to maximize some notion of cumulative reward over a trajectory. For example, in a video game, when Mario touches a coin, he wins points. From any given state, an agent sends output in the form of actions to the environment, and the environment returns the agent's new state (which resulted from acting on the previous state) as well as rewards if there are any. Rewards can be immediate or delayed. They effectively evaluate the agent's action.

Policy (π): A policy is a rule used by an agent to decide what actions to take. Roughly speaking, a policy is a mapping from perceived states of the environment to actions to be taken when in those states. In deep RL, we deal with parameterized policies: policies whose outputs are computable functions that depend on a set of parameters.

Value (V): It is the expected long-term return (considering the discount factor). As opposed to the short-term reward given by reward function R. $V_\pi(s)$ is the expected long-term return of the current state under policy π. We discount rewards or lower their estimated value, the further into the future they occur. See the discount factor. And remember Keynes: 'In the long run, we are all dead.' That's why you discount future rewards. It is useful to distinguish.

Q-value or action-value (Q): It is similar to Value, except that it takes an extra parameter, the current action a. $Q_\pi(s, a)$ refers to the long-term return of an action taking action under policy π from the current state S. Q maps state-action pairs to rewards. Note the difference between Q and policy.

Trajectory: A trajectory is a sequence of states and actions in the world. From the Latin ‘to throw across’. The life of an agent is but a ball tossed high and arching through space-time unmoored, much like humans in the modern world. Trajectories are also frequently called episodes or rollouts.

The agent-environment interaction process is shown in Fig 2.1. The agent takes actions and receives feedback from the environment in the form of *rewards* or *punishments*. The agent uses this feedback to adjust its behavior and improve its performance over time.

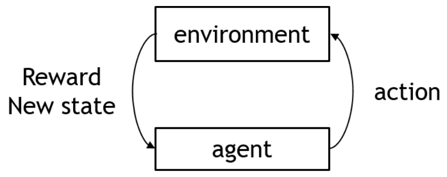


Figure 2.1: Block diagram of typical RL

.

An autonomous agent observes state $s(t)$ at time-step t, and then interacts with the environment using an action $a(t)$, reaching the next state $s(t+1)$ in the process. Once a new state has been achieved, the agent receives a reward correlated with that state $r(t+1)$. The agent's goal is to find an optimal policy, i.e., optimal action in any given state. Unlike other types of machine learning – such as supervised and unsupervised learning – reinforcement learning can only be thought about sequentially in terms of state-action pairs that appear one after the other.

RL assesses actions by the outcomes, i.e., states, they achieve. It is goal-oriented, and seeks to learn sequences of actions that will lead an agent to accomplish its goal or optimize its objective function. An example of the RL objective function is:

$$\sum_{t=0}^{t=\infty} \gamma^t r(s(t), a(t)) \tag{2.1}$$

This objective function measures all the rewards we will get from running through the states exponentially increasing the weight γ.

Two important concepts of RL are Monte Carlo learning, a naive idea where the agent interacts with the environment and learns about the states and rewards, and Temporal difference (TD) learning, i.e., updating the value in every time step and without requiring to wait to update the values till the end of the episode.

A Markov decision process (MDP) is defined as a tuple $\langle S, A, r, T, \gamma \rangle$ where S stands for a set of states; A, for actions; r, $S \times A \to R$, for the function specifying a reward of taking an action in a state; $T : S \times A \times S \to R$, for the state-transition function; and γ, for the discount factor implying that a reward obtained in the future is worth a smaller amount than an immediate reward. Solving an MDP involves finding a policy that determines the optimal action for each state, with the goal of maximizing the long-term discounted expected reward. This policy should be optimal with respect to the MDP's reward structure and discount factor.

Although it is difficult to make a standardized classification of RL algorithms due to their wide modularity, many current studies tend to divide them into value-based, policy-based, and actor-critic algorithms (see Fig 2.2).

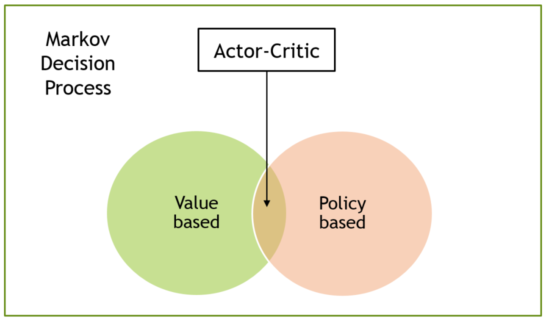


Figure 2.2: Types of RL algorithms

.

2.2 Deep Reinforcement Learning: An Overview

Deep Reinforcement Learning (DRL) merges the decision-making capability of Reinforcement Learning (RL) with the powerful representation learning ability of deep neural networks. This combination enables the solution of complex, high-dimensional problems that were previously intractable. DRL leverages the strengths of deep learning to handle raw, high-dimensional inputs such as images, and it uses reinforcement learning to learn optimal policies through trial and error.

Nevertheless, deep learning-based reinforcement learning has been a challenge. In particular, the triumphs of deep learning have been somewhat disconnected from the reinforcement learning problem setting. Hence, reinforcement learning requires that the agent, in its most practical form, be able to learn from a scalar reward signal that is sparse, noisy, and delayed. Perhaps most importantly, the ability to delay rewards for long sequences of time steps, sometimes thousands of time steps, is one particularly daunting fact as one compares this with the direct association of inputs and targets that is standard in supervised learning [6].

Deepmind [7] uses deep reinforcement learning for the ultimate in-world simulation, supporting companies to design their facilities, turn their contact centers into the superhuman workforce, design warehouses, and much more in an optimal way.

2.3 Value-Based Methods in DRL

Value-based methods in DRL focus on estimating the value function, which represents the expected cumulative reward of states or state-action pairs. The most common value-based methods include:

2.3.1 Q-learning

Q-Learning [8] is a value-based TD method of Reinforcement Learning that uses Q-values (also called state action values) to iteratively develop the actions of the learning agent. Q-learning learns the Bellman action-value function $Q(s, a)$: how good it is to take an action at a given state.

$$Q(s,a) = r(s,a) + \gamma \max_{a} Q\left(s', a\right). \tag{2.2}$$

The Bellman action-value equation describes how to calculate the Q-value for an action taken from a particular state, s. It is calculated as the sum of the immediate reward for the current state and the discounted optimal Q-value for the next state, denoted by γ. In Q-learning, the Q-value is updated using the following rule: the new Q-value is equal to the old Q-value plus the temporal difference error. This update can be framed as trying to minimize a loss function, such as the mean squared error loss:

$$\mathrm{L} \;= \left(r + \gamma \max_{a'} Q\left(s', a'; \theta'\right) - Q(s, a; \theta)\right)^2 . \tag{2.3}$$

2.3.2 SARSA

The SARSA [9] algorithm is a policy-based variant of the well-known Q-Learning algorithm. Unlike Q-Learning, which is an off-policy technique that learns the Q value using a greedy approach, SARSA is an on-policy technique that uses the action taken by the current policy to learn the Q value. To update Q values, we use:

$$Q(s, a) = Q(s, a) + \alpha \left[R + \gamma Q\left(s', a'\right) - Q(s, a)\right] . \tag{2.4}$$

SARSA is a type of on-policy reinforcement learning algorithms, meaning that it learns the value of actions based on the policy that is currently being followed. In SARSA, the next action, a', is chosen using the same epsilon-greedy policy as the action that led to the current state, s'. One advantage of this approach is that SARSA is able to learn a near-optimal policy while still allowing for some exploration. However, if the goal is to learn the optimal policy, it may be necessary to carefully tune the decay rate of the epsilon value in the epsilon-greedy action selection process. On the other hand, Q-learning is an off-policy algorithm that learns the optimal policy directly. While this can be more efficient in certain cases, it also has a higher per-sample variance and can be more difficult to converge when used with neural networks.

2.3.3 Deep Q-learning (DQN)

One problem with traditional Q-learning is that the size of the Q table grows exponentially with the number of states and actions, making it impractical for many problems. To address this, deep Q-learning (DQN) was introduced by Mnih et al. [6], which uses a neural network to approximate the Q-values. As a universal function approximator, the neural network is able to capture the relationships between states and actions more efficiently than the Q table.

However, one issue with using a neural network to learn the Q-values is that the update rule (Equation 4) depends on the values produced by the network itself, which can make convergence difficult. To address this, the DQN algorithm introduces the use of a replay buffer and target networks. The replay buffer stores past interactions as a list of tuples, which can be sampled to update the value and policy networks. This allows the network to learn from individual tuples multiple times and reduces dependence on the current experience. The target networks are time-delayed copies of the policy and Q networks, and their parameters are updated according to the following equations:

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'} \tag{2.5}$$

$$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'} \tag{2.6}$$

where $\theta^{\mu'}$ and $\theta^{Q'}$ denote the parameters of the policy and Q networks, respectively. The Figure 2.3 shows the overall workflow of Deep Q-Network.

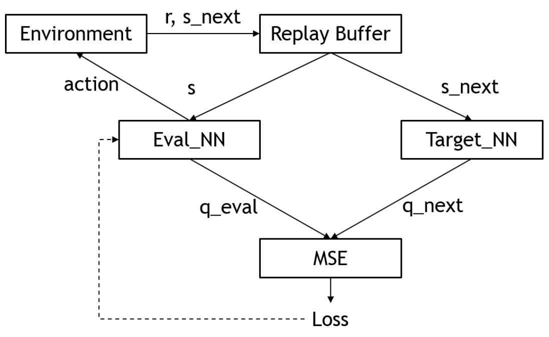


Figure 2.3: Flowchart of DQN

.

2.3.4 Double Deep Q-Learning (Double DQN)

One issue with the DQN algorithm is that it tends to overestimate the true rewards, leading to inflated Q-values. To address this, the Double DQN algorithm [10] introduces a modification to the Bellman equation used in DQN. Instead of using the same equation, the action selection and action evaluation are decoupled in the following way:

$$Q(s, a; \theta) = r + \gamma Q\left(s', \operatorname{argmax}_{a'} Q\left(s', a'; \theta\right); \theta'\right) \tag{2.7}$$

Here, the main neural network, θ, determines the best next action, a', while the target network is used to evaluate this action and compute its Q-value. This simple change has been shown to reduce over-estimations and lead to better final policies.

2.3.5 Dueling Deep Q-Learning(Dueling DQN)

The Dueling DQN algorithm introduced by Wang et al. [11] seeks to improve upon traditional DQN by decomposing the Q-values into two separate components: the value function, $V(s)$, and the advantage function, $A(s, a)$. The value function represents the expected reward for a given state, s, while the advantage function reflects the relative advantage of taking a particular action, a, compared to other actions. By combining these two functions, it is possible to compute the full Q-values for each state-action pair.

To implement this decomposition, the Dueling DQN algorithm introduces a neural network with two separate output layers, one for the value function and one for the advantage function. These outputs are then combined to produce the final Q-values. This modification allows the network to learn more efficiently in situations where the exact values of individual actions are not as important, as it can focus on learning the value function for the state.

2.4 Policy-based RL

Policy gradient (PG) methods are widely used reinforcement learning algorithms that are particularly well-suited to situations with continuous action spaces [12]. The goal of an RL agent using a PG method is to maximize the expected reward, $J\left(\pi_\theta\right)=\underset{\tau \sim \pi_a}{\mathrm{E}}[R(\tau)]$, by adjusting the policy parameters, θ. A standard approach to finding the optimal policy is to use gradient ascent, in which the policy parameters are updated according to the following rule:

$$\theta_{t+1}=\theta_t+\alpha \nabla J\left(\pi_{\theta_t}\right) \tag{2.8}$$

where α is the learning rate and $\nabla J\left(\pi_\theta\right)$ is the policy gradient. This gradient can be further expanded and reformulated as:

$$\nabla J\left(\pi_\theta\right)=\underset{\tau \sim \pi_\theta}{\mathrm{E}}\left[\sum_{t=0}^T \nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right) R(\tau)\right] . \tag{2.9}$$

In PG methods, the policy function, which maps states to actions, is learned explicitly and actions are selected without using a value function.

2.4.1 Vanilla Policy Gradient (VPG)

In RL, it is often more important to understand the relative advantage of a particular action, rather than its absolute effectiveness. The advantage function, $A_\pi(s, a)$, captures this idea by measuring how easier it is to take a specific action, a, in state s compared to randomly selecting an action according to the policy, π, considering that the policy will be followed indefinitely thereafter. Mathematically, the advantage function is defined as:

$$A_\pi(s, a)=Q_\pi(s, a)-V_\pi(s) \tag{2.10}$$

where $Q_\pi(s, a)$ is the action-value function and $V_\pi(s)$ is the state-value function for the policy π. Using this definition, the vanilla policy gradient (VPG) algorithm [12] can be written as:

$$\nabla J\left(\pi_\theta\right)=\underset{\tau \sim \pi_\theta}{\mathrm{E}}\left[\sum_{t=0}^T \nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right) A_\pi(s, a)\right] \tag{2.11}$$

The figure 2.4 describe the general workflow of vanilla policy gradient.

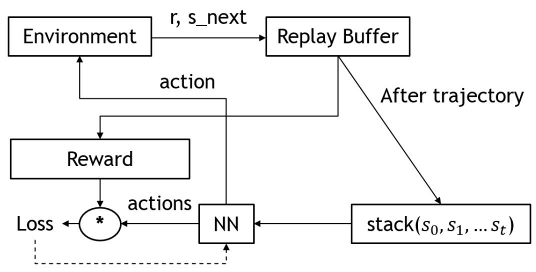


Figure 2.4: Flowchart of Vanilla Policy Gradient

.

2.4.2 Trust Region Policy Optimization (TRPO)

In policy gradient methods, we aim to optimize a policy objective function, such as the expected cumulative reward, using gradient descent. These methods are well-suited for continuous and large state and action spaces, but can be sensitive to the learning rate. A small learning rate may result in vanishing gradients, while a large learning rate may cause exploding gradients. Trust Region Policy Optimization (TRPO) [13] was introduced as a solution to this problem, by constraining the optimization of the policy to a trust region. This region is defined as the area where local approximations of the function are accurate, and the maximum step size is determined within it. The trust region is then iteratively expanded or shrunk based on how well the new approximation performs.

The policy update in TRPO is given by the following optimization problem, which uses the Kullback–Leibler (KL) divergence between the old and new policies as a measure of change:

$$\nabla J\left(\pi_\theta\right)=\underset{\tau \sim \pi_\theta}{\mathrm{E}}\left[\frac{\pi \theta\left(a_t \mid s t\right)}{\pi \theta_{\text {old }}\left(a_t \mid s t\right)} \hat{A} t\right] \tag{2.12}$$

$$\text{subject to } \underset{\tau \sim \pi\theta}{\mathrm{E}}\left[\mathrm{KL}\left[\pi\theta_{\text{old}}\left(\cdot \mid st\right), \pi\theta(\cdot \mid st)\right]\right] \leq \delta \tag{2.13}$$

where δ is the size of the trust region, and the KL divergence between the old and new policies must be less than δ. This optimization problem can be solved using the conjugate gradient method.

2.4.3 Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) [14] is an algorithm that aims to address the overhead issue of TRPO by incorporating the constraint into the objective function as a penalty. Instead of adding a separate constraint, the KL divergence between the old and new policies is subtracted from the objective function, multiplied by a constant C. This allows for the use of simple stochastic gradient descent to optimize the function:

$$\underset{\tau \sim \pi_\theta}{\mathrm{E}}\left[\frac{\pi\theta\left(a_t \mid st\right)}{\pi\theta_{\text{old}}\ \left(a_t \mid st\right)} \hat{A}t\right] - C \cdot \overline{\mathrm{KL}}\pi\theta_{\text{old}}\left(\pi\theta\right) \tag{2.14}$$

One challenge of PPO is choosing the appropriate value for C. To address this, the algorithm updates C based on the magnitude of the KL divergence. If the KL divergence is too high, C is increased, and if it is too low, C is decreased. This allows for effective optimization over the course of training.

2.5 Actor Critic

Actor-Critics [15] are RL algorithms that combine elements of both value-based and policy-based methods. In this approach, *the actor*, a policy network, proposes an action for a given state, while *the critic*, a value network, evaluates the action based on the state-action pair. The critic uses the Bellman equation to learn the Q-function and the actor is updated based on the Q-function to train the policy. This allows the actor-critic approach to take advantage of the strengths of both value-based and policy-based methods. The figure 2.5 illustrate the network architecture of actor critic.

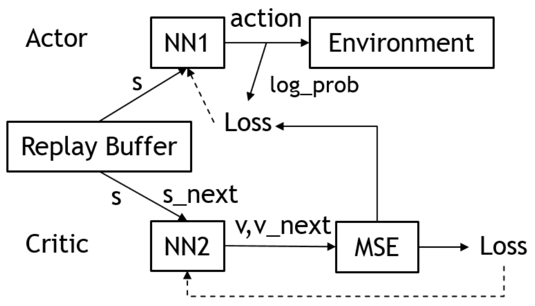


Figure 2.5: Flowchart of Actor Critic

2.5.1 Advantage Actor-Critic (A2C)

In the Advantage Actor-Critic (A2C) algorithm [16], the critic is trained to estimate the Advantage function instead of the Q-function. This allows the evaluation of an action to consider not only its success, but also how much better it is compared to other actions. Using the Advantage function can help reduce the high variance of policy networks and improve the stability of the model.

2.5.2 Asynchronous Advantage actor-Critic (A3C)

The Asynchronous Advantage actor-Critic (A3C) algorithm [16], released by DeepMind in 2016, is a highly efficient and effective reinforcement learning algorithm that has outperformed other methods such as DQN on many tasks. A key feature of A3C is its asynchronous nature, which allows multiple independent agents (networks) with their own weights to interact with different copies of the environment in parallel, allowing for more efficient exploration of the state-action space. A3C has proven to be a robust and reliable method, achieving high scores on standard reinforcement learning tasks.

2.5.3 Deep Deterministic Policy Gradient (DDPG)

Deep Deterministic Policy Gradient (DDPG) [17] is a reinforcement learning technique that combines both deep Q learning (DQN) [18] and deterministic policy gradients (DPG) [19]. DDPG is an actor-critic technique, it uses two neural networks: a deterministic policy network and a critic (Q) network. The policy network simply performs the gradient ascent to solve Equation (4). Note that the critic parameters are frozen as constants. The critic network is updated similarly to equation (3). Nevertheless, in DDPG, the updated Q values are calculated by Bellman equation (2) with the target Q network and target policy network. Then, we minimize the mean squared error loss between the original Q value and the updated Q value:

$$\mathrm{L} \ = \frac{1}{N} \sum_{i} \left(Q_{old} - \left(r(s,a) + \gamma \max_{a} Q_{target}\left(s',a\right) \right) \right)^{2} \tag{2.15}$$

Since the policy is deterministic, it suffers from inefficient exploration when the agent was to explore the environment. To improve DDPG policy, the authors add Ornstein-Uhlenbeck noise [20] to the selected actions during training. However, more recent research implies that uncorrelated, zero-mean Gaussian noise is effective. The figure 2.6 shows the update rule of Deep Deterministic Policy Gradient.

