NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

FORMING ADVERSARIAL EXAMPLE ATTACKS AGAINST DEEP NEURAL NETWORKS WITH REINFORCEMENT LEARNING

by

Matthew D. Akers

March 2023

Thesis Advisor: Armon C. Barton
Second Reader: Marko Orescanin

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 2023	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE FORMING ADVERSARIAL EXAMPLE ATTACKS AGAINST DEEP NEURAL NETWORKS WITH REINFORCEMENT LEARNING	**5. FUNDING NUMBERS**
6. AUTHOR(S) Matthew D. Akers	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Deep neural networks (DNN) are producing groundbreaking results in virtually all academic and commercial domains and will serve as the workhorse of future human-machine teams that will modernize the Department of Defense (DOD). As such, leaders will need to trust and rely on these networks, which makes their security a paramount concern. Considerable research has demonstrated that DNNs remain vulnerable to adversarial examples. While many defense schemes have been proposed to counter the equally many attack vectors, none have been successful at securing a DNN from this vulnerability. Novel attacks expose blind spots unique to a network's defense, indicating the need for a robust and adaptable attack, used to expose these vulnerabilities early in the development phase. We propose a novel reinforcement learning–based attack, Adversarial Reinforcement Learning Agent (ARLA), designed to learn the vulnerabilities of a DNN and generate adversarial examples to exploit them. ARLA was able to significantly degrade the accuracy of five CIFAR-10 DNNs, four of which used a state-of-the-art defense. We compared our method to other state-of-the-art attacks and found evidence that ARLA is an adaptive attack, making it a useful tool for testing the reliability of DNNs before they are deployed within the DOD.

14. SUBJECT TERMS reinforcement learning, machine learning, deep neural networks, adversarial examples, adversarial attacks, DNN	**15. NUMBER OF PAGES** 81
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

FORMING ADVERSARIAL EXAMPLE ATTACKS AGAINST DEEP NEURAL NETWORKS WITH REINFORCEMENT LEARNING

Matthew D. Akers
Lieutenant Commander, United States Navy
BS, Oklahoma State University, 2007

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 2023

Approved by: Armon C. Barton
Advisor

Marko Orescanin
Second Reader

Gurminder Singh
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Deep neural networks (DNN) are producing groundbreaking results in virtually all academic and commercial domains and will serve as the workhorse of future human-machine teams that will modernize the Department of Defense (DOD). As such, leaders will need to trust and rely on these networks, which makes their security a paramount concern. Considerable research has demonstrated that DNNs remain vulnerable to adversarial examples. While many defense schemes have been proposed to counter the equally many attack vectors, none have been successful at securing a DNN from this vulnerability. Novel attacks expose blind spots unique to a network's defense, indicating the need for a robust and adaptable attack, used to expose these vulnerabilities early in the development phase. We propose a novel reinforcement learning–based attack, Adversarial Reinforcement Learning Agent (ARLA), designed to learn the vulnerabilities of a DNN and generate adversarial examples to exploit them. ARLA was able to significantly degrade the accuracy of five CIFAR-10 DNNs, four of which used a state-of-the-art defense. We compared our method to other state-of-the-art attacks and found evidence that ARLA is an adaptive attack, making it a useful tool for testing the reliability of DNNs before they are deployed within the DOD.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AE	adversarial example
ARLA	Adversarial Reinforcement Learning Agent
ART	Adversarial Robustness Toolbox
AI	artificial intelligence
CNN	convolutional neural network
DNN	deep neural network
DQL	deep Q-learning
DQN	deep Q-network
DOD	Department of Defense
D3QN	double dueling deep Q-network (DQN)
DDQN	dueling deep Q-network (DQN)
FGSM	Fast Gradient Sign Method
ML	machine learning
MDP	Markov decision processes
PGD	Projected Gradient Descent
PGDAT	Projected Gradient Descent (PGD) Adversarial Trained
RGB	red, green, and blue
RL	reinforcement learning
TRADES	TRadeoff-inspired Adversarial DEfense via Surrogate-loss minimization

USN U.S. Navy

WRN Wide Residual Network

Acknowledgments

To my wife, Makenzie, thank you for all the love, strength and support that you have given me through this program and all our years together. I could not imagine doing any of this without you in my corner. To my incredible children, Evelyn and James, thank you for all the sacrifices that you've made and being patient through all of the "Daddy has to work" moments. I love you both tremendously.

To my advisor, Professor Armon Barton, thank you for giving me this incredible opportunity and driving me to do better. Thank you for your direction and giving me renewed focus when I needed it. Now, onto the next project!

To CAPT Clay Herring and LCDR Paul Edelman, thank you for all your trust and support in getting me here and into the CS program. Y'alls leadership has had a profound and irreversible impact on me, and I plan to carry it forward. LLTB.

When I was a junior enlisted Sailor, I discovered NPS and resolved to make it a part of the shadow box I was building in my mind. This thesis is not the culmination of the last two years, but of many little steps that can be traced back to the beginning of my military career. So my final thanks goes to the United States Navy and every leader that has helped me realize this longstanding goal. Non sibi sed patriae.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

1.1 Deep Learning and the DOD

If the U.S. Navy (USN) and Department of Defense (DOD) are serious about building an enduring technological advantage over our adversaries [1], they must be willing to integrate cutting-edge machine learning (ML) technologies into current systems and processes. ML, in which systems extract meaning and knowledge from raw data [2], has propelled the broader field of artificial intelligence (AI) into seemingly endless applications. One would be hard pressed to find a field, whether academic, commercial, or medical, that ML has not revolutionized. ML has been used to help identify automobile insurance fraud [3], to provide early detection of cervical cancer [4], and to detect and characterize the formation of ice on aircraft [5]. It is not the role of the ML model to make a decision in these scenarios, only to provide the human operator with better information. By applying ML in a similar manner, the DOD has a road map for evolving systems and processes into human-machine teams that adhere to the principles of ethical AI [6].

While ML can encompass a broad range of models used for making predictions, a subset known as deep learning is the driving force behind this AI summer. Unlike more simplistic ML techniques such as linear regression modeling and support vector machines, deep learning encompasses ML models that take advantage of deep neural networks (DNNs) which use many hidden layers of artificial neurons to learn complex concepts through data [2]. Though DNNs are utilized for many purposes, this thesis focuses on those specializing in image recognition.

1.2 Trusting the Machine

For the DOD to successfully transition to human-machine teams, military and civilian leaders must be able to trust and rely on the underlying technology. This is not a small ask for senior leaders. Unlike human analysts, whose thought process can be understood through dialog, there is no clear path to understanding how a DNN makes a decision based solely on

data. Trust, then, must be built upon a reasonable belief that the system is resistant to attack and that its results are consistent and reliable. Any concern regarding trustworthiness and reliability is more than justified, because a litany of research has demonstrated that DNNs are persistently vulnerable to adversarial examples.

An adversarial example (AE) is a benign input sample that has been malformed through the addition of perturbations resulting in the target DNN returning incorrect output. The purpose of an AE is to appear non-malicious while degrading the overall accuracy of the target network, which can have severe and life-threatening consequences. Consider, for example, autonomous driving and how crucial it is that a car does not confuse stop and yield signs. For military commanders, if a network is not robust against adversarial examples, trust in that system can easily be degraded and the system is disregarded for more traditional and time-consuming analysis. Imagine a system where the DNN correctly filters out 90% of images, leaving only 10% tagged for human review. Should that system be successfully attacked, then the human-machine team fails, and the analyst is quickly overwhelmed by the new workload.

1.3 Research Questions

Adversarial attack algorithms are, at their core, functions, where, given a benign input of X, an adversarial $\hat{X}$ is generated. Many attacks may require the sample's true label (y), or the target network or some approximation of it, but they are still just functions. A certain attack will therefore always output the same AE given a certain set of input variables. Deep learning is not a part of the attack itself, which means that there is no ML involved when creating adversarial examples. This algorithmic approach to generating AEs led us to consider the field of reinforcement learning (RL), where a DNN "agent" learns to behave optimally in a specific environment while pursuing a specific goal [7]. There have been a flood of successes coming from the RL research group DeepMind demonstrating that RL is capable of achieving super-human performance playing a variety of games [8]–[11]. In simplest terms, an RL agent learns through a pattern of observing the environment, playing an action for which it receives some reward, and then observing the subsequent state. The agent, attempting to maximize the total rewards it receives, eventually learns the best policy for behavior.

Considering RL and the threat that adversarial examples pose to DNNs led us to our first research question:

1) If an image is the environment, and pixel changes are playable actions, can a reinforcement learning agent learn to generate minimally perturbed adversarial examples?

For all the academic literature researching adversarial attacks, there is an equal amount covering adversarial defenses: A novel attack is proposed, followed sometime thereafter by a defense which counters it, and the cycle repeats itself. While a state-of-the-art defense may defend against all current attacks, there is no guarantee that a defense will be able to defend against an unknown attack. If an attack could be adaptable to any defense, it would assist researchers and developers in staying ahead of unknown attacks. Considering attack adaptability led us to our second research question:

2) Can a reinforcement learning-based adversarial attack be an adaptive attack?

By addressing these two questions, we blend the two fields of adversarial research and reinforcement learning for the first time.

1.4 Adversarial Reinforcement Learning Agent (ARLA)

This research introduces the first RL–based adversarial attack. Named Adversarial Reinforcement Learning Agent (ARLA), our attack uses a benign sample image as a learning environment to generate adversarial examples with the goal of finding the adversary with the shortest ℓ_2 distance from the original sample. ARLA uses double deep Q-learning (DQL), explained in Chapter 2, with an a improved deep Q-network (DQN) agent architecture, explained in detail in Chapters 2 and 3. Our results provide evidence that ARLA is an adaptive adversarial attack by showing significant attack success against all five models used for attack evaluation in this thesis. While our results are promising, more work will need to be done to stabilize how ARLA learns the optimal behavior policy.

The intent of our research was to give the DOD an effective tool for evaluating DNNs being developed by the armed services. Unlike other adaptive attacks which need to be tuned to a specific defense by a technical expert, an RL-based adversarial attack might be utilized with greater ease and minimal training. It is our hope that ARLA is such an attack, and becomes a

small, but valuable step in building institutional trust in the human-machine teams deployed as a part of future military systems.

CHAPTER 2:
Background and Related Work

In this chapter, we present background material and related research and make connections to our present work. Section 2.1 discusses convolutional neural networks and their utility with image recognition. Section 2.2 discusses adversarial examples and the threat they pose to deep neural networks. Section 2.3 discusses reinforcement learning, specifically deep Q-learning.

2.1 Convolutional Neural Networks

A DNN is any neural network that has two or more hidden layers within its architecture [12]. Figure 2.1 depicts one of the most recognizable examples of a DNN, the fully connected DNN, in which every neuron in one layer is connected to each neuron in the next. Though fully connected architectures have proven useful when applied to complex datasets [13], they come up short when applied to image recognition. This shortcoming is due to the structure of the input layer, where each node is a *feature* of the data. When the input is an image, each pixel becomes a feature, and the input layer quickly becomes unwieldy. For example, MNIST digits [14] are displayed as 28x28 grayscale images, but those pixels equate to an input layer of 728 features. More importantly, individual pixels are meaningless when removed from the larger picture. Pixels only reveal information when aggregated, and standard DNNs are not designed for such a task.

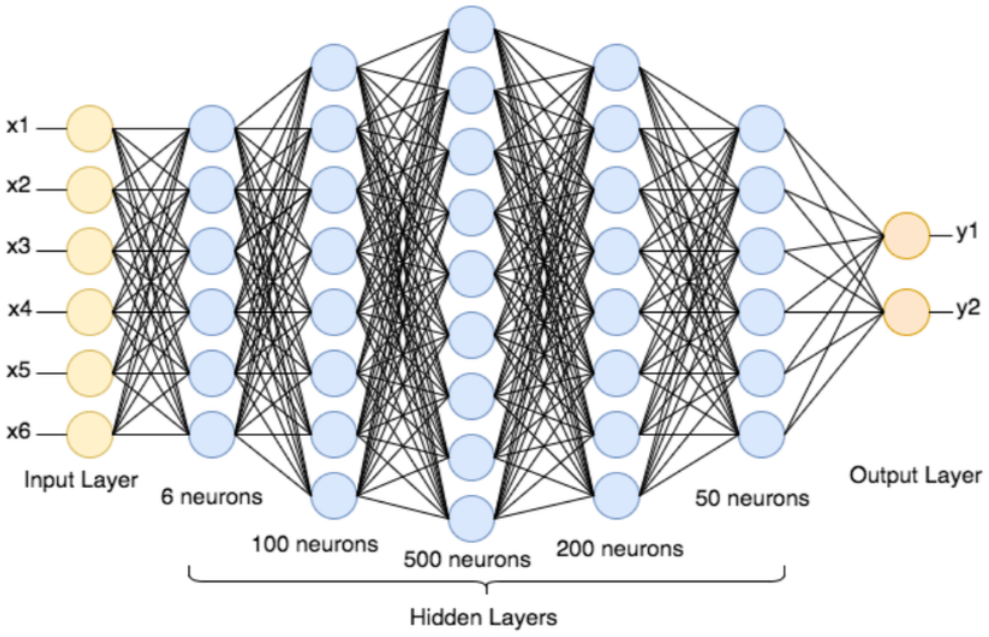

Figure 2.1. Example of a basic fully connected DNN, where each neuron is connected to all the neurons in the next layer. Source: [15].

Stemming from the introduction of the neocognitron in 1992, convolutional neural networks (CNNs) addressed the problem of image recognition in deep learning [16]. Based on the visual cortex of the human brain, a CNN extracts the most important features of an image before classifying it. Feature extraction is done primarily with the help of two layers unique to CNNs: *convolution* and *pooling* layers. Convolutional layers create a convolved feature, or *feature map*, which is the product of a *kernel filter* sliding over the input. This process is best understood by envisioning an image as a large grid, where each cell represents a pixel value and the kernel filter as a smaller grid where each cell represents a weight. As the filter traverses the image, it performs a computation on the pixels within its *receptive field* to produce a single value within the feature map [12].

Figure 2.2 depicts a simple example of this process, with three important points to note: First, the filter is using a *stride length* of one, meaning that the filter only shifts by one pixel with each iteration. Second, this example has no *padding*, meaning that the output is smaller than the input. In certain cases, padding the output leads to better performance of the model. Third, this example is only for a single-channel image, meaning it is color-mapped

in grayscale. A three-channel red, green, and blue (RGB) image would need a matching three-channel filter, one for each color, but would still result in the same output.

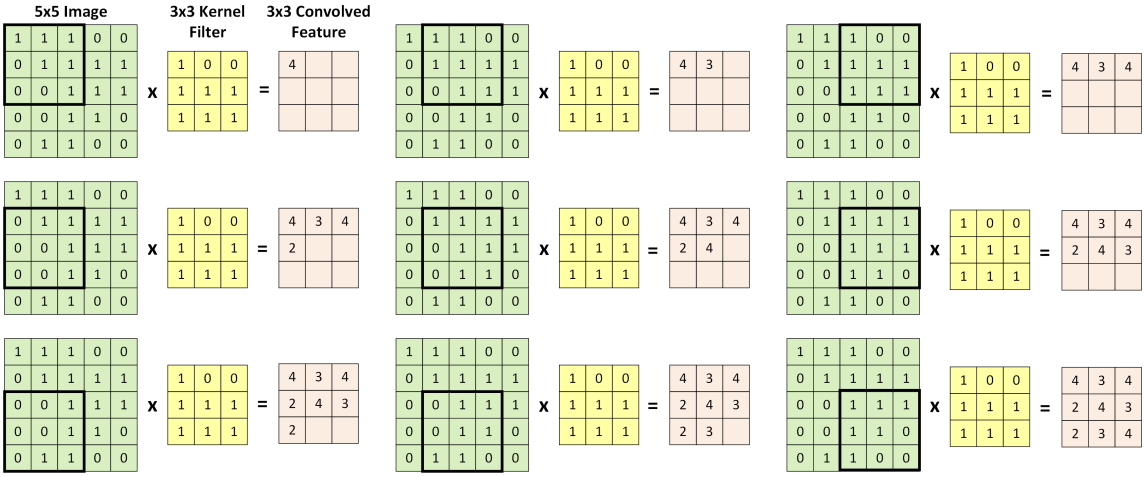

Figure 2.2. Process of creating a feature map of a 5x5 image using a 3x3 kernel filter, with a stride of 1 and no padding. Adapted from [17].

Following the application of one or more convolutional layers, a pooling layer reduces the dimensions of the image through *subsampling*. Reducing the dimensions of the output, typically in half, reduces the computational load, memory usage, and model parameters [12]. Pooling can be accomplished by taking either the max value or average of the pooling kernel. In Figure 2.3, we see a 2x2 pooling kernel reducing the input by half using either max or average operations.

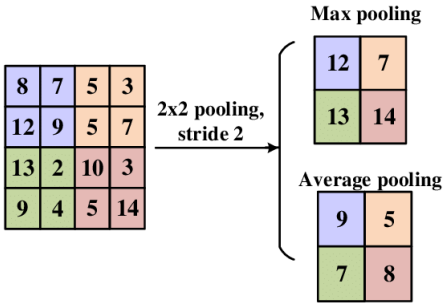

Figure 2.3. Pooling with a 2x2 kernel using either max or average operations. Source: [18].

Using any number of convolution and pooling layers, depending on the architecture of the model, the output is then transformed for the *classification* head of the model. In Figure 2.4, this transformation is taking place when the output from the last pooling layer is flattened and then fed into a fully connected network. The final layer has an output shape that matches the number of predictable classes.

