NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

A* ATTACK: A NOVEL PATH-FINDING APPROACH TO ADVERSARIAL EXAMPLES

by

Christopher D. Clark

September 2023

Thesis Advisor:	Armon C. Barton
Second Reader:	Geoffrey G. Xie

THIS PAGE INTENTIONALLY LEFT BLANK

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

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

4. TITLE AND SUBTITLE
A* ATTACK: A NOVEL PATH-FINDING APPROACH TO ADVERSARIAL EXAMPLES

5. FUNDING NUMBERS

6. AUTHOR(S) Christopher D. Clark

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)

This paper presents a novel approach to exploiting a key vulnerability of deep neural networks (DNNs) to adversarial examples with a focus on the black-box machine learning as a service (MLaaS) environment. We introduce A* Attack, a unique adversarial example attack that leverages the A* Search algorithm to find adversarial perturbations. This innovative approach is designed to overcome the challenges of both excessive model queries in decision- and score-based attacks and the limitations of transferability from white-box attacks. The A* Attack demonstrates competitive performance in the white-box setting and sets a new standard in the decision-based black-box setting, achieving high attack success rates with minimal queries. This represents a significant advancement in the field, offering a new approach to the black-box attack method. This paper provides a competitive evaluation of the A* Attack on CIFAR10 and ImageNet, comparing its performance against other leading attacks and defenses.

14. SUBJECT TERMS deep neural network, DNN, artificial intelligence, A*, A* search, A-Star, adversarial example, MLaaS, machine learning as a service, black-box attack			**15. NUMBER OF PAGES** 67 **16. PRICE CODE**
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** 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.

A* ATTACK: A NOVEL PATH-FINDING APPROACH TO ADVERSARIAL EXAMPLES

Christopher D. Clark
Captain, United States Marine Corps
BS, Ohio State University, 2016

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2023

Approved by: Armon C. Barton
Advisor

Geoffrey G. Xie
Second Reader

Gurminder Singh
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This paper presents a novel approach to exploiting a key vulnerability of deep neural networks (DNNs) to adversarial examples with a focus on the black-box machine learning as a service (MLaaS) environment. We introduce A* Attack, a unique adversarial example attack that leverages the A* Search algorithm to find adversarial perturbations. This innovative approach is designed to overcome the challenges of both excessive model queries in decision- and score-based attacks and the limitations of transferability from white-box attacks. The A* Attack demonstrates competitive performance in the white-box setting and sets a new standard in the decision-based black-box setting, achieving high attack success rates with minimal queries. This represents a significant advancement in the field, offering a new approach to the black-box attack method. This paper provides a competitive evaluation of the A* Attack on CIFAR10 and ImageNet, comparing its performance against other leading attacks and defenses.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AI	artificial intelligence
ART	Adversarial Robustness Toolbox
AT	adversarial training
CNN	convolutional neural network
CIFAR10	Canadian Institute For Advanced Research-10 Classes
CW	Carlini and Wagner
DNN	deep neural networks
GR	gradient regularization
HSJA	HopSkipJump Attack
IEEE	Institute of Electrical and Electronics Engineers
ML	machine learning
MLaaS	machine learning as a service
PGD	Projected Gradient Descent
ResNet	Residual Network
TRADES	Tradeoff-inspired adversarial defense via surrogate-loss minimization

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

Deep neural networks (DNNs) are increasingly used for military and commercial applications in classification, prediction, and decision-making [1], transforming how organizations understand data and automate complex processes. In 2004, a study by Dalvi et al. [2] explored the concept of adversarial classification, that an adversary could manipulate inputs to a deep learning model by introducing perturbations that evade a binary classifier. In 2014 Szegedy et al. [3] published "Intriguing Properties of Neural Networks," demonstrating that neural networks could be tricked by slightly altering the input, called adversarial examples, into misclassification.

The exploration of adversarial examples is an essential component for the development and verification of robustness DNN models. Easily tricked DNNs that are deployed in critical infrastructure and military systems could have extremely negative consequences.

1.1 Machine Learning

Machine learning (ML) is a field within artificial intelligence (AI) that enables computer models to be trained on various datasets to perform tasks without explicitly being programmed to do so. Some common applications for ML are image detection and classification [4], natural language processing [5], and in healthcare systems [6].

Convolutional neural networks (CNNs) are often used for the purpose of image classification. CNNs are a deep learning architecture that was designed primarily for image recognition and classification tasks [7]. This type of neural network is composed of several convolutional layers that work as filters to detect various portions of an image. Variations on the CNN architecture have resulted in significant breakthroughs in computer vision and image classifications [8].

1.2 Reliability of Neural Networks

The vulnerability posed by adversarial examples leads to concerns about the reliability and trustworthiness of DNNs, especially where AI systems are intended for critical systems or infrastructure, used in autonomous weapons systems, or play a role in national security decision making. In a paper by Wise and Plested [9] adversarial examples were used to create military camouflage capable of hiding military assets from observation by aircraft cameras that relied on computer vision processing by CNNs.

1.3 Problem Statement

The A* Search algorithm is a path-finding algorithm used in graph traversal to identify the shortest possible path from a start state to an end state [10]. This thesis explores the use a shortest-path algorithm in the adversarial example domain.

Hypothesis: A shortest-path search agent can be used to find minimal perturbations that are able to fool a neural network into misclassifying an object, while appearing unchanged to a human observer.

Research Questions:

1. Can A* Search be used to find optimal adversarial examples efficiently?
2. Does A* Search find adversarial examples that are closer to the original compared to L2 distance than other attacks?
3. Can A* Search adversarial examples compromise other defenses better than other attacks?

1.4 Benefits of the Study

To the author's knowledge, we are the first to leverage the A* Search algorithm to find adversarial perturbation—called A* Attack. With minor hyperparameter tuning, the A* Attack achieves competitive performance under the white-box setting and state-of-the-art performance under the decision-based black-box setting in which we show top attack success rates with minimal queries compared to other top query-based attacks in literature. We evaluate A* Attack on CIFAR10 and ImageNet and compare the performance against other state-of-the-art attacks including Carlini & Wagner (CW) [11], Projected Gradient

Descent (PGD) [12], AutoAttack [13], and HopSkipJump Attack (HSJA) [14], and defenses including adversarial training (AT) [15], gradient regularization (GR) [16], TRADES [17], RobustBench [18], and AI Guardian [19].

1.5 Terms and Definitions

There are a number of terms and definitions that are unique to ML- and AI-based attacks. The following provides a brief description of each as it pertains to this thesis:

- A* Search: A popular algorithm used for finding the best path, commonly used in map and gaming applications for route finding.
- Adversarial attack: An attack that uses adversarial examples that cause a neural network to misclassify on object.
- Adversarial perturbation: In our case, a small change to the input image pixels to craft adversarial examples
- Adversarial training: Training performed on neural networks to make them more robust against adversarial attacks.
- Targeted attack: The attack intends to craft an adversarial example that will cause the model to classify the output as a specified class.
- Untargeted attack: The final classification can be anything other than the correct class.
- Projected Gradient Decent (PGD): An iterative application gradient based attack that induces perturbations at multiple steps while limiting perturbations to a maximum bound.
- Deep Fool: Another iterative approach that induces perturbations in an image that have the largest effect on the classification until the model misclassifies.
- White-Box: An attack with complete knowledge and access to a model's architecture, parameters, and gradient.
- Black-Box: An attack that has no knowledge of a model's architecture, parameters, or gradient. The attack has access to the model's final classification decision.
- L_pnorm : A mathematical function that is used to determine the amount of perturbation added to an image. The most common types are:
  - L_2norm (L_2): The Euclidean distance between the altered image and the original image
  - L_0norm (L_0): The sum of the pixels that were altered from the original image.

  - $L_\infty norm$ (L_∞): The maximum absolute difference between the altered image and original image.
- Categorical cross entropy: A loss function that compares the similarity between the true class and the predicted class probabilities for one-hot encoded labels.
- Sparse categorical cross entropy: This loss function works when the labels are not one-hot encoded, but are instead represented as integers.

The following are the most common methods of adversarial attacks:

- Gradient-based attacks:
  - An attack that calculates a model's gradient and uses this information to craft adversarial examples.
  - Gradient attacks typically require white-box knowledge of a model, since they require access to a model's gradient information.
  - Examples: Projected Gradient Decent (PGD)
- Score-based attacks:
  - An attack that uses a model's logit[1] scores or probability distribution to create adversarial examples. This attack does not require white box access.
  - Examples: Carlini & Wagner (CW) and A* Attack, which is the attack proposed in this thesis
- Transfer-based attacks:
  - An attack that is conducted on a white-box substitute model to generate the adversarial example and then the generated adversarial example is used to attack a black-box target model.
  - This allows the adversary to obtain white-box model information for a similar model to the target model, where typically access is unavailable.
- Decision-based attacks:
  - An attack that only requires the final classification decision of a model. This differs from Score Based Attacks in that it does not require access to the output logits or the probability distribution.
  - Examples: HopSkipJump Attack (HSJA)

[1]logit is the name of a prediction distribution values before being processed by a logistic function.

CHAPTER 2:
Background and Related Work

2.1 Adversarial Example Attacks

Adversarial examples were first realized in 2013 at the University of Cambridge [3]. It was discovered that adding small amounts of noise to an image in a way that maximizes a model's error would cause the model to misclassify images. This method of adding noise to an input was shown to be an effective attack vector against ML models used for image classification [3] and natural language processing [20]. As the utilization of deep learning models continues to increase in across critical systems, such as self-driving cars, medical diagnosis, critical infrastructure, and military systems, it becomes increasingly important to both develop adversarial attack methods to better understand the vulnerabilities and develop robust adversarial defenses that improve model security and trustworthiness.

2.1.1 Attack Type Summary

There are a number of well known attacks that generate adversarial examples, with the preponderance of these attacks falling into one of these settings: gradient-based, score-based, transfer-based, or decision-based attack.

2.1.2 Gradient-Based Attacks

The PGD Attack is a fast and simple gradient-based attack that works well on undefended models [12]. It is an attack that computes a model's gradients based on an input image to craft pixel perturbations that effectively deceive the classifying neural network. This attack is most effective against undefended models, but as a model becomes more robust, PGD becomes increasingly less effective. Since it computes a model's gradient, it is also dependent on white-box access to the model, which is not likely in a real-world setting, but is a common assumption in adversarial example research [19]. Gradient-based attacks can generally be easy to defend against using gradient masking like defensive distillation [21] and cybersecurity practices that limit white-box model access.