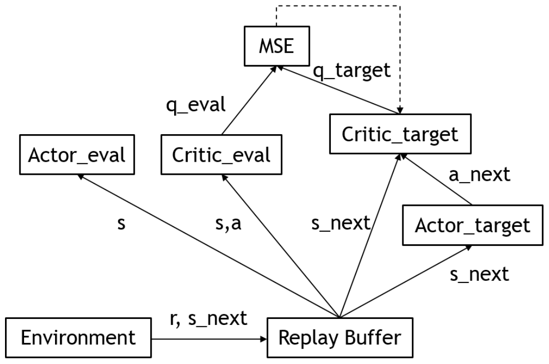


Figure 2.6: Flowchart of Deep Deterministic Policy Gradient

2.5.4 Twin Delayed Deep Deterministic Policy Gradients (TD3)

Twin Delayed Deep Deterministic Policy Gradients (TD3) [21] is the successor to the DDPG. Although DDPG is capable of providing excellent results, it has its drawbacks. Like many RL algorithms training DDPG can be unstable and heavily reliant on finding the correct hyper parameters for the current task. This is caused by the algorithm continuously over estimating the Q values of the critic (value) network. These estimation errors build up over time and can lead to the agent falling into a local optima or experience catastrophic forgetting. TD3 addresses this issue by focusing on reducing the overestimation bias seen in previous algorithms.

TD3 has three main features that help to solve the aforementioned problems. Firstly, TD3 uses two critic networks instead of one, and uses the smaller of the two Q-values as the targets in the Bellman error loss functions. This helps to reduce overestimation by ensuring that the Q-value targets are more conservative. Secondly, TD3 updates the policy (and target networks) less frequently than the Q-function, which is called twin delay. The paper suggests one policy update for every two Q-function updates. This helps to reduce over-fitting and improve the stability of the algorithm. Finally, TD3 adds noise to the target action, to make it harder for the policy to exploit Q-function errors by smoothing out Q along changes in action. This helps to improve the robustness of the algorithm and reduce its reliance on hyperparameter tuning.

2.5.5 Soft Actor-Critic (SAC)

Developed jointly by UC Berkeley and Google [22], Soft Actor-Critic (SAC) is a cutting-edge reinforcement learning algorithm. It employs a maximum entropy approach in which the goal is to determine the optimal policy that maximizes both the expected long-term reward and the long-term entropy [23]. The objective function for this maximum entropy RL is displayed below:

$$J\left(\pi_\theta\right)=E_{\pi_\theta}\left[\sum_{t=0}^{T-1} \gamma^t R\left(s_t, a_t\right)+\alpha H\left(\pi\left(. \mid s_t\right)\right)\right] \tag{2.16}$$

The maximum entropy reinforcement learning objective is used to encourage exploration. This is achieved by promoting policies that assign equal probabilities to actions that have similar or almost equal Q-values, which leads to higher entropy. By explicitly encouraging exploration in this way, SAC is able to effectively explore the state-action space and find optimal policies.

SAC makes use of three networks: a state value function V parameterized by

ψ, the second one is a policy function parameterised by ϕ, and the last network represents soft state-action value function parameterised by θ. We train the state Value network by minimizing the following error:

$$E_{\mathbf{s}_t\sim\mathcal{D}}\left[\frac{1}{2}\left(V_\psi\left(\mathbf{s}_t\right)-\mathbb{E}_{\mathbf{a}_t\sim\pi_\phi}\left[Q_\theta\left(\mathbf{s}_t,\mathbf{a}_t\right)-\log\pi_\phi\left(\mathbf{a}_t\mid\mathbf{s}_t\right)\right]\right)^2\right] \tag{2.17}$$

The loss function implies that across all the states that we sample from our experience replay buffer, we need to decrease the squared difference between the prediction of our value network and the expected prediction of the Q function minus the entropy of the policy function π. We train the soft Q network by minimizing the following error:

$$J_Q(\theta)=\mathbb{E}_{\left(\mathbf{s}_t,\mathbf{a}_t\right)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_\theta\left(\mathbf{s}_t,\mathbf{a}_t\right)-\hat{Q}\left(\mathbf{s}_t,\mathbf{a}_t\right)\right)^2\right] \tag{2.18}$$

where

$$\hat{Q}\left(\mathbf{s}_t,\mathbf{a}_t\right)=r\left(\mathbf{s}_t,\mathbf{a}_t\right)+\gamma\mathbb{E}_{\mathbf{s}_{t+1}\sim p}\left[V_{\bar{\psi}}\left(\mathbf{s}_{t+1}\right)\right] \tag{2.19}$$

In simple words, training is done by minimizing the squared difference between predicted Q value and the reward plus the discounted expectation of state-value of next state. Finally, policy network learning is based on:

$$J_\pi(\phi)=\mathbb{E}_{\mathbf{S}_t\sim\mathcal{D}}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi_\phi\left(\cdot\mid\mathbf{s}_t\right)\|\exp\left(Q_\theta\left(\mathbf{s}_t,\cdot\right)\right)\right)\right] \tag{2.20}$$

attempting to make the distribution of our policy function look more like the distribution of the exponentiation of our Q function.

After the discussion of RL algorithms, there are two possible approaches for finding the optimal policy: on-policy and off-policy. These two terms are used to describe how, in a general sense, the agent learns and behaves during the training

Table 2.1: The category of different RL algorithms.

Methods	**Value-based**	**Policy-based**	**Actor-critic**
On-Policy	Monte Carlo Learning/ SARSA	REINFORCE(PG)	A2C/ A3C/ TRPO/ PPO
Off-Policy	Q-learning/ DQN Double/ Dueling		DDPG/ TD3/ SAC

phase, as well as the two main ways that an agent can go about learning/behaving. Table 2.1 shows the category of different RL algorithms.

While RL has proven to be a powerful approach to solving a wide range of problems, there are several different algorithms that have been developed to address different types of environments and learning objectives. In this context, it is important to understand the strengths and limitations of these algorithms in order to make informed decisions about which algorithm to use for a particular application. Table 2.2 describes the strength and limitations of different RL algorithms.

2.6 Challenges in Deep Reinforcement Learning

Despite the remarkable successes and wide-ranging applications of Deep Reinforcement Learning (DRL), several significant challenges hinder its broader adoption and effectiveness. Addressing these challenges is crucial for advancing the field and realizing the full potential of DRL.

2.6.1 Sample Efficiency

One of the primary challenges in DRL is sample efficiency, which refers to the number of interactions with the environment required to learn an effective policy. Many DRL algorithms require millions of interactions to achieve good performance, making them impractical for real-world applications where obtaining such data can be expensive or time-consuming. Improving sample efficiency involves developing methods that can learn from fewer interactions, such as using model-based approaches where a model of the environment is learned and used to generate synthetic experiences.

2.6.2 Stability and Convergence

Training deep neural networks in the context of reinforcement learning can be unstable and may not always converge to an optimal solution. This instability arises from several factors, including the non-stationary nature of the environment, the correlation between consecutive experiences, and the high variance in gradient estimates. Techniques such as experience replay, target networks, and more sophisticated optimization methods like Proximal Policy Optimization (PPO) have been developed to mitigate these issues, but achieving stable and reliable convergence remains a challenging task.

2.6.3 Exploration and Exploitation

Balancing exploration (trying new actions to discover their effects) and exploitation (choosing actions that are known to yield high rewards) is a fundamental challenge in reinforcement learning. Effective exploration is crucial for discovering optimal policies, especially in environments with sparse rewards or high-dimensional action spaces. Strategies like ϵ-greedy, softmax action selection, and intrinsic motivation (where the agent is driven by curiosity) are used to address this trade-off, but finding the right balance remains an open problem.

2.6.4 Generalization

DRL models often struggle to generalize beyond the specific tasks and environments they were trained on. Overfitting to the training environment can result in poor performance in new, unseen environments. This lack of generalization limits the applicability of DRL in real-world scenarios where the environment may change or the agent may encounter novel situations. Research in meta-learning, transfer learning, and robust policy learning aims to enhance the generalization capabilities of DRL agents.

2.6.5 Reward Shaping

Designing appropriate reward signals is critical for guiding the learning process. Poorly designed rewards can lead to unintended behaviors, where the agent optimizes for the given reward but not the intended outcome. This challenge is particularly acute in complex tasks where defining a reward function that captures the desired behavior is difficult. Reward shaping, which involves adding auxiliary rewards to guide the agent towards desirable behaviors, can help but requires careful tuning to avoid introducing bias.

2.6.6 Computational Resources

DRL algorithms are computationally intensive, requiring significant processing power and memory. Training deep neural networks involves substantial computational overhead, and the need for extensive interaction with the environment further exacerbates this demand. Access to high-performance computing resources, such as GPUs and TPUs, is often necessary, making DRL research and deployment expensive and less accessible.

2.6.7 Scalability

Scalability is another challenge, particularly in multi-agent settings or large-scale environments. Coordinating multiple agents and ensuring efficient learning in environments with a vast number of states and actions is complex. Techniques such as hierarchical reinforcement learning, where the problem is decomposed into smaller, more manageable sub-problems, offer potential solutions but require further development.

2.6.8 Interpretability

Understanding the decision-making process of DRL agents is challenging due to the black-box nature of deep neural networks. This lack of interpretability makes it difficult to trust and validate the actions of DRL agents, especially in safety-critical applications like healthcare and autonomous driving. Research in explainable AI (XAI) aims to develop methods for interpreting and explaining the behavior of DRL agents, providing insights into their decision-making processes.

2.7 Summary

Deep Reinforcement Learning represents a powerful paradigm in machine learning, combining the strengths of reinforcement learning and deep neural networks. This chapter has provided an overview of the fundamental concepts of RL, the integration of deep learning into RL, and the various methods and challenges associated with DRL. Understanding these theoretical foundations is crucial for advancing the field and developing robust and efficient DRL algorithms capable of tackling complex decision-making tasks across diverse domains.

Table 2.2: The strength and limitations of different RL algorithms.

RL Algorithm	**Strengths**	**Limitations**
Q-Learning	Straightforward and easy to use	Suffers from slow convergence
SARSA	Manages stochastic environments and policies	Slow convergence
Deep Q-Networks (DQN)	Handles high-dimensional state spaces, directly learns from sensory data	Difficulty using in continuous action spaces, overestimation of Q-values
Policy Gradient	Handles continuous action spaces, learns stochastic policies	High variance in gradients, sensitive to hyperparameters
Actor-Critic	Combines the advantages of policy gradient and value-based approaches	Difficult to balance exploration and exploitation, high variance in gradients
DDPG	Handles continuous action spaces	Instability, overestimation bias
SAC	Optimizes entropy-regularized objectives	Computationally expensive

Chapter 3

Deep Reinforcement Learning for Robotic Manipulation

3.1 Introduction

Robotic manipulation has become a cornerstone in various industries, ranging from manufacturing and healthcare to logistics and space exploration [24]. The ability of robots to perform complex tasks such as assembling products, picking and placing objects, and performing surgeries has significantly enhanced productivity, reduced costs, and improved the quality of products and services [1]. In recent years, the integration of artificial intelligence (AI) and robotics has further accelerated these advancements, particularly through the application of deep reinforcement learning (DRL) [25].

Deep reinforcement learning, a combination of reinforcement learning and deep learning, has emerged as a powerful tool for training robots to perform manipulation tasks with high precision and adaptability [26]. Unlike traditional control methods, which rely on predefined rules and models, DRL allows robots to learn optimal behaviors through interaction with their environment. This capability is crucial for handling the uncertainties and dynamic nature of real-world scenarios.

Chapter 2 provided a comprehensive overview of the theoretical foundations of deep reinforcement learning, covering key concepts such as value-based methods, policy-based methods, and actor-critic approaches. Building on this theoretical background, Chapter 3 delves into the practical applications of DRL in robotic manipulation. This chapter presents a survey of recent advances in DRL algorithms specifically designed for robotic manipulation tasks, exploring how these algorithms address the challenges and complexities inherent in this field.

The focus of this chapter is to bridge the gap between theory and practice by examining how DRL algorithms are implemented in robotic manipulation. We begin with a discussion on the fundamentals of robotic manipulation, outlining the essential components and typical tasks. Following this, the chapter provides an overview of the key DRL algorithms used in robotic manipulation, highlighting their strengths and limitations.

To ensure a comprehensive and systematic review, we employed a rigorous search methodology to identify relevant publications. This methodology, detailed in Section 3.4, involved the use of specific keywords, search engines, and criteria for selecting and excluding studies. The subsequent sections then delve into the specifics of various DRL algorithms and their applications, discussing both the challenges encountered and the innovative solutions proposed in recent research.

Finally, the chapter concludes with an analysis of current trends and future directions in the application of DRL to robotic manipulation. This analysis aims to provide insights into the evolving landscape of this field and identify potential areas for further research and development. Through this exploration, we aim to highlight the transformative potential of DRL in enhancing the capabilities and versatility of robotic manipulation systems.

3.2 Fundamentals of Robotic Manipulation

Robotic manipulation is a subfield of robotics that focuses on the development of robots capable of interacting with objects in their environment. This capability is essential for performing a wide range of tasks across various industries, including manufacturing, healthcare, agriculture, logistics, and more. The primary goal of robotic manipulation is to create systems that can autonomously perform tasks that require dexterous and precise handling of objects, much like human hands. Figure 3.1 shows a classic robotic manipulation workflow. Moreover, Matt Mason provided a thorough and in-depth description of manipulation in the introduction of his 2018 review paper [27].

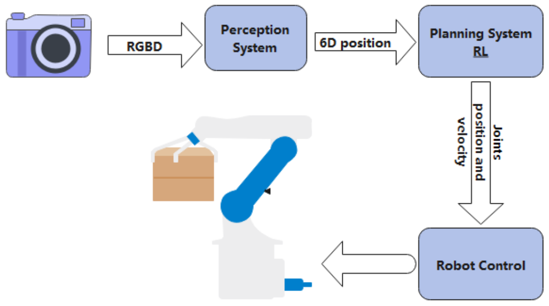


Figure 3.1: Classic robotic manipulation workflow

.

3.2.1 Definition and Scope of Robotic Manipulation

Robotic manipulation involves the control and coordination of robotic arms and end-effectors (such as grippers) to perform tasks such as picking, placing, assembling, and manipulating objects. These tasks can vary in complexity from simple pick-and-place operations to intricate assembly and surgical procedures. The scope of robotic manipulation extends to any application where robots are required to interact physically with their surroundings to achieve specific objectives.

3.2.2 Components of Robotic Manipulation Systems

A typical robotic manipulation system consists of three main components:

- Robot Arm: This is the primary manipulator that provides the necessary degrees of freedom (DoFs) to reach and interact with objects. Modern robot arms can range from simple two or three DoF systems to complex seven or more DoF systems, allowing for a wide range of movements and capabilities.

- Gripper or End-Effector: The end-effector is the component that directly interacts with objects. It can take various forms, including parallel grippers, suction cups, robotic hands with multiple fingers, or specialized tools for specific tasks (e.g., welding torches, surgical instruments). The design and functionality of the end-effector are critical for the successful execution of manipulation tasks.

- Control System: The control system coordinates the movements of the robot arm and end-effector. It translates high-level task commands into precise, executable actions. This system often includes sensors (e.g., cameras, force sensors) to provide feedback and ensure accurate and adaptive manipulation.

3.2.3 Challenges in Robotic Manipulation

Robotic manipulation poses several challenges that need to be addressed for successful implementation:

- High Dimensionality: Manipulation tasks often involve complex and high-dimensional state and action spaces. Efficiently exploring and learning in these spaces is crucial for developing effective manipulation policies.
- Real-World Variability: Real-world environments are dynamic and unpredictable, requiring robots to adapt to varying conditions and object properties (e.g., size, shape, texture).
- Perception and Sensing: Accurate perception and sensing are essential for reliable manipulation. Robots must be able to detect and recognize objects, estimate their properties, and adjust their actions accordingly.
- Precision and Dexterity: Many manipulation tasks require a high level of precision and dexterity, similar to human hand movements. Achieving this level of control is particularly challenging for robots.
- Safety and Reliability: Ensuring the safety and reliability of robotic manipulation systems is critical, especially when operating in close proximity to humans or handling fragile objects.

Robotic manipulation continues to evolve, driven by advancements in artificial intelligence, machine learning, and sensor technologies. The application of deep reinforcement learning (DRL) has shown significant promise in addressing many of the challenges in this field, enabling robots to learn and adapt to complex manipulation tasks through experience. The following sections will delve into the specific DRL algorithms and their applications in robotic manipulation, providing a comprehensive overview of the current state of the art in this exciting and rapidly advancing area of research.

3.3 Overview of Deep Reinforcement Learning in Robotic Manipulation

Deep reinforcement learning (DRL) has emerged as a powerful tool for enabling robots to learn and execute complex manipulation tasks. By combining the strengths of reinforcement learning (RL) and deep learning, DRL allows robots to autonomously learn optimal behaviors through interaction with their environment [28]. This capability is particularly valuable in robotic manipulation, where the diversity and complexity of tasks require flexible and adaptive solutions.

Robots are programmed with a set of rules and instructions that specify how they should interact with items in their surroundings in conventional methods of robotic manipulation. This approach is effective for simple activities, but it becomes more challenging as the difficulty of the tasks increases. Robots may manipulate objects in their surroundings using a process called deep reinforcement learning, which allows them to make mistakes and learn from them. Deep reinforcement learning provides a more flexible and adaptable method, allowing robots to learn from experience and change their behavior [29][30]. For instance, the robot receives positive reinforcement if it successfully picks up and moves an object to the desired location. If it drops the object or fails to transfer it to the desired location, a negative reward is provided. As it obtains the capacity to correlate some activities with good results and other actions with undesirable outcomes over time, the robot develops a strategy for accomplishing the task at hand [31]. Robotic manipulation using deep reinforcement learning has the potential to change a variety of industries, including healthcare and manufacturing. By allowing robots to learn from experience and adjust to changing situations, it enables them to perform tasks that are too difficult or dangerous for humans to complete. As research in this area advances, we could expect to see more capable and advanced robots that can manipulate objects more precisely and effectively.

3.4 Search Methodology

Since RL is more adaptable in highly dynamic and unstructured environments than more traditional or other AI control approaches, there has been a recent increase in interest in using RL to operate robotic manipulators[32]. These techniques have demonstrated impressive results in enabling robots to learn complex tasks and operate in dynamic environments. Moreover, the growing interest in robotic manipulation in reinforcement learning has been stimulated by the expanding availability of affordable and efficient robotic hardware as well as the rising demand for automation across a variety of industries. Applications of this technology include manufacturing, logistics, healthcare, and home automation, among others.

So, the purpose of this review is to present an overview of the major works using RL in robotic manipulation tasks and to analyze the future directions of this topic. In order to achieve this goal, a thorough review of the literature is conducted, and the content of more than 150 articles in relevant fields is searched and reviewed.

Given the enormous amount of literature on the subject, looking for papers in reinforcement learning for robotic manipulation can be a difficult task. Start by identifying the relevant keywords for the search, such as "reinforcement learning," "robotic manipulation," "manipulation tasks," "control policies," "deep learning," etc. These keywords will help narrow down the search to the most relevant papers. Use a specialized search engine such as Google Scholar, IEEE Xplore or ArXiv to search for papers related to reinforcement learning for robotic manipulation. These search engines allow filter the results from 2015 to 2022. This period's start was chosen because of the release of the RL and deep neural network-using AlphaGo program[33]. This innovation has made a significant contribution to the rapid growth of RL.

Meanwhile, studies that were not appropriate for the scope of this review had to be excluded even though they were relevant to the field of RL. Outdated and do not contribute to the current state of the field should be exclued. Additionally, the authors decided to exclude papers that are poorly written or have significant methodological flaws. An overview of the specified search criteria can be found in Table 3.1.

Table 3.1: An overview of the specified search criteria.

Criteria	**Description**
Keywords	”reinforcement learning” AND ”robotic manipulation” AND ”manipulation tasks” AND ”control policies”
Search engine	Google Scholar, IEEE Xplore or ArXiv
Time period	Between 2015 and the present.
Publication type	Academic conference papers and journal articles
Relevance	Exclude studies that are not appropriate for the scope of the review.
Outdated	Exclude old papers that are no longer relevant to the current state of the field.
Quality	Exclude poorly written or methodologically flawed papers.