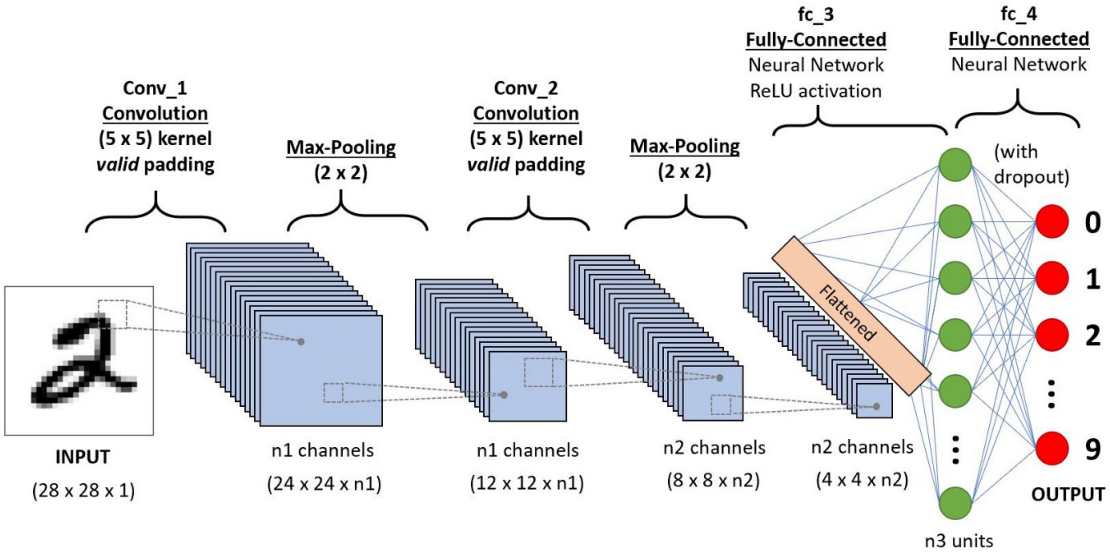

Figure 2.4. Convolutional neural networks extract features from an image, which are then fed into a deep neural network that returns an output of class probabilities. Source: [17].

2.2 Adversarial Examples

As powerful as CNNs are, they have proven quite vulnerable to malformed input. First described by Szegedy et al. [19], an AE is an image to which non-random perturbation (noise), indistinguishable to the human eye, is added to maximize the likelihood of that image being misclassified by a trained DNN. The goal of an AE is that the predicted label ($\hat{y}$) does not match the true label (y). Figure 2.5 depicts an adversarial example generated by the Fast Gradient Sign Method, which will be discussed later in this chapter. Two common attack methodologies for generating AEs are *gradient optimization* and *constrained optimization*. Gradient optimization attacks calculate the noise for each pixel based upon its gradient value, whereas constrained optimization attacks minimize perturbations by optimizing some similarity metric. All adversarial attacks, regardless of the overarching methodology they belong to, can be classified by the *transparency* of the method, and the *specificity* of the desired misclassification.

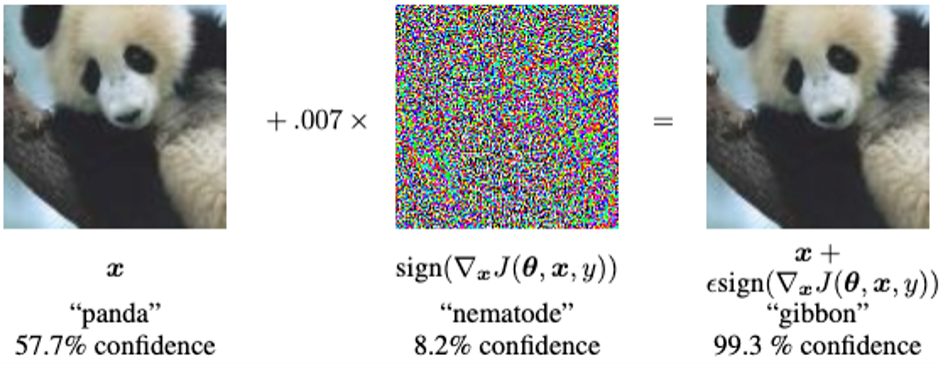

Figure 2.5. Example of how Fast Gradient Sign Method quickly generates an adversarial example. Source: [20].

Attack Transparency and Specificity

A *white box* attack is considered *transparent* because the attacker has direct access to the model they intend to attack along with detailed information such as its architecture, weights, output logits, and class probabilities. As such, the attacker may generate adversarial examples to attack the target model directly. *Black box* attacks are *non-transparent*. Without access to the target model, the attacker must instead generate adversarial examples on an

approximate DNN first. Then, those examples must be *transferred* to the target DNN to carry out the attack. Certain attacks are designed specifically as black box attacks, such as the Square attack, which only needs access to the output of the target DNN to craft adversarial perturbation [21]. Since black box attacks are first generated on a surrogate model, they generally have a much lower success rate when attacking the target model compared to their white box equivalents, though not always [21].

Specificity refers to whether an attack is attempting to induce the target model to misclassify an image as a specific (*target*) class, or as any (*non-targeted*) class. Where non-targeted attacks only require that the model's prediction differs from the truth label ($\hat{y} \neq y$), targeted attacks are only considered successful when the model predicts some specific class targeted by the adversary ($\hat{y} = y_{target}$). As one would expect, targeted attacks in general have a lower success rate than their non-targeted equivalents.

2.2.1 Adversarial Attacks

For our research, we compared six well-known attacks to our novel method: Fast Gradient Sign Method, Projected Gradient Descent, DeepFool, Square, Carlini-Wagner ℓ_2, and Auto attack.

Fast Gradient Sign Method (FGSM)

One of the earliest proposed attacks, FGSM was designed to quickly produce adversarial examples [20]. FGSM generates adversarial noise by first calculating the gradient map of the image with respect to the cost function ($\nabla_x J(\theta, x, y)$) and then multiplying the signs of those gradients by some noise parameter, ϵ [20]. The adversarial example, $\hat{X}$, is created by adding the noise map to the original image, such that

$$\hat{X} = X + \epsilon sign(\nabla_x J(\theta, X, y)).$$

FGSM-generated adversarial examples are non-optimal with regards to the distance metric. The name of the game with FGSM is speed, not optimization.

Projected Gradient Descent (PGD)

Best understood as a multi-step variation of FGSM, the PGD attack described in Madry et al. [22] can quickly find robust adversaries which are bound by either ℓ_∞ or ℓ_2 distance metrics. For ℓ_∞-bound adversaries, PGD first samples one or many times from the ℓ_∞-ball of the original image. With a small step size α, PGD then *ascends* the gradient to maximize loss for *non-targeted* attacks or *descends* the gradient to minimize loss for *targeted* attacks. Perturbation is clipped by ϵ after every step to enforce the ℓ_∞ bound. Through this method of projecting off the manifold by introducing noise to the benign image followed by multiple bounded steps, PGD can better constrain perturbations while finding more effective adversarial examples better than FGSM.

DeepFool

The DeepFool attack, proposed by Moosavi-Dezfooli et al. [23], was originally designed to generate ℓ_2-bounded adversaries, but later altered to fit any ℓ_p metric. Strictly a non-targeted attack, DeepFool makes an approximate calculation of the minimum distance to the nearest adversarial class and uses an adaptive step size to reach it [23]. Since this calculation is only an approximation, there is no guarantee that the perturbations introduced to $\hat{X}$ are optimal. However, this greedy approach yields very small perturbations that are good approximations of the optimum [23].

Square Attack

Strictly black box, the Square attack is a score-based non-targeted attack that can adhere to either ℓ_∞ or ℓ_2 constraints. Instead of relying on gradient information to generate adversarial noise, it finds minimal perturbations through randomized search at the approximate decision boundary for the proposed set of classes [21]. Importantly, these characteristics make Square impervious to gradient masking as a defensive technique. Compared to similar state-of-the-art black box attacks, the Square attack increases query efficiency by a maximum factor of three against various ImageNet classifiers [21]. Even more impressive, it has outperformed certain gradient-based white box attacks.

Carlini-Wagner ℓ_2 Attack

Carlini and Wagner [24] outlined a suite of attacks that optimize the perturbations for a sample based on various distance metrics (ℓ_0, ℓ_2, and ℓ_∞). By addressing the attack as an optimization problem, where the objective is to minimize the amount of adversarial noise while maximizing the loss between a model's prediction and target label, Carlini-Wagner attacks are capable of creating highly robust adversaries with minimal perturbations [24]. When compared with other gradient-based attacks, Carlini-Wagner AEs demonstrate a higher success rate while remaining closer to the benign original with regard to distance. This research exclusively used the ℓ_2 attack.

Auto Attack

The Auto attack is not a single attack but an ensemble of attacks used as a comprehensive approach to evaluating the robustness of a trained DNN. First, it uses a novel method of implementing PGD, called Auto PGD, which utilizes an adaptive method for better determining the number of k steps to take [25]. Second, the Auto Attack iterates through multiple attacks (Auto PGD, DeepFool, Square, etc.) in an attempt to induce misclassification of as many benign samples as possible across the ensemble [25].

2.2.2 Adversarial Defenses

This research utilizes a variety of adversarial defenses to compare the performance of our RL attack methodology to those just discussed.

Adversarial Training

Perhaps the simplest defense, adversarial training adds adversarial examples to the training set, each with the associated true label. Once trained, the network will be more likely to predict the correct class of an adversarial example. Easy to implement, this approach has a multitude of weaknesses. First, adversarial training can only defend against known attacks and is still susceptible to unknown attacks. Second, a well-defined training set would need to include an adversary for each image per known attack, and to be truly robust it would need to contain multiple adversaries for every benign sample, each derived from a unique perturbation bound. Given the breadth of attacks available, this strategy is not reasonable. Our research implemented adversarial training using PGD adversaries.

Gradient Regularization

Research has suggested that AEs exist at the edge of the decision boundary between classes, where small pixel changes are enough to push an image into the incorrect class [26]. Gradient regularization addresses this vulnerability by penalizing a DNN during training when small differences in pixel values lead to dramatic increases in loss [27]. That is, gradient regularization seeks to protect the network from minimally perturbed AEs by reinforcing the decision boundary for each class. While gradient regularization has not proven to be particularly effective against white box attacks, it has demonstrated robustness against black box attacks.

TRadeoff-inspired Adversarial DEfense via Surrogate-loss minimization (TRADES)

When training a network to defend against adversarial examples, there is a known but poorly understood trade-off between adversarial robustness and model accuracy. That is, when adversarial robustness is strong, accuracy tends to suffer, and vice versa. Winner of the NeurIPS 2018 Adversarial Vision challenge, TRADES offers a defensive technique that balances these competing goals by tightly bounding the robust error of a model. This balancing can be accomplished by decomposing the error into two parts: 1) a natural error and 2) a boundary error [28]. By regularizing both of these terms through the surrogate loss function, natural accuracy is promoted while the decision boundary is pushed away from the data, providing adversarial robustness [28].

PadNet

Proposed by Barton et al. [29], PadNet is unique relative to other defenses in two key areas: First, PadNet incorporates into the mode a *padding class* designed to detect likely AEs. In practical terms, if a network is designed for n number of classes, the padding class is $n + 1$. Training the padding class is similar to adversarial training but is refined by adding AEs generated from a mix of benign and PGD adversaries. This technique adds *boundary padding* to the model that is used to better define valid classes and remove decision space, which adversarial attacks might exploit. Second, PadNet further refines the decision boundary around valid classes by incorporating targeted gradient regularization to penalize gradients in the direction of the barrier class [29]. Again, this defense feature is important, as minimally perturbed adversarial examples are found at the edge of the

decision boundary for the incorrect class [26]. By combining boundary padding and targeted gradient regularization, PadNet has demonstrated remarkable resilience to a variety of attack methodologies [29].

2.3 Reinforcement Learning

Unlike other attacks, our novel method uses reinforcement learning to generate adversarial examples. As a field of research, reinforcement learning (RL) is concerned with learning behavior, mapped to a situation, in service of achieving a goal [7]. Examples of RL in action range from the simple, such as a smart thermostat learning to efficiently regulate temperatures [30], to the wildly complex task of air-to-air combat [31]. Unlike supervised learning, in which a network has access to the correct answer in the form of labeled data, or unsupervised learning, in which the DNN is seeking to find structure in unlabeled data, RL DNNs, or *agents*, learn through some form of observe-act-reward loop meant to reinforce behavior that returns the highest rewards [7]. Implemented correctly, RL agents can often learn to outperform human experts, without having prior human knowledge [9]–[11].

2.3.1 RL Structure and Training

To discuss the key elements and terminology of RL, let us take the example of trying to train an agent to play the game Pac-Man [32]. In this scenario, the *agent* is the DNN playing as the Pac-Man character, and the *environment* is the game maze, to include pellets, fruit, and ghosts. Training takes place over the course of many *episodes*, which to the observer is a single game played to completion, win or lose. At each *time step*, or simply *step*, the environment presents the agent with a *state*, which contains the current position of the agent, obstacles, and game elements. For this training example, there is no limit to the number of steps for a game, but this may not be the case for environments where speed could be an element of success. Given the state, the agent will predict and play the best *action* based on its current *policy* for playing. The total number of actions available to the agent is called the *action space*, which in this case is discrete (four actions: up, down, left, right) but may be continuous in other cases (as in the power output to a drone motor). Once an action is played, the environment evaluates the action and returns a *reward*, which signals how well an action performed for that specific state. The episode is over when the agent has reached a *terminal state*, which may be when all pellets and fruit have been consumed (success) or

when the agent runs out of lives (failure). At the end of an episode, training will occur to update the *target policy*, and if training is not complete, the environment will reset for the next episode.

The generalized learning regimen of an agent is graphically summarized in Figure 2.6:

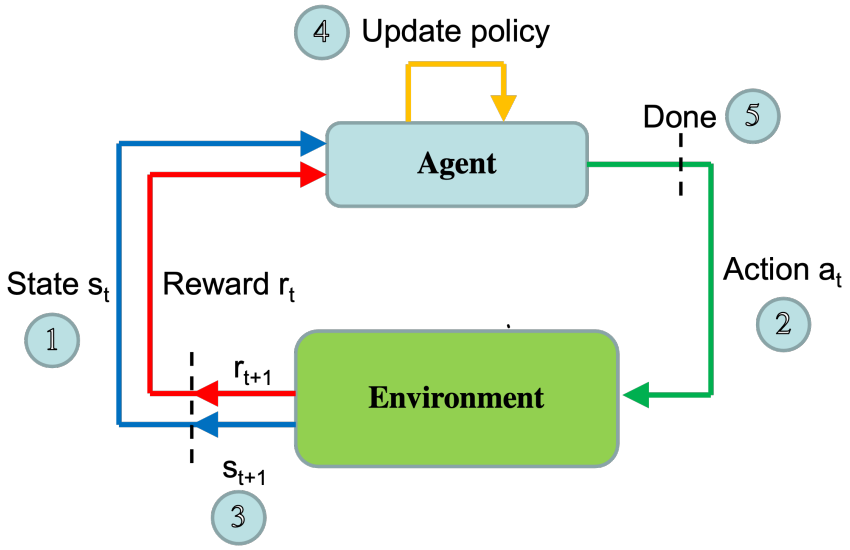

Figure 2.6. Basic observe-act-reward loop used with reinforcement learning. **1)** Environment presents its current state (s_t) to the agent for observation. **2)** Agent uses behavior policy to predict and play the best action for the current state. **3)** Environment returns a reward for the played action (r_t) and the next state (s_{t+1}). Repeat steps 1–3 until terminal phase or max time steps reached. **4)** Agent updates policy in an effort to maximize rewards. **5)** Environment resets if there are more training episodes, else the program terminates. Adapted from [7].

On-Policy versus Off-Policy Algorithms

When discussing agent policies, we need to understand the distinctions between the behavior and target policies mentioned above. The target policy represents the optimal policy that the agent is trying to learn, while the behavior policy is what it uses to select actions. For *on-policy* algorithms, the policies are the same. For *off-policy* algorithms, the policies are distinct, and the behavior policy is updated less frequently than the target. Off-policy

algorithms tend to do better at exploring the environment, which can lead to a better target policy. The tradeoff between exploration versus exploitation is discussed in more detail in section 2.3.2.

On the Importance of Rewards and Future Rewards

As rewards are arguably the most critical part of RL, let us briefly return to the reward function and its impact on agent learning. With RL a "best action" must be defined in the context of immediate versus future rewards. Since "best" is therefore relative to the task, the reward function needs to be carefully designed to reinforce only good behaviors. With RL, the emphasis is on accomplishing the objective or goal, and not simply teaching specific actions, since it is the job of the agent to learn potentially unexpected strategies for optimizing a task [7]. Depending on the environment and the task, the best reward function can range from simple to complex. For our Pac-Man example, a sparse reward might be +1 point for every time step survived, with the goal being to maximize time steps played. However, this strategy may be too simplistic and may train a policy that prioritizes longevity over game completion. A more complex reward signal may be to give +1 for pellets, +5 for fruit, and -5 for dying.

Almost as important as the reward function itself is the *discount factor* (γ) used for evaluating the importance of future rewards. Represented as a continuous value between 0 and 1, the discount factor determines how much emphasis is put on future versus immediate rewards. Values closer to 1 indicate that the agent should heavily weight future rewards, encouraging the agent to perhaps forgo immediate but small rewards for large rewards in the future. Low γ values signal to the agent that the immediate reward is the most important.

In the next sections, we will discuss the method of reinforcement learning used in this research.

2.3.2 Q-learning with Deep Neural Networks

Described in the 1950s by mathematician Richard Bellman [12], Markov decision processes (MDP) describe the means by which an agent can choose one of many possible actions, and by receiving some reward, reinforce behavior that will maximize its performance (evaluated by overall reward). Derived from Markov chains, which have no memory of past states, MDP

transition probabilities only rely on the current and next possible state (s_t, s_{t+1}). Figure 2.7 shows a simple example of an MDP represented as a state diagram [12].