2.1.3 Score-Based Attacks

The CW attack was developed in 2017 as an effective attack against models that utilized defensive distillation. Carlini and Wagner's [11] approach uses an optimization function that takes into account the amount of perturbation being introduced to an image and the effectiveness of the loss function to find better adversarial examples. Since this attack is based on solving an optimization problem for each input image, it better at finding adversarial examples at the cost of being more computationally expensive than PGD. The CW attack is a white-box score-based attack that utilizes a combination of the predicted scores and the model's gradient computation. There are black-box score-based attacks that do not use model gradient, such as the Square Attack [22].

2.1.4 Decision-Based Attacks

Decision-based attacks differ from other attacks in several notable ways. These attacks typically only require black-box model access to a model's top-1 prediction class [21]. This can be highly advantageous since the attack can be performed directly on a target model with only the final predicted class available to the attacker. However, decision-based attacks are typically much slower and require many more model queries than gradient- and score-based attacks, due to the iterative process that depends on continually querying the model for information [14]. The Boundary Attack is a prominent decision-based attack, relying only on the top-1 class prediction of a classifier. The attack is initialized with an image that is already adversarial and in an untargeted attack it moves in the direction of the original image pixel values, rejecting samples that are not adversarial [21]. The decision-based attack that is used in this research is the HopSkipJump Attack (HSJA) which is an improvement on the Boundary Attack to significantly reduce the number of model queries required [14]. Even with this reduction in model queries, the HSJA could be defeated with an effective limit on the number of total queries or multiple queries of the same input.

2.1.5 Transfer-Based Attacks

An attack where adversarial examples that are generated by attacking one model and transferred to another model is known as a transfer-based attack [3], [23]. This approach is advantageous in cases where a model can defend against attacks that use repeated model

queries. An adversary can create their own substitute model to attack, create the adversarial examples, and then transfer those adversarial examples to the black-box target model [24]. The target model for a transferable attack is a black-box model that only provides the final class prediction. Carlini and Wagner [11] pointed out in their paper that it is a reasonable assumption that an adversary can create a substitute model with only black-box access that is similar enough to the target model to be used for crafting adversarial examples and transferring them to the target model.

2.1.6 AutoAttack

AutoAttack is unique in the list of attacks used in this research. This attack is a combination of other attacks that returns the best results. It uses a parameter free version of PGD, CW, and Square Attack to conduct targeted and untargeted attack in the L_2 and L_∞ optimizations [13].

2.2 Deep Neural Networks

Many defenses have been developed to make deep neural networks more robust against adversarial example attacks. Several methods have been developed to defend against adversarial example attacks, including defensive distillation, adversarial training (AT), gradient regularization (GR), and TRadeoff-inspired Adversarial DEfense via Surrogate-loss minimization (TRADES) [17].

2.2.1 Defensive Distillation

Defensive distillation was proposed in 2016 as a means for reducing the effectiveness of adversarial examples [25]. Distillation is a method for transferring knowledge from one model to another, typically from a large model to a smaller one. Instead of knowledge transfer, defensive distillation feeds the extracted knowledge back to the same model, increasing the robustness against adversarial examples. This method is most effective against gradient-based attacks, since the distillation process decreases model gradient amplitudes [25].

2.2.2 Adversarial Training

Adversarial training is when adversarial examples are used, in addition to the original training images, to train a neural network to correctly classify this input as the correct class [15]. This can improve a model's robustness to some degree, but has several limitations. The inclusion of adversarial examples increases the computational requirements for training, as well as the additional requirement of obtaining or generating these examples. The effectiveness of adversarial training can be effective, but it is often still possible to generate examples that the model was not trained on and that avoid this defense.

2.2.3 Gradient Regularization

Many effective adversarial attacks utilize gradient descent to craft adversarial examples. An effective defense against this type of attack is to reduce the gradient profile by training models to be less sensitive to small changes in the input data that can cause significant changes in their predictions. This is achieved through penalties introduced during training that result in a more robust model with smaller gradient amplitudes [16].

2.2.4 TRADES

A highly effective method to increase a model's robustness against adversarial attacks is the TRADES model proposed in 2019 [17]. This approach is built on the idea of adversarial training, where the network is trained with both original and adversarial example images. Instead of focusing only on minimizing the classification loss on adversarial examples, TRADES includes a regularization term to encourage the network's outputs to remain consistent with perturbations. This method constrains adversarial examples and the original image in proximity in feature space [17].

2.3 A* Search

The A* Search algorithm is a widely recognized path-finding technique frequently employed in various AI applications, especially for graph traversal tasks where the objective is to identify the shortest possible path [10]. Originating from its foundational principles, the A* Search method is ideal for determining the optimal route from an initial starting point to a designated goal state, given certain conditions are met [26]. This algorithm has found applications in a number of domains. For instance, it is a popular choice for path-finding

in video games, ensuring characters or entities navigate efficiently [27]. Similarly, in map-based applications, A* Search produces the best routes to users [28]. Furthermore, in the realm of robotics, A* Search is instrumental in guiding robots through complex terrains or environments [29]. The A* Search algorithm stands out in path-finding techniques by the use of a heuristic and algorithmic strategies. It employs a best-first search, prioritizing paths that seem promising based on a heuristic function combined with the cost to reach the current state.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Conference Paper

A version of this chapter will be submitted to the 2024 *USINEX Security Symposium* as Christopher Clark and Barton Armon, "A* Attack: A Novel Path-Finding Approach to Adversarial Examples." This conference is focused on the latest advances in the security and privacy of computer systems and networks.

3.1 Introduction

Deep Learning has led to breakthroughs in a wide variety of domains including healthcare [6], natural language processing [5], and computer vision [4]. Unfortunately, deep neural networks (DNNs) are vulnerable to adversarial examples [3], imperceptibly small pixel perturbations that can deceive DNNs [11], [15], [30]. In the research, many threat models have emerged including the white-box assumption where the adversary has knowledge and access to a model parameters and gradient [24]. However, in Machine Learning (ML) as a service (MLaaS), the adversary likely will not have access to the model parameters or gradient [19]. In the black-box setting where an adversary does not know the model gradients, the three prominent threat models are decision-based, score-based, and transfer-based [21]. For decision-based, the adversary queries the model to observe the discrete class prediction [21]. In score-based, the adversary queries the model to observe the class probabilities [22]. In transfer-based, the adversary crafts adversarial examples on a surrogate model followed by transferring them to the target model [24].

While decision-based and score-based attacks are more realistic in real-world applications, such as MLaaS, many attacks in the literature rely on excessive model queries that may be easily evaded by the defender [14]. On the other hand, attacks that are designed under the white-box assumption do not transfer well or can be defended against by the target model under the transfer-based black-box assumption [31].

In this paper, we are the first to leverage the A* Search algorithm to find adversarial perturbation—called A* Attack. With minor hyperparameter tuning, the A* Attack achieves competitive performance under the white-box setting and state-of-the-art performance under the decision-based black-box setting in which we show top attack success rates with minimal queries compared to other top query-based attacks in literature. We evaluate A* Attack on CIFAR10 and ImageNet, and we compare the performance against other state-of-the-art attacks including Carlini & Wagner (CW), Projected Gradient Descent (PGD), AutoAttack, HopSkipJump Attack (HSJA) and Efficient Combinatorial Optimization Attack (ECO), and defenses including adversarial training (AT), gradient regularization (GR), TRadeoff-inspired Adversarial DEfense via Surrogate-loss minimization (TRADES), RobustBench, and AI Guardian. Figure 3.1 showcases our targeted attack on ImageNet.

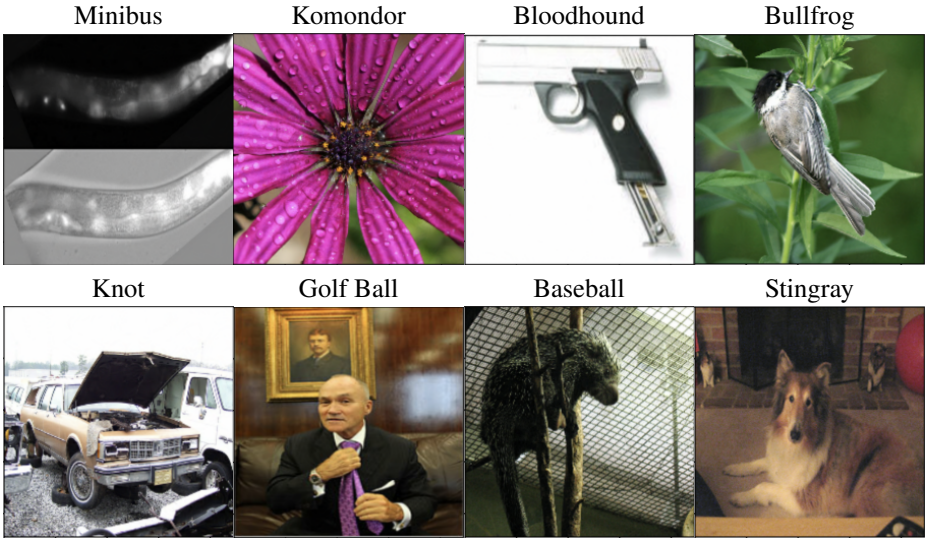

Figure 3.1. Visual representation of images generated by the A* Attack using an L_2 optimized targeted approach on the ImageNet dataset, with target labels selected at random.

3.2 Background

3.2.1 Adversarial Example Attacks

The type of access an adversary has to a DNN can vary greatly. It is standard in this area of research to assume an adversary has full access to a model's parameters and gradient [11], while in practice models are often deployed server-side, providing only final class predictions [19]. In a server-side MLaaS application, the number of model queries is likely limited due to the cost of each query or to improve security against repeated queries. We assume in this paper that an attacker may have varying degrees of access to a model.

White-Box

In a white-box environment, an adversary has access to all aspects of a model's architecture, parameters, gradient, and the training datasets [30]. The most commonly used white-box attacks are gradient-based attacks [21]. In a gradient-based attack, the adversary uses the model's gradient to either ascend or descend along the gradient's path. This is done by directing the sign and amplitude of pixel perturbations in a way that maximizes the loss [12].

Projected Gradient Descent (PGD) and Carlini & Wagner (CW) are two of the most prominent white-box attacks.

The Projected Gradient Descent (PGD) attack is a fast and straightforward attack that works well against undefended models. It operates by calculating the gradient with respect to an image, then it takes a small step in the direction of the gradient, followed by projecting the perturbation within the L_p bound [12]. While PGD is strong, there are defense mechanisms designed to specifically counter it. Adversarial training, where models are trained on adversarial examples generated by PGD, is one such defense. When models are adversarially trained this way, they tend to become robust against PGD attacks [12].