3.5 Applications and Implementations of DRL in Robotic Manipulation

This section explores the practical applications and implementations of deep reinforcement learning (DRL) algorithms in robotic manipulation. While Chapter 2 provided a theoretical foundation, this section focuses on how these algorithms are applied in real-world scenarios, including reward engineering techniques such as imitation learning, behavior cloning, and hierarchical reinforcement learning.

3.5.1 Practical Implementations of DRL Algorithms

In robotic manipulation, value-based methods like Q-learning and Deep Q-Networks (DQN) have been applied to tasks such as grasping and object manipulation. These methods estimate the value of taking specific actions in given states, helping robots learn optimal policies through experience. Due to the quantity of samples necessary for precise estimations, learning continuous control in high-dimensional, sparse reward environments like robotic manipulation is a difficult challenge. For these tasks, Rammohan et al. [34] recommend value-based reinforcement learning, more especially RBF-DQN. For robotic manipulation tasks, they show that RBF-DQN converges more quickly than existing state-of-the-art algorithms like TD3, SAC, and PPO, and that RBF-DQN is also amenable to improvement methods like HER and PER. The authors contend that compared to policy-gradient approaches, value-based systems may be more susceptible to data augmentation and replay buffer sampling strategies. The use of equivariant neural networks in Q-learning and actor-critic reinforcement learning is investigated by Wang et al. [35]. Equivariant models are suited to robotic manipulation issues because they enforce symmetry in their convolutional layers and can considerably increase sampling efficiency when learning an equivariant or invariant function. The authors suggest equivariant variations of the DQN and SAC algorithms that take advantage of this structure and show through tests that they can be more sample-efficient than rival algorithms on certain robotic manipulation challenges.

Methods such as Proximal Policy Optimization (PPO) and Trust Region

Policy Optimization (TRPO) are particularly suited for tasks requiring continuous control. These algorithms directly learn a policy that maps states to actions, enabling smooth and precise movements. Chen et al. [36] aimed to address multiple robotic manipulation tasks, such as grasping, button-pushing, and door opening, using reinforcement learning (RL), state representation learning (SRL), and imitation learning. To do so, the authors self-built simulated environments in PyBullet and explored three different learning-style methods using wrapped PPO and DQN algorithms from the OpenAI baseline. These methods were successfully applied to solve diverse missions in the self-constructed environments. Clegg et al. [37] investigate the application of reinforcement learning to create robotic dressing aides that can predict human movements. The robot offers a client with an open sleeve of a medical gown, and the individual puts their arm into the sleeve, according to the researchers' models of human behavior during dressing assistance. The system develops a model of human behavior using the TRPO algorithm that can effectively train for three distinct robot-assisted dressing procedures and can insert the arm into the sleeve. The purpose of the project is to simulate how individuals may aid with dressing.

Combining the strengths of value-based and policy-based methods, actor-critic approaches like Soft Actor-Critic (SAC) and Deep Deterministic Policy Gradient (DDPG) are used for complex manipulation tasks. The actor proposes actions while the critic evaluates them, balancing exploration and exploitation. Vecerik et al. [38] developed a model-free reinforcement learning approach for real robotics with sparse rewards by using demonstrations and extending the Deep Deterministic

Policy Gradient (DDPG) algorithm. Both demonstrations and actual interactions were used to populate a replay buffer, and a prioritized replay mechanism [39] automatically adjusted the sampling ratio between demonstrations and transitions. Kilinc et al. [40] proposed a novel framework for robotic manipulation that does not rely on human demonstrations. They framed every robotic manipulation task as a locomotion task from the perspective of the manipulated object, and used a physics simulator to obtain an object locomotion policy. This policy was then used to generate simulated locomotion demonstration rewards (SLDRs), which enable the learning of the robot manipulation policy. Yang et al. [41] developed a Pybullet engine re-implementation of the OpenAI multi-goal robotic manipulation environment with additional APIs for joint control, customizable camera goals and image observations, and on-hand camera access. The authors also created a series of challenging robotic manipulation tasks with sparse rewards, long horizons, and multi-step and multi-goal objectives to inspire new goal-conditioned reinforcement learning algorithms. Vulin et al. [42] tackled the challenge of exploration in deep reinforcement learning for robotic manipulation by introducing an intrinsic reward based on the sum of forces between the robot's force sensors and manipulation objects, encouraging physical interaction. They also proposed a contact-prioritized experience replay sampling scheme that prioritizes contact-rich episodes and transitions.

There are lots of RL techniques for robot manipulator to apply. Figure 3.2 categorize the research papers by using different RL algorithms in robotic manipulation from 2015 to 2022.

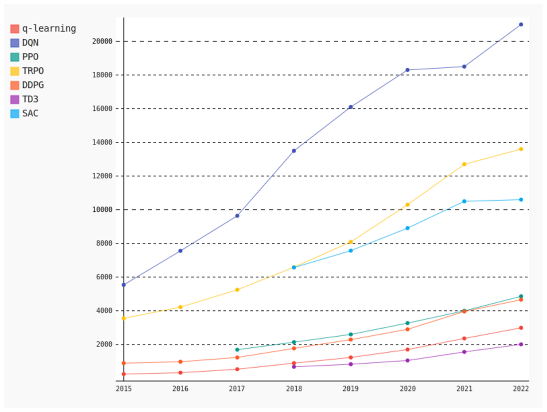


Figure 3.2: The trend of published papers using different RL algorithms in robotic manipulation

.

3.5.2 Reward Engineering

Reward engineering is crucial for designing effective DRL systems for robotic manipulation. It involves shaping the reward function and learning techniques to ensure that the learning agent can efficiently achieve the desired behavior.

In imitation learning, the goal is to learn a policy that can mimic the behavior of an expert. The expert's behavior is represented as a set of demonstrations, which can be used to train the policy. The policy is typically learned by minimizing the distance between the expert's behavior and the policy's behavior, using a distance measure such as the KL divergence or the maximum mean discrepancy. The advantage of imitation learning is that it does not require a reward function to be specified, which can be difficult or infeasible in some cases. However, it may not generalize well to situations that are different from those encountered by the expert. The classification of imitation learning can be seen in Fig 3.3.

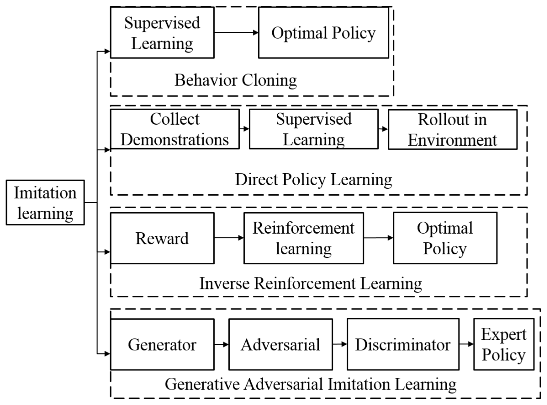


Figure 3.3: Classification of imitation learning [1]

.

One popular algorithm for imitation learning is behavior cloning, where the goal is to learn a policy that can mimic the expert's behavior. This can be done by collecting a dataset of expert demonstrations and using it to train a supervised learning algorithm to predict the expert's actions given the current state. The learned policy can then be used to execute the desired behavior in the environment. Another approach is inverse reinforcement learning, where the goal is to learn the reward function that the expert is optimizing, and then use this reward function to train a policy using RL techniques. This allows the agent to not only mimic the expert's behavior, but also adapt and improve upon it. An important example of behaviour cloning is ALVINN [43], a vehicle equipped with sensors that has learnt to map the sensor inputs into steering angles and drive autonomously. Dean Pomerleau initiated this research in 1989, and it was also the first implementation of imitation learning in general.

In Direct policy learning (DPL), the goal is to learn a policy that can replicate the expert's behavior as closely as possible. This is done through an iterative process where the policy is trained using supervised learning on a dataset of demonstrations provided by the expert. The trained policy is then implemented in the environment and evaluated through interactions with the expert. Any additional data collected during these interactions is then added to the training dataset and the process is repeated until the policy converges to a satisfactory level of performance. RL-teacher [44] is a specific implementation of DPL that allows for the learning of novel behaviors without the need for a pre-defined reward function or the ability for the expert to demonstrate the desired behavior directly.

Inverse reinforcement learning (IRL) [45] is a kind of imitation learning in which we are given a policy or a history of behavior from an agent and use reinforcement learning to try to discover a reward function that explains the behavior. IRL, like RL, is seen as an issue as well as a category of techniques. However, there are two issues with discovering a reward function that is best for observed behavior. First, for most observations of behavior there are many fitting reward functions. Many degenerate solutions exist in the set of solutions, such as providing 0 reward to all states. Second, the IRL algorithms are based on the assumption that the observed behavior is ideal. This is similar to over-fitting in supervised learning.

Generative adversarial imitation learning (GAIL) [46] combines IRL with generative adversarial networks (GAN). GAIL's purpose is to train generators that behave similarly to given experts. Meanwhile, the discriminators may be used as reward functions for RL, which determines if the actions match those of experts.

GAIL can learn from a small number of expert trajectories. GAIL is capable of handling difficult issues and generalizes effectively to unknown scenarios. GAIL is not exactly IRL, because GAIL learns the policy, rather than the reward function, directly from the data.

Goal-conditioned imitation learning (GCIL) [47] is a combination of the GAIL and hindsight experience replay (HER) [48] algorithms. GCIL utilizes the benefits of both algorithms, such as the ability of GAIL to quickly learn from a few demonstrations at the start of a task and the ability of HER to generalize and learn new tasks through hindsight relabeling. In a goal-reaching task, the data distribution includes both states, actions, and attempted goals. HER improves the data distribution by replacing the initially desired goals with the actually achieved goals, since a robot's failure to achieve a desired goal is still a success in terms of achieving the goal it actually reached. By optimizing this non-parametric data distribution, GCIL can improve the efficiency and effectiveness of imitation learning.

Curriculum learning [49] is a training method in which the complexity of the data samples used increases gradually over time. The original formulation of curriculum learning was based on the idea that it mirrors the natural way in which people learn. While it may seem that curriculum learning is solely about increasing the complexity of the training experience, it is also about leveraging knowledge gained from simpler tasks to reduce the exploration needed in more complex tasks through generalization.

Teacher-student curriculum learning (TSCL) [50] is a method for automating the process of curriculum learning in reinforcement learning. It involves using two RL agents: a student and a teacher. The student is responsible for learning and completing tasks, while the teacher is responsible for selecting appropriate sub-tasks to help the student learn more effectively. By gradually increasing the complexity of the tasks presented to the student, the teacher helps the student learn more efficiently and effectively. TSCL is a useful approach for tackling difficult tasks that may not be able to be learned directly, by breaking them down into smaller sub-tasks that can be learned more easily.

Different from the teacher-student framework, two agents are doing very different things. In asymmetric self-play [51], the two agents Alice and Bob both train on the main task directly. Alice's role is to propose challenging goals for Bob to achieve, while Bob's role is to try to reach those goals. This interaction between Alice and Bob creates an automated curriculum of progressively challenging tasks, and because Alice has already completed the task before presenting it to Bob, it is guaranteed to be achievable. Asymmetric self-play is a method for goal discovery, in which the two agents work together to find and achieve new goals within the main task.

Hierarchical reinforcement learning (HRL) [52] is a computational approach that allows an agent to learn how to perform tasks at different levels of abstraction. It involves multiple sub-policies working together in a hierarchical framework, rather than just one policy trying to accomplish the overall goal. This method has several benefits, such as improved exploration and sampling efficiency, and can also be used for transfer learning, where low-level policies or sub-policies can be reused for multiple tasks. For example, if an agent has learned how to make coffee, it can reuse that knowledge when learning how to make a cappuccino by separating the process into making coffee and then warming and frothing the milk. HRL can therefore speed up the learning process for new tasks.

For robot manipulators, there are multiple reward engineering strategies available. Figure 3.4 shows the trend of reward engineering used in robotic manipulation from 2015 to 2022.

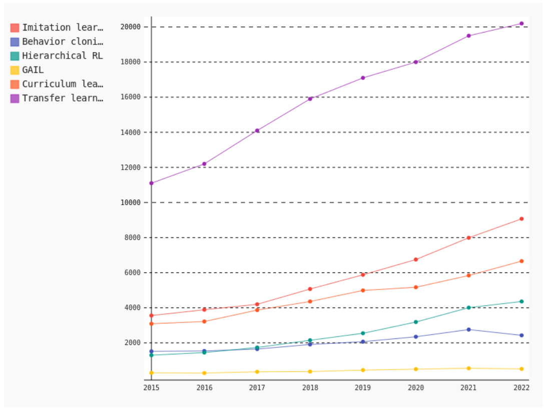


Figure 3.4: The trend of published papers using different reward engineering in robotic manipulation

.

3.5.3 Graph Neural Network Architectures in DRL for Robotic Manipulation

This section explores the Graph neural network architectures employed in deep reinforcement learning (DRL) for robotic manipulation. Graph neural networks (GNNs) can be used as the network architecture for reinforcement learning problems in relational environments.

Table 3.2 demonstrate a list of papers about GNN implementation relative to RL Algorithms and learning techniques. Janisch et al. [53] proposed a deep RL framework based on GNNs and auto-regressive policy decomposition that is well-suited to these types of problems. They demonstrate that their approach, which uses GNNs, can solve problems and generalize to different sizes without any prior knowledge.

Table 3.2: A list of papers about GNN implementation relative to RL Algorithms and learning techniques.

Reference papers	**RL algorithms**	**Learning techniques**
[54]	DQN and GNN	Evaluate on OTN routing map
[55]	Blueprint policy and PPG	Curriculum learning
[53]	GNN and A2C	Behavior cloning
[56]	PPO	Imitation learning
[57]	GNN and SAC	Sequential curriculum learning
[58]	Model-based method	Imitation learning
[59]	MDP and GAT	Hierarchical imitation learning
[60]	PG and GCN	GAIL
[61]	PPO and GCN	Relational inductive bias

To address the issue of limited generalisation in current DRL-based networking solutions, Almasan et al. [54] present a Deep Reinforcement Learning (DRL) agent including Graph Neural Networks (GNN). The proposed GNN-based DRL agent can learn and generalize over diverse network topology since GNNs are built to generalize over networks of various sizes and architectures. The agent outperforms cutting-edge methods in topology not seen during training in a routing optimization use case in optical networks.

Richard Li et al. [57] present a reinforcement learning system that can stack 6 blocks without any demonstrations or task-specific assumptions. They use the Soft Actor-critic (SAC) algorithm as their base learning algorithm because it is more robust to hyperparameter choices and random seeds than other off-policy learners such as Deep Deterministic Policy Gradient (DDPG). They find that training a policy represented by an attention-based graph neural network (GNN) enables successful curriculum learning in multi-object manipulation tasks.

Yunfei Li et al. [55] study a challenging assembly task in which a robot arm must build a feasible bridge based on a given design. The authors divide the problem into two steps. First, they use an attention-based neural network as a "blueprint policy" to assemble the bridge in a simulation environment. Then, they implement a motion-planning-based policy for real-robot motion control.

Lin et al. [56] presents a method for robot manipulation using graph neural networks (GNNs). The authors propose a GNN-based approach for solving the object grasping problem, in which the robot must select a suitable grasp pose for a given object. The approach combines a graph representation of the object with a GNN to learn a grasp quality function, which predicts the success of a grasp based on the object's geometry and the robot's kinematics. The authors evaluate their method on a dataset of synthetic objects and a real-world grasping task, and demonstrate that it outperforms previous approaches in terms of both efficiency and interpretability.

3.5.4 Current Trends and Future Directions

The field of deep reinforcement learning (DRL) for robotic manipulation is rapidly evolving, with several key trends shaping its development. These trends highlight the ongoing innovations and the directions in which current research is focused.

Sim-to-Real Transfer: One of the significant trends is improving the transfer of learned policies from simulation to real-world environments. Techniques such as domain randomization, which involves varying the simulation parameters to create a more robust policy, and progressive networks, which help in adapting learned behaviors to new domains, are gaining traction. These methods aim to bridge the gap between simulated training environments and the variability of real-world scenarios, making DRL-trained robots more adaptable and reliable.

Hierarchical Reinforcement Learning (HRL): HRL continues to be a prominent approach for managing complex tasks by breaking them down into simpler subtasks. This hierarchical structure not only simplifies the learning process but also enhances the robot's ability to handle long-horizon tasks. HRL is particularly useful in tasks such as multi-step assembly processes and autonomous exploration, where the robot needs to plan and execute a series of actions to achieve a high-level goal.

Multi-Task and Meta-Learning: There is a growing interest in developing DRL algorithms that can generalize across multiple tasks and learn new tasks quickly. Meta-learning, which focuses on learning how to learn, enables robots to adapt to new tasks with minimal training data by leveraging prior knowledge. Multi-task learning aims to train robots on several tasks simultaneously, improving their ability to generalize and transfer skills across different applications.

Incorporation of Advanced Sensing and Perception: The integration of advanced sensing technologies, such as tactile sensors, depth cameras, and LIDAR, is enhancing the perception capabilities of robots. These sensors provide rich, multimodal data that can be leveraged by DRL algorithms to make more informed decisions. Improved perception allows robots to interact more effectively with their environment, handle objects with greater dexterity, and perform tasks with higher precision.

Human-Robot Interaction (HRI): Enhancing the interaction between humans and robots is a critical trend. Research is focusing on making robots more intuitive and easier to control through natural language commands, gesture recognition, and shared autonomy. By improving HRI, robots can better assist humans in collaborative tasks, leading to safer and more efficient operations in environments such as manufacturing, healthcare, and domestic settings.

Future research in deep reinforcement learning (DRL) for robotic manipulation will focus on improving sample efficiency through techniques like model-based reinforcement learning, ensuring robustness and safety with fail-safe mechanisms and robust policy learning, and integrating DRL with other AI fields such as computer vision and natural language processing. There will be a push towards scalable and distributed learning frameworks leveraging cloud and edge computing, and developing personalized learning approaches that tailor robot behaviors to individual users. Addressing the ethical and societal implications of deploying robots, such as ensuring transparency, fairness, and accountability in DRL algorithms, will also be crucial. These advancements aim to create intelligent, adaptable, and reliable robotic systems capable of performing a wide range of tasks across various industries.

3.6 Summary

In this chapter, we have explored the application of deep reinforcement learning (DRL) algorithms in robotic manipulation, a field that has seen significant advancements in recent years. Starting with an overview of robotic manipulation and its components, we delved into the practical implementations of DRL algorithms, highlighting value-based, policy-based, and actor-critic methods. These methods have demonstrated their efficacy in handling complex tasks that require precision, adaptability, and continuous control.

We also examined the critical role of reward engineering in shaping effective DRL systems. Techniques such as imitation learning, behavior cloning, inverse reinforcement learning, and hierarchical reinforcement learning were discussed in detail. These methods help in designing reward functions and learning policies that can efficiently solve manipulation tasks by leveraging demonstrations, decomposing tasks, and learning from structured experiences.

Current trends in DRL for robotic manipulation, such as sim-to-real transfer, hierarchical reinforcement learning, multi-task and meta-learning, advanced sensing and perception, and human-robot interaction, were identified. These trends underscore the ongoing innovations aimed at enhancing the capabilities and robustness of robotic systems.

In summary, the application of DRL in robotic manipulation has the potential to revolutionize numerous industries by enabling robots to perform complex and precise tasks autonomously. Continued research and innovation in this field will lead to more intelligent, versatile, and reliable robotic systems, capable of transforming both industrial processes and everyday life. This chapter has provided a comprehensive overview of the state of the art in DRL for robotic manipulation, laying the groundwork for future advancements and applications in this exciting and rapidly evolving field.