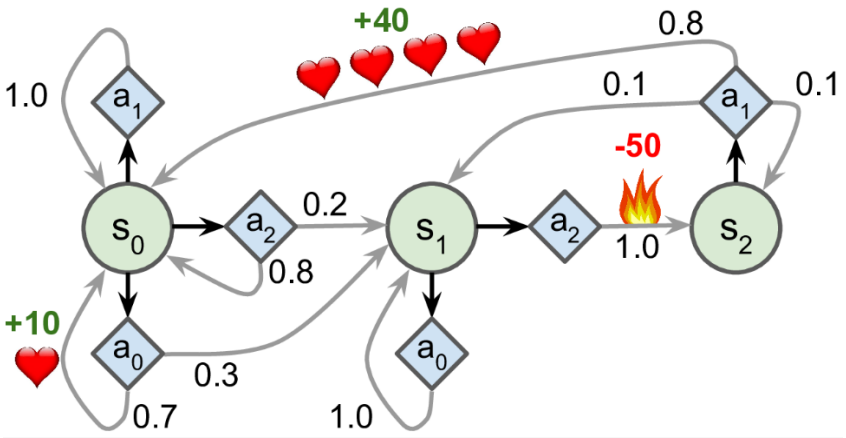

Figure 2.7. Example of a Markov decision process. Source: [12].

Q-learning

Derived from MDP, Q-learning assumes that all initial transition probabilities and associated rewards are unknown, and therefore must be explored [12]. Without any knowledge about the goal or proper behavior, an agent must first play randomly to learn the value of state-action pairs, represented as $Q(s, a)$. The Q-learning algorithm is expressed with Equation 2.1, which states that the Q-value of each state action pair is computed from the reward of that state-action pair plus discounted future rewards expected [12].

$$Q(s,a) \underset{\alpha}{\leftarrow} r + \gamma \cdot \max_{a'} Q(s',a') \tag{2.1}$$

Exploration vs. Exploitation

Recall that initially an agent has no knowledge about its environment and all the possible state-action pairs that exist within it. For the agent to be able to eventually *exploit* the environment, it first has to adequately *explore* it, typically through random actions [7], [12]. While this concept seems simple enough, each action must be played for one particular

state for the agent to get a good approximation of the reward it can expect. Extrapolate that pattern over all possible states and it becomes clear that such a task will be computationally expensive and require extremely long periods of training. One solution to the *exploration-exploitation* dilemma is to use an *ε-greedy policy*, where ε is the probability of acting randomly, and 1-ε is the probability of the agent predicting the greedy action using its current behavior policy. At the start of training, ε will be 1, meaning that all actions are random. As time progresses, ε will slowly decay towards some minimum value, such as 0.01. The ε-greedy policy allows the agent to almost exclusively explore the environment early in training and then gradually transition to exploitation [12]. For example, an ε of 0.6 means that 60% of actions in an episode will be random choice, with the remaining 40% chosen by the behavior policy. To avoid confusion with AE nomenclature, where ε refers to the noise found in an adversary, we simply use the term ε-greedy (ε_g) to refer to the exploration-exploitation probability.

Deep Q-learning with Deep Q-networks

Though the theory behind Q-learning is sound, its greatest drawback is that it does not scale well to environments with complex MDPs. DeepMind showed that this problem could be solved by combining Q-learning with deep learning [8]. This approach is called deep Q-learning (DQL) and uses a specialized DNN referred to as a deep Q-network (DQN).

An important feature of DQL is the use of *experience replay*, which allows the DQN to learn from prior experiences. An *experience* is defined as the tuple $(s_t, a_t, r_{t+1}, s_{t+1})$, where s_t is the state at time step t, a_t is the played action for that step, r_{t+1} is the returned reward, and s_{t+1} is the subsequent state. For each time step t, an experience is saved to the *replay memory*. At the training step, the DQN will randomly sample a mini-batch of experiences to train the target policy. Deep Q-learning is an off-policy algorithm, as it requires a behavior policy that allows for environment exploration (such as ε-greedy).

The DQL algorithm calculates the target Q-value to use when training the DQN [12], and can be expressed as

$$Q_{target}(s, a) = r + \gamma \cdot \max_{a'} Q_\theta(s', a'). \tag{2.2}$$

While equation 2.2 shares many similarities with the original Q-learning algorithm (equation

2.1), the future rewards are only an *expectation* based on the policy of the agent at that time. While this method has proven capable of learning to exploit a given environment, the drawback was that a single target network often overestimates Q-values. DeepMind addressed this issue by proposing a variation to DQL - double Q-learning.

Double DQN

DeepMind researchers found that using a single *target network*, represented as θ_T in equation 2.3, for evaluating both action selection and action evaluation when calculating the Q-learning error often resulted in overestimating Q-values [33].

$$Y_t^Q = R_{t+1} + \gamma Q(S_{t+1}, \arg\max_a Q(S_{t+1}, a; \theta_T); \theta_T) \tag{2.3}$$

A solution was discovered by decoupling the action selection and evaluation functions and assigning them to separate DQNs, which in practice is referred to as double DQN. An *online model* would be responsible for action selection using the greedy policy, while a *target model* would evaluate the overall policy. The double Q-learning error is shown in Equation 2.4, where the online and target models are represented as θ_T and θ'_T, respectively [33].

$$Y_t^{DoubleQ} \equiv R_{t+1} + \gamma Q(S_{t+1}, \arg\max_a Q(S_{t+1}, a; \theta_T);\ \theta'_T) \tag{2.4}$$

Another important distinction between the online and target models is the manner in which their weights are updated. Since the online model is the policy the agent uses for its action selection, the weights are updated with each training step. However, this is not the case for the target model, which is evaluating the online policy. By only periodically updating the weights for the target model with those of the online model, the evaluation function is stabilized, which leads to increased performance [33]. Figure 2.8 depicts how the two DQNs work in tandem to stabilize learning.

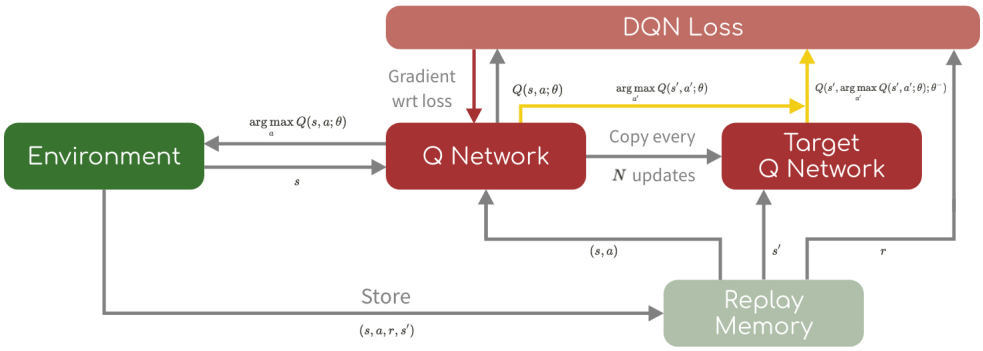

Figure 2.8. Double DQN. Source: [34].

Dueling DQN

Shortly after the publication of double DQN, DeepMind introduced the dueling deep Q-network (DQN) (DDQN) architecture, depicted in Figure 2.9. This dueling architecture shares the same convolutional layers but separate estimator networks: one for learning state values and another for learning action-advantages [35]. These two networks are then aggregated to produce the Q-values for a given state.

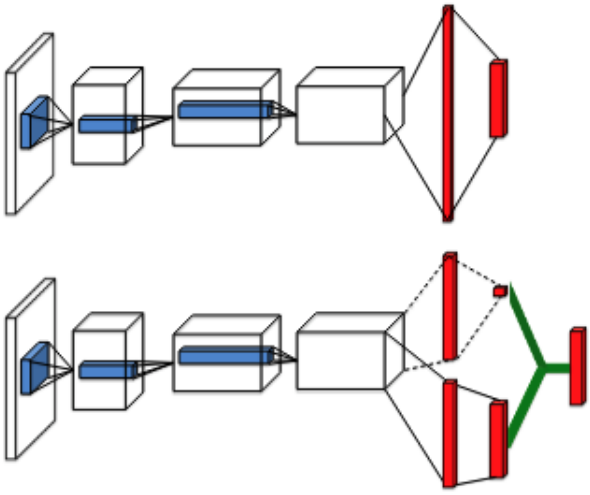

Figure 2.9. Dueling DQN architecture below traditional DQN. In the DDQN the state-value and action-advantages are calculated in separate networks and then aggregated. Source: [35].

The benefit of this approach is that by using separate estimators, DDQN does a better job of learning which states are irrelevant to the goal, meaning that it does not have an appreciable effect regardless of the action played [35]. This leads to more generalized learning, which results in better policy evaluation for states where all actions produce similar results.

2.4 Summary

In this chapter, we discussed the value of convolutional neural networks to deep learning, specifically image and object recognition. As powerful as they are, convolutional neural networks are highly susceptible to attacks from malformed input images, known as adversarial examples. We briefly described the adversarial attacks and defenses used for comparison in this research. Finally, we discussed reinforcement learning, and more specifically deep Q-learning and deep Q-networks, which power our attack.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3: Methodology

In this chapter, we describe the research methodology and experiment design used for this thesis. Section 3.1 reviews terminology used in this research. Section 3.2 is a generalized discussion of our method, an ARLA. Section 3.3 discusses the dataset used to form adversarial examples in this research. Section 3.4 discusses ARLA's learning environment. Section 3.5 discusses the ε-greedy policy used to determine if ARLA is going to explore or exploit the environment. Section 3.6 discusses our reward function and approach for determining correct behavior. Section 3.8 covers our experiments for comparing ARLA to other well-known attacks.

3.1 Terminology

The following is a review of terminology used in our research.

- α: ARLA pixel noise per step (Section 3.2)
- ε-greedy: exploration-exploitation probability
- γ: discount factor for future rewards
- ε: ℓ_2 bound for noise in adversarial example
- C: Confidence of adversarial examples, lower number results in AE closer to original

3.2 A Reinforcement Learning Approach

We propose an Adversarial Reinforcement Learning Agent (ARLA) as a means of generating minimally perturbed adversarial examples. As an agent, ARLA is a double dueling deep Q-network (DQN) (D3QN), as described in Section 2.3.2. Figure 3.1 illustrates the dueling architecture used for both the online and target models.

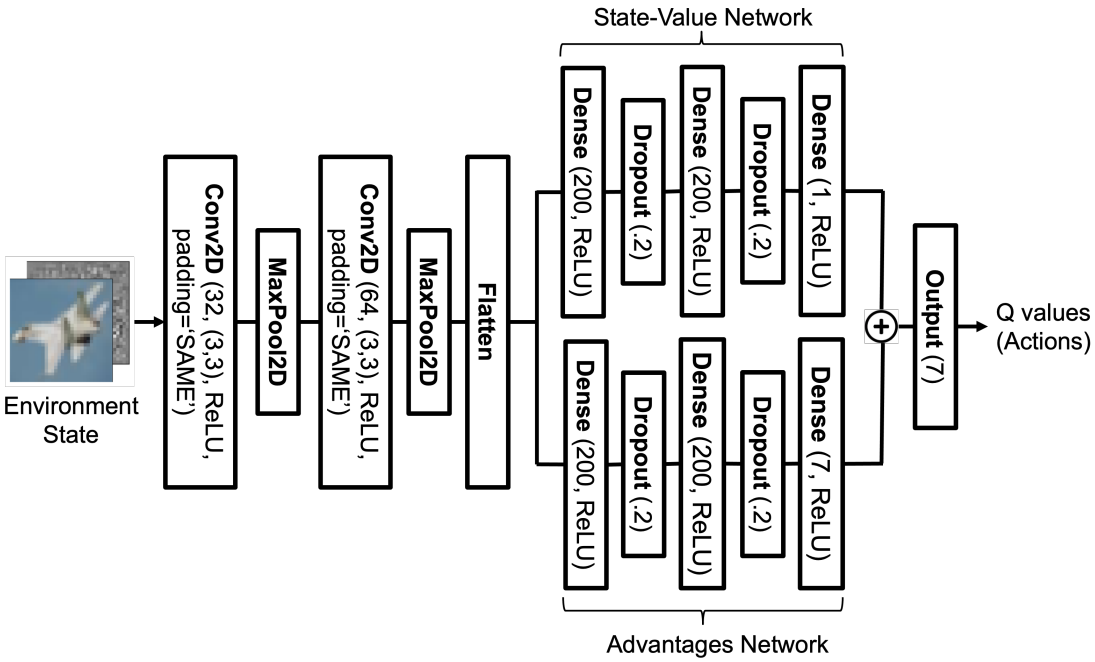

Figure 3.1. Dueling DQN architecture used for ARLA models. Environment consists of color and gradient map channels. Adapted from [36].

ARLA's goal is to interact with an image to learn adversarial perturbations. It does so by traversing the 2D image space much like a Pac-Man agent would traverse a 2D grid. At any given state, the ARLA agent occupies a single pixel location. A state transition occurs when the agent takes one action from the following action space: move_up, move_down, move_left, move_right, change_pixel. For an image with three color channels red, green, and blue, the change_pixel action is extended to include change_red, change_green, change_blue making a total of seven actions in the action space. Pixel changes are based upon the sign of the gradient for that pixel multiplied by some noise parameter, α. The major difference between non-target and target attacks is the label ARLA uses to generate the gradient maps and the direction of the change for that pixel. For the non-targeted attack, the gradient is calculated with the true label, and the change *ascends* the gradient in order to maximize loss between the input image and the truth label. The targeted attack takes the opposite approach, where the gradient is first calculated by the target label, and the change descends the gradient to minimize loss between the input image and the targeted label. Equations 3.1

and 3.2 represent these non-target and target methods, respectively.

$$Non - Target : pixel_{new} = pixel_{old} + (\alpha \cdot sign(\nabla_x(f(x), y_{true})))[pixel_{index}] \quad (3.1)$$

$$Target : pixel_{new} = pixel_{old} - (\alpha \cdot sign(\nabla_x(f(x), y_{target})))[pixel_{index}] \quad (3.2)$$

During the course of this research, a third attack was considered, which will be referred to as the *hybrid* attack. While being a non-targeted attack, the hybrid attack combines the gradient calculation and pixel change of the target attack with the success condition of the non-target attack ($\hat{y} \neq y_{true}$). In another words, ARLA drives towards a target label but stops as soon as it classifies to any incorrect label. This allows ARLA to find distinct adversarial example space that would not be found using our targeted or non-targeted approach.

During the course of an episode, ARLA saves an experience, $(S_t, a_t, r_{t+1}, S_{t+1})$, for each time step to a fixed-size replay memory. The replay memory uses a simple deque data structure, which is first in, first out, so newer experiences push out older ones. Once the replay memory has filled to a certain number of episodes, training occurs after each episode with a randomly sampled mini-batch of experiences.

3.3 Image Dataset

To generate adversarial examples, we chose the CIFAR-10 dataset [36], which is well known in image classification and adversarial research. Images in this dataset have dimensions of 32x32x3 (color) and belong to ten classes. An example of each class can be seen in Figure 3.2. Pixel values are normalized between 0.0 and 1.0.

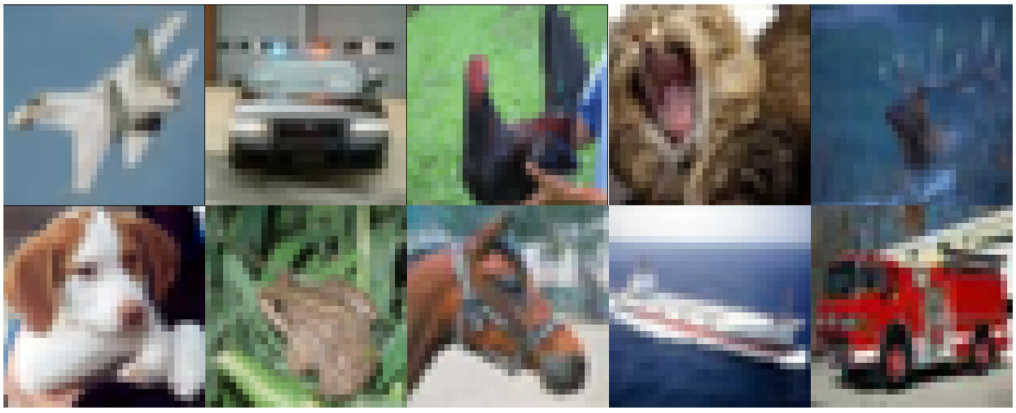

Figure 3.2. Example of each class within the CIFAR-10 dataset. From top left to bottom right: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck. Adapted from [36].

3.4 Learning Environment

Built using the OpenAI Gym API [37], ARLA's environment requires two major components - a *classifier network* and a *benign image*. In the context of this learning environment, the classifier network is not necessarily the *target network*, which we are ultimately trying to attack. For a white box attack, yes, the classifier network is the target network. However, in the case of a black box attack where the target network is hidden, the classifier network is just the best approximation of the real-world target. In this research, black box target networks share the same architecture as their white box counterparts, but with different weights.

Using the benign image and classifier network, the environment creates six channels, which can be considered as two distinct sets. The first set consists of the RGB color channels of the image, and the second set consists of the gradient channels calculated by the classifier network for each color channel. This process is illustrated with Figure 3.3. Each time ARLA adds noise to a pixel, the environment state is updated to reflect the new gradient values.