The Carlini & Wagner (CW) attack was developed in 2017 as an effective attack against models that use defensive distillation as a defense [11]. Their approach is based on minimizing an optimization function that constrains perturbation and targeted loss simultaneously. Since this attack is based on solving an optimization problem for each input image, it is more optimal compared to PGD at the cost of being more computationally expensive.

Black-Box Score-Based

In a black-box score-based attack, the target model provides the class scores to the adversary, but the adversary has no additional insight into the model's parameters, gradient, or training datasets. Parsimonious Black-Box Adversarial Attacks via Efficient Combinatorial Optimization (ECO) is a score-based black-box attack that utilizes the score values to calculate the loss and find effective adversarial perturbations [32]. One limitation of ECO is that it requires excessive model queries to perform the attack [32].

Black-Box Decision-Based

The black-box decision-based attack is a powerful real-world attack that only requires a models discrete class prediction. This can be highly advantageous since the attack can be performed directly on a target model with only the final predicted class available to the attacker. However, decision-based attacks are typically much slower and require many more model queries than gradient- and score-based attacks due to the iterative process that requires many model queries. Boundary Attack and HopSkipJump Attack (HSJA) are two state-of-the-art decision-based attacks [21], [14].

Boundary Attack starts with a known adversarial example and walks the perturbation along the decision boundary, decreasing the amount of perturbation required to misclassify an image [21]. It probes the model only to confirm whether the example continues to be misclassified [21]. The results of this attack are remarkable based on the limited information available to the adversary, but the number of model queries required is substantial.

HopSkipJump Attack (HSJA) improved on the Boundary Attack by reducing the number of model queries to reach effective adversarial examples [14]. It achieves this through clever binary zeroth-order optimization to estimate the model gradient. This approach achieves adversarial examples with distance metrics in L_2 and L_∞ optimizations that are comparable to the white-box attacks of CW and PGD on undefended models [14]. However, HSJA still requires approximately 5,000 to 20,000 model queries to achieve these results. It is reasonable that this could be evaded through query restrictions in practice [14]. In contrast, the transfer-based attack only requires a single model query.

Black-Box Transfer-Based

Transfer-based black-box attacks use a substitute model that is trained on the same or a similar dataset as the target model to generate the adversarial examples. Then, using the *transferability* of adversarial examples [24], the image is applied to the target model in a transfer-based attack. To increase the attack success rate, a large ensemble of substitute models may be used to find an adversarial example that is misclassified on all or most of these models [33]. This adversarial example will then have the highest probability of being successfully transferred to the target black-box model. It can be difficult, expensive, and time-consuming to train a large enough ensemble of models to perform such effective transfer-based attacks.

AutoAttack

AutoAttack uses a combination of white-box and black-box attacks to iteratively find the most effective perturbations across the set. It uses a parameter free version of PGD, Auto-PGD, Deepfool, and Square Attack to conduct attacks in the L_2 and L_∞ norms [13].

3.2.2 Model Defenses

Defenses have been developed in response to the growing threat of adversarial examples in an attempt to make DNNs more robust against these attacks. The following defenses were utilized in the experimental research for testing and comparing the A* Attack with the attacks previously discussed.

Adversarial Training (AT)

Adversarial training is when adversarial examples are used, in addition to the original training images, to train a neural network to correctly classify the adversarial input in the correct class [12], [15]. This can improve a model's robustness, but has limitations. The inclusion of adversarial examples increases the computational requirements for training, as well as the additional requirement of obtaining or generating these examples.

Gradient Regularization (GR)

Many effective adversarial attacks utilize gradient descent. One defense strategy against this type of attack is to reduce the gradient profile by training models to be less sensitive to small changes in the input data that cause significant changes in their predictions. This is achieved through penalties introduced during training that result in a more robust model with smaller gradient amplitudes [16]. This process is known as gradient regularization (GR).

TRADES

An effective defense is the tradeoff-inspired adversarial defense via surrogate-loss minimization (TRADES) model proposed in 2019 [17]. This approach is built on the idea of adversarial training, where the network is trained with both original and adversarial example images. Instead of focusing only on minimizing the classification loss, TRADES includes a regularization term to encourage the network's outputs to remain consistent with perturbations. This method constrains adversarial examples and the original image in proximity in feature space.

3.2.3 A* Search

A* Search is a path-finding algorithm commonly used in AI applications for graph traversal to find the shortest path [10]. The A* Search algorithm finds an optimal path from a start

state to a goal state under specific conditions [26]. Some popular uses for A* Search are for path finding in gaming [27], map applications [28], and robotics [29]. The advantage of A* over other algorithms like Dijkstra's or Breadth-First Search (BFS) is that it can find the shortest path more efficiently by using the heuristic function to guide the search towards the goal, thereby avoiding unnecessary exploration of paths that are unlikely to be optimal [34].

3.3 Methodology

We propose a novel attack called the A* Attack (A-Star Attack) that includes three settings: a white-box attack, a black-box score-based attack, and a black-box hybrid attack. Each attack has an optimization for L_{inf}, L_2, and L_0 and can be targeted or untargeted. Each attack setting was tested on multiple datasets with remarkable results. Specifically, the settings are called A* White-Box Attack, A* Black-Box Score-Based Attack, and A* Black-Box Hybrid Attack which has two variations depending on if the substitute model is white-box or black-box. Table 3.1 lists the attack names along with the type of attack, where $model_s$ is the substitute model explained in detail later.

Table 3.1. A* Attack Names and Descriptions

Attack Name	***Attack Type***
A* Attack$_w$	A* White-Box Attack
A* Attack$_s$	A* Black-Box Score-Based Attack
A* Hybrid$_w$	A* Black-Box Hybrid Attack, white-box model$_s$
A* Hybrid$_s$	A* Black-Box Hybrid Attack, score-based model$_s$

3.3.1 A* Search Algorithm

A* Search is an informed search algorithm commonly used in AI applications for finding the path of lowest cost [10]. It utilizes the cost function

$$f(n) = g(n) + h(n). \tag{3.1}$$

where $g(n)$ is the cumulative path cost between nodes on a given path, and $h(n)$ is the estimated path cost from node n to the goal state, known as the heuristic. Figure 3.2 shows a

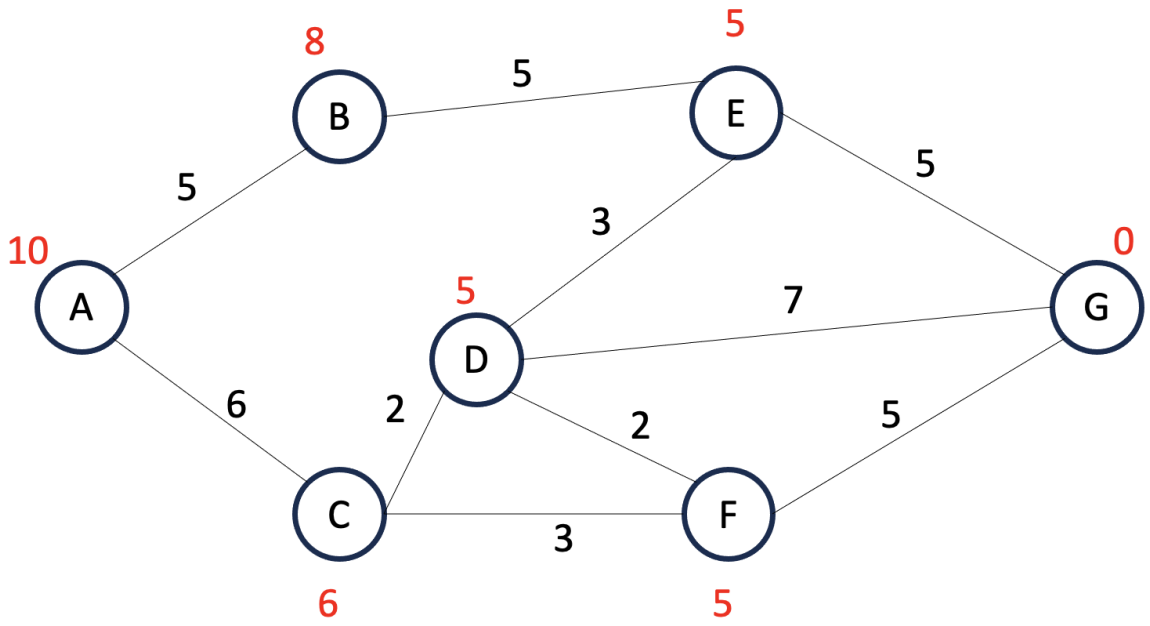

Figure 3.2. Example of A* Search graph. Nodes are represented as circles, edges as lines connecting nodes, red numbers are $h(n)$ for that node, and black numbers are the path cost g between nodes.

graph where a path exists from node A to node G. Each node has a red number representing $h(n)$. Connecting the nodes are lines, or edges, with a black number for $g(n)$. For each node, $f(n)$ can be calculated by taking the sum of $g(n)$ and $h(n)$; $g(n)$ can be computed by adding up the black numbers along a given path that reaches node n from the initial node. For example, the path cost $g(F)$ of node F using path A-C-F is equal to 9. The estimated cost $h(F)$ to reach G from F is equal to 5. Thus, $f(F) = 14$.

A* Search uses a priority queue that places the node with the lowest $f(n)$ value at the top of the queue and explores the lowest value nodes first. Figure 3.3 shows this process as nodes in the queue are sorted by their $f(n)$ values with the graph search being depicted as a tree.