Chapter 4

Enhancing Semantic Segmentation with Reinforced Active Learning

4.1 Introduction

Semantic segmentation is a critical step in computer vision whereby a class label is assigned to each pixel in an image to partition effectively an image into meaningful segments. Unlike image classification, where the whole image is labeled with only one class, semantic segmentation can provide much finer information that would enable us to interpret and understand visual information at the pixel level.

On the other hand, training a large semantic segmentation model brings two major issues: dataset imbalance and efficiency of the annotation process. Moreover, dataset imbalances caused by biases related to age and gender in clinical contexts or skewed representation in natural images may exert significant impacts on the performance of the model. Furthermore, tasks related to segmentation in general and those pertaining to semantic segmentation, in particular are expensive in terms of time and resources during annotation. These challenges form the basis for this research study, which investigates reinforced active learning methodologies for developing semantic segmentation. Advanced methods to be integrated within the proposed framework include Dueling Deep Q-Networks, Prioritized Experience Replay, Noisy Networks, and Emphasizing Recent Experience. These are intended to optimize the efficiency and precision of the segmentation process. Therefore, while the former accelerates the annotation process by selectively picking the most informative and representative images, the latter alleviates dataset imbalances by using strategies of mitigating biases.

The key aims of this research are to develop a high-performance model for achieving accurate and efficient semantic segmentation in various domains and robust frameworks applicable in cases with imbalanced datasets. Extensive experiments and evaluations show the improvement but also the limitation of diverse approaches, which will give insight toward developing more sophisticated and precise segmentation algorithms.

In short, this chapter presents a comprehensive study on enhancing semantic segmentation through reinforced active learning. This will make it evident how the balancing of exploration and exploitation strategies in reinforcement learning can firmly push forward the field of semantic segmentation toward more effective and efficient image analysis under different application scenarios.

4.2 Background

4.2.1 Semantic Segmentation

Semantic segmentation is an essential task in computer vision in which one attempts to classify every pixel in an image into a predefined class. Such dense classification allows the understanding of visual content and, enabled by this ability, has furthered applications in various fields, from autonomous driving to medical imaging and robotics [62]. Unlike in image classification, where only one label for the entire image is given, semantic segmentation provides labeling at the pixel level, which enriches spatial information concerning object boundaries and relations within the image. Semantic segmentation could suffer from data imbalance.

In many real-world cases, however, this is not the situation: there are underrepresented classes and biased model predictions that reduce performance for these classes [63]. For instance, in clinical imaging, there could be an overabundance of images from a specific demographic, while other parts of a population might be underrepresented. In natural images, some object categories may overshadow the dataset, compared to less frequent ones [64]. Such imbalances can lead the learning process of the model to be biased toward the dominant classes while performing poorly for the underrepresented classes [65]. The need for pixel-wise annotations in semantic segmentation is highly time-consuming and labor-intensive. Human annotation for each image of the dataset with pixel-level detail is highly costly and also error-prone [66]. The problem of annotations rises manifold if the dataset is of a very high magnitude, such as an effective way of minimizing human effort towards quality-assured annotations.

4.2.2 Active learning

Active learning is an ML paradigm that aims to minimize the amount of labeled data by selectively annotating the most informative samples. Traditional active learning protocols prescribe how a model at each iteration chooses a subset of samples from an unlabeled dataset; then if annotated, it would yield a significant improvement in performance over models trained on an equivalently sized, randomly selected sample. This approach reduces annotation effort while focusing on the most valuable data points [67].

Active learning strategies can be broadly classified into three categories:

- Membership Query Synthesis: The model generates new samples that are most informative for learning.
- Stream-Based Selective Sampling: The model evaluates each incoming sample and decides whether to label it based on its informativeness.
- Pool-Based Sampling: The model selects the most informative samples from a large pool of unlabeled data.

One of the reasons active learning for semantic segmentation has gotten little attention compared to image classification is that this task is more costly and complex, as images need to be annotated per pixel.

4.2.3 Reinforced Active Learning

Reinforced active learning combines active learning with reinforcement learning principles into a strategy that optimizes how samples are selected. Here, an RL agent selects samples following a developed policy based on past experiences of this specific task. The agent gets feedback as rewards, which are derived from improvements in the model's performance after selecting and incorporating labeled samples [68]. Reinforced active learning has been shown to work well in many contexts because the task is one of trading off between exploration and exploitation: one needs to select diverse samples but focus on maximal performance gain samples. It works particularly well in low-data contexts and tasks with a high annotation cost [69].

4.2.4 Significance of Reinforced Active Learning in Semantic Segmentation

This work combines the benefits of reinforced active learning with semantic segmentation to handle dataset imbalances and annotation efficiency. The presented framework here capitalizes on state-of-the-art RL techniques in prioritizing the selection of informative and representative samples to alleviate the annotation burden and enhance the model over all classes, whether balanced or imbalanced.

In summary, the chapter has given a broad view of semantic segmentation, the challenges around it, and how active and reinforced learning methodologies could offer the potential to mitigate those challenges. The following sections will discuss the related work, proposed methodology, experimental setup, and results showing how effective the given framework of reinforced active learning is in improving semantic segmentation.

4.3 related work

Active learning is a dedicated methodology that focuses on maximal performance gain with a minimal number of labeled samples. The primary goal is to search within the unlabeled dataset for samples that contain the maximum amount of information and present them consecutively to an oracle, like human annotators, for labeling. This effectively minimizes the labeling cost while ensuring sustained performance. Active learning strategies can be broadly categorized into membership query synthesis [70], [71], stream-based selective sampling [72], [73], and pool-based [74] strategies based on the applicability scenario [75]. Many methods combine different approaches to improve overall active learning performance. For example, Shui et al. [76] combine the diversity and uncertainty of the query samples and try to find a balance between these two criteria. Further research on classical query strategies is also carried out in [77]. However, despite the vast research on active learning, it still faces significant difficulty in making it applicable to high-dimensional data such as images, text, and video [78]. This has meant that most of the problems considered by active learning research are low-dimensional [79].

However, in the recent past, there has been increasing interest in reinforcement learning as an approach to acquiring a labeling policy that directly optimizes the performance of the active learning algorithm. For instance, Dhiman et al. [80] proposed an automated annotation model for Multimedia Streaming Applications (MAS) that addresses current challenges, which include slow speeds and inefficiencies in access to multimedia content. Apply Multi-modal Active Learning (MAL) and Convolutional Recurrent Neural Network (CRNN) with the help of Deep Reinforcement Learning (DRL), and therefore, the retrieval accuracy and performance metrics surpass those of this model. Gong et al. [81] proposed a Meta Agent Teaming Active Learning (MATAL) framework to effectively reduce the laborious efforts involved in annotations of poses. Sadigh et al. [82] designed an active learning algorithm for inverse reinforcement learning using human-provided preferences between two sample trajectories. Kunapuli et al. [83] maintain a level of information from a human expert through preference elicitation to take actions within a designated state. Ezzeddine et al. [84] pooled the feedback from a human trainer, especially when demonstrations provided are suboptimal.

Recent work in active learning also considered semantic segmentation [85]. Uncertainty-driven active learning flags data samples with high aleatoric uncertainty. Kampffmeyer et al. [86] attempt to maximize the average standard deviation of the predicted probabilities. Jain et al. [87] incorporate measures defined over hand-crafted heuristics, encouraging diversity and representativeness of labeled samples. Some methods exploit unsupervised super pixel-based over-segmentation [88], [89] that relies heavily on the precision of superpixel segmentation. Other works focus on foreground-background segmentation of biomedical images [90], [91] using likewise designed heuristics. The importance of self-consistency that should employ simple transformations, not altering the observation, is focused by Golestaneh et al. [92].

The arrival of DQN was a significant breakthrough, but since then, many limitations of the algorithm have appeared, and different extensions have been proposed: Double DQN [10] reduces the overestimation bias of Q-learning [93] by decoupling the selection from the evaluation of the bootstrap action. Prioritized experience replay improves data efficiency by an increased replay frequency of more informative transitions [39]. The dueling network architecture helps to generalize over the action by independently representing state values and action advantages [11]. Learning from multi-step bootstrapping targets, as in A3C [94], adjusts the bias-variance trade-off while speeding the propagation of the newly observed rewards to the states visited earlier. Noisy DQN introduces stochastic network layers in the network to facilitate exploration [95]. To the best of our knowledge, our work is the first to examine an agent that integrates all these components addressed to date in solving the problem of active learning for semantic segmentation.

4.4 Methodology

4.4.1 Active learning with reinforcement learning for semantic segmentation

Following Casanova et al. [96], we follow their architecture to train the segmentation network. We formulate this as a Markov decision process. In what follows, an active learning process is developed to improve the performance of a segmentation network, f, with parameters θ under a limited number of labeled samples budget B. At each iteration, a query network, π, parametrized by ϕ, is used to select K regions from the sizeable unlabeled set, U_t. These regions are then sent to an oracle to be labeled, thus augmenting the labeled set, L_t. The segmentation network f is trained on the augmented L_t, and its performance is measured based on the IoU metric. This process iterates further until the target budget of B is reached. In this way, it optimizes segmentation network performance by judiciously choosing representative regions for labeling, and consequently, high-quality segmentation can be achieved with a small amount of labeled data effectively. The approach is data-centric, and the model learns the selection strategy from the previous AL instances.

We use four unique data splits in this setup. We train the query network π by designating a subset of the labeled data D_T and using it for multiple iterations of the active learning process, yielding an effective acquisition function that optimizes performance within a B region budget. We evaluate the query network on the data that has been split away, D_V. Further, we use a different subset D_R to construct the reward signal, for which we evaluate the performance of the segmentation network. We also employ the set D_S, where D_S is not more significant than D_T, in constructing the representation of the current state.

The Markov decision process is characterized as the tuple (S, A, r, T, γ), where S indicates a set of states and A is the actions composed of K sub-actions, or the same, relies on segmentation network, labeled, and unlabeled sets. Each sub-action consists of soliciting the annotation of a specific region; $r : S \times A \to \mathbb{R}$ for the function based on improvement in mean IoU per class of taking an action in a state; $T : S \times A \times S \to \mathbb{R}$ for the state-transition function; and γ for the discount factor, implying that a reward obtained in the future is worth a smaller amount than an immediate reward. Figure 4.1 describes this training workflow. In our framework, an episode stops as soon as the labeling budget B of regions is exhausted, after which the segmentation network f is reinitialized with weights θ_0, and a new episode starts. Query policy π training involves the process of simulating multiple episodes with weight updates after each time step by sampling transitions $(s_t, a_t, r_{t+1}, s_{t+1})$ from an experience replay buffer, E.

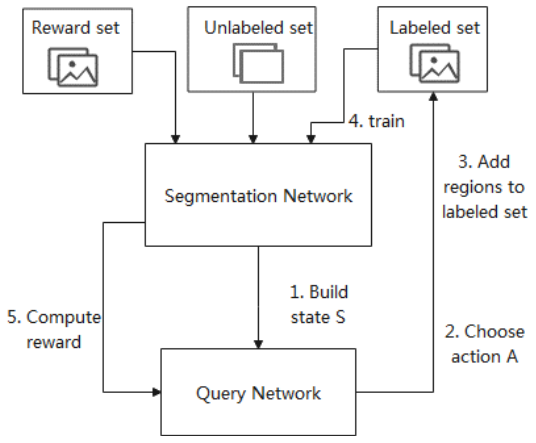


Figure 4.1: The overall workflow of the active learning with Reinforcement learning in semantic segmentation.

4.4.2 Extensions to DQN

The development path of Deep Q-Networks (DQN) naturally extended into several significant modifications that tried to solve their shortcomings while improving the overall performance. These modifications significantly expanded the functionality of the original DQN framework, allowing more effective and robust learning in complex environments. As we have already talked about Double DQN and Dueling DQN in the previous chapter, we will now talk in detail about the rest of the four extensions used.

Prioritized experience replay: The implementation presented by Schaul et al. [39] gives rise to a scheme called Prioritized Experience Replay. This is achieved by adding data structure that holds the priority of each transition.

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha} \tag{4.1}$$

Now, experiences are sampled concerning the priorities associated with them. Where α is the hyper-parameter determining how much sampling bias is applied. These priorities are based on the temporal difference error of the agent in its most recent training on that experience. Thus, the agent utilizes more learning resources on less accurate predictions, yielding improvements in weak areas and significant sample efficiency improvement. The new transitions will have a very high priority, so sampling from the replay buffer gives a bias to the recent ones. Moreover, it is interesting to note that stochastic transitions might still be preferred even when there is little remaining knowledge to learn from.

Emphasizing Recent Experience: Although mainly proposed to speed up the convergence rate of Soft Actor-Critic (SAC) [97], the method may theoretically be extended to a wide variety of algorithms and tasks that intrinsically benefit from the faster learning of more recent experiences, mainly when they consist of several elements. The basic idea is to sample the original mini-batch from the data set stored in the replay buffer. Updating parameters are used, and then for every new mini-batch after that, the range is closer, meaning one focuses more on recent data points. The scheme works with two principles: (i) A datum close to the present point is sampled with higher frequency, and (ii) There is a systematic arrangement of upgradation such that an older datum does not write over the more recent one. The Experience Replay Emphasis strategy directly addresses the challenge by providing a way for the agent to potentially prioritize recent transitions while taking into account previously learned policies using a simple but effective sampling method.

Adaptive epsilon greedy: Epsilon-greedy balances exploitation and exploration. For instance, when epsilon is set at 0.3, an action among the possible actions is picked at random with a probability of 0.3, while another is chosen greedily as the output action based on $\arg\max(Q)$ with a probability of 0.7. A more elaborate variant of the epsilon-greedy approach is the Adaptive-epsilon-greedy [98]. In this method, for instance, a policy is trained for N epochs/episodes, which could differ depending on the problem. The method initializes the epsilon with p_{init} (for instance, $p_{\text{init}} = 0.6$) and decays slowly down to $\epsilon = p_{\text{end}}$ ($p_{\text{end}} = 0.1$) over a designated number of training epochs/episodes (n_{step}). Especially in the first training phase, the model is allowed to have more freedom to explore (e.g., $p_{\text{init}} = 0.6$), after which epsilon slowly decreases according to the training epochs/episodes according to the formula below:

$$r = \max\left(\frac{N - n_{\text{step}}}{N_0}\right) \tag{4.2}$$

$$\epsilon \leftarrow (p_{\text{init}} - p_{\text{end}}) \cdot r + p_{\text{end}} \tag{4.3}$$

This is flexible; it allows the conclusion of training with a very low probability of exploration, p_{end} after n_{step}, thus quickly shifting towards a larger focus on exploitation or a greedier approach towards the very end of training. The probability of exploration remains small to explore as the policy approaches convergence.

Noisy Network: Noisy networks are employed in many places instead of the epsilon-greedy method because they encourage more efficient and dynamic exploration during training. A noisy network, as opposed to the epsilon-greedy technique, influences the parameters in a network by leading to the injection of stochasticity directly, which is how a more delicate, continuous process of exploration is achieved. The Noisy Network [95] proposes a new idea of a noisy linear layer, combining deterministic and noisy parts:

$$y = (b + Wx) + (b_{\text{noisy}} \odot \epsilon_b + (W_{\text{noisy}} \odot \epsilon_w)x) \tag{4.4}$$

Where ϵ_w and ϵ_b are random variables and $\odot$ represents the Hadamard product. Over the long term, the network learns to ignore the noisy stream; however, it does so at different rates for different areas of the state space, allowing state-dependent exploration with a type of intrinsic self-annealing. More precisely, such a dynamic strategy of exploration can fine-tune the relationship between exploitation and exploration in ways that favor adaptability and learning efficiency in complex settings.

Soft Update for Target Network: The soft update target network is a key concept in the field of deep reinforcement learning [17]. It refers to a technique used to stabilize and improve the training of deep neural networks in reinforcement learning tasks. Unlike hard updates, which involve periodically copying the parameters of the main network to the target network, soft updates gradually blend the parameters of the target network towards those of the main network. This process helps to mitigate the issue of drastic changes in the target network, which can lead to instability during the learning process. The value of τ is used. In the paper which proposed an algorithm called DPG, they used $\tau = 0.001$. The target network is updated as follows:

$$\theta_{\text{target}} = \theta \cdot \tau + \theta_{\text{target}} \cdot (1 - \tau) \tag{4.5}$$

Due to the small value of the parameter τ, allowing the target network to smoothly adjust towards the Q-network's value. Frequent updates are needed for this adjustment to be effective. This strategy enables the target network to more smoothly track changes in the main network, providing a more stable and effective learning process. It is particularly useful in complex reinforcement learning tasks where maintaining stability during training is crucial for optimal performance.

4.5 Experimental Setup

This section details the experimental setup for assessing the proposed reinforced active learning framework for semantic segmentation. It covers the datasets, data collection and preprocessing techniques, evaluation metrics, comparative baselines, and the hardware and software configurations employed.

4.5.1 Dataset Description

The primary dataset used in our experiments is CamVid [99], which consists of 360 x 480 street scene images categorized into 32 classes, with sample images shown in Figure 4.2. These images were captured from a moving vehicle using high-resolution cameras mounted on the streets, enabling the observation of various objects. Our study focuses on the segmentation of 11 key classes. In the urban context, the Road, Building, and Sky classes dominate the frames, comprising about 15.81%, 27.35%, and appearing in nearly all frames. Other significant classes, such as Car and Pedestrian, consistently appear throughout the frame sequence but occupy smaller areas, approximately 3.93% and 0.64%, respectively. Additionally, our analysis includes other classes listed in Table 4.1, which highlights a notable imbalance among the different classes, a challenge addressed in our study. The video sequences were recorded during the day and at dusk, where objects in the scene remain recognizable but appear darker. Daytime sequences were captured in sunny weather, featuring mixed urban and residential settings.

The training, validation, and test sets comprise 370, 104, and 234 images, respectively. Within the training set, we utilized 100 labeled images to construct D_T and 10 images for the state set D_S. For the baseline evaluation set D_V, 260 images were used. D_S has a similar class distribution to D_T to accurately represent it. The original validation set was used to build D_R. The test set was employed to train the model for final segmentation results. Each image was divided into K regions (where $K = 24$) with a resolution of 80x90. For implementation, we conducted 5 different runs with random seeds to calculate the mean and standard deviation. Data augmentation techniques, such as horizontal flips and random crops of 224x224, were applied.

In addition to CamVid, we utilized the Grand Theft Auto V (GTA) dataset [100] for pre-training. The GTA dataset includes synthetic images with pixel-level semantic annotations, generated from the open-world video game Grand Theft

Table 4.1: Statistics for each class used in this study.

Class name	Percentage (%)	Occurrence
Road	27.3	701
Building	22.7	687
Sky	15.8	699
Tree	10.4	636
Sidewalk	6.33	672
Car	3.4	643
Column_Pole	0.98	698
Fence	1.43	363
Pedestrian	0.64	640
Bicyclist	0.53	365
Sign_Symbol	0.12	416

% shows the ratio of pixels.

Occurrence shows the number of occurrences over all images.

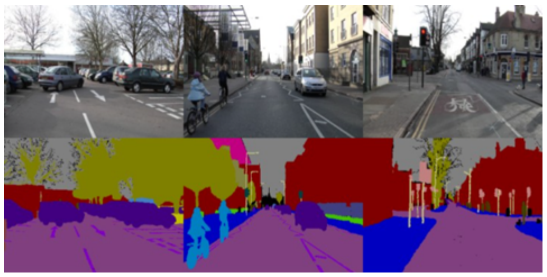


Figure 4.2: Labeled frames from the video at 1Hz.

Auto V. This dataset offers a variety of urban scenes from a car's perspective, providing diverse and richly annotated data.