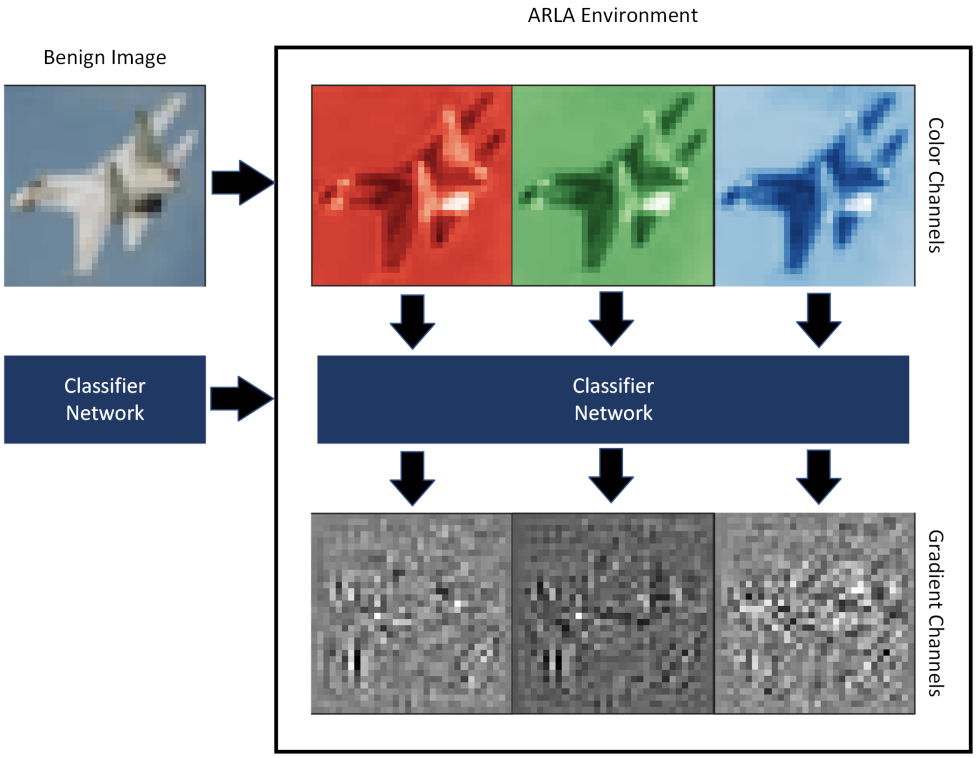

Figure 3.3. Creation of ARLA environment using benign image and target network. Adapted from [36].

3.5 Epsilon-Greedy Policy

To find the middle ground between exploration and exploitation within the environment, ARLA uses a simple ε-greedy policy for determining the action to play. Algorithm 1 outlines this policy. For each step, a random number between 0 and 1 is generated (line 1). If that number is less than the ε-greedy value, then action is chosen at random, otherwise the action is predicted by the online model for the given state (lines 2-5).

Algorithm 1 Epsilon-Greedy Policy

Variables	
S	environment state
A_O	ARLA Online Model
ε_g	ε-greedy value
Functions	
$random()$	returns random float between 0 and 1
$randint(i)$	returns random integer between 0 and i exclusive
$A_O.predict(S)$	A_O predicts action based on state

function $eg\_policy(A_O, S, \varepsilon_g)$
    $rand \leftarrow random()$
    if $rand < \varepsilon_g$ **then**
        $action = randint(7)$
    else
        $action = A_O.predict(S)$
    end if
    return $action$

3.6 Reward Function

A common theme in reinforcement learning is carefully crafting the reward function to reinforce good behaviors. For ARLA, "good" behavior was envisioned as rewarding minimal pixel changes that led to large loss values, which would result in an image being misclassified with only minimal perturbation. A good reward function would be one where a successful adversarial example with the highest score also has the shortest ℓ_2 distance from the benign original.

Algorithm 2 outlines our approach to this problem. We used a parameterized loss threshold to differentiate between good and bad pixel changes. For pixel changes that generate a high level of loss, the new_loss will be much greater than the old_loss, resulting in a loss ratio greater than 1. Therefore, if the ratio is greater than the threshold (we used 1.001) we can say that the pixel change was a good action. We then multiply the threshold decimal by 1,000 to scale the reward to the amount of loss incurred (greater loss = greater ratio = greater reward). Likewise, if the loss ratio fell below that threshold, that pixel change was considered a bad action and received a fixed negative reward (-4). In an attempt to neither

punish nor reward movements which are necessary to explore the environment to find the most meaningful pixels, our reward function is only called when a pixel is changed. Agent movements (up, down, left, right) automatically receive a score of 0.

Algorithm 2 Calculate Reward

Variables	
A	y probability
B	$\hat{y}$ probability
C	success_bonus
$done$	boolean

function REWARD($A, B, C, done$)
    $new\_loss = CategoricalCrossEntropy(A, B)$
    $loss\_ratio = \frac{new\_loss}{old\_loss}$
    if $loss\_ratio >= 1.001$ **then**
        $r1 = round(loss\_ratio - 1)$
        $r1 = int(r1 * 1000) + 1$
    else
        $r1 = -4$
    end if
    $old\_loss \leftarrow new\_loss$
    $r2 = done * C$
    $reward = r1 + r2$
    return $reward$
end function=0

3.7 ARLA Algorithm

Now that we have covered all the individual components of ARLA, we can construct a holistic view of our approach. Algorithm 3, adapted from the double DQN algorithm [33], decomposes ARLA's method for generating adversarial examples during its training loop. First, ARLA's online model is instantiated (line 1) with the architecture discussed in Section 3.2. The target model is a direct copy of the online model (line 2), to include the weights. Replay memory is instantiated using a deque data structure, where $N\_Experiences$ is the maximum number of experiences the memory can hold (line 3). As described in Section 3.4, ARLA's environment is instantiated using a original image (X), the true label (y), the classifier network being attacked (C), and a categorical crossentropy loss object (CCE) used for calculating the reward, as defined in Section 3.6 (line 4). The outer for loop iterates through a total of $N\_Episodes$, where each episode starts with the initial state, and calculates the ε-greedy value (ε_g) for that episode (lines 5-7). The inner for loop iterates through N_Steps, where actions are played within the environment until an adversarial

example is found or the total number of steps have been played. Within this for loop, an action is first derived through the ε-greedy policy, defined previously with Algorithm 1 (line 9). The action is played in the environment, which returns the next state, the reward, and a boolean value *done*, which represents whether an adversary was found (line 10). The state, action, reward and next state are then stored in the replay memory as a tuple (line 11). The next state is now the current state for the next step (line 12). If *done* is true, then the RGB channels of the state are saved as an adversarial example and the episode is over (line 13-14). If enough episodes have elapsed, then a batch of experiences is sampled from the memory (line 18). The target Q value is calculated, as described with Figure 2.8 and Equation 2.4, and a gradient descent step is performed on ARLA's online model (lines 19-20). The target model weights are then updated incrementally using some percentage of the online model weights (line 21). For our research we used $\tau = 0.99$.

3.8 Experiments

As this research was centered on generating minimally perturbed adversarial examples, our experiments were designed to compare ARLA's success rate against six well-known attacks, discussed in Section 2.2.1: FGSM, PGD, DeepFool, Carlini-Wagner ℓ_2, Square, and Auto attack. To ensure consistency and correctness of our comparison methods, we used the Adversarial Robustness Toolbox (ART) [38] for adversary generation. Parameters for both ARLA and ART attacks are covered in Table 3.1. For benign samples we used the first 100 images of the CIFAR-10 test set. ARLA was trained on a given benign image for 100 episodes, and the AE with the shortest ℓ_2 distance was saved for comparison with other attacks. At the end of those 100 episodes, ARLA was reinitialized with new weights and a new environment for the next image.

For target networks, we used seven variations of a Wide Residual Network (WRN) [39] model, each trained with a different style of defense, described in Section 2.2.2: 1) undefended, 2) Projected Gradient Descent (PGD) Adversarial Trained (PGDAT), 3) Gradient Regularization, 4) TRADES, and 5) PadNet. White box attacks are straightforward enough—the target network is loaded into ARLA's environment, as outlined in Section 3.4, with those resulting adversaries used to attack the same network. For black box attacks, we used the same adversaries but attacked a similar network. "Similar" in this case is defined

Algorithm 3 ARLA

Variables	
X	original image
y	true label for X
S_t	current state
S_{t+1}	Next state
A_O	ARLA online model
θ	ARLA online model weights
A_T	ARLA target model
θ_T	ARLA target model weights
C	classifier network
CCE	Categorical Crossentropy
RM	replay memory
τ	Update parameter θ_T
Functions	
$Environment(X, Y, C, CCE)$	creates ARLA environment
$MSE(Q^*, Q_\theta)$	Mean squared error between target and predicted Q values
$Double\_DQN(A_O, A_T, batch)$	Defined with Equation 2.4

$A_O \leftarrow dueling\_dqn()$
$A_T \leftarrow copy(A_O)$
$RM \leftarrow deque(N\_Experiences)$
$env \leftarrow Environment(X, y, C, CCE)$
for $episode$ in $N\_Episodes$ **do**
    $S_t \leftarrow env.reset()$
    $\varepsilon_g \leftarrow \max(1 - \frac{episode}{N\_Episodes}, 0.01)$
    for $step$ in N_Steps **do**
        $a_t \leftarrow eg\_policy(A_O, S_t, \varepsilon_g)$
        $S_{t+1}, r_t, done \leftarrow env.step(action)$
        $RM.append((S_t, a_t, r_t, S_{t+1}))$
        $S_t \leftarrow S_{t+1}$
        if $done$ **then**
            save $(S_t[r, g, b])$ and break
        end if
    end for
    if episode >= training_start **then**
        $batch \leftarrow RM.sample(batch\_size)$
        $Q^*(S_t, a_t) \approx r_t + \gamma Q_\theta(S_{t+1}, \underset{a}{argmax}\ Q(S_{t+1}, a_t; \theta); \theta_T)$
        Perform gradient descent step on $MSE(Q^*, Q_\theta)$
        $\theta_T \leftarrow (\tau * \theta_T) + ((1 - \tau) * \theta)$
    end if
end for

as sharing the same architecture but with different weights.

Table 3.1. Attack parameters for ARLA and ART attacks. It is important to note that for ART parameters, "Confidence" is exclusive to the Carlini-Wagner attack, and "Max_iter" is exclusive to the Square attack.

ARLA Parameters			
Total Episodes	100	**Batch Size**	32
Steps per Episode	500	**Training Steps**	1
α	0.031	**Training Start**	10
γ	0.95	**Loss Function**	MSE
Memory Size	20,000	**Learning Rate**	0.006
ART Attack Params			
ε	0.5	**Norm**	2
Confidence	0	**Max_iter**	5,000

3.9 Summary

In this chapter, we discussed our methodology and line of experiments used in our research. ARLA is a deep Q-learning agent designed to generate minimally perturbed adversarial examples, enhanced with double Q-Learning and a dueling deep Q-network (DQN) architecture. Target images being used come from the CIFAR-10 dataset, which is composed of ten distinct classes [36]. ARLA's learning environment requires a single image and target network, which are then used to calculate gradient maps. Our reward function was one of the greatest challenges of this research and premised on the concept that increases in loss should be rewarded. Finally, we discussed how our experiments are designed to compare ARLA to other well-known attacks.

CHAPTER 4:
Results

In this chapter, we present the results from our experiments comparing the performance of ARLA to other attacks. Section 4.1 discusses ARLA's performance relative to other attacks against an undefended model using two different α sizes. Section 4.2 discusses our results against PGD Adversarial Training; Section 4.3, against Gradient Regularization; Section 4.4, against TRADES; and Section 4.5, against PadNet. Overall, ARLA performed well against all models, in terms of both attack success and average ℓ_2 distance compared to the other attacks. However, ARLA significantly stood out compared to the other attacks when tested against the PadNet defense in which ARLA was the top performing attack.

4.1 Undefended WRN

For our initial experiment, we varied the α parameter used by ARLA to create perturbations, which highlighted the drastic impact it has on attack performance. Table 4.1 contains observed performance metrics for ARLA compared to other attacks when targeting the undefended model. For the first set of attacks, ARLA used a α value of 0.1. This large α produced considerably higher success rates with our white box attacks compared to our black box attacks. While not the overall best performer, ARLA did outperform other attacks in the white box category, at least in terms of success rate. However, using a large α resulted in average ℓ_2 distances for ARLA-generated AEs roughly two to three times higher than those for the best-performing attack, PGD. We successfully constrained the adversarial perturbations by setting the α parameter to 0.031. Using this smaller α resulted in lower success rates for ARLA against the undefended model overall, but also pushed the average ℓ_2 distance below that of most other attacks. Regardless of which α value used, ARLA's hybrid attack outperformed the targeted attack, which was a common result across our experiments.

Table 4.1. Comparative attack performance against the undefended WRN model.

	Undefended Model											
	White Box (Benign Accuracy = 0.91)						Black Box (Benign Accuracy = 0.92)					
	Non-Targeted		*Targeted*		*Hybrid*		*Non-Targeted*		*Targeted*		*Hybrid*	
	Success Rate	Avg ℓ_2 Dist	Success Rate	Avg ℓ_2 Dist	Success Rate	Avg ℓ_2 Dist	Success Rate	Avg ℓ_2 Dist	Success Rate	Avg ℓ_2 Dist	Success Rate	Avg ℓ_2 Dist
ARLA (α=0.1)	87%	0.934	66%	0.953	79%	0.911	13%	0.339	14%	0.69	16%	0.652
ARLA (α=0.031)	56%	0.427	43%	0.416	53%	0.399	11%	0.122	11%	0.264	13%	0.221
FGSM (ε=0.5)	75%	0.497	78%	0.498	N/A	N/A	42%	0.497	41%	0.498	N/A	N/A
PGD (ε=0.5)	93%	0.5	88%	0.464	N/A	N/A	68%	0.5	58%	0.449	N/A	N/A
DeepFool (ε=0.5)	93%	19.327	N/A	N/A	N/A	N/A	77%	20.345	N/A	N/A	N/A	N/A
Square (ε=0.5)	21%	0.499	N/A	N/A	N/A	N/A	10%	0.494	N/A	N/A	N/A	N/A
C&W ℓ_2 (C=0.0)	57%	0.421	50%	0.311	N/A	N/A	15%	1.139	14%	0.639	N/A	N/A
Auto Attack (ε=0.5)	94%	0.5	94%	0.5	N/A	N/A	70%	0.5	71%	0.5	N/A	N/A

While this table provides useful metrics for understanding ARLA's performance, to fully appreciate the results, we need to visualize them. Subsections 4.1.1 and 4.1.2 provide examples of benign original images, followed by the shortest ℓ_2 adversary ARLA generated for that image, and then the isolated noise. For example, regarding Figure 4.1, the first row depicts the benign sample. As indicated at the bottom of each column, the model predicted the correct class of the original sample (O) with some probability. This probability indicates how certain the model is that the image belongs to that class. The second row shows the shortest ℓ_2 AE that ARLA generated for that image during 100 episodes of training. As indicated by the text, the model predicts that the adversarial image (A) is an incorrect prediction with some probability. Recall that the predicted class must only have a higher probability than all other classes. The third row is the adversarial noise isolated from the AE, and enhanced to five times its original value for easier viewing. Distance is the ℓ_2 distance between the benign and adversarial images.

4.1.1 Large α Value

With a large α value, adversarial noise is fairly pronounced and easily visible to a human observer. Figures 4.1, 4.2, and 4.3 show examples of ARLA-generated AEs below their benign originals for the non-targeted, targeted, and hybrid attacks, respectively.

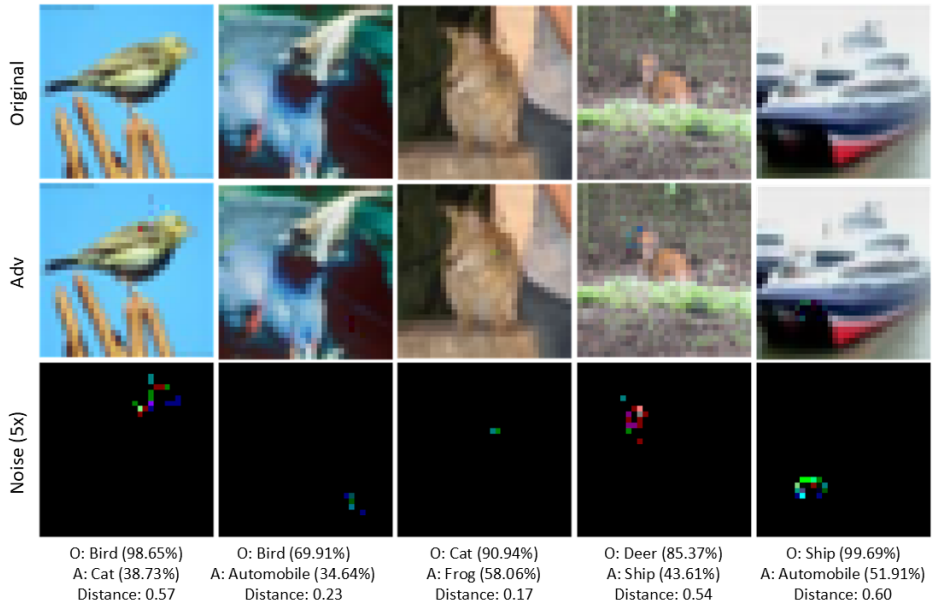

Figure 4.1. ARLA non-targeted attack against undefended WRN model. Using α=0.1, adversarial noise is pronounced. Adapted from [36].