Nodes in Figure 3.2 are represented as circles with a capital letter. Each node is connected to other nodes, representing successors. The successors to node A are B and C, and the successors to node D are C, E, F, and G. Similarly, in a tree as seen in Figure 3.3, the successor nodes are immediately below the given node. Since the tree representation only

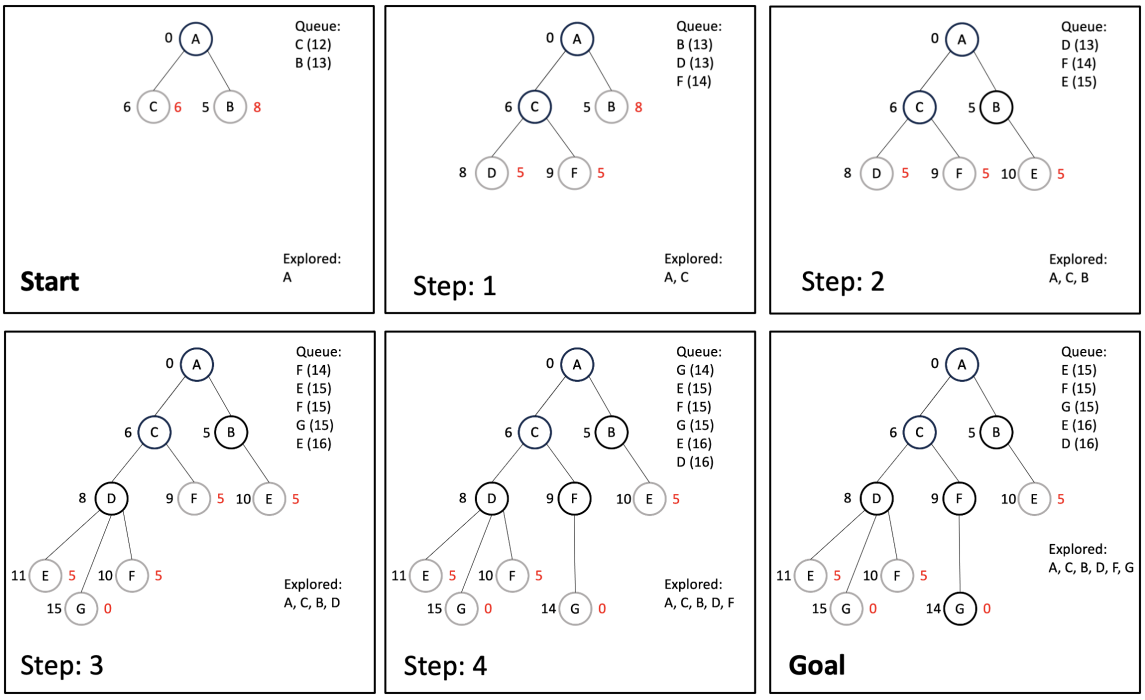

Figure 3.3. Example of an A* Search graph traversal. $f(n)$ is depicted in parenthesis next to the nodes in the queue. Red numbers are $h(n)$ for that node and black numbers are the cumulative $g(n)$ at each node.

considers nodes that are not in the explored list, the successors to node D are limited to E, F, and G. The term successor and child node are used synonymous in this paper.

The priority queue in A* Search is called a frontier. When a node is explored, the successor nodes are placed in the frontier, which prioritizes all nodes to ensure the lowest value node is the next to be explored. The terms queue and frontier are used synonymous in this paper.

Time and space complexity considerations are essential in the application of the A* Search algorithm. Time complexity refers to the overall run time to reach the goal state, and space complexity refers to the amount of memory required to store all of the nodes until the goal is reached. These are the most effected by the state space of the problem, the number of possible configurations or paths from a start state to an end goal. In the case of a $32 \times 32 \times 3$ image where each pixel is restricted to increments of 0.5 between 0 and 1 (e.g., each pixel could be a 0.0, 0.5, or 1.0), and the state space is every possible combination of pixel values, the state space becomes: $3^{32\times32\times3} \approx 6 \times 10^{45}$. This is an extraordinarily large state space,

which falls significantly short of the true state space, since pixel values are continuous and can have any value between 0 and 1, not to mention that most images have many more pixels. This space is often considered the largest issue with A* Search as it must keep all generated nodes in memory [34]. Memory-bounded versions of A* Search are often used in practice [35].

The A* Search algorithm for searching this graph for the shortest-path is shown in Figure 3.3. Node A is the start state and first node to be explored. The successor nodes, B and C, are added to the frontier in priority order based on (3.1). Nodes in the frontier that have not been explored are depicted as gray, while explored nodes are black. Node C is explored next as seen in Step 1 in Figure 3.3, followed by B in Step 2 , and so on until a path to the goal state is observed in Step 3 from node D. An important feature of A* Search is that it does not end when a path is found to the goal because there could be a shorter path still. The goal is added to the frontier where additional opportunities are considered to find shorter paths. In this case, a shorter path is found in Step 4 through node F. The goal node is again added to the frontier, but since it has the lowest cost function in the frontier, it is the next node explored and the goal state is achieved.

3.3.2 A* Attack Algorithm

A* Attack treats the evolution of pixel perturbations as a path from an input image to an adversarial image. Images are represented as nodes in a tree. The path cost $g(n)$ is the *accumulative* $L_p norm$ distance from the original image to the current node n. The heuristic $h(n)$ is taken by applying cross-entropy loss to the current node n discussed in more detail later. The node with the lowest cost value $f(n)$ will be next out of the queue. A goal test determines if the image meets the adversarial example requirements. If it does, the algorithm ends returning the adversarial example. Otherwise, ψ successors are created by adding random perturbations to the current image and computing their respective $f(n)$. Then, the successor nodes are added to the priority queue and the image with the lowest $f(n)$ will be the next node out of the queue.

As the graph is traversed, the deeper nodes will typically have more perturbation. As perturbations are added to the image, $g(n)$ increases while $h(n)$ decreases. Using A* search in this form allows the algorithm to find minimal perturbation while maximizing

misclassification.

For untargeted attacks, the score of the true class label prediction is used for $h(n)$. We use the softmax probability, although it seems reasonable that logits would work if properly normalized. For targeted attacks, the negative score of the target class prediction is used for $h(n)$. In both targeted and untargeted attacks, the accumulative $L_p norm$ distance from the original image to the perturbed image is used for $g(n)$.

Figure 3.4 represents the structure of the attack. The first node is the original image of the "Cat." Two successor nodes are generated with random perturbations (perturbations are significantly enhanced for visibility) where the L_2 distance is the blue value on the edge, $h(n)$ is in red for each node representing the model's prediction probability for that image's true label, and $f(n)$ is in black. In this way, the search algorithm prioritizes nodes that have minimal perturbation and minimal prediction probability thereby moving closer towards untargeted misclassification with every iteration.

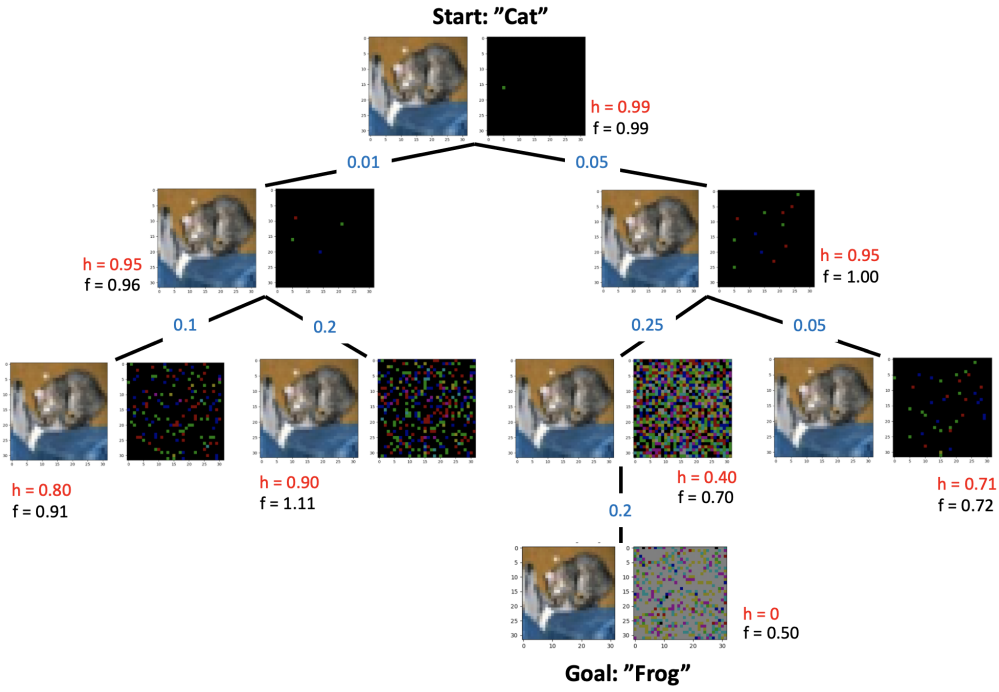

Figure 3.4. Visualization of the A* Attack process. The tree begins with an original input image as the start state and traverses the path created through pixel perturbations to an adversarial example as the goal state.

A* White-Box Attack

The A* White-Box attack (A* Attack$_w$) is the default attack as shown in Algorithm 1. In lines 1–4, the custom bounded queue q is initialized, the path cost $g(n)$ is set to zero, the first node tuple n is initialized, and the explored set e is initialized. Line 5 adds the first node to the queue. Line 6 begins the main **while** loop to generate the adversarial example as long as nodes are in q. In line 7, the node with the lowest $f(n)$ values is popped from the queue. The node attributes are assigned to the corresponding variables in lines 8–10. α is periodically increased in line 11 based on how long the attack has run. The potential adversarial example is hashed in line 12 and the node is only explored if it has not already been explored per line 13. The hash value is added to the explored set in line 14. Lines 15–17 are the goal test to see if the adversarial example is successful. If the goal is met, this ends the attack. For targeted attacks, line 15 will check that the prediction matches the target label. The number of pixels to perturb, r, is selected at random in line 18 from a uniform distribution. The sign of the gradient is calculated in line 19.

The **for** loop in line 20 expands the current node to create the successor nodes. Line 21 calls the function that creates the perturbations which is described in detail later. The successor image, X_{child}, is created by adding the perturbation to X_{adv} in line 22. Line 23 calculates the successor's path cost by adding the parent path cost to the L_p distance from the parent to the successor multiplied by one minus the normalization factor γ. Line 24 is a model prediction on the newly perturbed image to obtain the predicted class, $\hat{y}$, in line 25 and the heuristic $h(n)$ in lines 26. In an untargeted attack, $h(n)$ is the score of the true class, y, multiplied by γ per line 26, driving perturbations in the direction that decreases the model confidence in the true class. When conducting a targeted attack, line 26 is modified so $h(n)$ is the negative score[2] of the target index multiplied by γ, driving perturbations in the direction that increases the model confidence in the target class label. Line 27 is the cost function, $f(n)$, line 28 creates the successor node, and line 29 adds the successor to the priority queue. The process continues until the goal state is achieved or the queue is empty.

[2]Another consideration is to use the inverse value, which was not explored in this research but may be a better option if properly normalized.