4.5.2 Data Collection and Preprocessing

The CamVid dataset was captured using a digital film camera under fixed conditions without auto-zoom, focus, or adjustments during the collection process. The cameras were set to a fixed gain and shutter speed, and the aperture was maximized to prevent overexposure of white objects.

Preprocessing steps included:

- Annotation: Each image in the dataset was manually annotated to assign pixel-level semantic labels.

- Data Augmentation: To enhance the robustness of the model, data augmentation techniques such as horizontal flips and random crops (224×224 pixels) were applied to the images.

- Region Division: Each image was divided into K regions (K = 24) with a resolution of 80×90 pixels for the active learning process.

4.5.3 Evaluation Metrics

We trained the active learning agent on DT with approximately 0.5k regions to optimize the selection of regions that would enhance performance in data-scarce scenarios. The model was then evaluated using DV, where it accessed an increasing number of images within different fixed budgets. Once the fixed budget was reached, the segmentation network was trained with LT until it satisfied the early stopping condition in DR. The segmentation network ff for all algorithms was pre-trained with the GTA dataset, a synthetic dataset, and DT. Finally, the segmentation model's performance was measured on the CamVid test set using the Intersection over Union (IoU) score.

4.5.4 Hardware and Software Configurations

The experiments were conducted using a single NVIDIA RTX A5000 GPU with 24 GB of VRAM. The following software configurations were used:

- Operating System: Ubuntu 20.04 LTS
- Deep Learning Framework: PyTorch
- Reinforcement Learning Libraries: OpenAI Gym, Stable Baselines3

Training the active learning agents required approximately 18 hours for 5 runs, while training the segmentation models to evaluate the active learning algorithms took a total of 8 hours for 5 runs at each of the 6 budgets.

4.6 Results

This section details the experimental outcomes of our reinforced active learning framework for semantic segmentation. We evaluate the performance of different active learning techniques, examine the enhancements introduced by our proposed method, and explore the implications of these results.

In Figures 4.3 and 4.4,We evaluate various methods across increasing budgets of labeled 128x128 pixel regions. The x-axis, marked as "Budget," indicates the additional number of regions in thousands and the percentage of utilized unlabeled data. The plots show the means and standard deviations over 5 runs. The segmentation network used in these methods has been pre-trained with the GTA dataset and portions of their respective target datasets. The dashed line indicates 96% of the best performance (Intersection Over Union) achieved by the segmentation network trained with all available labels. Since the performance of the preceding work exceeds that of other baseline models, we will use it as the new baseline model for comparisons with other methods.

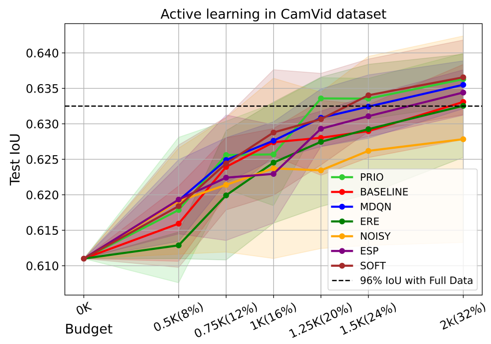


Figure 4.3: Comparisons of various active learning methods.

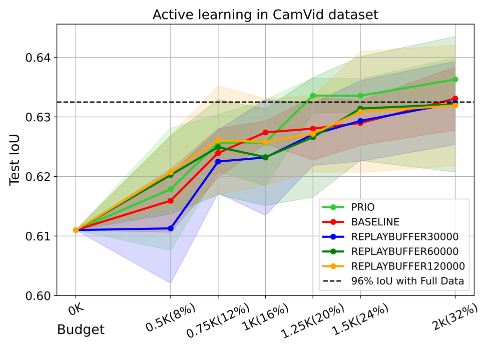


Figure 4.4: Compare PRIO method with varying replay buffer sizes.

In detail, Figure 4.3 illustrates the performance of various methods, including Prioritized Experience Replay (PRIO) [39], the reproduced DQN baseline (BASELINE) [96], Dueling Deep Q-network (MDQN) [11], Emphasizing Recent Experiences (ERE) [97], and Noisy Network (NOISY) [95], Adaptive Epsilon Greedy (ESP) [98], Soft Update for Target Network [17]. It's significant to highlight that at 1.5k regions, the performance of some methods surpasses 96% of the maximum achieved with fully supervised training (with access to all labels). In these experiments, the NOISY model performs the worst, indicating that acquiring new labels does not provide substantial additional information to the model. PRIO and SOFT outperform other methods, including the baseline, across all budget scenarios except for the 1K case for the PRIO method. They achieve this without overfitting the training model, while the other methods produce similar results. This suggests that effective active learning, through selective labeling or incorporating additional information, can help the segmentation model avoid local minima and achieve better performance.

In Figure 4.4, a comparison is made between the baseline and PRIOR methods, considering different replay buffer pool sizes: 600, 30,000, 60,000, and 120,000. Performance remains relatively stable for both 60,000 and 120,000. Interestingly, PRIOR, despite having a smaller replay buffer (around 600 compared to 30,000, 60,000, and 120,000), outperforms the others by a significant margin.

As the experimental results show, implementing a soft target network update can enhance performance. This improvement is because soft updates allow the target network to track changes in the main network more smoothly, resulting in a more stable learning process. This stability helps the algorithm converge to a better policy, leading to improved performance with better results and more reliable Q-value estimations. The Dueling DQN's improvement arises from its ability to separately estimate the value and advantage functions of Q-values. By decoupling these functions, it can better assess the significance of actions in various states, enhancing learning and generalization. This architecture enables a more effective understanding of state value and action advantage, leading to better action selection and performance. Additionally, this separation minimizes variance in learned action-values, stabilizing learning and ensuring more accurate estimations, thus addressing issues of Q-value overestimation or underestimation.

Prioritized Experience Replay (PER) significantly enhances reinforcement learning tasks by improving sample efficiency, stabilizing the learning process, and promoting effective exploration of the state space. By prioritizing experiences based on higher probabilities for transitions with greater TD errors, PER accelerates convergence and fosters efficient learning. Emphasizing rare events helps the agent handle critical scenarios adeptly. The stability of the learning process is maintained through effective policy updates, leading to swift learning and improved convergence. Thorough exploration of the state space, facilitated by focusing on high learning potential transitions, enhances the agent's decision-making and overall performance. PER's use reduces bias from uniform sampling and mitigates high variance issues, leading to more accurate and stable updates of Q-values, thereby enhancing the learning process and performance in various reinforcement learning tasks.

Our experiments revealed that certain techniques, such as incorporating Noisy Networks designed to foster diverse segmentation strategies, adversely affected segmentation performance by introducing instability in the learning process and decreasing accuracy in specific scenarios. In complex environments with sparse rewards or high-dimensional state spaces, challenges arise where adjusting the exploration rate alone may not ensure effective exploration. Inadequate tuning of the exploration rate adaptation and neglect of specific learning dynamics can disrupt the balance between exploration and exploitation, even with adaptive epsilon-greedy methods, posing challenges in achieving the optimal exploration-exploitation equilibrium, particularly in certain environments. Moreover, overemphasizing recent experiences in a Deep Q-Network (DQN) can hinder reinforcement learning performance by reducing sample efficiency, impeding generalization, introducing increased variance, and destabilizing the learning process, thereby compromising the agent's convergence to an optimal policy. This emphasis also limits exploration across the state space, restricting the discovery of critical, infrequently encountered states. Thus, to enhance performance, it is essential to strike a balance between prioritizing recent experiences and maintaining a diverse set of samples that facilitate effective learning and exploration across the entire state space.

4.7 Summary

This chapter advances the field of semantic segmentation by showcasing the benefits of integrating active learning with reinforcement learning. By tackling issues related to dataset imbalances and annotation efficiency, the proposed framework provides a practical solution for enhancing semantic segmentation tasks across various domains. The insights from this research contribute to the development of more sophisticated and precise segmentation algorithms, thereby advancing the broader fields of computer vision and machine learning.

In conclusion, reinforced active learning emerges as a promising approach to addressing significant challenges in semantic segmentation. This research not only offers a robust framework for current applications but also lays the groundwork for future innovations in the field.

Chapter 5

Securing Image Classifiers Against Model Extraction Attacks

5.1 Introduction

In recent years, machine learning has made substantial strides, particularly in image classification. However, as these technologies advance, so do the threats from adversarial attacks. One significant threat is model extraction attacks, where an adversary attempts to replicate a machine learning model's functionality without accessing its internal parameters or training data. These attacks pose serious risks, including intellectual property theft and potential subsequent adversarial attacks.

This chapter examines the efficacy of various learning techniques in conducting model extraction attacks on image classifiers. The primary focus is on evaluating how Deep Q-network (DQN) extensions can enhance the performance of surrogate models, which are replicas of the original models created through these attacks. Additionally, the chapter explores synthetic data generation techniques and their role in facilitating efficient model extraction.

We start with a comprehensive overview of the background and related work in the field, highlighting progress and existing challenges. The methodology section then details the experimental setup, including the DQN extensions and synthetic data generation methods used in our study. Following this, we present the results of our experiments, comparing the performance of different techniques and analyzing the implications of our findings.

By the end of this chapter, we aim to provide valuable insights into the vulnerabilities of image classification models and the effectiveness of various model extraction strategies. This knowledge is crucial for developing robust defense mechanisms to protect machine learning models from such adversarial threats.

5.2 Background and related works

The rise of adversarial attacks targeting image classification models has become a major issue, exposing vulnerabilities that can be exploited to deceive these models into making incorrect predictions or classifications [101, 102]. Among the various categories of adversarial attacks, the black-box adversarial attack problem is particularly significant due to its practical relevance and the realistic constraints it imposes on attackers. Unlike white-box attacks, which assume attackers have complete information about the architecture and parameters of the model, black-box attacks operate under the assumption that the attacker has little to no understanding of the target model's internal workings [103]. This scenario is more common in real-world settings, where attackers usually do not have direct access to proprietary or confidential models.

5.2.1 Overview of Model Extraction Attacks

Model extraction attacks involve an adversary attempting to create a surrogate model that mimics the behavior of a target model by systematically querying it. These attacks can be categorized into black-box and white-box attacks. In black-box attacks, the attacker only has access to the input-output pairs of the model, whereas in white-box attacks, the attacker has full knowledge of the model's architecture and parameters. Black-box attacks are particularly concerning due to their practicality in real-world scenarios, where attackers often do not have access to the internal workings of the models they are targeting.

A key challenge in executing successful model extraction attacks lies in the strategic utilization of inquiries to the target model. Each query provides feedback regarding the model's output for a given input, but excessive querying can not only raise suspicion but also become impractical due to rate limits or cost considerations [104]. Prior work in this domain has explored various strategies to generate adversarial examples under these constraints, yet there remains a significant gap in effectively utilizing query feedback [105]. This inefficiency in feedback utilization leads to suboptimal attack strategies that require a large number of queries, thereby increasing the risk of detection and reducing the feasibility of the attack [106]. By systematically querying the model and analyzing its responses, attackers can infer valuable information, allowing them to construct a surrogate model that closely resembles the original [107, 108].

Figure 5.1 provides a standard example of an adversarial model extraction attack, illustrating how the surrogate model is refined using feedback from queries to more accurately mirror the victim model.

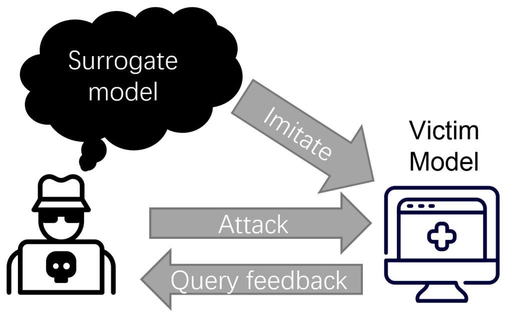


Figure 5.1: Black-box adversarial model extraction attack.

5.2.2 Previous Studies on Adversarial Attacks and Machine Learning Security

The field of adversarial machine learning has seen extensive research aimed at understanding and countering various types of attacks. Han et al. [109] provide a comprehensive survey on adversarial attacks across different domains, highlighting the vulnerabilities of machine learning models to these attacks and the importance of developing robust defense strategies. Other studies, such as those by Pitropakis et al. [110], have classified and reviewed different types of adversarial attacks, emphasizing the need to understand these threats to develop effective countermeasures.

Recent research has also focused on specific adversarial attack methodologies. For instance, Dong et al. [101] explored momentum-based iterative algorithms to enhance the effectiveness of adversarial attacks. Similarly, Fang et al. [111] demonstrated the practical viability of black-box attacks on MNIST classification models, highlighting the ease with which these attacks can be conducted in practice.

5.2.3 Reinforcement Learning and Active Learning for Model Extraction

Reinforcement learning (RL) and active learning (AL) have emerged as powerful tools in the context of model extraction attacks. RL, particularly through Deep Q-networks (DQN) and its extensions, has shown promise in optimizing the sample selection process during model extraction. By leveraging RL techniques, attackers can strategically choose the most informative samples to query the target model, thereby enhancing the efficiency of the attack.

Active learning, on the other hand, focuses on selecting the most informative data points to train models, which is especially useful in scenarios with limited query budgets. Combining RL and AL techniques allows for a more refined and adaptive approach to model extraction, as demonstrated by recent studies such as those by Zhang et al. [112] and Chen et al. [113].

Significant progress has been made in understanding and developing techniques for model extraction attacks. The use of DQN extensions, such as Double DQN [93], Dueling DQN [11], Noisy Network [95], and Prioritized Experience Replay (PER) [39], has been explored to enhance the efficiency of these attacks. Additionally, synthetic data generation techniques, including Jacobian-based methods [114], Linf-projected Gradient Descent (LinfPGD) [115], and Fast Gradient Sign Method (FGSM) [116], have been investigated for their potential to improve surrogate model training.

Despite these advancements, challenges remain in fully understanding the vulnerabilities of machine learning models and developing robust defenses. This chapter builds on the existing body of research by providing a comprehensive evaluation of DQN extensions and synthetic data generation techniques in the context of model extraction attacks. It offers new insights and potential directions for future research.

5.3 Methods

5.3.1 Problem Definition

The main challenge in model extraction attacks is creating a surrogate model that can closely replicate the performance of a sophisticated victim online image classification model using a reduced dataset. This issue arises due to limited access to the victim model or API and the constraints of computational costs, data privacy, or proprietary restrictions that limit the amount of data available for training. The task involves selecting an optimal subset of images that maintains the diversity and complexity of the larger dataset to effectively train the surrogate model.

The surrogate model's goal is to minimize loss measures, such as mean squared error (MSE), between the surrogate and victim models' predictions while achieving similar accuracy, precision, and recall as the victim model. Our research investigates various methods for image selection, data augmentation, and model training that maximize the potential of limited data, thereby reducing the reliance on large datasets without compromising performance. Our aim is to provide meaningful insights into the effectiveness of sample selection and data-free tactics in conducting model extraction attacks.

5.3.2 Active Learning with Reinforcement Learning for Model Extraction Attack

Our goal is to select an optimal subset of images that captures the full diversity and complexity of a larger dataset, enabling effective training of the surrogate model. This task can be framed as an active learning problem. We approach this challenge using a Markov decision process model, inspired by studies such as Sener et al. (2017) [117], Casanova et al. (2020) [96], and Gao et al. (2022) [118]. The policy network π, acting as an agent in reinforcement learning, is represented by a deep Q-network [6]. This method allows the model to develop selection strategies by leveraging outputs from both the victim and surrogate models, guided by data-driven insights. Our approach differs from existing methods in several key areas: the problem we address, our definitions of states, actions, and rewards, and the specific reinforcement learning algorithm used to determine the best policy.

We aim to enhance the performance of surrogate models, S, parameterized by θ, using a smaller number of images from a large dataset. This process is iteratively conducted until a specified threshold is met. At each iteration, t, a policy network π, parameterized by ϕ, selects a single image from a randomly sampled subset U for training the surrogate model. The selected images are also processed by the victim model to acquire its outputs. To assess performance, we employ a straightforward metric: the percentage of images from the total dataset that are correctly classified.

In our methodology, we start by training the victim model using the entire training dataset to establish it as the target. We then divide the training data into three distinct segments. A specific portion, designated as D_t, is used for pre-training the surrogate models. To mitigate overestimation issues during the policy network's training phase, we initialize ten different surrogate models, all sharing the same architecture. For training the policy network π, another subset, D_r, is employed to facilitate the active learning process over multiple episodes. This process focuses on refining an acquisition function designed to enhance the surrogate model's performance using a carefully selected set of K images. We utilize a separate data partition, D_e, to evaluate the policy network, where the surrogate models are trained up to the defined budget B. The total query budget is $K * B$, but for simplicity, we refer to it as B.

Figure 5.2 describes the main workflow of training for the Policy Network and surrogate models. We split the MNIST dataset into three parts. In Phase 1, we use the split D_t to pre-train the surrogate models. In Phase 2, U images are sampled uniformly from the split D_r. The concatenated U images feature, computed using a CNN feature extractor, serves as the state representation. The policy π selects K images and provides them to the surrogate and victim models. The reward is obtained from the prediction MSE of the surrogate and victim models. After training the policy π, we fix it and continue training the surrogate model in Phase 3. Phases 2 and 3 are iteratively trained until the final budget B is achieved.

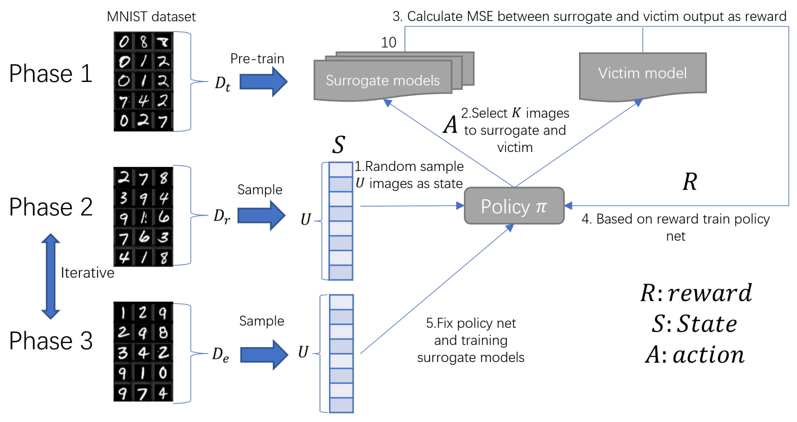

MNIST dataset
Phase 1
Phase 2
Phase 3
Iterative
Pre-train
D_t
Sample
D_r
D_e
U
S
10
Surrogate models
Victim model
3. Calculate MSE between surrogate and victim output as reward
A
2.Select K images to surrogate and victim
1.Random sample U images as state
Policy π
R
4. Based on reward train policy net
5.Fix policy net and training surrogate models
R: reward
S: State
A: action

Figure 5.2: The main workflow of training for Policy π and surrogate models.

The MDP framework is defined by a sequence of transitions $(s_t, a_t, r_{t+1}, s_{t+1})$. Within this framework, for any given state $s_t \in \mathcal{S}$, the agent can choose an action $a_t \in \mathcal{A}$, which involves selecting specific images from the unlabeled set U. Each action is associated with the selection of a particular image. The agent then receives a reward r_{t+1}, calculated based on the feedback from both the surrogate and victim models. It is important to note that the definitions of states and actions are not contingent on the specific architecture of the surrogate models. Our goal is to develop a policy that enhances sample selection to improve the accuracy of the surrogate model. To achieve this, we utilize a DQN architecture, employing transitions from an experience replay memory ε to train the policy π.