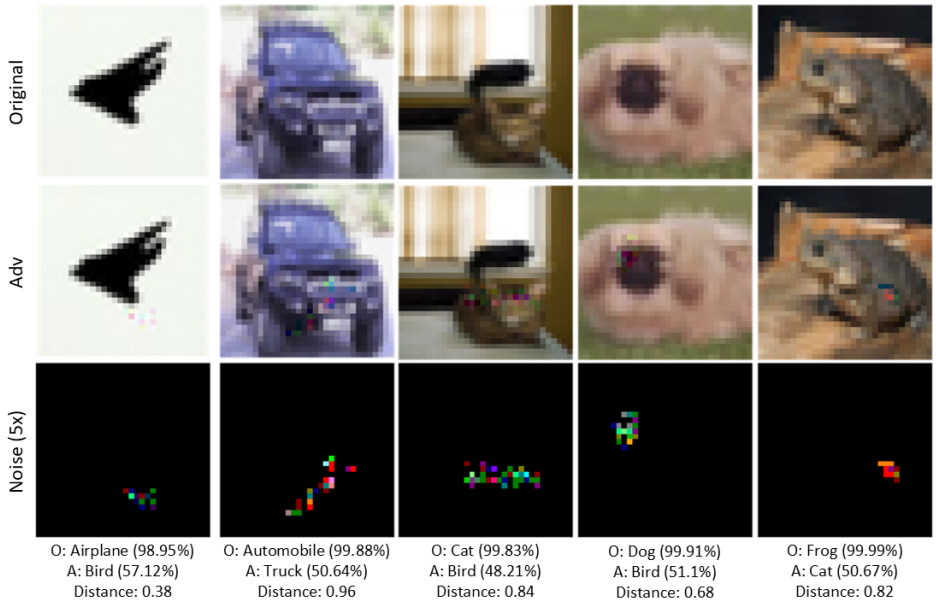

Figure 4.2. ARLA targeted attack against undefended WRN model, where α=0.1. Adapted from [36].

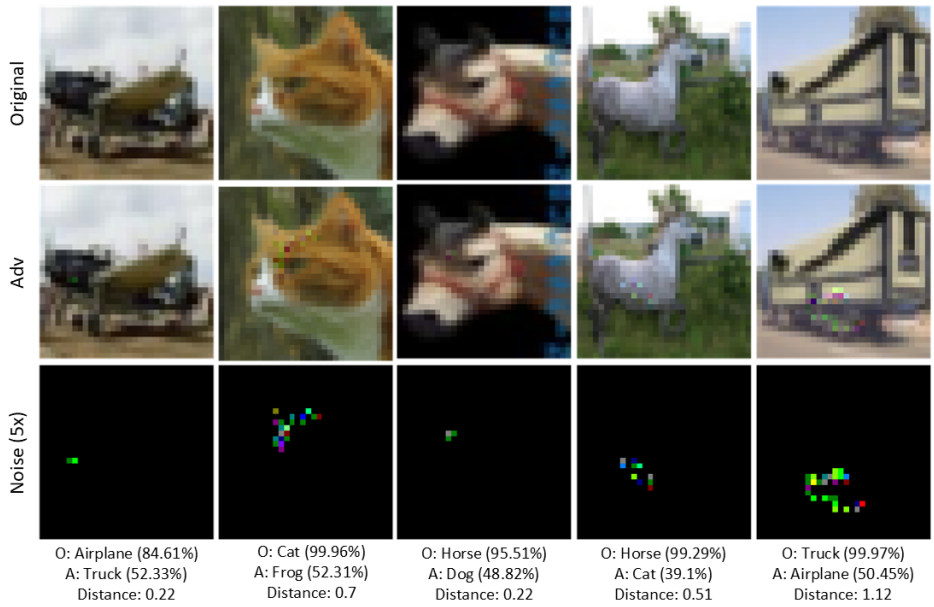

Figure 4.3. ARLA hybrid attack against undefended WRN model, where α=0.1. Adapted from [36].

4.1.2 Small α Value

By lowering the α parameter to one-third its original value, we observed a decrease in visually identifiable perturbations, and in certain cases, an increase in adversarial certainty. Figures 4.4, 4.5, and 4.6 show ARLA AEs using the non-targeted, targeted, and hybrid methodologies, respectively. By using the same benign samples from Section 4.1.1, we can see that, with a smaller α, ARLA can induce misclassification with fewer perturbations, albeit at a lower success rate. Based on these results, our other experiments exclusively used the lower α rate since it produced higher-quality AEs.

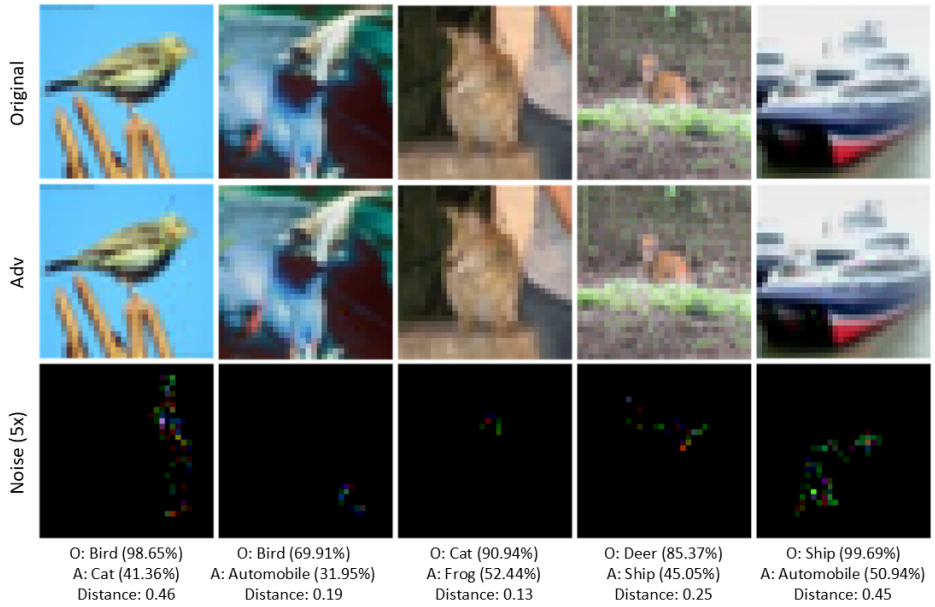

Figure 4.4. ARLA non-targeted attack against undefended WRN model. Using α=0.031, noise is constrained and difficult to detect. Adapted from [36].

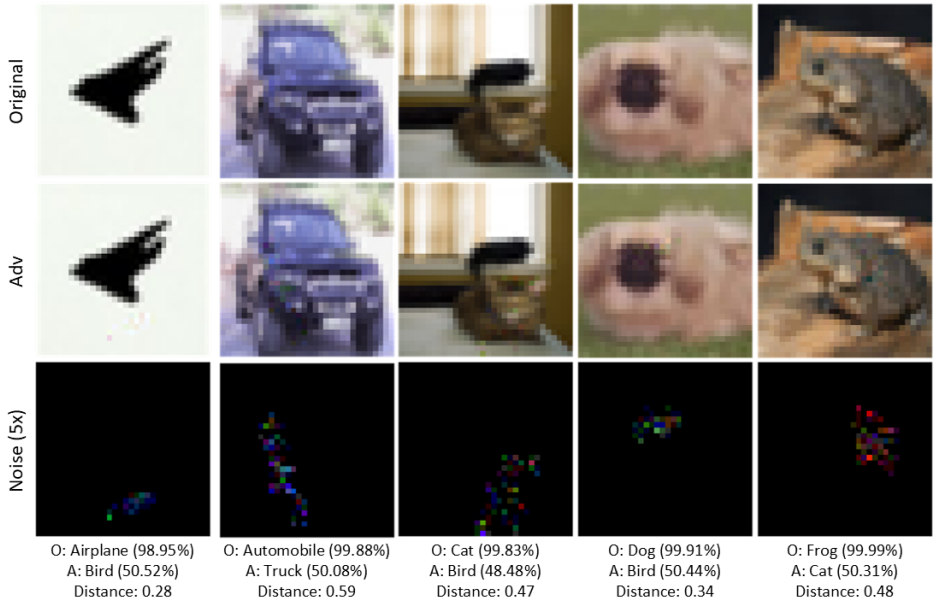

Figure 4.5. ARLA targeted attack against undefended WRN model, where α=0.031. Adapted from [36].

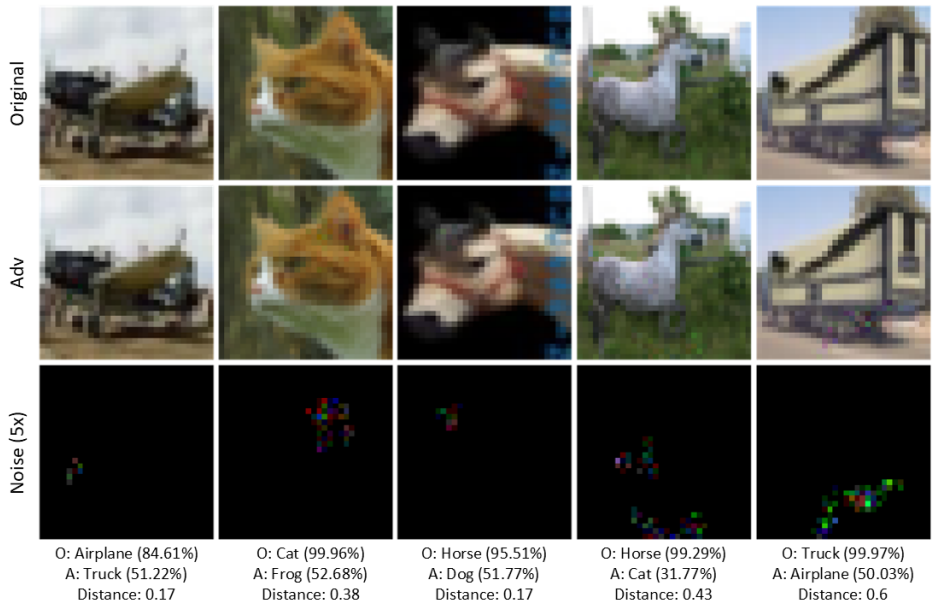

Figure 4.6. ARLA hybrid attack against undefended WRN model, where α=0.031. Adapted from [36].

4.2 PGD Adversarial Trained Model

Table 4.2 contains the performance metrics for ARLA compared to other attacks when targeting the Projected Gradient Descent (PGD) Adversarial Trained model. Again, for this experiment and those remaining, we set the α parameter to 0.031. Results show that ARLA significantly degraded the accuracy of the PGDAT model, though at a lower rate than more successful attacks, such as FGSM, PGD, and Auto. ARLA's average ℓ_2 distance was comparable to other attacks, and in some cases superior. The exception to this was the Carlini-Wagner attack, which had similar success rates to ARLA, but with considerably fewer perturbations. For black box attacks, ARLA was again a poor performer compared to other attacks. We attribute ARLA's low distance scores to the natural error of the black box model itself (8%).

The example adversaries shown in Figures 4.7, 4.8, and 4.9 demonstrate the relationship observed between predicted certainty and potential for misclassification. For example, the non-targeted adversaries in Figure 4.7 all have ℓ_2 distances commensurate with their original

Table 4.2. Comparative attack performance against the Projected Gradient Descent (PGD) Adversarial Trained model.

PGD Adversarial Trained Model

	White Box (Benign Accuracy = 0.9)						Black Box (Benign Accuracy = 0.92)					
	Non-Targeted		*Targeted*		*Hybrid*		*Non-Targeted*		*Targeted*		*Hybrid*	
	Success Rate	**Avg ℓ_2 Dist**	**Success Rate**	**Avg ℓ_2 Dist**	**Success Rate**	**Avg ℓ_2 Dist**	**Success Rate**	**Avg ℓ_2 Dist**	**Success Rate**	**Avg ℓ_2 Dist**	**Success Rate**	**Avg ℓ_2 Dist**
ARLA (α=0.031)	51%	0.456	36%	0.514	45%	0.449	11%	0.204	10%	0.223	11%	0.154
FGSM (ε=0.5)	73%	0.498	71%	0.498	N/A	N/A	31%	0.498	29%	0.498	N/A	N/A
PGD (ε=0.5)	89%	0.499	82%	0.499	N/A	N/A	38%	0.5	27%	0.5	N/A	N/A
DeepFool (ε=0.5)	91%	17.762	N/A	N/A	N/A	N/A	77%	17.934	N/A	N/A	N/A	N/A
Square (ε=0.5)	10%	0.499	N/A	N/A	N/A	N/A	13%	0.498	N/A	N/A	N/A	N/A
C&W ℓ_2 (C=0.0)	58%	0.22	51%	0.24	N/A	N/A	12%	0.167	11%	0.133	N/A	N/A
Auto Attack (ε=0.5)	89%	0.5	88%	0.5	N/A	N/A	39%	0.5	35%	0.5	N/A	N/A

certainty: greater certainty is often associated with greater ℓ_2 distance. This is not necessarily true for all benign images but has been observed as a consistent trend.

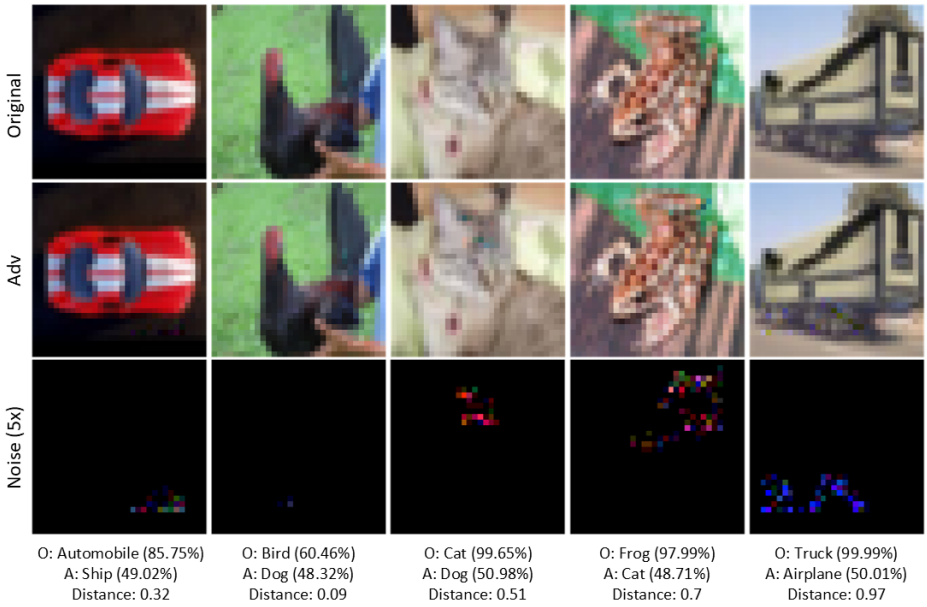

Figure 4.7. ARLA non-targeted attack against the PGDAT model, where α=0.031. As the original image with the lowest certainty, the bird requires less noise to induce misclassification. Adapted from [36].

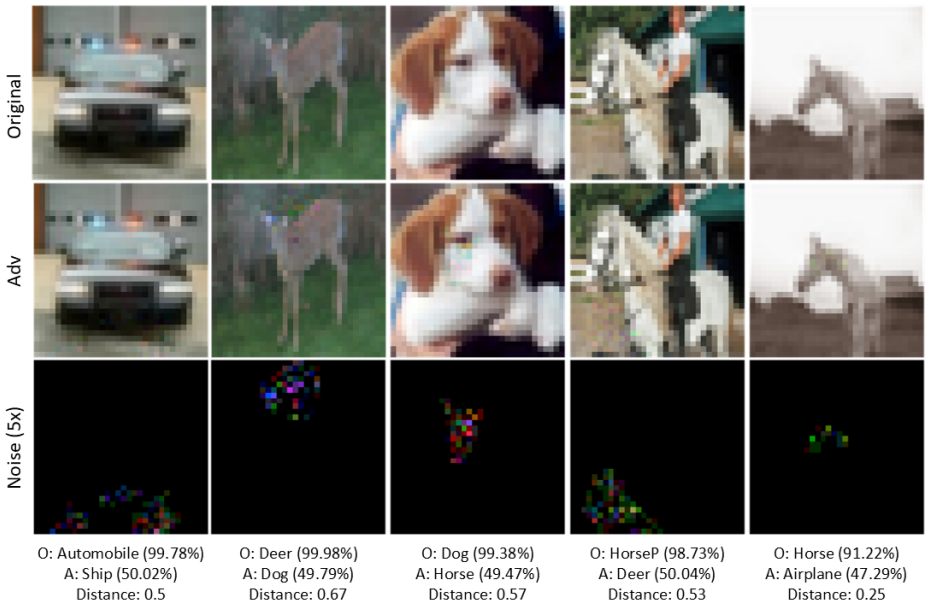

Figure 4.8. ARLA targeted attack against the PGDAT model, where α=0.031. Adapted from [36].

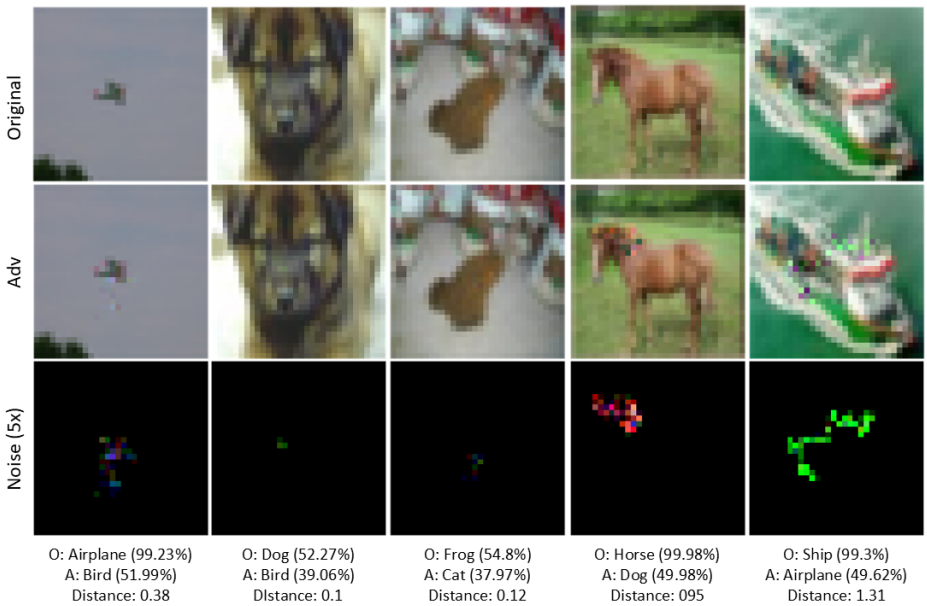

Figure 4.9. ARLA hybrid attack against the PGDAT model, where α=0.031. Adapted from [36].