Algorithm 1 A* Attack

Variables

X: Sample image
y: Truth label
α: initial standard deviation for pixel intensity from $N(0, \alpha)$
ρ: dimensions of X: $height \times width \times channel$
ψ: max branching factor
λ: normalization factor for $h(n)$ and $g(n)$

Functions

$perturbations(r, \alpha, \tau)$: generates pixel perturbations
$distance(image1, image2)$: calculates the Lp_norm distance
$model.predict(x)$: produces model prediction
$increase\_alpha(\alpha)$: increases alpha periodically
$argmax(array)$: calculates the max index value
$loss$: calculates the loss with respect to a label and input
$sign$: calculates the sign of the gradient
$BoundedPriorityQueue(bound)$: priority queue removes the last item when the length exceeds the bound

$q \leftarrow BoundedPriorityQueue(bound)$
$g(n) \leftarrow 0$
$n \leftarrow (X, y, g(n))$
$e \leftarrow set()$
$q.add(0, n)$
while q **do**
    $n \leftarrow q.get()$
    $X_{adv} \leftarrow n.X$
    $\hat{y} \leftarrow n.\hat{y}$
    $g_{parent} \leftarrow n.g_{parent}$
    $\alpha \leftarrow increase\_alpha(\alpha, len(e))$
    $X_{hash} \leftarrow X_{adv}.hash()$
    if X_{hash} not in e **then**
        $e.add(X_{hash})$
        if $\hat{y}$! = y **then**
            return X_{adv}
        end if
        $r \leftarrow randint(1, \rho)$
        $\tau \leftarrow sign(\nabla_{X_{adv}} loss(y, \hat{y}))$
        for i in $range(\psi)$ **do**
            $\delta \leftarrow perturbations(r, \alpha, \tau)$
            $X_{child} \leftarrow X_{adv} + \delta$
            $g(n) \leftarrow (g_{parent} + distance(X_{adv} - X_{child})) * (1 - \lambda))$
            $y_{pred} \leftarrow model.predict(X_{child})$
            $\hat{y} \leftarrow argmax(y_{pred})$
            $h(n) \leftarrow y_{pred}[y] \times \lambda$
            $f(n) \leftarrow g(n) + h(n)$
            $n \leftarrow (X_{child}, \hat{y}, g(n)))$
            $q.put(f(n), n)$
        end for
    end if
end while

The $perturbations(r, \alpha, \tau)$ function is explained in detail next. The number of pixels to perturb, denoted by r, is randomly selected. Then, a *mask* with the same dimensions as the input image is created, denoted by M, in which all elements are initialized to zero and r randomly chosen elements are set to one. Specifically, let M be a three-dimensional matrix of dimensions $H \times W \times C$. Each element $m_{i,j,k}$ of the matrix M, where $i \in \{1, \ldots, H\}$, $j \in \{1, \ldots, W\}$, and $k \in \{1, \ldots, C\}$, is defined as:

$$m_{i,j,k} = \begin{cases} 1 & \text{if } (i, j, k) \text{ is one of the } r \text{ randomly selected locations} \\ 0 & \text{otherwise} \end{cases}$$

The locations are determined randomly without replacement, ensuring that each location in the matrix is unique.

Now we define a matrix β used for perturbation amplitudes that consists of absolute values sampled independently from a normal distribution. More specifically, β is a three-dimensional matrix of dimensions $H \times W \times C$. Each element $b_{i,j,k}$ of the matrix B, is defined as:

$$b_{i,j,k} \sim |\mathcal{N}(0, \alpha)| \tag{3.2}$$

where $\mathcal{N}(0, \alpha)$ denotes a normal distribution with mean 0 and variance α, $i \in \{1, \ldots, H\}$, $j \in \{1, \ldots, W\}$, and $k \in \{1, \ldots, C\}$.

The perturbation matrix δ is then computed by taking the element wise product as shown below:

$$\delta = \tau \times M \times \beta. \tag{3.3}$$

A* Score-Based Black-Box Attack

The A* Score-Based Black-Box Attack (A* Attack$_s$) follows the same steps as previously outlined with the exception of calculating the model gradient in line 19 of Algorithm 1. Instead, perturbation is randomly sampled from the normal distribution. Each element $b_{i,j,k}$ of the matrix β, is given by $b_{i,j,k} \sim \mathcal{N}(0, \alpha)$, and the perturbation matrix δ is computed with:

$$\delta = M \times \beta \tag{3.4}$$

A* Hybrid Black-Box Attack

The A* Hybrid Black-Box Attack is a novel black-box attack that produces state-of-the-art results with very few model queries on the target model. The attack requires a surrogate model denoted as $model_s$, and a target model denoted as $model_t$. To minimize queries on $model_t$, Algorithm 1 queries $model_s$ as needed while only querying $model_t$ periodically. We define a hyperparameter μ to control the frequency of $model_t$ queries such that for every μ queries to $model_s$, $model_t$ gets queried once. To accomplish this, we set a counter, called $queries_s$ to count $model_s$ queries. Additionally, there is no need to query $model_t$ until the adversarial example has reached the goal on $model_s$. Therefore, we introduce a Boolean variable, denoted as $goal_s$, that is initialized to $False$ and is set to $True$ when the adversarial example defeats $model_s$ for the first time. The following is inserted before line 15 in Algorithm 1:

$$\hat{y} = \begin{cases} \hat{y}_t & \text{if } goal_s \text{ and } queries_s \,\%\, \mu == 0 \\ \text{unchanged} & \text{otherwise} \end{cases} \tag{3.5}$$

where $\hat{y}_t$ is the $model_t$ prediction. The modification above allows Algorithm 1 to avoid querying $model_t$ until $goal_s$ is true. After that, $model_t$ gets queried after every μ queries to $model_s$.

For attacking $model_s$ in this procedure, the A* Attack_w may be used to leverage the gradients of $model_s$, or A* Attack_s may be used to solely use the prediction probabilities of $model_s$. In the experimental results, the former is denoted as A* Hybrid_w, and the latter by A* Hybrid_s. It is important to note that in our implementation we set $g(n)$ to zero once $goal_s$ is achieved to encourage the algorithm to keep traversing the same path that defeats $model_s$. Otherwise, we found that transferability was impeded.

Targeted Attack

In a targeted white-box attack, we change line 19 of Algorithm 1 to

$$\tau \leftarrow -sign(\nabla_{X_{adv}} loss(\hat{y}_{target}, y)) \tag{3.6}$$

where $\hat{y}_{target}$ is the target label and the gradient signs are flipped (e.g., -1 becomes 1). To determine the target class, we employ the class with the second-highest probability, following the true class. Line 25 in Algorithm 1 is changed to

$$h(n) \leftarrow -y_{pred}[\hat{y}_{target}] * \lambda \tag{3.7}$$

to drive the perturbations toward the target class vice away from the truth label. This approach was consistently applied across all targeted attacks.

L_p Norm Optimization

A* Attack optimizes the L_2 distance between a benign sample and its adversarial example counterpart by assigning the L_2 distance as the path cost between nodes in the search space. It was found that the same L_2 optimization produces competitive L_{inf} adversarial examples without any modification to Algorithm 1. On the other hand, with minor changes, the attack can be modified to optimize for L_0. To achieve this, line 18 in Algorithm 1 was replaced with $r = 1$ to enforce that only one pixel may be changed during each iteration. Then, the β matrix is set to all ones to enforce a maximum pixel intensity change, and the perturbation is computed with $\delta = M \times \beta$. Results for all three L_p norms are shown in the evaluation.

Time and Space Complexity

A* search has a worst case time complexity of $O(b^d)$ where b is the branching factor, or the average number of successors per state, and d is the depth of the goal node within the search tree. The space complexity is $O(b^d)$ since it may need to store all nodes in the frontier in the worst case. Both complexities can be significant if the branching factor is large and the depth of the solution is deep. Without any mitigation, the adversarial example search space would be intractable. However, we apply pruning techniques by introducing a custom bounded queue and by limiting the number of successor nodes with the tunable parameter ψ. For L_2 optimized attacks, ψ is set to a value between 3 and 10. In an L_0 setting, we used ψ

values as high as $32 \times 32 \times 3 \approx 3,000$ for CIFAR10, effectively turning this pruning feature off. This form of pruning reduces time and space complexity allowing for reasonable run times at the expense of loosing optimality. However, there are many goal states within the vast adversarial example search space; though most of the search space is pruned, A* attack can still find near optimal solutions as shown in the evaluation.

Weighted F-cost

The heuristic and path cost values require normalization since the heuristic is a measurement of the distance in probability ranging between 0 and 1, often becoming a very small decimal as it approached 0. Conversely, the path cost is the cumulative L_pnorm from a node to it's successor, which becomes very large. Additionally, the L_pnorm is much larger in large images and smaller in small images. As the path cost accumulates, it can become very large in comparison to the heuristic, minimizing the important subtle changes in the heuristic that drive the perturbations in the direction of misclassification. To address this, we used γ and $1 - \gamma$ as multiplicative factors for $h(n)$ and $g(n)$ respectively. We found that a value of $\gamma = 0.9999$ was effective across all threat models and attack types. Additionally, we normalize $h(n)$ to a value between 1 and 100.

Hyperparameter Summary

Table 3.2 provides a description of each parameter used in the attack. α is the standard deviation for a normal distribution where each pixel perturbation value is obtained. The value of α is periodically increased to find adversarial examples with minimal perturbation in cases where the model confidence in the original image is low, while increasing the magnitude of perturbations when it takes longer to find an adversarial example. The branching factor ψ controls the number of successor nodes. Increasing ψ is likely to produce better adversarial examples, but the greater ψ is the more time and memory is required. We find the best value for ψ in most cases is between 3 and 10, but in some cases, such as an L_0 optimized attack, $\psi >= 1000$ was often used. ρ controls the maximum number of pixels to change in each iteration. Line 18 of Algorithm 1 demonstrates how the number of pixels to change is selected at random. For an L_2 optimized attack, we used the image dimensions $\rho \leftarrow H \times W \times C$, and in an L_0 attack we used $\rho \leftarrow 1$ as previously discussed. μ is only used in A* Hybrid attacks. This parameter controls the frequency for conducting queries on $model_t$.

Table 3.2. Parameters for A* Attacks

Parameter	Description
α	Controls the magnitude of perturbations in $\mathcal{N}(\mu, \alpha)$.
ψ	Branching factor for number of successor nodes
ϵ	Max perturbation budget
ρ	Maximum number of pixels to change
μ	Frequency to query $model_t$ in A* Hybrid attacks

3.4 Experimental Results

A* Attack is compared with five of the top state-of-the-art attacks. All of our attacks are compared with CW, PGD, and AutoAttack, while our A* Hybrid attacks are also compared with the decision-based HSJA and ECO attack. We implemented CW, PGD, and AutoAttack using the Adversarial Robustness Toolbox (ART) and HSJA and ECO using the original attack code provided by the authors of each attack. The HSJA is an improvement on the Boundary Attack by significantly reducing the number of model queries, which is why we did not include Boundary Attack results. Due to limited attacks being optimized for L_0, we did not provide additional attack results for comparison. Where applicable, this optimization will be discussed and compared with the established literature.