We begin by assigning a set of initial weights θ_0 to the ten surrogate models, which have been trained using the dataset D_t.The following steps are performed during each iteration t:

1. A subset U is randomly sampled from D_r.
2. The state s_t is computed from the feature extractor using U.

3. The policy agent chooses action a_t using an ϵ-greedy policy, with each action corresponding to the selection of one image for training.

4. The surrogate models and the victim model generate output based on the selected image.

5. The agent receives the reward r_{t+1}, calculated as the performance differential between the surrogate model and the victim models.

6. The surrogate models are trained for one iteration on the recently selected images.

5.3.3 DQN

The desired agent is designed to follow an optimal policy, associating each state with an action that maximizes total future rewards. We conduct our DQN training using a divided dataset D_r. The agent observes the current state U of the environment. Using a neural network, the agent selects actions (images) based on the current Q-values, which estimate the quality of a state-action combination. After taking an action, the agent receives a reward and transitions to a new state. The Q-values are updated based on the reward received and the maximum predicted Q-values for the new state, using the Bellman equation. To stabilize learning, DQN uses a replay buffer where past experience tuples (state, action, reward, next state) are stored. The agent then samples from this buffer to break the correlation between sequential observations.

The development of Deep Q-Networks (DQN) has led to numerous important advancements, each aimed at addressing specific shortcomings and enhancing the overall effectiveness of the algorithm. These advancements have significantly improved the functionality of the target DQN framework, enabling more effective and resilient learning in complex settings. In this segment, we introduce the enhancements to the DQN model that were evaluated. In the actual implementation, we replace the DQN model with its extensions solely to evaluate and compare their performance. We also test different combinations of these extensions to determine which synergies are most effective, providing insights into how these extensions interact.

Deep Q-networks (DQN) are a reinforcement learning technique that optimizes decision-making processes by approximating the optimal action-value function. In the context of model extraction, DQNs can be employed to select the most informative samples for querying the target model. We explore several extensions to the standard DQN to enhance its performance:

- Double DQN (DDQN): DDQN mitigates the overestimation bias present in standard DQN by decoupling the action selection and action evaluation processes. This is accomplished by employing two separate networks: one to select the best action and another to evaluate the value of that action.

- Dueling DQN: This extension separates the estimation of state value and advantage functions, enabling the network to discern valuable states from advantageous actions, thereby enhancing the stability and performance of the learning process.

- Noisy Network: Noisy networks introduce randomness into the network's weights, promoting more effective exploration during training by encouraging the network to dynamically try different actions.

- Soft Update: This technique incrementally updates the target network's weights using a small parameter τ, which gradually blends the weights of the main network. This approach ensures a more stable training process compared to hard updates.

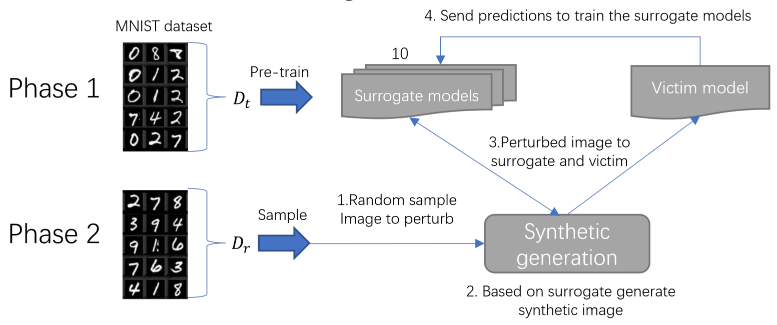
- Prioritized Experience Replay (PER) enhances learning efficiency by prioritizing experiences with higher temporal difference (TD) errors, ensuring that more informative samples are replayed more frequently during training.

- Emphasizing Recent Experience (ERE) prioritizes recent experiences by gradually increasing the sampling frequency of new data points, allowing the model to adapt more effectively to recent changes.

5.3.4 Synthetic Data Generation Techniques

In training surrogate models using synthetic generation techniques, our workflow consists of two main phases to utilize the image dataset effectively. In Phase 1, a portion of the dataset, denoted as D_t, is used to pre-train the surrogate models. This initial training phase helps establish a foundational capability for the surrogate models to approximate the behavior of the target or victim model. In Phase 2, we uniformly sample images from a different dataset split, D_r, to start the synthetic data generation process. The pre-trained surrogate model generates synthetic image data by perturbing the original images to create modified versions.

These altered images are then inputted into the victim model, which processes them and provides predictions based on its learned parameters. The combination of the perturbed images and the victim model's predictions is used to further train the surrogate models. This methodology ensures that the surrogate models are trained on both real and synthetically generated adversarial examples, fine-tuning them to closely mimic the victim model's behavior. In Figure 5.3 demonstrates the overall workflow. We split the MNIST dataset into two parts. In Phase 1, split D_t is used to pre-train the surrogate models. In Phase 2, images are uniformly sampled from split D_r, synthetic image data is generated using the surrogate model, and the altered images are submitted to the victim model to obtain predictions. The perturbed images and their associated predictions are then used to further train the surrogate models.



4. Send predictions to train the surrogate models
MNIST dataset
10
Phase 1
Pre-train
D_t
Surrogate models
Victim model
3.Perturbed image to surrogate and victim
1.Random sample Image to perturb
Sample
Phase 2
D_r
Synthetic generation
2. Based on surrogate generate synthetic image

Figure 5.3: The main workflow of training surrogate models using synthetic generation techniques.

Jacobian-Based method: The Jacobian-based technique is a gradient-based strategy that creates synthetic samples for model extraction by utilizing the target model's Jacobian matrix [114]. The sensitivity of the model's outputs to its inputs is captured by the Jacobian matrix, which provides valuable information for generating informative examples. For a target model f and an input sample x are used to calculate the Jacobian matrix $J(x)$.

$$J(x) = \frac{\partial f(x)}{\partial x}$$

The partial derivatives of the model's outputs with respect to each input feature are contained in the Jacobian matrix $J(x)$, which essentially captures the local behavior of the algorithm around the input value x.

We can apply a perturbation onto the data x in the way of the Jacobian matrix to produce synthetic samples. In particular, a synthetic sample x' can be acquired as follows:

$$x' = x + \epsilon \cdot \text{sign}(J(x))$$

where the step size that regulates the perturbation's magnitude is ϵ, and $\text{sign}(\cdot)$ is the sign function applied element-wise to the Jacobian matrix.

The underlying idea of this approach is that perturbing the input sample in the direction indicated by the Jacobian matrix tends to generate synthetic samples that are highly sensitive to the target model's outputs. These synthetic samples can then be utilized to train a replica model, potentially enhancing its ability to approximate the target model with improved performance and accuracy. To optimize the process of generating synthetic samples, various strategies can be applied, such as experimenting with different step sizes, merging multiple Jacobian-based samples, or incorporating additional constraints or regularization techniques.

It is important to note that the Jacobian-based method depends on having access to the target model's gradients, which may not always be feasible. In such scenarios, alternative gradient estimation techniques or other synthetic data generation methods might be required.

The Fast Gradient Sign Method: The Fast Gradient Sign Method (FGSM) [116] is a gradient-based approach for creating adversarial examples, which can also be adapted for generating synthetic data in model extraction attacks. FGSM is both efficient and computationally inexpensive, producing artificial samples by utilizing the target model's gradients.

Given a target model f, an input sample x, and a loss function $\mathcal{L}(\cdot, \cdot)$, By perturbing x in the orientation of the gradient of the loss function in relation to the input, the FGSM creates a synthetic sample x':

$$x' = x + \epsilon \cdot \text{sign}(\nabla_x \mathcal{L}(f(x), y))$$

where y is the ground truth label or target output for x. The sign function applied element-wise to the gradient is $\text{sign}(\cdot)$, and a tiny variable ϵ regulates the size of the perturbation.

The core idea of FGSM is to move in the direction of the gradient of the loss function with respect to the input, creating a synthetic sample x' that maximizes the loss function $\mathcal{L}(f(x'), y)$. This perturbation aims to produce samples that are informative and challenging for the target model, potentially improving model extraction performance when used to train a replica model. FGSM is efficient as it requires only a single gradient computation per input sample. However, it may be vulnerable to gradient masking or obfuscation techniques employed by the victim model as a defense against model extraction attacks. To enhance the effectiveness of FGSM for model extraction, various extensions and modifications can be explored, such as iterative methods (e.g., Projected Gradient Descent), incorporating additional constraints or regularization techniques, or combining FGSM with other synthetic data generation methods.

Linf-Projected Gradient Descent: An iterative gradient-based method for generating adversarial examples is the Linf-projected Gradient Descent (LinfPGD) [115]. This technique can be adapted to create synthetic data for model extraction attacks. LinfPGD is an extension of the Fast Gradient Sign Method (FGSM) and aims to produce more robust and effective synthetic samples. Given a victim model f, an input sample x, a loss function $\mathcal{L}(\cdot, \cdot)$, and a maximum perturbation budget ϵ, Under the L_∞ norm constraint, LinfPGD iteratively updates x in the path of the direction of the gradient of the loss function to produce a synthetic sample x':

$$x't + 1 = \Pi\epsilon \left(x'_t + \alpha \cdot \text{sign}(\nabla_x \mathcal{L}(f(x'_t), y)) \right)$$

where y is the ground truth label or target output for x, α is the step size, $\text{sign}(\cdot)$ is the sign function applied element-wise to the gradient, and $\Pi_\epsilon(\cdot)$ is the projection operator that projects the disturbance to the L_∞ sphere with radius ϵ.

The iterative nature of LinfPGD allows for more sophisticated and effective synthetic sample generation compared to FGSM. By frequently updating the synthetic sample in the direction of the loss function gradient, LinfPGD can explore a larger region of the input space, creating samples that are more informative and challenging for the target model. The L_∞ norm constraint ensures that the synthetic samples remain closely bounded to the original input sample, preserving the semantic and structural characteristics of the input data.

LinfPGD offers several advantages over FGSM, including improved robustness to gradient masking or obfuscation techniques, and the ability to generate more diverse and informative synthetic samples. However, it is computationally more expensive than FGSM due to its iterative nature. Various modifications and extensions to LinfPGD can be explored, such as adjusting the step size, incorporating momentum or restart strategies, or combining it with other synthetic data generation techniques to further enhance its effectiveness for model extraction.

Data-Free Model Extraction: A technique called "Data-Free Model Extraction" [119] aims to create a functional duplicate of a target model without access to its original training data. This method raises significant privacy concerns, as it allows an adversary to replicate a model's functionality solely through querying, without needing any proprietary data from the model owner.

The data-free model extraction process typically involves the following steps: In the absence of training data, the adversary generates a set of synthetic input samples to query the victim model. These artificial samples can be created using gradient-based approaches, Generative Adversarial Networks (GANs), or other data synthesis techniques. The synthetic input samples are used to query the victim model, and the corresponding output predictions or labels are collected. The gathered input-output pairs from the target model queries replace the original training data. These query responses are used to train a replica model, effectively extracting the functionality of the target model.

The primary challenge in data-free model extraction is generating synthetic input samples that can effectively capture the target model's behavior across a variety of inputs. Techniques such as gradient-based methods or GANs can be employed to create synthetic samples that provide useful information and accurately represent the decision boundaries of the target model.

5.4 Experiments

This section begins by detailing the objectives that guide our experiments. It then provides a summary of the datasets used to evaluate our methodology, describes the experimental setup, and presents the comparative analysis. We use the MNIST dataset to evaluate the algorithm, with the results presented here. Each model was trained using a single NVIDIA RTX A5000 GPU with 24 GB of VRAM.

5.4.1 Experiment Objectives

To Evaluate the Effectiveness of DQN Extensions for Model Extraction Attacks: The objective is to determine the impact of various Deep Q-network (DQN) extensions—such as Double DQN, Dueling DQN, Noisy Networks, and Prioritized Experience Replay (PER)—on the efficiency of surrogate models in model extraction attacks. This involves evaluating the extensions' ability to select informative samples that optimize adversarial benefits under limited query budgets.

To Assess Synthetic Data Generation Techniques: This study aims to evaluate the performance of various synthetic data generation methods, including the Jacobian-based method, Linf-projected Gradient Descent (LinfPGD), and Fast

Gradient Sign Method (FGSM), in training surrogate models for model extraction. The focus is on these methods' ability to enhance the performance of adversary models. Additionally, the study explores the feasibility and effectiveness of data-free model extraction attacks, which attempt to replicate a target model's functionality without access to the original training data. This involves assessing the performance of such attacks in constrained query environments.

5.4.2 Experimental Setup

MNIST: In the fields of computer vision and machine learning, the MNIST dataset (Modified National Institute of Standards and Technology dataset) [120] is a well-known benchmark for image processing systems and algorithms. It consists of 70,000 28×28 pixel grayscale images of handwritten digits from 0 to 9, with 10,000 images reserved for testing and 60,000 images used for training.

For our purposes, we divided the training set using uniform sampling into three subsets: 1,000 images to create D_t (used to obtain the pre-trained surrogate model), 21,000 images to create D_r, and the remaining images to create D_e. D_e is used to continue training the surrogate model while evaluating the fixed trained policy. The performance of the surrogate model is then evaluated using the test set.

Implementation Details: From the training set D_r, we sample a subset of $U = 20$ images to construct the state representation. A Convolutional Neural Network (CNN) feature extractor is used to process these images, and the extracted features are concatenated to form a state. The policy network's action corresponds to the index number of the chosen image within U.

The split D_r is used to optimize the hyper-parameters, which are selected based on the best configuration for our technique as well as for baseline methods. Additionally, D_r is used to create the rewards for the DQN. The reward is determined by the Mean Squared Error (MSE) difference between the victim model's and the surrogate model's predictions, and it is always negative. To address the overestimation problem within the policy network, we employ 10 surrogate models and calculate the reward by taking the mean of the 10 MSE values.

The victim model chosen for our study is ExquisiteNetV2 (2021) [121], which achieved an accuracy of 99.71% on the MNIST dataset.

In synthetic data generation attacks, we continue to use images from D_r. An image is randomly selected for perturbation using the previously mentioned techniques. This perturbed image is then fed into the victim model to produce a prediction. The surrogate model is subsequently trained using pairs of the perturbed image and its corresponding prediction.

For the data-free model extraction method, we used randomly generated noise images as inputs to train the surrogate model, following a similar approach to the other techniques. Additionally, we employed the entropy method, a purely active learning approach without reinforcement learning, to serve as a benchmark against the synthetic data generation techniques.

Evaluation: The evaluation process consists of two stages: assessing the performance of the policy network and evaluating the surrogate model's effectiveness using our approach. With a set budget B, the policy network π is trained using D_r to prioritize selecting images that will enhance performance in a sparse data environment. We test the learned acquisition function and the synthetic data generation techniques on D_e, training the surrogate models for various budgets until the budget is exhausted.

Once the budget is reached, we fix the policy network and proceed to evaluate the surrogate model with the test data. The objective of our work is to demonstrate that through the application of the described method, the surrogate model can produce diverse outcomes. Our goal is to compare this method with others to identify the most efficient approach, one that requires fewer images to achieve comparable performance. Essentially, this involves training the surrogate model with the same number of images while achieving superior results compared to alternative methods.

5.4.3 Results

Figure 5.4 illustrates the performance of various DQN models, including Double DQN (DDQN), Noisy Network, Dueling DQN, Soft Update, Emphasizing Recent Experiences (ERE), and Prioritized Experience Replay (PER). Each line represents a different DQN model, with the x-axis indicating training surrogate model budgets from 1,000 to 10,000 in intervals of 1,000, and the y-axis displaying the model's accuracy. The "Budget" labeled x-axis shows the number of queries used to train the surrogate model. We set specific policies and trained the surrogate models accordingly. Following this, we evaluated the surrogate models using a test dataset.

Given the significant variability inherent to reinforcement learning (RL) algorithms, we conducted 10 trials to average the accuracy across various trained policy models. The policy models were trained with $K = 4$ and a budget of 3,000. Since the DQN model was initially trained using a budget of 3,000, the total budget range represented in this figure spans from 3,000 to 13,000.

We also integrated all DQN extensions to identify the configuration that achieves optimal performance. The findings reveal that this combination of methods outperforms each method when used individually. Specifically, among the standalone applications, PER demonstrated the highest performance.

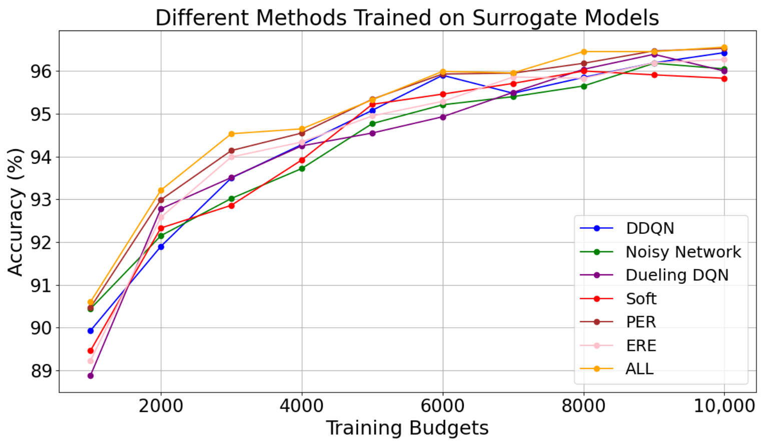


Figure 5.4: Performance of different DQN policies on surrogate models.

We also investigate how an increase in budget for training policy models affects the performance of our method. Figure 5.5 presents the results of varying the budget for different trained policy performances. The x-axis represents the training budget for the surrogate model, ranging from 100 to 1,000 in increments of 100, while the y-axis shows the model's accuracy. The policy variants evaluated include

DoubleDQN (DDQN), Noisy Network, Dueling DQN, Soft Update, Emphasizing Recent Experiences (ERE), and Prioritized Experience Replay (PER).

It is important to note that direct comparison across the three subfigures is not appropriate since the query budgets used for training the DQN models differ in each case. The total budgets are 4,000 for the first subplot, 5,000 for the second, and 6,000 for the third. Due to limited accessibility to the victim model, we focus on evaluating the surrogate model with a constrained budget of 1,000 queries. This evaluation budget includes the queries used during the training of the policy models, so the total query budget will be 1,000 plus additional allocations for the different policy model variants.

Across all models, Prioritized Experience Replay (PER) generally yields higher accuracy compared to other methods, with Double DQN also performing well. We evaluated the performance with various sample sizes of selected images, specifically for $K = 1$ and $K = 3$. However, these configurations did not outperform $K = 4$. Therefore, we present our best performance results achieved with a sample size of $K = 4$.

For synthetic data generation methods, the experiments aimed to evaluate the effectiveness of various techniques in producing samples that enhance model performance in model extraction attacks. Specifically, we examined the Jacobian-based method, Linf-projected Gradient Descent (LinfPGD), Fast Gradient Sign Method (FGSM), Entropy, and a data-free model extraction approach. To ensure robust and fair analysis, the experiment was repeated 10 times, and the mean of the results was calculated to account for variability. We also included one DQN method for comparison with the synthetic data generation methods.

Figure 5.6 illustrates the performance of the Jacobian-based method, Linf-projected Gradient Descent (LinfPGD), Fast Gradient Sign Method (FGSM), Entropy, and the data-free model extraction method. The x-axis indicates training budgets from 1,000 to 10,000 in intervals of 1,000, while the y-axis displays the model's accuracy. For comparison, we also included the PER method trained using the DQN model. Since a budget of 3,000 was used to train the DQN model, its performance is plotted from a budget of 3,000 up to 10,000.