4.3 Gradient Regularization Model

Table 4.3 contains the performance metrics for ARLA compared to other attacks when targeting the Gradient Regularization model. ARLA had the most difficulty attacking this model compared to other experiments. For white box metrics, ARLA's success rates were 30-50% lower than high-performing attacks. ARLA remained a poor performer in black box attacks compared to other attacks, with success rates two to three times lower than high-performing attacks.

Table 4.3. Comparative attack performance against the Gradient Regularization model.

Gradient Regularization Model												
	White Box (Benign Accuracy = 0.9)						**Black Box (Benign Accuracy = 0.91)**					
	Non-Targeted		***Targeted***		***Hybrid***		***Non-Targeted***		***Targeted***		***Hybrid***	
	Success Rate	**Avg ℓ_2 Dist**	**Success Rate**	**Avg ℓ_2 Dist**	**Success Rate**	**Avg ℓ_2 Dist**	**Success Rate**	**Avg ℓ_2 Dist**	**Success Rate**	**Avg ℓ_2 Dist**	**Success Rate**	**Avg ℓ_2 Dist**
ARLA (α=0.031)	48%	0.461	37%	0.503	43%	0.434	15%	0.311	14%	0.391	15%	0.381
FGSM (ε=0.5)	78%	0.498	77%	0.498	N/A	N/A	31%	0.499	32%	0.499	N/A	N/A
PGD (ε=0.5)	91%	0.5	87%	0.5	N/A	N/A	48%	0.5	34%	0.5	N/A	N/A
DeepFool (ε=0.5)	89%	11.647	N/A	N/A	N/A	N/A	71%	13.01	N/A	N/A	N/A	N/A
Square (ε=0.5)	18%	0.495	N/A	N/A	N/A	N/A	11%	0.496	N/A	N/A	N/A	N/A
C&W ℓ_2 (C=0.0)	81%	0.333	72%	0.401	N/A	N/A	16%	0.313	18%	0.534	N/A	N/A
Auto Attack (ε=0.5)	91%	0.5	90%	0.5	N/A	N/A	49%	0.5	47%	0.5	N/A	N/A

Figures 4.10, 4.11, and 4.12 show examples of ARLA's non-targeted, targeted, and hybrid attack, tailored to the Gradient Regularization model. Though our goal was to create minimally perturbed adversaries, we recognize that this quality is subjective to the image being altered. For example, the AE generated for the ship in Figure 4.12 has an ℓ_2 distance nearly double the hybrid attack average. However, we can still consider this a minimally perturbed AE, as the perturbations blend well with the benign image.

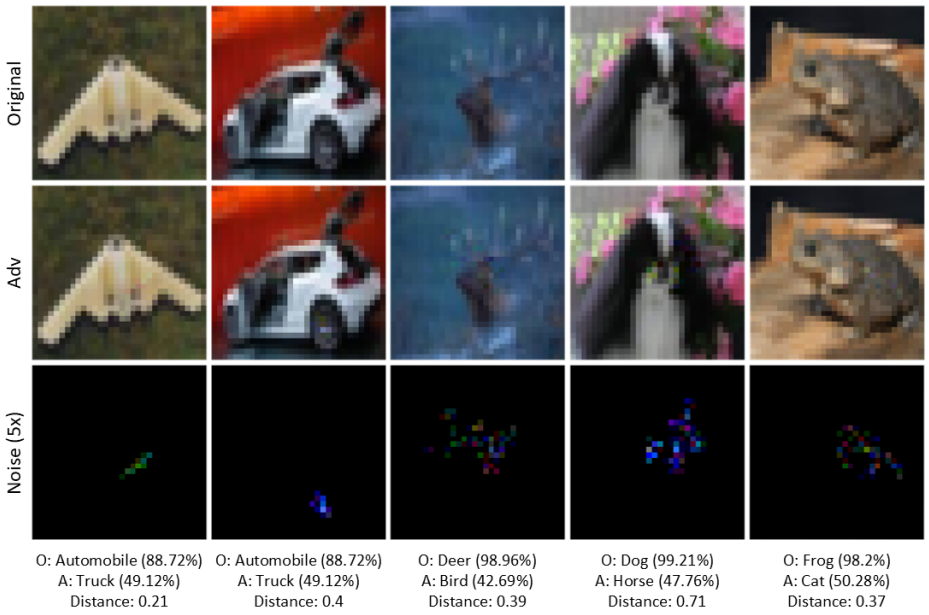

Figure 4.10. ARLA non-targeted attack against the Gradient Regularization model, where α=0.031. Adapted from [36].

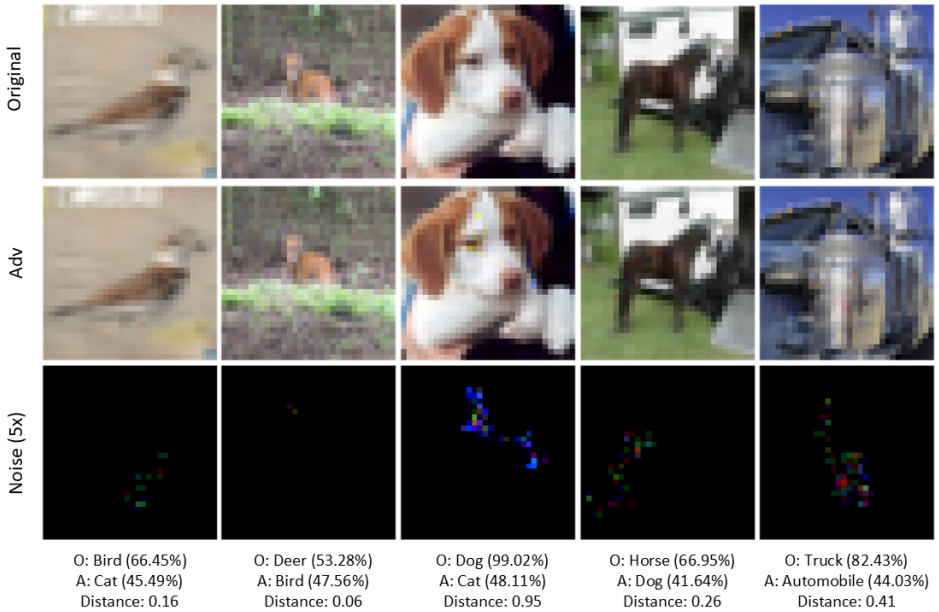

Figure 4.11. ARLA targeted attack against the Gradient Regularization model, where α=0.031. Adapted from [36].

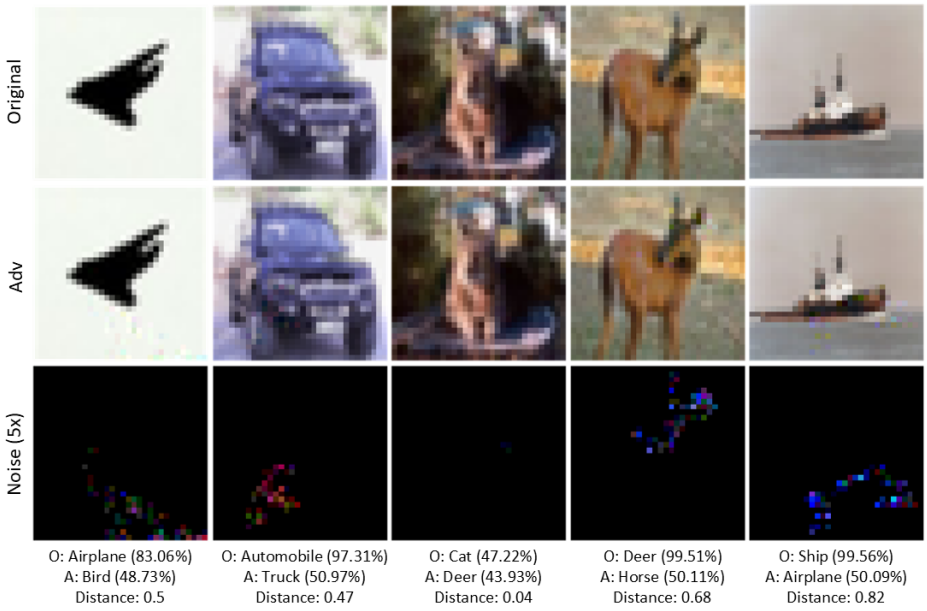

Figure 4.12. ARLA hybrid attack against the Gradient Regularization model, where α=0.031. Adapted from [36].

4.4 TRADES Model

Table 4.4 contains the performance metrics for ARLA compared to other attacks when targeting the TRADES model and highlights a performance increase for certain attacks compared to previous experiments. For white box attacks, ARLA's non-targeted attack had a considerably higher success rate than previous experiments, while keeping adversarial noise constrained. PGD and Auto still outperformed ARLA in terms of success rate, and the Carlini-Wagner attack had a matching success rate with fewer perturbations. ARLA's white box targeted attack was the worst of its three attacks, both in terms of attack success and average ℓ_2 distance. ARLA's hybrid attacked outperformed its targeted attack with a 16% higher success rate and 0.1 lower average distance. For black box attacks, ARLA remained a poor performer compared to most other attacks, despite this experiment being its second best for black box attacks.

Table 4.4. Comparative attack performance against the TRADES model.

	TRADES Model											
	White Box (Benign Accuracy = 0.87)						Black Box (Benign Accuracy 0.84)					
	Non-Targeted		*Targeted*		*Hybrid*		*Non-Targeted*		*Targeted*		*Hybrid*	
	Success Rate	**Avg ℓ_2 Dist**	**Success Rate**	**Avg ℓ_2 Dist**	**Success Rate**	**Avg ℓ_2 Dist**	**Success Rate**	**Avg ℓ_2 Dist**	**Success Rate**	**Avg ℓ_2 Dist**	**Success Rate**	**Avg ℓ_2 Dist**
ARLA (α=0.031)	64%	0.49	45%	0.568	61%	0.467	21%	0.381	19%	0.429	20%	0.312
FGSM (ε=0.5)	62%	0.499	64%	0.499	N/A	N/A	42%	0.499	38%	0.5	N/A	N/A
PGD (ε=0.5)	87%	0.5	85%	0.5	N/A	N/A	48%	0.5	42%	0.5	N/A	N/A
DeepFool (ε=0.5)	89%	14.688	N/A	N/A	N/A	N/A	72%	15.554	N/A	N/A	N/A	N/A
Square (ε=0.5)	24%	0.493	N/A	N/A	N/A	N/A	16%	0.496	N/A	N/A	N/A	N/A
C&W ℓ_2 (C=0.0)	64%	0.296	61%	0.271	N/A	N/A	21%	0.359	20%	0.268	N/A	N/A
Auto Attack (ε=0.5)	87%	0.5	87%	0.5	N/A	N/A	48%	0.5	47%	0.5	N/A	N/A

Figures 4.13, 4.14, and 4.15 show examples of ARLA's non-targeted, targeted, and hybrid attacks against the TRADES model.

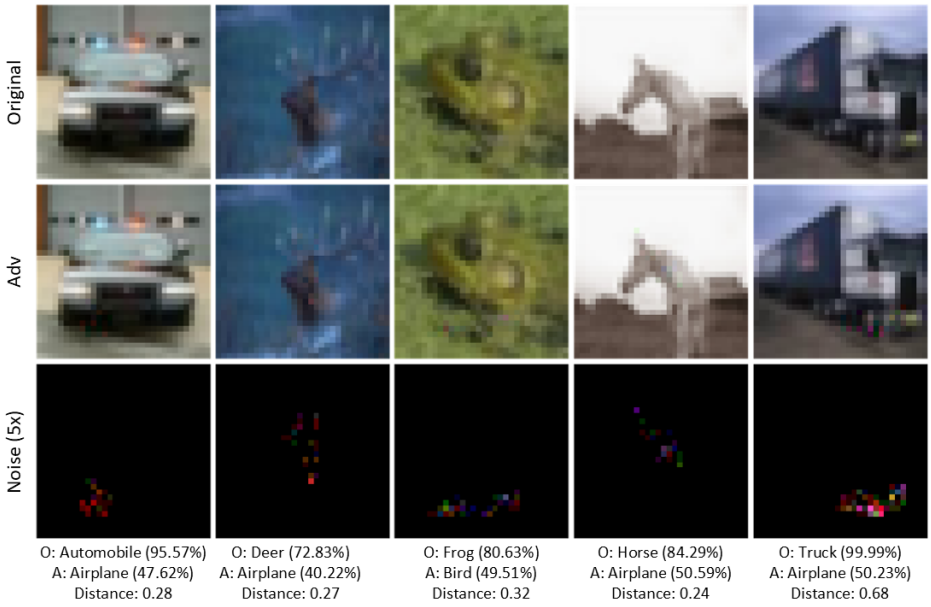

Figure 4.13. ARLA non-targeted attack against the TRADES model, where α=0.031. Adapted from [36].

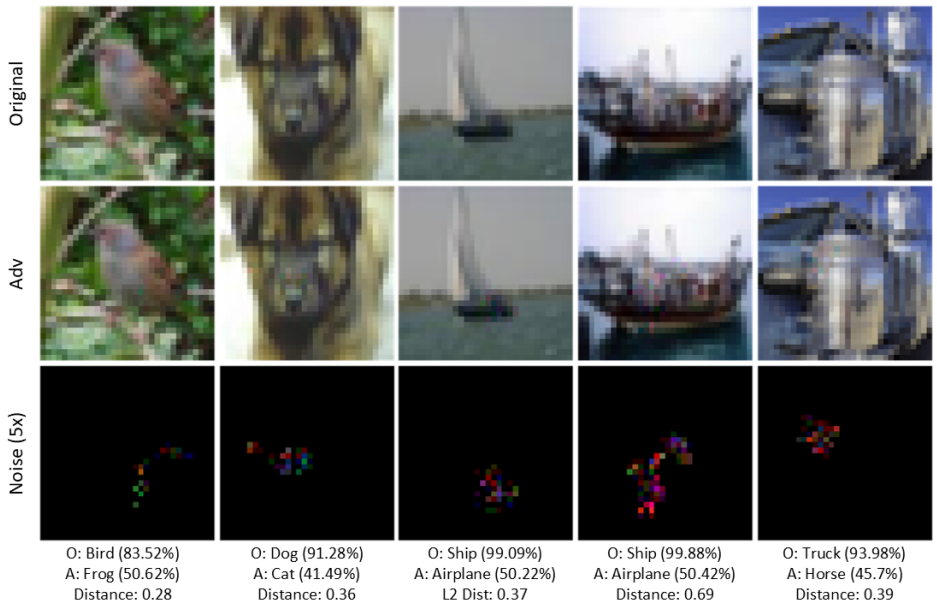

Figure 4.14. ARLA targeted attack against the TRADES model, where α=0.031. Adapted from [36].

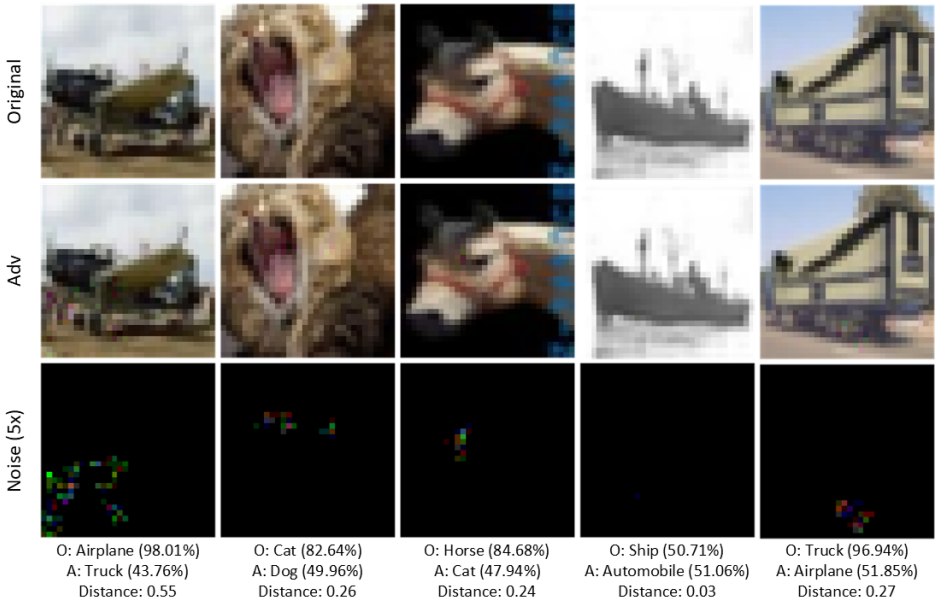

Figure 4.15. ARLA hybrid attack against the TRADES model, where α=0.031. Adapted from [36].

4.5 PadNet Model

Table 4.5 contains the performance metrics for ARLA compared to other attacks when targeting the PadNet model. For the white box non-targeted attack, ARLA was the top performing attack, with a success rate 17% higher than the next best attack (Auto) while also having a shorter average ℓ_2 distance by 26%. These results are significant because they indicate that ARLA was able to avoid the padding class in a way that other non-targeted attacks could not. In the white box targeted category, ARLA was only outperformed by PGD, though our attack had a lower ℓ_2 average. Clearly ARLA had trouble finding pixel changes that placed the AE in the decision boundary for the target class, but more work will need to be done to determine the cause. Two hypotheses exist: 1) the α value was too high and the target class was overshot, or 2) ARLA was not able to change enough pixels to find space in the target class decision boundary. ARLA's white box hybrid attack scored 5% lower than its non-targeted attack with a marginally lower ℓ_2 average.