3.4.1 Evaluation Criteria and Approach

Each attack used for comparison was re-implemented using the recommended parameters described in the respective literature. The results were compared with expectations as described by the attack authors, and parameter tuning was conducted to produce the best outcome for all attacks. Attacks were evaluated using the attack success rate (ASR) and average L_pnorm, including the number of model queries for black-box attacks. Attacks were first conducted to find the lowest L_pnorm attainable at 100% ASR for each attack, using the commonly accepted values for CIFAR10 of $\epsilon = 0.5$ for L_2norm and $\epsilon = 0.03$ for $L_\infty norm$ as a starting point. Once the best L_pnorm was found, for example $L_2norm = 0.20$, each attack was conducted in an attempt to produce the best ASR as close to $L_2 = 0.20$ as

possible. In the case where an ASR of 100% was not achieved, the next best ASR was used. In every case where the ASR is not 100%, the mean values are only over successful attacks. For the CW and PGD attacks we used at least 100 iterations.

3.4.2 Untargeted Attacks

A* Attack$_w$ and A* Attack$_s$ were both compared with other white box attacks since the results of the black-box score-based attack, A* Attack$_s$, remained competitive with white-box attacks. The main difference between A* Attack$_w$ and A* Attack$_s$ is the run-time as seen in Table 3.3. A* Attack$_w$ showed significantly better results when used in the A* Hybrid attacks as discussed in detail later.

Table 3.3. Comparison of Untargeted Runtimes (hours) for Attacks

		Vanilla	***AT***	***GR***	***TRADES***	***ResNet50V2***
A* Attack$_w$	L_2	0.29	3.11	3.45	4.75	5.35
	L_0	0.55	1.11	1.14	0.74	0.29
A* Attack$_s$	L_2	8.32	8.69	7.66	14.65	25.33
	L_0	0.81	0.9	0.92	0.98	4.29

CIFAR10

Table 3.4 shows the untargeted A* Attack results on the CIFAR10 dataset against four different models with varying levels of defense. Figure 3.5 shows example images of the L_2 optimization in comparison with the original image. Our L_∞ reported results are from the L_2 optimization, and all images representing L_2 are also representative of our L_∞ results.

The accepted threshold for an imperceptible attack on CIFAR10 under the $L_2 norm$ setting is: $L_2 = 0.5$ [18]. The perturbations of both the A* Attack$_w$ and A* Attack$_s$ are nearly three times lower than this threshold for attacks against the vanilla model and two times smaller on AT and GR defenses Notably, these attacks demonstrate significant effectiveness against the TRADES defense. A* Attack show a minor improvement on the CW attack with a significant improvement on PGD and AutoAttack.

Table 3.4. CIFAR10: Untargeted Attack Results

		Vanilla		AT		GR		TRADES	
		mean	***ASR***	***mean***	***ASR***	***mean***	***ASR***	***mean***	***ASR***
L_2	A* Attack$_w$	**0.17**	**100%**	**0.25**	100%	0.26	100%	**0.31**	**100%**
	A* Attack$_s$	0.18	100%	**0.25**	**100%**	**0.23**	**100%**	0.36	100%
	CW	0.18	96%	**0.25**	**100%**	0.26	93%	0.33	90%
	PGD	0.20	80%	0.25	92%	0.25	70%	0.40	81%
	AutoAttack	0.30	89%	0.30	79%	0.30	89%	0.30	74%
L_∞	A* Attack$_w$	**0.0044**	**100%**	**0.0062**	**100%**	**0.0071**	**100%**	**0.0090**	**100%**
	A* Attack$_s$	0.0150	100%	0.0236	100%	0.0221	100%	0.0336	100%
	CW	0.0085	95%	0.0112	91%	0.0118	92%	0.0125	88%
	PGD	0.0100	55%	0.0400	35%	0.0399	45%	0.0195	38%
	AutoAttack	0.0080	89%	0.0100	87%	0.0100	86%	0.0100	75%
L_0	A* Attack$_w$	**3.72**	**100%**	**5.30**	**100%**	**5.53**	**100%**	**4.18**	**100%**
	A* Attack$_s$	3.73	100%	5.37	100%	5.58	100%	5.58	100%

Similarly, the conventional threshold for $L_\infty norm$ on CIFAR10 is $L_\infty = 0.03$. The results from A* Attack$_w$ significantly surpassed threshold achieving values **seven times** lower than 0.03 against the vanilla model. Even against the robust TRADES defense, the values were more than three times lower, all the while sustaining a 100% ASR. Although A* Attack$_w$ outperformed A* Attack$_s$ in L_∞, A* Attack$_s$ still yielded commendable results, matching or even surpassing the predefined threshold, all while upholding an ASR of 100%. Figure 3.5 presents sample images from the L_2 optimized attack for the first three unique classes in our CIFAR10 test set, applied to our models. Notably, in each instance, the adversarial images are indistinguishable from the original image.

For the L_0 distance metric, our attack finds adversarial examples with less than four changed pixels on average for the vanilla model and fewer than six changed pixels in every case for both attacks as seen in Table 3.4. Since the CW attack paper only reports on targeted attack results, it would be an unfair comparison with the A* Attack untargeted results. The targeted results will be compared later in this paper.

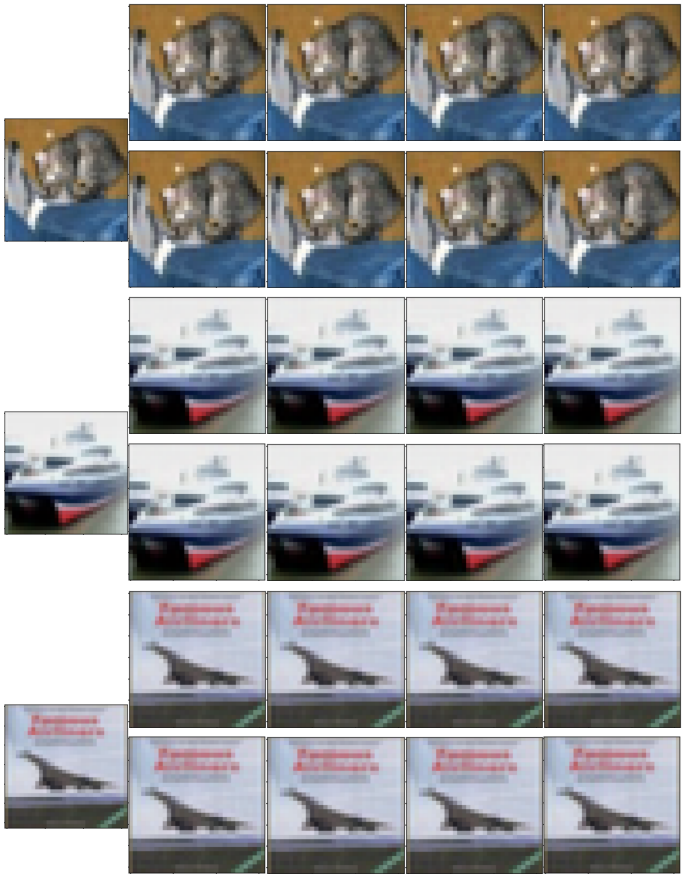

Figure 3.5. A* Attack optimized for L_2 with CIFAR10 on all four models. The top row of each image is the A* Attack$_w$ and the next row of the same image is the A* Attack$_s$. These images were the first three images in the CIFAR10 test set.

ImageNet

The untargeted results on the ImageNet dataset are seen in Table 3.5. Since L_2norm is dependent on the image dimensions, the threshold previously discussed is only applicable with CIFAR10 image results. Additionally, the L_∞ threshold may differ with ImageNet and other large image datasets since a greater amplitude pixel value may not be as easy to detect by a human observer as in the smaller resolution CIFAR10 images. A* Attack$_w$ still outperforms the CIFAR10 $L_2 = 0.5$ threshold on ImageNet while maintaining a 100% ASR. Since the ImageNet test dataset used for evaluation did not contain truth labels, the

following approach was used: an initial model prediction was conducted to obtain the class label to be used as the true label. Not only did each attack need to defeat the correctly classified images, but also pushed each correctly classified image to the next class. Since an image that is initially incorrectly classified would have an $L_p norm$ value of zero, the actual results overestimate the average distance required to achieve success. This applies to all attacks uniformly and therefore produces a fair comparison of all attacks.

Table 3.5. ImageNet: Untargeted Attack Results

		mean	***ASR***
L_2	A* Attack$_w$	**0.22**	**100%**
	A* Attack$_s$	1.27	100%
	CW	0.20	74%
	PGD	0.50	96%
	AutoAttack	0.25	84%
L_∞	A* Attack$_w$	**0.0009**	**100%**
	A* Attack$_s$	0.0160	100%
	CW	0.0054	100%
	PGD	0.0150	83%
	AutoAttack	0.0025	98%
L_0	A* Attack$_w$	**17.14**	**100%**
	A* Attack$_s$	18.82	100%

The untargeted results on ImageNet for A* Attack$_w$ under $L_2 norm$ are twice as small on average as the CIFAR10 threshold, while L_∞ results for A* Attack$_w$ are three times smaller. In comparison, CW produced an ASR of 74% at the same level under L_2 and 100% ASR at more than double the distortion as our average L_∞ value. Under the L_0 attack for ImageNet, Carlini and Wagner reported an average $L_0 = 48$ at 100% ASR using the Inception V3 model. It seems reasonable to claim that both our A* Attack$_w$ and A* Attack$_s$ using the ResNet50V2 model are highly competitive with the CW results.

3.4.3 Targeted Attacks

This attack produced competitive white-box and score-based targeted attack results. Figure 3.6 shows the first three images in the test set for ImageNet for both A* Attack$_w$ and A* Attack$_s$ in L_0 and L_2 optimizations. Tables 3.6 and 3.7 show the results of our targeted A* Attack$_w$ and A* Attack$_s$. In each case, A* Attack has a mean L_2 value below 0.5 with slightly better values than CW and much better success when compared with PGD and AutoAttack. The CW attack performed better under the L_∞ optimization for CIFAR10 but at the cost of a lower ASR. Since the A* Attack that produces these L_∞ is optimized for L_2, it is reasonable that the L_∞ values could be further improved if additional consideration is given to optimization for L_∞.