Among these techniques, PER emerged as the superior method, consistently outperforming the others in terms of achieving higher accuracy in the model extraction process. Within the synthetic data generation methods, LinfPGD outperformed the other methods. Conversely, the data-free method demonstrated significantly lower performance, suggesting that the allocated query budget may have been insufficient for generating realistic and useful images for training purposes. The effectiveness of synthetic data generation in model extraction attacks appears to be highly dependent on the ability to produce quality training data, a criterion where the data-free method fell short under the constraints of a limited query budget.

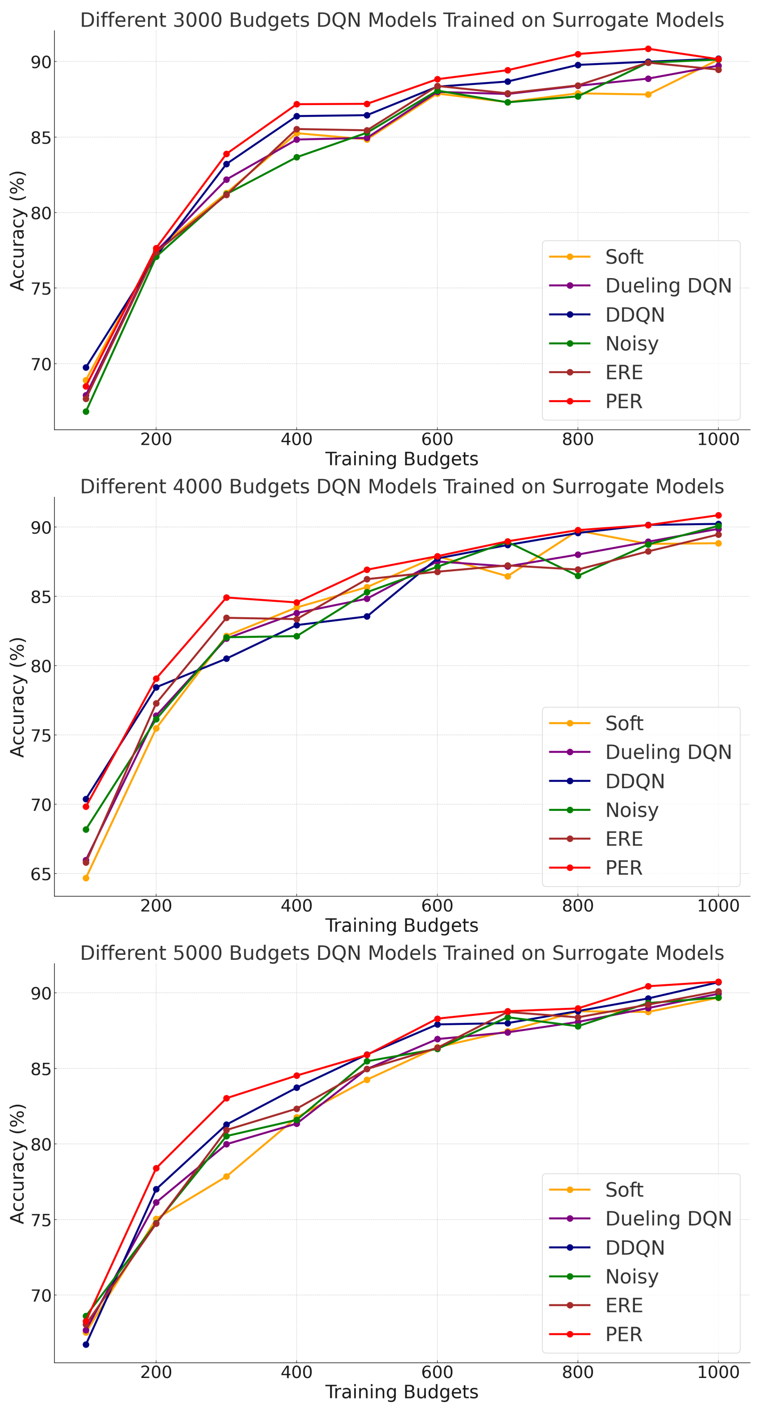


Figure 5.5: Performance of various DQN policy variants on surrogate models as the training budget increases.

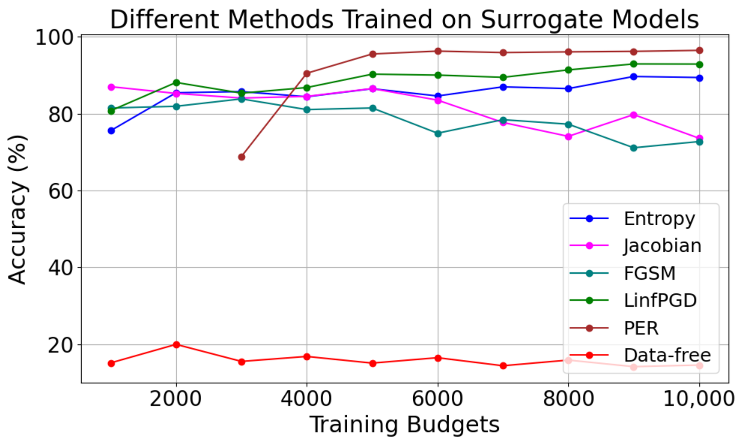


Figure 5.6: Performance of different synthetic data generation techniques with PER method on surrogate models.

5.4.4 Ablation Studies

We have demonstrated that several enhancements to DQN can achieve better performance. Subsequently, we conducted additional experiments to test combinations of these extensions and assess how the removal of each DQN extension affects model extraction performance. To better understand the influence of each component on the DQN agent, we performed ablation studies. In each study, we systematically removed one component from the complete set of extensions. Note that ERE is implemented based on PER, so when PER is removed in our ablation study, ERE is also omitted.

Figure 5.7 illustrates the comparison of the accuracy scores of the full combination of extensions to five ablated variants. The x-axis indicates training budgets from 100 to 1,000 in intervals of 100, while the y-axis displays the model's accuracy. Given that a budget of 3,000 was used to train the DQN model, the total budget is 4,000.

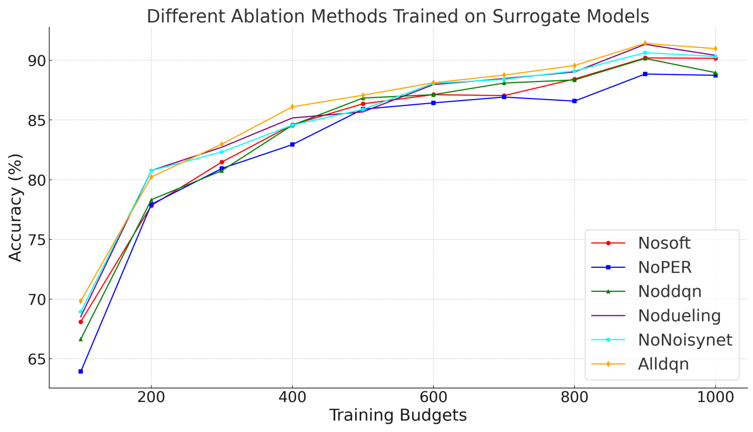


Figure 5.7: Performance of different ablation methods trained on surrogate models.

The findings indicate that Prioritized Experience Replay (PER) was the most critical element among the DQN enhancements, as its removal significantly reduced performance. The soft update and double DQN were next in importance, as shown by the results of other ablations. Notably, in the initial stages of learning, specifically during the first 200 queries, there was a marked difference between the ablations and the full agent. Overall, removing the dueling network and Noisy net from the combined setup did not result in a significant performance change. Both the dueling DQN and Noisy net tend to introduce instability, complicating the maintenance of a stable learning trajectory and convergence to an optimal policy.

5.4.5 Generalizability

In addition, we conducted preliminary experiments on the CIFAR-100 dataset to demonstrate the robustness of our model. The CIFAR-100 dataset is a widely used benchmark in machine learning, especially for image recognition tasks. Developed by the Canadian Institute for Advanced Research (CIFAR) [122], this dataset contains 60,000 32 $\times$ 32 color images categorized into 100 classes, with each class comprising 600 images.

We divided the training set using uniform sampling into three subsets: 10,000 images to create D_t (used to obtain the pre-trained surrogate model), 21,000 images to create D_r, and the remaining images to create D_e. This setup makes the CIFAR-100 dataset particularly challenging and useful for developing and testing advanced image classification algorithms that require fine-grained recognition capabilities.

All the training and evaluation processes followed the previous methods. To simplify, we conducted five trials to average the accuracy across the policy models. The victim model selected for our study is EffNet-L2 (2020) [123], which achieved an accuracy of 96.08

Figure 5.8 compares our combination method with five different DQN extensions. The x-axis indicates training budgets from 2,000 to 20,000 in intervals of 2,000, and the y-axis displays the model's accuracy. Given that a budget of 3,000 was utilized to train the DQN model, the total budget ranges from 3,000 to 23,000. The policy models were trained with $K = 4$ and a budget of 3,000. We doubled the queries for better results, so the total budget range represented in this figure spans from 3,000 to 23,000.

From experiments with the CIFAR-100 dataset, we observe that larger victim and surrogate models increase the required runtime. The results show that performance on the CIFAR-100 dataset is relatively lower. This dataset is inherently more complex than simpler ones such as MNIST because it comprises 100 classes. This level of granularity necessitates a model with greater capacity to adequately distinguish between the numerous classes. Notably, as model sizes increase and datasets become more complex, performance tends to decrease, requiring additional queries to effectively extract the model.

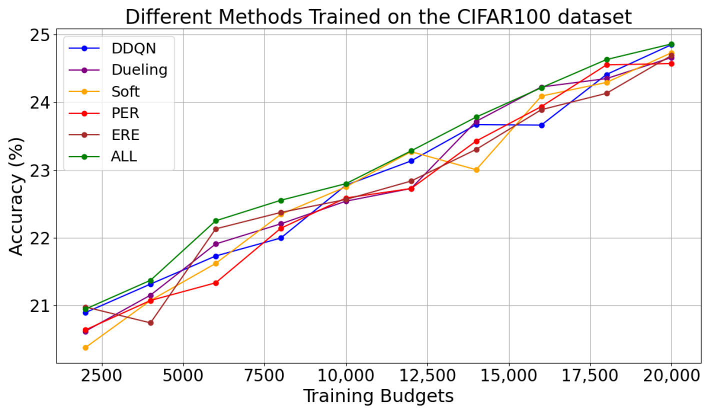


Figure 5.8: Performance of different methods trained on CIFAR100 dataset.

5.5 Discussion

In this section, we analyze the primary results of the tests and discuss the findings of the experiment. Across all experiments, utilizing DQN models outperformed synthetic data generation methods. This superiority stems from the fact that DQN models leverage the original data to train the surrogate model, thereby incorporating the strongest knowledge derived from the dataset. The experimental findings indicate that employing the Prioritized Experience Replay (PER) technique leads to improved performance across all experiments.

This suggests that the PER approach, which likely involves more sophisticated decision-making or optimization processes, is more effective at utilizing the training budget to enhance model performance. PER ranks experiences according to their temporal difference (TD) error, or the discrepancy between the expected and actual rewards. Experiences with higher TD errors are considered more valuable for learning because they indicate situations where the model's predictions were significantly off. By focusing on these experiences, PER enables the model to learn and adapt more quickly to critical aspects of the environment.

By concentrating on experiences that the model currently finds most challenging, PER can speed up the learning process, leading to faster convergence towards optimal policies. This targeted learning approach can be particularly effective in complex environments where certain states and actions are more critical to mastering the task. Additionally, PER includes safeguards against the model focusing too much on a small number of experiences, thereby maintaining a balance between exploring new knowledge and exploiting known information. This balance helps achieve not only faster but also more stable learning outcomes.

DDQN ranks second because it addresses the overestimation of action values that can occur with standard DQN. By decoupling the selection of actions from their evaluation, DDQN reduces bias in the learning updates, leading to more stable and reliable learning outcomes. This refinement helps it perform well, though it lacks the targeted efficiency improvements of PER.

ERE ranks third and enhances learning by focusing on a mix of recent experiences and the entire history, gradually concentrating more on recent interactions. This approach helps adapt to changes and optimize the model's performance over time by balancing between exploiting learned knowledge and exploring new information. However, it may not be as targeted or efficient in prioritizing critical learning opportunities as PER.

The Noisy Network ranks the lowest among the tested methods. While it introduces noise into the network parameters to encourage exploration by diversifying the policy's behavior, this can also lead to less stable training and potentially poorer performance when the task requires precise adjustments based on subtle cues from the environment, as is the case in model extraction attacks. The randomness introduced might not always align with the most informative learning paths, unlike the more structured approaches of PER, DDQN, and ERE.

In summary, PER's potential superiority lies in its ability to prioritize learning from the most informative experiences. This optimizes the learning process, accelerates convergence, and eventually results in enhanced DQN model performance in model extraction tasks.

One critical aspect of handling CIFAR-100 is the need for a model with greater capacity. This may involve exploring deeper or more complex neural network architectures capable of capturing a wide range of features across diverse image types. Models such as convolutional neural networks (CNNs) that are deeper and equipped with layers like residual connections might be particularly effective. These models can learn rich and hierarchical feature representations, making them more adept at managing the complexities of such a dataset.

The observed low performance on CIFAR-100 could also result from overfitting to specific features that do not generalize well across the entire dataset. This issue can be mitigated by employing techniques such as data augmentation, regularization, and dropout, which enhance the model's ability to generalize rather than memorize the training data.

Another approach to improving performance is increasing the number of queries during training. This method, often associated with active learning, involves iteratively selecting the most informative samples from the dataset to train the model. By strategically increasing the number of queries, it is possible to refine the training process to focus on the most beneficial examples, potentially improving model accuracy and robustness.

In situations where obtaining the original dataset is challenging, the use of synthetically generated data becomes necessary. Among the synthetic data generation methods evaluated, the LinfPGD approach demonstrated superior performance compared to the other techniques. This method iteratively fine-tunes adversarial examples, carefully exploring the space around the original input within the constraints of the L_∞ norm. This methodical exploration allows it to identify perturbations that are both subtle and highly effective at deceiving the model, leading to more potent adversarial examples. Its systematic approach to exploring adversarial space, coupled with controlled perturbations and iterative optimization, enables LinfPGD to achieve superior performance in creating synthetic data for model extraction.

The Jacobian-based method follows LinfPGD in effectiveness. It leverages the model's Jacobian matrix to determine the directions in which the input features should be perturbed to maximize output changes, focusing on the areas where the model is most sensitive. This method effectively uses gradient information to create informative adversarial examples, which can significantly aid in model extraction by highlighting the model's decision boundaries. However, because it is generally a single-step approach, it might not capture as nuanced a view of the model's vulnerabilities as the iterative LinfPGD method, resulting in slightly less effective synthetic data for training.

FGSM is a more straightforward and faster approach than both LinfPGD and the Jacobian-based method, as it involves just a single step of perturbation along the gradient of the loss. This makes FGSM computationally cheaper and quicker to apply, but it also tends to be less refined. FGSM's single-step nature means it might not find the optimal perturbation for the most informative adversarial examples, especially against well-regularized or complex models. It often leads to examples that are either too easy for the model to classify correctly or too aggressive, pushing the examples out of the data distribution and making them less effective for training robust surrogate models.

Data-free methods rely on randomly generated noise as input images to train the surrogate model. However, the results are poor because the input-output pairs consist of random noise images paired with the victim model's outputs, which lack meaningful correspondence.

5.6 Summary

This chapter examined the efficacy of various learning techniques in model extraction attacks on image classifiers, focusing on enhancing the performance of surrogate models through the use of Deep Q-network (DQN) extensions and synthetic data generation techniques. The findings of this study offer significant insights into the vulnerabilities of machine learning models and the effectiveness of different strategies for conducting model extraction attacks.

Our comprehensive evaluation demonstrated that DQN extensions, particularly Prioritized Experience Replay (PER), consistently outperformed other methods in optimizing sample selection for model extraction. The use of PER led to higher accuracy and efficiency in surrogate model training by prioritizing more informative experiences, thus enhancing the overall learning process.

Among synthetic data generation techniques, the Linf-projected Gradient Descent (LinfPGD) method emerged as the most effective. Its iterative approach to creating adversarial examples within a controlled perturbation budget proved highly beneficial for training robust surrogate models. The Jacobian-based methods and Fast Gradient Sign Method (FGSM) also showed promise, although they were slightly less effective than LinfPGD.

The data-free model extraction approach, while innovative, demonstrated significantly lower performance under the constraints of limited query budgets. This highlights the critical importance of access to meaningful training data, whether real or synthetically generated, for successful model extraction.

The insights gained from this research highlight the critical need to continuously improve the security of machine learning models against adversarial threats. By thoroughly evaluating different learning techniques for model extraction attacks, this study enhances our understanding of machine learning security and guides the development of more resilient and robust models. As machine learning technology progresses, it's essential to stay vigilant and proactive in tackling the new challenges and threats posed by adversarial attacks.

Chapter 6

Conclusion

This dissertation has explored the transformative potential of Deep Reinforcement Learning (DRL) across three pivotal domains: robotic manipulation, enhanced semantic segmentation, and the security of image classifiers. By systematically addressing the challenges inherent in these areas, this research has contributed significant advancements and practical insights, reinforcing the versatile and powerful capabilities of DRL.

The exploration into robotic manipulation provided a comprehensive analysis of various DRL algorithms, including value-based, policy-based, and actor-critic methods. This detailed examination illuminated the specific strengths and limitations of each method, offering a granular understanding that aids in selecting appropriate algorithms for different robotic applications. The research also highlighted the importance of integrating multiple learning paradigms to enhance robotic adaptability and performance across diverse tasks. The proposed new directions for future research in robotic manipulation emphasize the necessity for such integration to tackle the complex nature of real-world environments where robots operate.

In the realm of enhanced semantic segmentation, this dissertation developed a robust framework utilizing reinforced active learning methodologies. The framework optimizes annotation processes and addresses imbalanced datasets, integrating advanced techniques such as Dueling Deep Q-Networks (DQN), Prioritized Experience Replay, Noisy Networks, and Emphasizing Recent Experience. The comparative experiments demonstrated the robustness and efficiency of the proposed approach across various domains, particularly excelling in scenarios with constrained annotation budgets. These findings contribute significantly to the practical deployment of enhanced semantic segmentation methods, providing a foundation for more precise and efficient image segmentation in diverse applications.

Securing image classifiers presented another critical challenge addressed in this research. The development of surrogate models capable of replicating proprietary image classification models under stringent constraints was a primary focus. The introduction of an open-source framework that integrates popular DQN extensions demonstrated their effectiveness in enhancing attack methodologies against neural networks. The evaluation of various synthetic data generation techniques identified best practices and guidelines for practitioners, refining the approach to generating synthetic data crucial for training robust adversarial models. This research not only advances the understanding of effective attack strategies in AI security but also underscores the importance of ethical considerations in conducting model extraction attacks.

Throughout this research, several overarching themes emerged. The importance of improving sample efficiency and stability in DRL algorithms was a recurring focus. Techniques such as Prioritized Experience Replay and soft updates for target networks proved essential in achieving these improvements. Additionally, the need for DRL algorithms to generalize effectively across diverse environments and tasks was emphasized. Rigorous experimental evaluations and the development of robust frameworks have enhanced the applicability of DRL in real-world scenarios, addressing the challenges of generalization and robustness.

Ethical and practical considerations also played a significant role in this research. The methodologies developed aimed to minimize interaction with victim models, mitigating ethical and legal risks associated with model extraction attacks. The practical implications of this research extend to various fields, including autonomous vehicles, robotics, and AI security, providing actionable insights that guide the deployment of DRL technologies in these areas.

Looking forward, this dissertation paves the way for several future research directions. One significant avenue is the integration of multi-paradigm learning, exploring the combination of supervised, unsupervised, and reinforcement learning paradigms to further enhance the adaptability and performance of DRL algorithms. Expanding evaluation frameworks to include a broader range of datasets and model architectures will validate the generalizability of the findings and refine the developed methodologies.