This experiment was the most successful for ARLA's black box attacks, where ARLA won the non-targeted category by 9% with an average ℓ_2 32% shorter. In the black box targeted attack category, ARLA came in a distant second behind PGD. Unlike ARLA's white box hybrid attack, the black box hybrid attack had a higher ℓ_2 average compared to the black box non-targeted attack.

Table 4.5. Comparative attack performance against the PadNet model.

	PadNet Model											
	White Box (Benign Accuracy = 0.89)						**Black Box (Benign Accuracy = 0.9)**					
	Non-Targeted		***Targeted***		***Hybrid***		***Non-Targeted***		***Targeted***		***Hybrid***	
	Success Rate	**Avg ℓ_2 Dist**	**Success Rate**	**Avg ℓ_2 Dist**	**Success Rate**	**Avg ℓ_2 Dist**	**Success Rate**	**Avg ℓ_2 Dist**	**Success Rate**	**Avg ℓ_2 Dist**	**Success Rate**	**Avg ℓ_2 Dist**
ARLA (α=0.031)	62%	0.322	51%	0.39	57%	0.314	29%	0.252	21%	0.44	22%	0.296
FGSM (ε=0.5)	0%	N/A	0%	N/A	N/A	N/A	0%	N/A	0%	N/A	N/A	N/A
PGD (ε=0.5)	0%	N/A	60%	0.5	N/A	N/A	0%	N/A	56%	0.5	N/A	N/A
DeepFool (ε=0.5)	33%	15.628	N/A	N/A	N/A	N/A	30%	17.191	N/A	N/A	N/A	N/A
Square (ε=0.5)	21%	0.495	N/A	N/A	N/A	N/A	13%	0.493	N/A	N/A	N/A	N/A
C&W ℓ_2 (C=0.0)	11%	0.012	21%	0.023	N/A	N/A	9%	0.005	12%	0.0327	N/A	N/A
Auto Attack (ε=0.5)	45%	0.433	37%	0.419	N/A	N/A	20%	0.4	15%	0.433	N/A	N/A

Figures 4.16, 4.17, and 4.18 show examples of ARLA's non-targeted, targeted, and hybrid attack, tailored to the PadNet model.

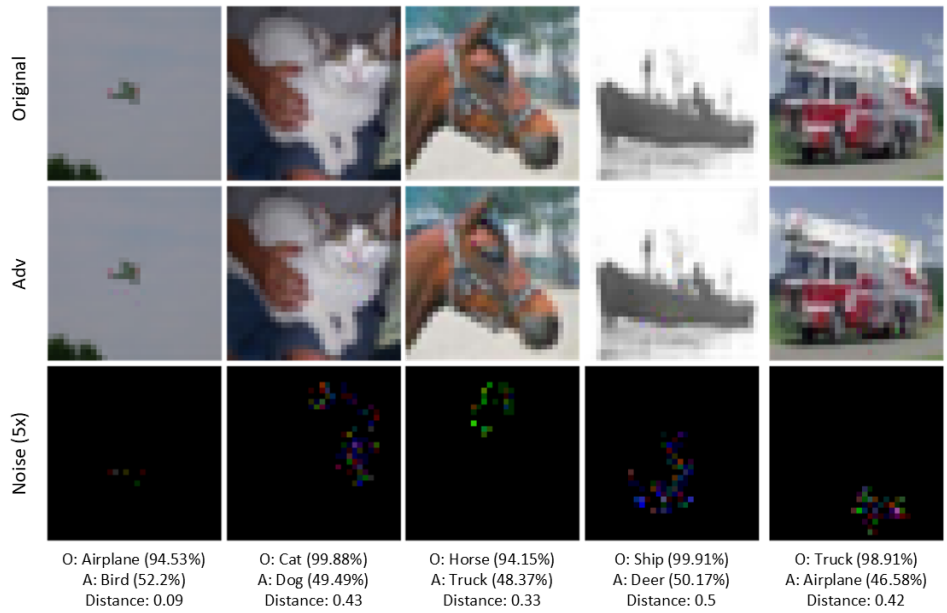

Figure 4.16. ARLA non-targeted attack against the PadNet model, where α=0.031. Adapted from [36].

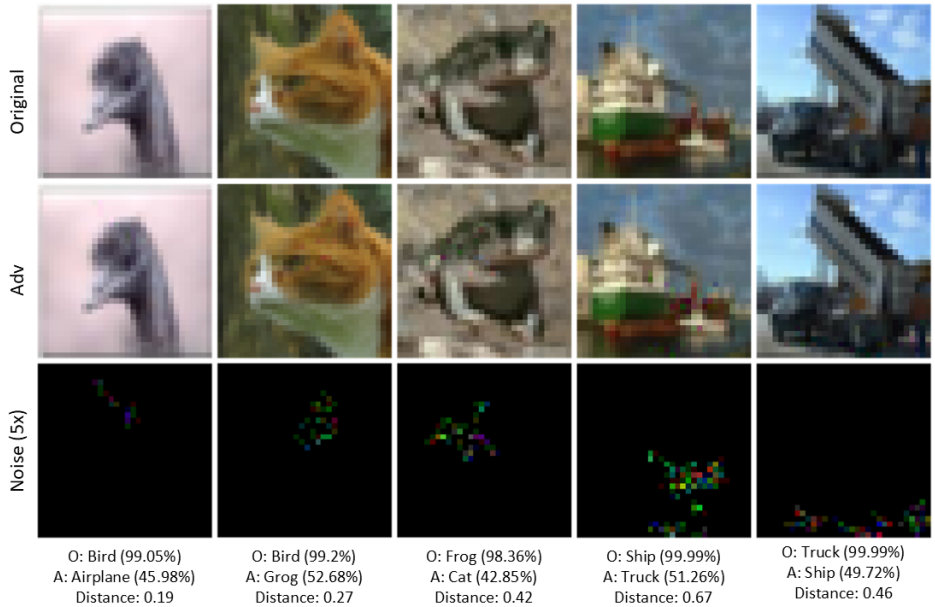

Figure 4.17. ARLA targeted attack against the PadNet model, where α=0.031. Adapted from [36].

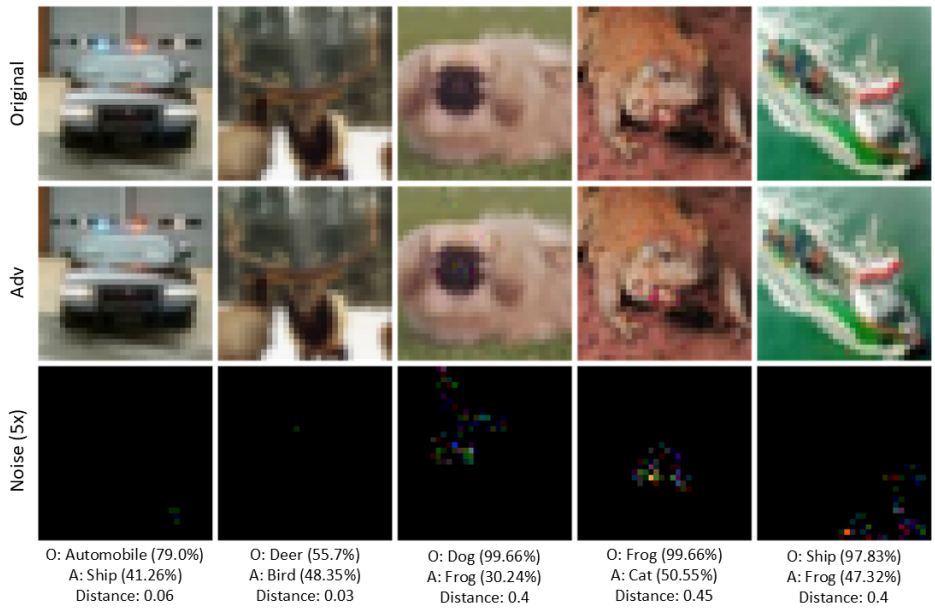

Figure 4.18. ARLA hybrid attack against the PadNet model, where α=0.031. Adapted from [36].

4.6 Summary

In this chapter, we presented the results from our experiments. For each attacked model, we discussed ARLA's performance metrics, attack success rate, and average ℓ_2 distance compared to other well-known attacks. We also showed examples of each of ARLA's attacks (non-targeted, targeted, and hybrid) for each model. ARLA proved to be a successful attack across all models, but was the best-performing attack against the PadNet defense.

CHAPTER 5:
Discussion and Future Work

In this chapter, we discuss ARLA's performance as an attack and summarize our implementation of deep Q-learning in this research space. Section 5.1 discusses ARLA as a potentially adaptive attack, one that performs well against all defenses. Section 5.2 discusses ARLA's overall performance in white box versus black box attacks. Section 5.3, discusses the performance trends of ARLA's hybrid attack compared to its non-targeted and targeted attacks. Section 5.4 examines ARLA's learning trends and analyzes bad and good examples of learning. Finally, Sections 5.5, 5.6, and 5.7 propose improvements for ARLA to be pursued in future research.

5.1 Attack Adaptability

The most significant finding of our research is that ARLA is a viable adversarial attack methodology, and that it may in fact be universally adaptive to any defense. Adversarial research is difficult due to how quickly new attacks and defenses are formed. A defense may claim to be robust against an array of attacks but is then quickly defeated by a new, stronger attack. That said, developing a new attack is often unnecessary, as almost all defenses are susceptible to existing attacks that have been modified [40]. Such attacks are considered *adaptive*, in that they can be adapted to exploit the weaknesses of a specific defense. The success of such attacks has made them the standard for evaluating whether a defense is truly robust against attack [40].

It is our continued hypothesis that an RL based attack like ARLA can be universally adaptive. Unlike static adaptive attacks, ARLA does not require humans to tune it for a specific defense. Using an RL methodology allows our attack to optimize pixel changes in the pursuit of higher rewards, so that regardless of the defense method, ARLA can learn how to exploit the vulnerabilities inherent in the model. The best evidence for this claim can be seen in our last experiment, in which we attacked the PadNet defense. While our comparison attacks were far less successful against this defense than against previous models, ARLA did remarkably well. For non-targeted white box attacks, ARLA outperformed the second

best attack by 17% with a 25% lower average ℓ_2. ARLA's targeted white box attack was its highest performing for all experiments, only outperformed by PGD, which had an average ℓ_2 28% higher than ARLA. These results indicate that ARLA was able to learn how to exploit the PadNet defense without any human input.

If ARLA is in fact universally adaptive, then it would be a prime choice for probing the weaknesses of any DNN before being deployed in the real world. As promising as these results are, more work needs to be done before this claim can be made definitively.

5.2 Attack Limitations

A caveat to ARLA's potential adaptability is that this property seems only to apply to white box attacks, not black box. Our results clearly indicate that ARLA is a poor attack method when the target model is unknown to the attacker. For all experiments, save PadNet, ARLA showed a significant performance drop relative to its white box attacks but also in contrast to the comparison attacks. In most cases, ARLA's black box success rates were two to three times lower than those of the comparison attacks. We attribute this performance gap to how ARLA is changing the picture. The intent behind ARLA is to make the minimum number of changes necessary to induce a misclassification: that is, only a small subset of the image pixels will be altered. These minimal pixel changes are clear from viewing the enhanced noise shown in our Chapter 4 figures. So if ARLA alters a pixel that has high importance in the white box model, there is no guarantee that it will have the same importance in the black box model. By contrast, most of our comparison attacks add a small amount of perturbation to a larger subset of the image pixels, if not all the pixels. This approach makes it far more likely that the most important pixels in an image will be altered enough to induce a large amount of loss, leading to a misclassification. Based on these results, we believe that future research should focus on ARLA solely as a white box methodology.

5.3 Hybrid Attack Performance

As described in Chapter 3, ARLA's hybrid attack takes a different approach from its non-targeted attack. The hybrid attack has three main features: 1) gradient maps are calculated using a target label, 2) pixel changes are made using gradient descent, and 3) success is determined by the image being classified as any incorrect class ($\hat{y} \neq y$). Table 5.1

captures the performance gaps between ARLA's non-targeted and hybrid attacks. Across all experiments, the non-targeted attack had a 3–6% higher success rate than the hybrid attack, averaging 4% higher overall. Comparing average ℓ_2 distances, the hybrid attack produced AEs that were 1.5–6.6% shorter than those produced by the non-targeted attack. Analyzing the percentage of hybrid attack AEs that were misclassified to the target label, we found that it averaged to 21% across all models. While it was not in the scope of our current research, there may be an adversarial defense for which the gradient descent approach to pixel changes makes ARLA's hybrid attack a more powerful attack methodology.

Table 5.1. Comparison metrics for ARLA hybrid attack compared to its non-targeted and targeted attacks.

	Success Rate			**Average ℓ_2 Distance**			**Target Label**
	Non-Tgt	**Hybrid**	**% Diff**	**Non-Tgt**	**Hybrid**	**% Diff**	**Hit Rate**
Undefended	56%	53%	-3%	0.427	0.399	-6.6%	34%
PGDAT	51%	45%	-6%	0.456	0.449	-1.5%	36%
Grad. Regularization	48%	43%	-5%	0.461	0.434	-5.9%	16%
TRADES	64%	61%	-3%	0.49	0.467	-4.7%	15%
PadNet	62%	57%	-5%	0.322	0.314	-2.5%	5%
Average	N/A	N/A	**-4%**	N/A	N/A	**-4.2%**	**21%**

5.4 Learning Limitations

While analyzing the adversarial examples generated by ARLA, we observed that the AE with the shortest ℓ_2 distance was often generated early in the 100 training episodes on each image. Table 5.2 displays this pattern, where on average the shortest AE was found with an ε-greedy value between 0.723 and 0.841, indicating that ARLA did not produce the shortest AE as a result of a learned behavior policy but simply as a byproduct on the way to learning a policy. Table 5.2 also shows a trend in which the best-scoring AE is generated somewhere in the middle of training, with the last AE generated using relatively few random actions. From this result we can discern that a policy is being learned, but not for the desired behavior.

To further explore this potential learning deficiency, we had ARLA train on a single environment for 5,000 episodes, at 500 steps per episode, with all other parameters similar to our experiments. In an optimal policy, the AE with the shortest ℓ_2 distance from the original

Table 5.2. ARLA's average ε-greedy values for shortest ℓ_2, best scoring, and last generated adversarial examples. LG and SM refer to the large and small α values ARLA used for pixel changes.

Average ε-Greedy Values

	Non-Target			Target			Combined		
Model	**Shortest**	**Best**	**Last**	**Shortest**	**Best**	**Last**	**Shortest**	**Best**	**Last**
Undefended LG	0.796	0.554	0.219	0.771	0.457	0.221	0.784	0.501	0.251
Undefended SM	0.801	0.576	0.164	0.78	0.435	0.137	0.781	0.486	0.158
PGDAT	0.816	0.592	0.17	0.816	0.528	0.135	0.814	0.566	0.161
Grad Reg	0.811	0.636	0.16	0.811	0.629	0.243	0.812	0.647	0.188
TRADES	0.735	0.572	0.093	0.723	0.446	0.151	0.747	0.548	0.124
PadNet	0.841	0.632	0.173	0.796	0.517	0.227	0.821	0.536	0.178

image should also have the highest score. This was not the case for ARLA, and we can refer to Figure 5.1 to analyze an example of sub-optimal learning. Here, looking at the top left image, we see that in 5,000 episodes, ARLA generated 2,613 adversaries. The shortest AE (center image) was generated using actions that were randomly selected for 95% of the steps played, which using our ε-greedy policy means that it was generated around episode 250. This shortest AE should have been the solution that ARLA was attempting to optimize. In the top right image, which is the final adversary generated, we can observe that for this environment, ARLA learned that the best policy was simply to change the red and green pixels. The first rewards plot, "Original Rewards," plots all the rewards for the 5,000 episodes. However, this plot does not give us a clear idea about which rewards are associated with AEs, so the plot below, "AE vs Non-AE Rewards," color codes successful and unsuccessful episodes to clearly identify trends between the two. As the ε-greedy value approached the minimum value of 0.01, ARLA did not have a strong policy, and its performance, defined as high rewards, tapered off. Once ARLA begins to make more predictions per episode (50+%), bad experiences began to fill its replay memory, making it more unlikely that ARLA would be able to sample good experiences for learning. The "Normalized Rewards" and "AE Only Rewards" plots support this analysis. If we wanted rewards to increase over the course of training, then optimal behavior would have the ℓ_2 distances decreasing at the same rate. The distance plots, bottom-middle and bottom-right, show that the opposite happened, before ARLA's online model experienced catastrophic forgetting. The bottom-middle plot in particular supports our analysis that ARLA only learned to bias pixel changes, leaving

agent movement to random actions.

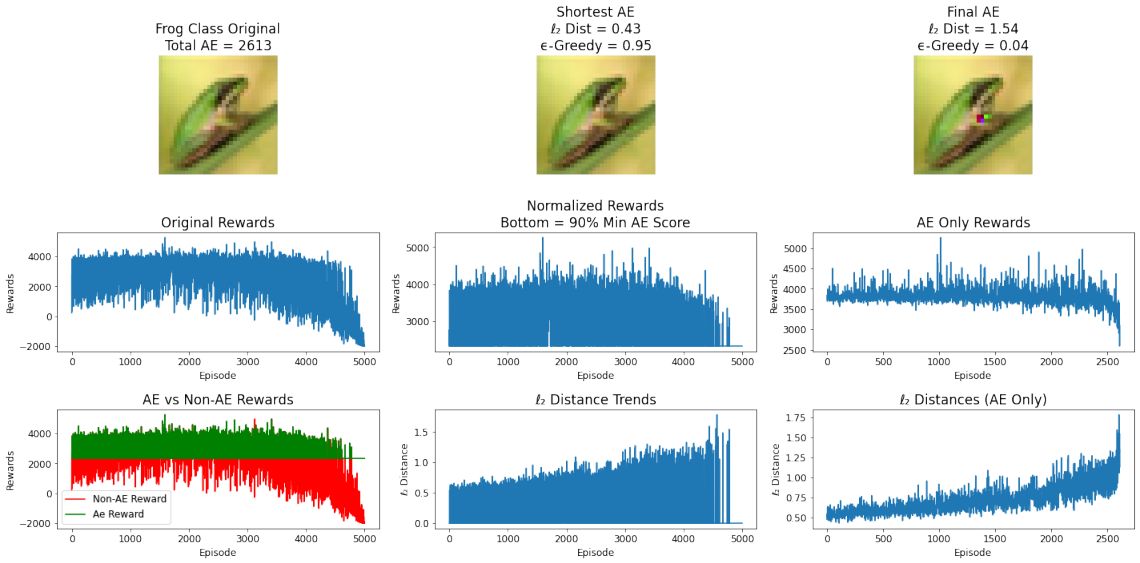

Figure 5.1. Example of ARLA being unable to learn a good behavior policy, leaving it unable to figure out this frog. Adapted from [36].