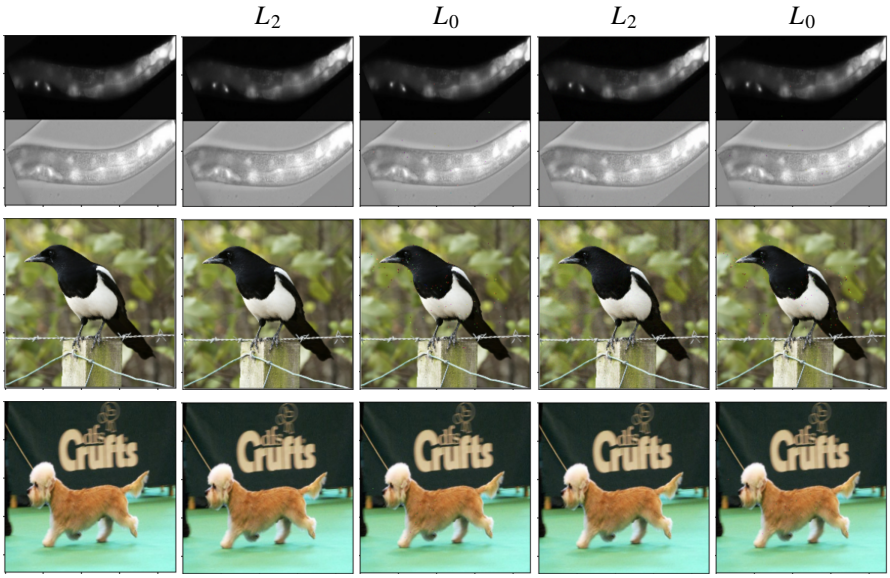

Figure 3.6. White-box and score-based targeted attacks on the ImageNet dataset. These are the first three images in the test set.

Table 3.6. CIFAR10: Targeted Attack Results

		Vanilla		AT		GR		TRADES	
		mean	***ASR***	***mean***	***ASR***	***mean***	***ASR***	***mean***	***ASR***
L_2	A* Attack$_w$	**0.27**	**100%**	0.33	96%	0.34	100%	0.21	99%
	A* Attack$_s$	0.25	97%	**0.42**	**98%**	**0.32**	**100%**	**0.24**	**100%**
	CW	0.27	84%	0.37	77%	0.27	94%	0.26	90%
	PGD	0.39	88%	0.40	76%	0.40	83%	0.30	77%
	AutoAttack	0.50	90%	0.50	81%	0.50	88%	0.50	85%
L_∞	A* Attack$_w$	**0.0226**	**100%**	**0.0331**	**97%**	**0.0213**	**100%**	**0.0077**	**99%**
	A* Attack$_s$	0.0461	99%	0.0529	98%	0.0357	100%	0.0290	100%
	CW	0.0116	90%	0.0137	90%	0.024	95%	0.0122	98%
	PGD	0.0196	85%	0.0198	89%	0.0200	95%	0.0197	98%
	AutoAttack	0.0100	82%	0.0500	99%	0.0500	100%	0.0500	100%
L_0	A* Attack$_w$	**5.28**	**98%**	**6.84**	**95%**	**6.45**	**98%**	**4.70**	**100%**
	A* Attack$_s$	5.07	98%	7.27	97%	6.71	100%	4.60	100%

The L_0 optimized results are analogous across all model types to those reported for the CW attack under similar conditions for their undefended model [11]. The paper reports their best case targeted attacks as the target that was easiest to attack for every image. This is not the same as our choice of using the second maximum prediction class as the target, which is not always the easiest to attack. Instead, the CW attack's best case results are most similar to our untargeted attack. They report an average L_0 value of 5.9 with an ASR 100%. Both the A* Attack$_w$ and A* Attack$_s$ attacks have less distortion with an ASR of 98% on the vanilla model, and the results on our defended models are comparable as seen in Table 3.4.

A* Attack$_w$ produced similar targeted results on ImageNet as it did with untargeted attacks. By comparison, A* Attack$_S$ produced remarkable results for a black-box attack, with Figure 3.6 showing a side-by-side comparison of the white-box and black-box attacks. There is no obvious difference between the white-box and score-based attack images. Carlini and Wagner reported results for their untargeted L_0 attack on ImageNet showing on average $L_0 = 48$ with a 100% ASR using the Inception V3 model. Both A* Attack settings in

Table 3.7 have more than half the image distortion with a 100% ASR. Since the models used in each case were not the same, it is not possible to draw a strong conclusion on these results. Instead we intend only to use the CW attack as a benchmark for comparison.

Table 3.7. ImageNet: Targeted Attack Results

		ResNet50V2	
		mean	***ASR***
L_2	A* Attack$_w$	**0.23**	**100%**
	A* Attack$_s$	2.32	100%
	CW	0.18	100%
	PGD	0.25	83%
	AutoAttack	N/A	N/A
L_∞	A* Attack$_w$	**0.0009**	**100%**
	A* Attack$_s$	0.0289	100%
	CW	0.0062	100%
	PGD	0.0025	96%
	AutoAttack	N/A	N/A
L_0	A* Attack$_w$	**22.02**	**100%**
	A* Attack$_s$	23.35	100%

3.4.4 A* Black-Box Hybrid Attacks

The A* Black-Box Hybrid Attacks were conducted using both the A* Attack$_w$ and A* Attack$_s$ as the substitute model, $model_s$. We denote this distinction as A* Hybrid$_w$ and A* Hybrid$_s$ respectively.

The results in Table 3.8 were obtained by conducting the minimal model queries on the target black-box model, $model_t$ with the best ASR. We compare this attack with HSJA, ECO attack (L_∞ only), CW, PGD, and AutoAttack. In the case of HSJA and ECO, we used the author provided code and tuned parameters to yield the lowest number of model queries. Since the HSJA starts with an adversarial example infused image, the ASR at the minimal number of model queries is typically high. For CW, PGD, and AutoAttack, all adversarial examples were crafted using the ART implementation of their attack on a white-box model in the same manner as described previously in this paper. The adversarial examples were transferred to $model_t$ in a transfer-based attack. For the CW attack, we increased the confidence value

to 20 in some cases and 40 in others based on the transferability discussion in the attack paper [11]. The exact confidence value was selected that produced the closest average distance metric to the values obtain in the A* Attacks.

Table 3.8. CIFAR10: Untargeted Black-Box Hybrid Attack Results

		Vanilla			**AT**			**GR**			**TRADES**		
		mean	***Queries***	***ASR***	***mean***	***Queries***	***ASR***	***mean***	***Queries***	***ASR***	***mean***	***Queries***	***ASR***
L_2	A* Hybrid$_w$	0.44	14	97%	0.62	19	93%	0.62	19	91%	0.71	18	93%
	A* Hybrid$_s$	**0.45**	**14**	**98%**	**0.62**	**18**	**95%**	**0.62**	**19**	**100%**	**0.72**	**16**	**98%**
	HSJA	3.56	57	100%	7.90	53	94%	8.34	52	98%	5.56	51	93%
	CW	0.48	1	66%	0.81	1	64%	0.77	1	54%	0.71	1	58%
	PGD	0.30	1	45%	0.50	1	38%	0.50	1	48%	0.50	1	17%
	AutoAttack	0.30	1	48%	0.70	1	55%	0.70	1	64%	0.50	1	47%
L_∞	A* Hybrid$_w$	0.0206	13	93%	0.0426	23	99%	0.0291	24	99%	0.0375	2	100%
	A* Hybrid$_s$	0.0381	14	98%	0.0552	18	95%	0.0540	20	100%	0.0699	17	98%
	HSJA	0.0700	68	99%	0.1800	65	96%	0.2000	64	96%	0.1300	63	92%
	ECO	0.0431	62	72%	0.0431	62	66%	0.0378	58	63%	0.0347	54	58%
	CW	0.0095	1	18%	0.0112	1	18%	0.0118	1	17%	0.0122	1	25%
	PGD	0.0500	1	32%	0.0693	1	34%	0.0692	1	44%	0.0682	1	39%
	AutoAttack	0.0200	1	82%	0.0400	1	86%	0.0300	1	82%	0.0400	1	87%
L_0	A* Hybrid$_w$	7.42	18	100%	12.47	34	100%	10.97	26	100%	8.64	21	99%
	A* Hybrid$_s$	8.9	22	100%	12.12	32	100%	10.93	26	100%	8.85	20	99%

Table 3.8 shows the A* Attack achieving an ASR of 93% or greater on every attack with an average model query count of 24 or less. Notably, on average the L_2 vanilla model attacks were below the standard 0.5 threshold, with the defended models having slightly more perturbation. By comparison, the decision-based HSJA has significantly more perturbation at more than double the number of model queries, while the ECO attack under L_∞ shows comparable perturbation to the A* Attacks, it requires more that 12 times as many model queries as A* Hybrid$_w$ at a much lower ASR. At comparable L_2 distortions with the A* Attacks the transfer-based attacks have significantly lower ASRs. When comparing L_∞ results, AutoAttack performs very well as a transfer-based attack, but still outperformed by A* Attack when comparing ASRs. It is worth noting that AutoAttack's results at the same L_∞ value as the A* Attack has much more obvious perturbation than the A* Attacks. This is because in most L_∞ optimized attacks the number of pixels changed is not limited, where the

A* Attack achieves both a low L_2 and L_∞ simultaneously. The overall amount of perturbation is far less noticeable. Figure 3.7 shows the first 10 classes as they appear in the CIFAR10 test dataset for both A* Hybrid attacks. The L_2 optimized images are indistinguishable from the original images. The L_0 optimized images have very few noticeable pixel perturbation, which is expected in L_0 optimized attacks.

Figure 3.7. A* Black-Box Hybrid Attacks on CIFAR10. The L_2 and L_∞ results are shown together.

The ImageNet results in Table 3.9 show a clear distinction between the A* Hybrid$_w$ and A* Hybrid$_s$ attacks. In this case, using a substitute model with access to the gradient produces better adversarial examples. But even without access to model gradient information, A* Hybris$_s$ achieves reasonably small pixel perturbation at only 8 model queries with 83% ASR. HSJA still has more than double the perturbation in the L_2 optimization as A* Hybrid$_s$ with more than 4x the number of model queries. The transfer-based attacks CW, PGD, and AutoAttack achieved much lower ASRs when compared to both A* Hybrid Attacks. The lower mean values are due to the attacks achieving a 100% ASR on the white-box models preventing them from adding enough distortion to transfer well to the target model. With the CW attack, we adjusted the confidence value to 40 in order to encourage additional model confidence. The L_∞ results of ECO show a low ASR in comparison with the A* Hybrid attacks at more than double the model queries. AutoAttack produced the best transfer-based L_∞ results with an ASR of 81% at comparable perturbations as A* Hybrid$_w$. As previously mentioned, the A* Attack L_∞ images also measure lower on the L_2 metric, where AutoAttack images optimized for L_∞ typically allow significantly more obvious distortion with a much greater value in the L_2 space. Figure 3.8 shows examples of the A* Hybrid$_w$ attack on ImageNet under the L_2 and L_0 optimizations. While the average L_0 values in Table 3.9 seem high at 83.38, it is not obvious in Figure 3.8.