Another critical area for future research is the development of robust defense mechanisms to counteract the growing sophistication of model extraction attacks. As these attacks become more advanced, creating and evaluating effective defense strategies will be crucial in ensuring the security and integrity of machine learning models. Additionally, leveraging advanced data generation techniques, such as Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs), can further improve the quality and effectiveness of synthetic data in training adversarial models.

In conclusion, the insights gained from this research underscore the transformative potential of Deep Reinforcement Learning in driving innovations across diverse fields. By addressing key challenges and advancing the understanding of DRL methodologies, this dissertation contributes to the broader goal of developing intelligent systems capable of tackling complex decision-making tasks. As the field of artificial intelligence continues to evolve, the findings of this research will play a crucial role in shaping the future of DRL and its applications, ensuring that it remains an indispensable tool in the pursuit of technological advancements and solutions to real-world problems.

References

[1] J. Hua, L. Zeng, G. Li, and Z. Ju, Sensors **21**, 1278 (2021).

[2] J. Kerbel, Journal of Robotics Research **10**, 123 (2024).

[3] P. Lee, T.-B. Chen, H.-Y. Lin, L.-R. Yeh, C.-H. Liu, and Y.-L. Chen, Bioengineering **11**, 548 (2024).

[4] H. Yadavari, V. T. Aghaei, and S. I. GLU, Journal of Robotics and Control (JRC) **5**, 117 (2024).

[5] L. P. Kaelbling, M. L. Littman, and A. W. Moore, Journal of artificial intelligence research **4**, 237 (1996).

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, CoRR **abs/1312.5602**, 1 (2013).

[7] C. Beattie, J. Z. Leibo, D. Teplyashin, T. Ward, M. Wainwright, H. Küttler, A. Lefrancq, S. Green, V. Valdés, A. Sadik, J. Schrittwieser, K. Anderson, S. York, M. Cant, A. Cain, A. Bolton, S. Gaffney, H. King, D. Hassabis, S. Legg, and S. Petersen, CoRR **abs/1612.03801**, 1 (2016).

[8] C. J. Watkins and P. Dayan, Machine learning **8**, 279 (1992).

[9] G. A. Rummery and M. Niranjan, *On-line Q-learning using connectionist systems* (University of Cambridge, Department of Engineering, Cambridge, UK, 1994), Vol. 37.

[10] H. Van Hasselt, A. Guez, and D. Silver, in *Proceedings of the AAAI Conference on Artificial Intelligence* (AAAI Press, Palo Alto, California, USA, 2016), No. 1.

[11] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, in *International Conference on Machine Learning*, PMLR (PMLR, New York, NY, USA, 2016), pp. 1995–2003.

[12] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, in *Advances in Neural Information Processing Systems* (MIT Press, Denver, Colorado, USA, 2000), pp. 1057–1063.

[13] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, in *International Conference on Machine Learning*, PMLR (PMLR, Lille, France, 2015), pp. 1889–1897.

[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, CoRR **abs/1707.06347**, 1 (2017).

[15] V. R. Konda and J. N. Tsitsiklis, in *Advances in Neural Information Processing Systems* (MIT Press, Denver, Colorado, USA, 2000), pp. 1008–1014.

[16] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, in *International Conference on Machine Learning*, PMLR (PMLR, New York, NY, USA, 2016), pp. 1928–1937.

[17] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, arXiv preprint arXiv:1509.02971 **abs/1509.02971**, 1 (2015).

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, nature **518**, 529 (2015).

[19] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, in *International Conference on Machine Learning* (PMLR, Beijing, China, 2014), pp. 387–395.

[20] G. E. Uhlenbeck and L. S. Ornstein, Physical review **36**, 823 (1930).

[21] S. Fujimoto, H. Hoof, and D. Meger, in *International Conference on Machine Learning*, PMLR (PMLR, Stockholm, Sweden, 2018), pp. 1587–1596.

[22] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, *et al.*, arXiv preprint arXiv:1812.05905 **abs/1812.05905**, 1 (2018).

[23] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, in *International Conference on Machine Learning*, PMLR (PMLR, Sydney, Australia, 2017), pp. 1352–1361.

[24] D. Han, B. Mulyana, V. Stankovic, and S. Cheng, Sensors **23**, 3762 (2023).

[25] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, Business & information systems engineering **6**, 239 (2014).

[26] A. Sigov, L. Ratkin, L. A. Ivanov, and L. D. Xu, Information Systems Frontiers **24**, 1 (2022).

[27] M. T. Mason, Annual Review of Control, Robotics, and Autonomous Systems **1**, 1 (2018).

[28] A. Hafiz and M. A. H. Hassaballah, Computer Systems Science and Engineering **45**, 123 (2023).

[29] A. M. Hafiz, M. Hassaballah, and A. Binbusayyis, Applied Sciences **13**, 723 (2023).

[30] E. F. Morales, R. Murrieta-Cid, I. Becerra, and M. A. Esquivel-Basaldua, Intelligent Service Robotics **14**, 773 (2021).

[31] M. Rubagotti, B. Sangiovanni, A. Nurbayeva, G. P. Incremona, A. Ferrara, and A. Shintemirov, IEEE Control Systems Magazine **43**, 44 (2023).

[32] Í. Elguea-Aguinaco, A. Serrano-Muñoz, D. Chrysostomou, I. Inziarte-Hidalgo, S. Bøgh, and N. Arana-Arexolaleiba, Robotics and Computer-Integrated Manufacturing **81**, 102517 (2023).

[33] F.-Y. Wang, J. J. Zhang, X. Zheng, X. Wang, Y. Yuan, X. Dai, J. Zhang, and L. Yang, IEEE/CAA Journal of Automatica Sinica **3**, 113 (2016).

[34] S. Rammohan, S. Yu, B. He, E. Hsiung, E. Rosen, S. Tellex, and G. Konidaris, arXiv preprint arXiv:2107.13356 **abs/2107.13356**, 1 (2021).

[35] D. Wang and R. Walters, in *International Conference on Learning Representations* (ICLR, Virtual Conference, 2022).

[36] H. Chen, in *Proceedings of the AAAI Conference on Artificial Intelligence* (AAAI Press, Palo Alto, California, USA, 2021), No. 18, pp. 15769–15770.

[37] A. Clegg, W. Yu, J. Tan, C. C. Kemp, G. Turk, and C. K. Liu, arXiv preprint arXiv:1709.07033 **abs/1709.07033**, 1 (2017).

[38] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, arXiv preprint arXiv:1707.08817 **abs/1707.08817**, 1 (2017).

[39] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, arXiv preprint arXiv:1511.05952 **abs/1511.05952**, 1 (2015).

[40] O. Kilinc, Y. Hu, and G. Montana, arXiv preprint arXiv:1910.07294 **abs/1910.07294**, 1 (2019).

[41] X. Yang, Z. Ji, J. Wu, and Y.-K. Lai, arXiv preprint arXiv:2105.05985 **abs/2105.05985**, 1 (2021).

[42] N. Vulin, S. Christen, S. Stevšić, and O. Hilliges, IEEE Robotics and Automation Letters **6**, 2194 (2021).

[43] D. A. Pomerleau, Advances in neural information processing systems **1**, 305 (1988).

[44] P. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, arXiv preprint arXiv:1706.03741 **abs/1706.03741**, 1 (2017).

[45] A. Y. Ng, S. J. Russell, *et al.*, in *Proceedings of the International Conference on Machine Learning* (PMLR, Stanford, California, USA, 2000), Vol. 1, pp. 663–670.

[46] J. Ho and S. Ermon, Advances in neural information processing systems **29**, 4565 (2016).

[47] Y. Ding, C. Florensa, M. Phielipp, and P. Abbeel, arXiv preprint arXiv:1906.05838 **abs/1906.05838**, 1 (2019).

[48] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, arXiv preprint arXiv:1707.01495 **abs/1707.01495**, 1 (2017).

[49] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, in *Proceedings of the 26th Annual International Conference on Machine Learning* (PMLR, Montreal, Quebec, Canada, 2009), pp. 41–48.

[50] T. Matiisen, A. Oliver, T. Cohen, and J. Schulman, IEEE transactions on neural networks and learning systems **31**, 3732 (2019).

[51] S. Sukhbaatar, Z. Lin, I. Kostrikov, G. Synnaeve, A. Szlam, and R. Fergus, arXiv preprint arXiv:1703.05407 **abs/1703.05407**, 1 (2017).

[52] R. S. Sutton, D. Precup, and S. Singh, Artificial intelligence **112**, 181 (1999).

[53] J. Janisch, T. Pevnỳ, and V. Lisỳ, arXiv preprint arXiv:2009.12462 **abs/2009.12462**, 1 (2020).

[54] P. Almasan, J. Suárez-Varela, K. Rusek, P. Barlet-Ros, and A. Cabellos-Aparicio, Computer Communications **196**, 184 (2022).

[55] Y. Li, T. Kong, L. Li, Y. Li, and Y. Wu, arXiv preprint arXiv:2108.02439 **abs/2108.02439**, 1 (2021).

[56] Y. Lin, A. S. Wang, E. Undersander, and A. Rai, arXiv preprint arXiv:2102.13177 **abs/2102.13177**, 1 (2021).

[57] R. Li, A. Jabri, T. Darrell, and P. Agrawal, in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE (IEEE, Paris, France, 2020), pp. 4051–4058.

[58] M. Sieb, Z. Xian, A. Huang, O. Kroemer, and K. Fragkiadaki, in *Conference on Robot Learning*, PMLR (PMLR, Virtual Conference, 2020), pp. 979–989.

[59] F. Xie, A. Chowdhury, M. De Paolis Kaluza, L. Zhao, L. Wong, and R. Yu, Advances in neural information processing systems **33**, 2327 (2020).

[60] J. Liang and A. Boularias, in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE (IEEE, New York, NY, USA, 2023), pp. 1807–1813.

[61] M. Oliva, S. Banik, J. Josifovski, and A. Knoll, in *2022 International Joint Conference on Neural Networks (IJCNN)*, IEEE (IEEE, Padua, Italy, 2022), pp. 1–9.

[62] Y. Huang, Z. Huang, and T. Jin, Applied Sciences **14**, 4724 (2024).

[63] V. Ofitserov and A. Konushin, International Journal of Open Information Technologies **12**, 57 (2024).

[64] S. V. Armstrong, S. Pallickara, S. Pallickara, S. Ghosh, J. F. Breidt, *et al.*, Journal of Machine Learning Research **12**, 123 (2024).

[65] Y. Bi, Z. Chen, C. Liu, T. Liang, and F. Zheng, Machine Vision and Applications **35**, 74 (2024).

[66] P. Varangaonkar and S. Rode, Multimedia Tools and Applications **80**, 1 (2024).

[67] H. Hajiabadi, C. Gerking, L. Hilbert, and A. Koziolek, Journal of Systems and Software **211**, 111986 (2024).

[68] C. Yu, J. Zhu, and X. Li, arXiv preprint arXiv:2402.10074 **abs/2402.10074**, 1 (2024).

[69] Y. Zhang, H. Tong, Y. Xia, Y. Zhu, Y. Chi, and L. Ying, in *Proceedings of the AAAI Conference on Artificial Intelligence* (AAAI Press, Palo Alto, California, USA, 2022), No. 8, pp. 9118–9126.

[70] D. Angluin, Machine learning **2**, 319 (1988).

[71] R. D. King, K. E. Whelan, F. M. Jones, P. G. Reiser, C. H. Bryant, S. H. Muggleton, D. B. Kell, and S. G. Oliver, Nature **427**, 247 (2004).

[72] I. Dagan and S. P. Engelson, *Machine Learning Proceedings 1995* (Elsevier, Amsterdam, The Netherlands, 1995), pp. 150–157.

[73] V. Krishnamurthy, IEEE Transactions on Signal Processing **50**, 1382 (2002).

[74] D. D. Lewis, in *ACM SIGIR Forum*, ACM (ACM, New York, NY, USA, 1995), No. 2, pp. 13–19.

[75] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, B. B. Gupta, X. Chen, and X. Wang, ACM computing surveys (CSUR) **54**, 1 (2021).

[76] C. Shui, F. Zhou, C. Gagné, and B. Wang, in *International Conference on Artificial Intelligence and Statistics*, PMLR (PMLR, Online, 2020), pp. 1308–1318.

[77] B. Settles, *Active Learning*, Vol. 6 of *Synthesis Lectures on Artificial Intelligence and Machine Learning* (Morgan & Claypool, San Rafael, CA, USA, 2012), pp. 1–114.

[78] B. Settles, in *Active learning and experimental design workshop in conjunction with AISTATS 2010*, JMLR Workshop and Conference Proceedings (JMLR: Workshop and Conference Proceedings, Cambridge, MA, USA, 2011), pp. 1–18.

[79] J. M. Hernández-Lobato and R. Adams, in *International Conference on Machine Learning*, PMLR (PMLR, Lille, France, 2015), pp. 1861–1869.

[80] G. Dhiman, A. V. Kumar, R. Nirmalan, S. Sujitha, K. Srihari, N. Yuvaraj, P. Arulprakash, and R. A. Raja, Multimedia Tools and Applications **82**, 5343 (2023).

[81] J. Gong, Z. Fan, Q. Ke, H. Rahmani, and J. Liu, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (IEEE, New York, NY, USA, 2022), pp. 11079–11089.

[82] D. Sadigh, A. D. Dragan, S. Sastry, and S. A. Seshia, *Active preference-based learning of reward functions* (Springer, New York, NY, USA, 2017).

[83] G. Kunapuli, P. Odom, J. W. Shavlik, and S. Natarajan, in *2013 IEEE 13th International Conference on Data Mining*, IEEE (IEEE, New York, NY, USA, 2013), pp. 409–418.

[84] A. Ezzeddine, N. Mourad, B. N. Araabi, and M. N. Ahmadabadi, Expert Systems with Applications **112**, 331 (2018).

[85] S. Mittal, J. Niemeijer, J. P. Schäfer, and T. Brox, arXiv preprint arXiv:2302.04075 **1**, 1 (2023).

[86] M. Kampffmeyer, A.-B. Salberg, and R. Jenssen, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (IEEE, New York, NY, USA, 2016), pp. 1–9.

[87] S. D. Jain and K. Grauman, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, New York, NY, USA, 2016), pp. 2864–2873.

[88] A. Vezhnevets, J. M. Buhmann, and V. Ferrari, in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE (IEEE, New York, NY, USA, 2012), pp. 3162–3169.

[89] K. Konyushkova, R. Sznitman, and P. Fua, in *Proceedings of the IEEE International Conference on Computer Vision* (IEEE, New York, NY, USA, 2015), pp. 2974–2982.

[90] J. Aklilu and S. Yeung, in *Machine Learning for Healthcare Conference*, PMLR (PMLR, Cambridge, MA, USA, 2022), pp. 892–911.

[91] X. Shu, Y. Yang, R. Xie, J. Liu, X. Chang, and B. Wu, SSRN Electronic Journal **2022**, 1 (2022).

[92] S. A. Golestaneh and K. M. Kitani, arXiv preprint arXiv:2008.01860 **abs/2008.01860**, 1 (2020).

[93] H. Hasselt, Advances in neural information processing systems **23**, 2613 (2010).

[94] M. Babaeizadeh, I. Frosio, S. Tyree, J. Clemons, and J. Kautz, arXiv preprint arXiv:1611.06256 **abs/1611.06256**, 1 (2016).

[95] M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, C. Blundell, and S. Legg, Noisy Networks for Exploration, 2019.

[96] A. Casanova, P. O. Pinheiro, N. Rostamzadeh, and C. J. Pal, arXiv preprint arXiv:2002.06583 **abs/2002.06583**, 1 (2020).

[97] C. Wang and K. Ross, arXiv preprint arXiv:1906.04009 **abs/1906.04009**, 1 (2019).

[98] M. Tokic and G. Palm, in *Annual Conference on Artificial Intelligence* (Springer, Berlin, Heidelberg, 2011), pp. 335–346.

[99] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla, in *Computer Vision–ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part I*, Springer (Springer, Berlin, Heidelberg, 2008), pp. 44–57.

[100] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, in *European Conference on Computer Vision (ECCV)*, Vol. 9906 of *LNCS*, edited by B. Leibe, J. Matas, N. Sebe, and M. Welling (Springer International Publishing, Cham, Switzerland, 2016), pp. 102–118.

[101] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, New York, NY, USA, 2018), pp. 9185–9193.

[102] M. Soltani, A. Bonakdar, N. Shakourifar, R. Babaei, and K. Raahemifar, Frontiers in Oncology **11**, 661123 (2021).

[103] Y. Dong, S. Cheng, T. Pang, H. Su, and J. Zhu, IEEE Transactions on Pattern Analysis and Machine Intelligence **44**, 9536 (2021).

[104] Z. Wei, J. Chen, H. Zhang, L. Jiang, and Y.-G. Jiang, in *Proceedings of the 2022 International Conference on Multimedia Retrieval* (ACM, New York, NY, USA, 2022), pp. 587–593.

[105] J. Yang, Y. Jiang, X. Huang, B. Ni, and C. Zhao, Advances in Neural Information Processing Systems **33**, 12288 (2020).

[106] A. Ilie, M. Popescu, and A. Stefanescu, in *International Conference on Neural Information Processing*, Springer (Springer, Cham, Switzerland, 2021), pp. 188–200.

[107] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, in *25th USENIX Security Symposium (USENIX Security 16)*, USENIX Association (USENIX Association, Berkeley, CA, USA, 2016), pp. 601–618.

[108] X. Zhang, C. Fang, and J. Shi, CoRR **abs/2104.05921**, 1 (2021).

[109] H. Xu, Y. Ma, H.-C. Liu, D. Deb, H. Liu, J.-L. Tang, and A. K. Jain, International Journal of Automation and Computing **17**, 151 (2020).

[110] N. Pitropakis, E. Panaousis, T. Giannetsos, E. Anastasiadis, and G. Loukas, Computer Science Review **34**, 100199 (2019).

[111] Y. Fang, Y. Zeng, B. Li, L. Liu, and L. Zhang, Plos one **15**, e0231626 (2020).

[112] S. Zhang, X. Xie, and Y. Xu, IEEE Access **8**, 128250 (2020).

[113] K. Chen, S. Guo, T. Zhang, X. Xie, and Y. Liu, in *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security* (ACM, New York, NY, USA, 2021), pp. 307–319.

[114] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security* (ACM, New York, NY, USA, 2017), pp. 506–519.

[115] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, arXiv preprint arXiv:1706.06083 **abs/1706.06083**, 1 (2017).

[116] I. J. Goodfellow, J. Shlens, and C. Szegedy, arXiv preprint arXiv:1412.6572 **abs/1412.6572**, 1 (2014).

[117] O. Sener and S. Savarese, arXiv preprint arXiv:1708.00489 **abs/1708.00489**, 1 (2017).

[118] W. Gao, X. Li, Y. Wang, and Y. Cai, Frontiers in Public Health **10**, 879639 (2022).

[119] J.-B. Truong, P. Maini, R. J. Walls, and N. Papernot, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (IEEE, New York, NY, USA, 2021), pp. 4771–4780.

[120] L. Deng, IEEE Signal Processing Magazine **29**, 141 (2012).

[121] S.-Y. Zhou and C.-Y. Su, arXiv preprint arXiv:2105.09008 **abs/2105.09008**, 1 (2021).

[122] A. Krizhevsky, G. Hinton, *et al.*, Technical Report **1**, 1 (2009).

[123] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur, arXiv preprint arXiv:2010.01412 **abs/2010.01412**, 1 (2020).