During this analysis, we did find environments where ARLA was able to learn good behavior, if not optimal. Figure 5.2 shows one such example. Here we can see that the shortest AE was produced early, but the final AE was generated using the minimal ε-greedy value allowed per our policy, 0.01. Distance trends are still not good, but we can observe a steep decrease towards the end as rewards begin to spike. Unfortunately, the shortest AE did not receive the highest reward, but the learned policy is much closer to this ideal than in our previous example of the frog.

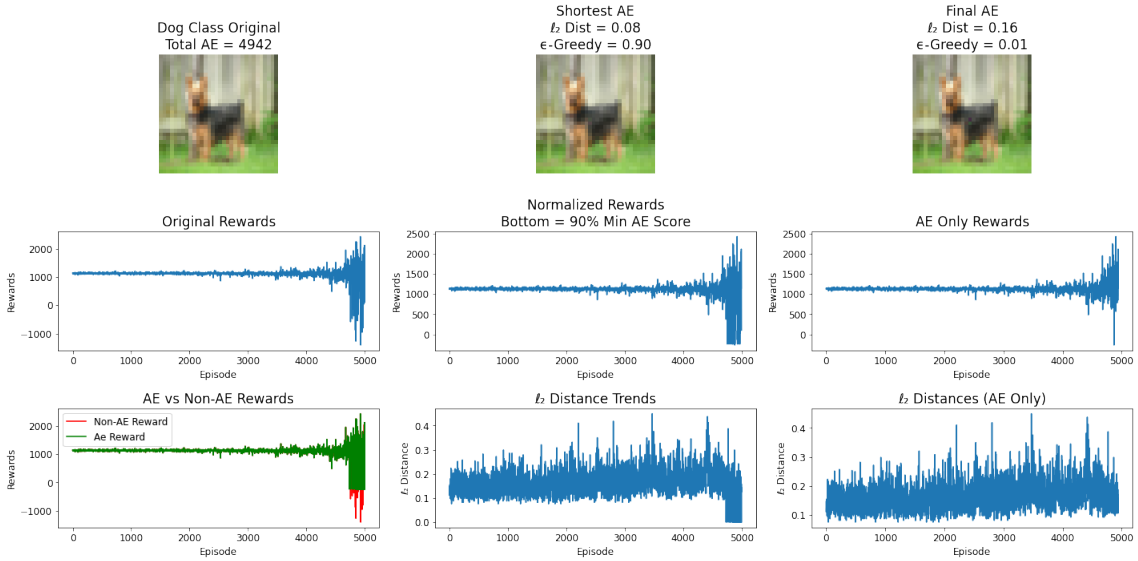

Figure 5.2. An example of ARLA learning a good policy for this benign example of the dog class. Adapted from [36].

Our results have shown that ARLA can learn, but at this stage of development, it is far too inconsistent. To improve its performance, we propose three key areas of future research in the sections below.

5.5 Future Work —Reward Function

Since the reward function is what helps an agent discern good versus bad actions in the pursuit of a goal, the majority of future research should be spent developing a better reward function for ARLA. To reiterate our goal for rewards, the highest-scoring episode should also be the one that produces the AE with the shortest ℓ_2 distance from the benign original. The reward function used in our research came up well short of that goal and allowed ARLA to disassociate high-reward episodes from those where minimally perturbed AEs were generated. In rare cases, ARLA was able to learn a policy where the highest-scoring episodes did not produce an adversarial example at all. The challenge that this task represents should not be underestimated, because while the concept is simple (short_ℓ_2 + high_loss = great job!), formulating a mathematically complete function is no small task.

That said, one potential improvement was immediately evident in the analysis of our results,

and could be easily implemented. In our current function, loss ratios that fall below our positive reward threshold are simply given a static reward of -4, while high loss ratios are given a dynamic reward based on the amount of loss they induced. Using a static negative reward is a mistake because it can lead to a behavior that we refer to as *pixel mining*. Pixel mining occurs when ARLA makes a change to a pixel that causes both a large amount of error and flips the sign of that pixel's gradient. If ARLA changes that pixel again, it will be in the opposite direction, and with the gradient returning to its original value and ARLA receiving a reward of -4. However, that negative reward does not matter much, since changing that pixel a third time will generate the same large reward it experienced the first time. Repeating this cycle, ARLA can accumulate large positive rewards that are only offset marginally by much smaller negative rewards. Instead, negative rewards should also be dynamic, canceling out any positive reward just received which would disincentivize pixel mining.

5.6 Future Work —Prioritized Experience Replay

In this research, we used a deque, which is a very simple data structure, to store a finite number of previous experiences, which allowed newer experiences to replace older ones once the memory reached capacity (first in, first out). Experiences were stored as tuples with no way to differentiate whether an experience was meaningful, such that it would lead to faster learning or not. As such, experiences were sampled uniformly from the memory, which made it impossible to know whether the agent is learning meaningful transitions. This problem was compounded by the fact that many pixels in an image may not actually be that important for inducing loss, which allowed the memory to become saturated with poor experiences. This in turn led to a situation where, as the ε-greedy value decayed, ARLA's behavior policy made consistently poor predictions and filled the memory with more poor experiences. Any good experiences caused through random action were eventually pushed out of memory to make room for newer predicted experiences. An example of this behavior is the frog from Figure 5.1.

Instead of using this basic experience replay, we propose that future work incorporate *prioritized experience replay*. Introduced by Schaul et al. [41] in 2015, experiences are sampled not at random, but based on their *priority*, defined by a transitions temporal

difference error (TD-error). Using TD-error as a general approximation for how much an agent can learn from a specific state transition, meaningful experiences can be replayed more often. For each training step, a new TD-error is calculated, and priorities are updated for all stored experiences. Research has shown that of the improvements that can be made to DQN, prioritized experience replay is one of the most valuable in terms of increased agent performance [42].

5.7 Future Work —Environment

An issue that we have considered for future research is how ARLA will scale to images with larger dimensions. For larger images, the time needed to sufficiently explore the environment will exponentially increase. While this research attempted to address the exploration-exploitation problem by determining the best starting location based on gradient map values, that approach may not be sufficient for ImageNet [43], where the average dimensions are 469x387 pixels. We propose cropping the image to the most important pixels, which might help constrain ARLA by presenting a smaller portion of the image as the observable state, e.g., the 16x16 portion of the image with the highest gradients. By altering the environment this way, we hypothesize that ARLA will be able to use any size image to learn how to generate strong AEs.

5.8 Summary

In this chapter we explained that ARLA is a viable attack methodology capable of generating minimally perturbed adversarial examples and that it shows potential for being adaptable to any defense methodology. We outlined the performance discrepancy between white box and black box attacks and concluded that ARLA is best suited as a white box attack. We described the performance differences between ARLA's hybrid attacks compared to its non-targeted and targeted attacks and concluded that the hybrid attack is a slightly worse-performing non-targeted attack with better ℓ_2 distance metrics. We analyzed ARLA's capacity to learn a good behavior policy, and concluded that ARLA is not learning the desired behavior, leaving more work to be done. Finally, we recommended the next steps of this research and outlined three proposed lines of research.

CHAPTER 6:
Conclusion

For the USN and DOD to successfully transition current systems and processes into highly-effective human-machine teams, senior leadership will need to field equipment that utilizes deep neural networks. For critical applications, the trustworthiness and reliability of these networks is paramount. While integrating ML and DNNs can make a transformative difference in performance, there has been a bevy of research showing that they are widely susceptible to perturbed or noisy data. This is especially true for DNNs used in image recognition, where adversarial examples pose a substantial risk. As we discussed in Chapter 1, degraded performance in systems that use image recognition can have severe and potentially life threatening consequences. For this reason, DNNs used in military systems will need to use robust testing protocols to identify their vulnerability to adversarial attacks. In this thesis we presented a novel reinforcement learning–based attack, ARLA, that could be used for such a purpose.

Unlike the comparison attacks we used, ARLA was able to successfully attack all five CIFAR-10 models used in this research, four of which were defended in some manner. This is strong evidence that ARLA is a universally adaptive attack, able to exploit a wide variety of defenses, which would make it a highly useful tool for evaluating DNNs in development. Future work should test this statement by incorporating a more diverse set of defensive techniques. We also found that the behavior policies that ARLA learns are inconsistent, and at times contrary to the goal of creating AEs. Future work needs to be done to stabilize ARLA and optimize the behavior policy being learned, through a combination of a better reward function and prioritized experience replay.

As the DOD incorporates cutting-edge machine learning technologies into future systems, its leaders should be realistic about the benefits and the threats. No system can be perfectly hardened against malicious behavior, and deep neural networks are no exception. To build a better defense, continuous work will need to be done to find new exploitable weaknesses. It is our hope that our research plays some small part in that journey.

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

[1] U.S. Department of Defense, *Fact Sheet: 2022 National Defense Strategy*, 2022 [Online]. Available: https://media.defense.gov/2022/Mar/28/2002964702/-1/-1/1/NDS-FACT-SHEET.PDF?source=GovDelivery

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, 1st ed. Cambridge, MA, USA: The MIT Press, 2016.

[3] Y. Wang and W. Xu, "Leveraging Deep Learning with LDA-based Text Analytics to Detect Automobile Insurance Fraud," *Decision Support Systems*, vol. 105, Jan 2018 [Online]. doi: https://doi.org/10.1016/j.dss.2017.11.001.

[4] K. Rayavarapu and K. K. Krishna, "Prediction of Cervical Cancer using Voting and DNN Classifiers," *2018 International Conference on Current Trends towards Converging Technologies (ICCTCT)*, 2018 [Online]. doi: 10.1109/ICCTCT.2018.8551176.

[5] Y. Dong, "An Application of Deep Neural Networks to the In-flight Parameter Identification for Detection and Characterization of Aircraft Icing," *Aerospace Science and Technology*, vol. 77, Jun 2018 [Online]. doi: https://doi.org/10.1016/j.ast.2018.02.026.

[6] U.S. Department of Defense, "DOD Adopts Ethical Principles for Artificial Intelligence," 2020 [Online]. Available: https://www.defense.gov/News/Releases/Release/Article/2091996/dod-adopts-ethical-principles-for-artificial-intelligence/

[7] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: The MIT Press, 2018.

[8] V. Mnih, K. Kavukcuoglu, A. G. David Silver, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing Atari with Deep Reinforcement Learning," 2013 [Online]. Available: http://arxiv.org/abs/1312.5602

[9] 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.*, "Human-level Control through Deep Reinforcement Learning," *Nature*, vol. 518, no. 7540, 2015 [Online]. doi: https://doi.org/10.1038/nature14236.

[10] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the Game of Go with Deep Neural Networks and Tree Search," *Nature*, vol. 529, no. 7587, 2016 [Online]. doi: https://doi.org/10.1038/nature16961.

[11] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the Game of Go without Human Knowledge," *Nature*, vol. 550, no. 7676, 2017 [Online]. doi: https://doi.org/10.1038/nature24270.

[12] A. Gron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. Sebastapol, CA, USA: O'Reilly Media, Inc.

[13] M. Iliadis, L. Spinoulas, and A. K. Katsaggelos, "Deep Fully-connected Networks for Video Compressive Sensing," *Digital Signal Processing*, vol. 72, Jan 2018 [Online]. doi: https://doi.org/10.1016/j.dsp.2017.09.010.

[14] Y. LeCun, C. Cortes, and C. Burges, "MNIST Handwritten Digit Database," *AT&T Labs*, vol. 2, 2010 [Online]. Available: http://yann.lecun.com/exdb/mnist

[15] M. Bahi and M. Batouche, "Deep Learning for Ligand-Based Virtual Screening in Drug Discovery," *2018 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS)*, pp. 1–5, 10 2018 [Online]. doi: 10.1109/PAIS.2018.8598488.

[16] K. Fukushima, "Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Visual Pattern Recognition Unaffected by Shift in Position," *Biological Cybernetics*, vol. 36, 1980 [Online]. doi: https://doi.org/10.1007/BF00344251.

[17] S. Saha, "A Comprehensive Guide to Convolutional Neural Networks - the ELI5 Way," Towards Data Science, Nov 2022 [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[18] H. Yingge, I. Ali, and K.-Y. Lee, "Deep Neural Networks on Chip - A Survey," *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Feb 2020 [Online]. doi: 10.1109/BigComp48618.2020.00016.

[19] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing Properties of Neural Networks," unpublished, 2013 [Online]. Available: https://arxiv.org/abs/1312.6199

[20] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," unpublished, 2014 [Online]. Available: https://arxiv.org/abs/1412.6572

[21] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square Attack: A Query-Efficient Black-Box Adversarial Attack via Random Search," *Computer Vision – ECCV 2020*, Aug 2020 [Online]. doi: https://doi.org/10.1007/978-3-030-58592-1_29.

[22] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards Deep Learning Models Resistant to Adversarial Attacks," unpublished, 2017 [Online]. Available: https://arxiv.org/abs/1706.06083

[23] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: a Simple and Accurate Method to Fool Deep Neural Networks," unpublished, 2015 [Online]. Available: https://arxiv.org/abs/1511.04599

[24] N. Carlini and D. Wagner, "Towards Evaluating the Robustness of Neural Networks," unpublished, 2016 [Online]. Available: https://arxiv.org/abs/1608.04644

[25] F. Croce and M. Hein, "Reliable Evaluation of Adversarial Robustness with an Ensemble of Diverse Parameter-free Attacks," in *Proceedings of the 37th International Conference on Machine Learning*, 2020 [Online]., pp. 2206–2216. Available: https://proceedings.mlr.press/v119/croce20b.html

[26] X. Cao and N. Z. Gong, "Mitigating Evasion Attacks to Deep Neural Networks via Region-based Classification," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, 2017 [Online]. Available: https://doi.org/10.1145/3134600.3134606

[27] A. Ross and F. Doshi-Velez, "Improving the Adversarial Robustness and Interpretability of Deep Neural Networks by Regularizing Their Input Gradients," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018 [Online]. doi: 10.1609/aaai.v32i1.11504.

[28] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. E. Ghaoui, and M. Jordan, "Theoretically Principled Trade-off between Robustness and Accuracy," in *Proceedings of the 36th International Conference on Machine Learning*, 2019 [Online]. Available: https://proceedings.mlr.press/v97/zhang19p.html

[29] A. Barton *et al.*, "Defending Neural Networks Against Adversarial Examples," Ph.D. dissertation, Dept. of Comp. Sci., University of Texas Arlington, Arlington, TX, USA, 2018.

[30] A. Haji Hosseinloo, A. Ryzhov, A. Bischi, H. Ouerdane, K. Turitsyn, and M. A. Dahleh, "Data-driven Control of Micro-climate in Buildings: An Event-triggered Reinforcement Learning Approach," *Applied Energy*, Nov 2020 [Online]. doi: https://doi.org/10.1016/j.apenergy.2020.115451.

[31] A. P. Pope, J. S. Ide, D. Micovic, H. Diaz, D. Rosenbluth, L. Ritholtz, J. C. Twedt, T. T. Walker, K. Alcedo, and D. Javorsek, "Hierarchical Reinforcement Learning for Air-to-Air Combat," unpublished, 2021 [Online]. Available: https://arxiv.org/abs/2105.00990

[32] "Pac-Man," Arcade, NAMCO, 1980.

[33] H. van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-learning," unpublished, 2015 [Online]. Available: https://arxiv.org/abs/1509.06461

[34] L. Taek, "Reinforcement Learning Key Paper - Double DQN," TISTORY, Sep 2020 [Online]. Available: https://taek-l.tistory.com/36

[35] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling Network Architectures for Deep Reinforcement Learning," unpublished, 2015 [Online]. Available: https://arxiv.org/abs/1511.06581

[36] A. Krizhevsky, G. Hinton *et al.*, "Learning Multiple Layers of Features from Tiny Images," University of Toronto, Toronto, ON, Canada, Tech. Rep., 2009 [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[37] OpenAI, "OpenAI Gym," Accessed January 8, 2023 [Online]. Available: https://www.gymlibrary.dev/

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

[39] S. Zagoruyko and N. Komodakis, "Wide Residual Networks," arXiv preprint, 2016 [Online]. Available: https://arxiv.org/abs/1605.07146

[40] F. Tramèr, N. Carlini, W. Brendel, and A. Madry, "On Adaptive Attacks to Adversarial Example Defenses," arXiv preprint, 2020 [Online]. Available: https://arxiv.org/abs/2002.08347

[41] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized Experience Replay," unpublished, 2015 [Online]. Available: https://arxiv.org/abs/1511.05952

[42] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining Improvements in Deep Reinforcement Learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018 [Online]. doi: https://doi.org/10.1609/aaai.v32i1.11796.

[43] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, 2015 [Online]. doi: https://doi.org/10.1007/s11263-015-0816-y.

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California