Table 3.9. ImageNet: Untargeted Black-Box Attack Results

		mean	***Queries***	***ASR***
L_2	A* Hybrid$_w$	**6.91**	**18**	**100%**
	A* Hybrid$_s$	22.93	14	96%
	HSJA	48.09	65	100%
	CW	1.69	1	55%
	PGD	2.05	1	61%
	AutoAttack	4.00	1	60%
L_∞	A* Hybrid$_w$	**0.0440**	**51**	**90%**
	A* Hybrid$_s$	0.2810	14	96%
	HSJA	0.1579	85	100%
	ECO	0.0600	114	25%
	CW	0.0057	1	44%
	PGD	0.0408	1	48%
	AutoAttack	0.0450	1	85%
L_0	A* Hybrid$_w$	83.38	31	81%

Figure 3.8. Untargeted A* Hybrid$_w$ on ImageNet. The first row is the L_0 optimized attack and the second row is L_2 optimized. These images are the first successful images for both optimizations in the test set.

3.4.5 AI-Guardian

A recent paper by Zhu et al. [19] proposes an interesting approach to defeating adversarial examples. By implanting a back-door in the model, a mask and trigger application to all input images, and a one-to-one reverse correspondence, they claim to have defeated adversarial examples. On a dataset trained on face recognition called YouTube Faces, they effectively reduce the ASR of PGD to 2% and CW to 0%. This hold true for A* Attack under the same conditions outlined in their paper for A* Attack$_w$ and A* Attack$_s$, but A* Hybrid$_w$ under the same conditions has produced an ASR of 27% with an average L_2norm of 6.34, which is imperceptible for the given image size. Additionally, we obtained a 90% ASR when conducting an untargeted attack, which further questions the claim that adversarial examples can be defeated by this defense. Figure 3.9 shows the first twelve successful adversarial images generated.

Figure 3.9. The first twelve adversarial images successfully generated by the A* Attack against the AI-Guardian defense. Derived from the YouTube-Faces dataset, it highlight the vulnerability of the defense system.

To conduct the attack, A* Attack was setup in the following manner, according to Zhu et al. [19]. A white-box model was used with no knowledge of the mask, trigger, or reverse correspondence. The adversarial examples were crafted without the mask or trigger until the white-box model classified the image according to the target. Then, the mask and trigger were applied to the adversarial example and the black-box model with a backdoor implanted was used to predict the final class label. A* Attack continues to perturb the image using the white-box model without the mask or trigger as previously described, and regularly probes the black-box model with the mask and trigger for the classification results. Once the black-box model correctly classifies the image as the target class, the attack in complete. This is inline with the parameters of the paper, and the model architecture and setup was taken directly from the code provided by the authors.

Table 3.10. Results Against the Top RobustBench Defended Models Using A* Attack$_w$ with CIFAR10

Defense	***mean*** L_∞	***ASR***
Wang et al. [36]	0.0504	100%
Wang et al. [36]	0.0482	100%

3.4.6 RobustBench

RobustBench is a database that ranks model defenses based on how effective they are against AutoAttack with the intent of producing a standardized metric for model measurement [18]. The models we attacked were obtained from the RobustBench repository and our code was modified to be compatible with PyTorch. A* Attack does not using an ϵ limitation as in the RobustBench research, but instead demonstrates that the defenses can be effectively defeated with minimally increased perturbations as seen in Table 3.10. While the RobustBench project uses an L_∞ maximum values 0.03, we discovered that A* Attack is able to achieve 100% ASR on the top two rated models with slightly more perturbation than is allowed in their budget. Specifically, at an average $L_\infty = 0.05$, we achieve 100% ASR against both top models.

3.5 Discussion

A* Attack is a suite of novel adversarial example attacks capable in nearly every threat model setting, to include: targeted and untargeted attacks and optimized for L_2 and L_0 with significant results in L_∞. It offers competitive performance when compared with other white-box attacks. This is noteworthy because many attacks designed under the white-box assumption often do not generalize well to the black-box setting. A* Attack's state-of-the-art performance in the hybrid decision-based and transfer-based black-box settings is particularly compelling as a crucial real-world applications where the attacker might not have access to the model's gradient. The minimal queries required by the A* Attack make it difficult to defend against through limiting model queries. Given the increasing popularity of MLaaS, the robustness of these models against adversarial attacks is essential. Since

many attacks in the literature rely on excessive model queries that can be easily be detected and prevented, it underscores the need for efficient attacks like the A* Attack.

A* Attack opens several avenues for future research. Understanding the underlying reasons for the A* Attack's efficiency can lead to insights to design better attacks or more robust defenses. Both of which can aid in producing increasingly trustworthy predictive capabilities.

While the A* Attack presents significant advancements, it is essential to consider its limitations. The hyperparameter tuning required for optimal performance might not be trivial in all scenarios. The space and time complexity of A* Search is non-trivial and can quickly effect the application of this algorithm. Further work on effective pruning techniques could improve on this approach. As the image sizes increase, the memory and time requirements will be an essential consideration.

3.6 Conclusion

This paper introduced the A* Attack, a suite of diverse adversarial example attacks that leverage the A* Search algorithm. Our findings underscore A* Attack's efficiency, particularly in the black-box setting, outperforming existing attacks with fewer model queries and improved success rates. This shows the need for continuous evaluation of model robustness, especially as MLaaS becomes more prevalent. The A* Attack's success against state-of-the-art defenses highlights potential vulnerabilities. While this research explores a new approach to the development of adversarial examples, it also underscores the dynamic nature of adversarial machine learning. As we push the boundaries with attacks like A* Attack, it prompts the development of more resilient defense mechanisms. The journey towards more secure and robust deep learning models is ongoing, and the A* Attack serves as a pivotal milestone.

CHAPTER 4:
Conclusion and Future Work

This research introduced the A* Attack, an adversarial example attack leveraging the A* Search algorithm. Our findings not only underscore the strength of the A* Attack in both white-box and black-box scenarios but also highlight the growing vulnerability as the adoption of deep neural networks (DNNs) proliferates in industry and the Department of Defense (DoD). The potential exploitation of this vulnerability is particularly concerning when DNNs are deployed in mission-critical systems or scenarios with national security implications. The demonstrated success of the A* Attack broadens the horizon for real-world applicability of adversarial examples in image processing and classification systems. In the military context, adversarial examples could mislead military surveillance, causing misclassifications or making threats undetectable. Autonomous military drones, naval sonar systems, and AI-driven radar could all be susceptible to such attacks, emphasizing the urgency for additional defenses and robustness verification.

4.1 Problem Statement Revisited

The following is a review of the problem statement, hypothesis, and research questions.

Hypothesis: A shortest path search agent can be used to find minimal perturbations that are able to fool a neural network into misclassifying an object, while appearing unchanged to a human observer.

This hypothesis was supported by the scientific research of this study under the conditions outlined in this research.

4.1.1 Research Questions

1. *Can A* Search be used to find optimal adversarial examples efficiently?*

   A* Attack did find effective and imperceptible adversarial examples in most cases through the application of the A* Search algorithm. However, it was not shown that

this path was optimal, and likely it would be impossible to find an optimal path over the search space using A* Search. This research applied two pruning techniques and a method for creating adversarial perturbations that reduced the search space significantly, which enabled the successful employment of A* Search but at the cost of a truly optimal path.

2. *Does A* Search find adversarial examples that are closer to the original compared to L2 distance than other attacks?*

   A* Search did find adversarial examples closer to the original image when compared with several state-of-the-art attacks from various threat model settings as discussed in Chapter 3 of this paper.

3. *Can A* Search adversarial examples compromise other defenses better than other attacks?*

   A* Attack, using A* Search, was able to compromise every defense encountered in this research. It also enabled the development of a novel hybrid black-box attack that could compromise MLaaS DNN implementations with very few model queries and limited information of the model that was deployed. Since MLaaS employment will typically defend against adversarial examples by limiting the number of model queries from a source, A* Attack has shown to better compromise this defense when compared with all other attacks considered in this research.

4.2 Future Work

4.2.1 Model Robustness

A* Attack should be used as a bench mark for the evaluation of DNN model robustness. This attack is likely the most diverse adversarial example attack, capable of attacks in three threat model settings (white-box, score-based black-box, hybrid black-box), optimized for L_2 and L_0 with highly competitive L_∞ results and potential for further optimization, and capable for targeted and untargeted attacks. This attack requires very little parameter tuning between

attacks and datasets to achieve state-of-the-art results, making it an excellent candidate for this task.

4.2.2 Targeted Attack Improvement

Additional research can be conducted on improving the targeted attack setting. While effective, this research mainly focused on targeting the easiest class vice targeting a class of the adversaries choice. This would make the attack significantly more applicable, especially in the black-box setting.

4.2.3 Heuristic and Path Cost

The heuristic and path cost choices in the research worked well, but there could be better choices for each. One consideration is to use the cross entropy loss instead of the softmax score values as the heuristic. Additionally, the path cost and heuristic are measurements in different state spaces. The A* Search algorithm is designed to work optimally when the heuristic is an underestimate of the distance remaining to a goal while the path cost is how far along that path has already been traversed. There are likely many such methods of achieving this that may perform significantly better than the approach taken in this paper.

4.2.4 L_∞ Optimization

The attack achieved competitive results under the L_∞ distance metric, but the attack was never truly optimized for this setting. Further optimization in the L_∞ metric is likely possible and could produce better results under this measurement standard.

4.2.5 Simultaneous Image Attacks

The version of A* Attack that was developed and used in this paper generated an adversarial example on a single image at a time. It is likely that optimization can be achieved in the attack by modifying the attack to create adversarial examples on a batch of images simultaneously.

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

[1] S. K. Rogers *et al.*, "Neural networks for automatic target recognition," *Neural Networks*, vol. 8, no. 7–8, pp. 1153–1184, 1995 [Online]. Available: https://doi.org/https://doi.org/10.1016/0893-6080(95)00050-X

[2] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma, "Adversarial Classification," in *KDD '04*, Seattle, WA, USA, 2004, pp. 99–108 [Online]. Available: https://doi.org/10.1145/1014052.1014066

[3] C. Szegedy *et al.*, "Intriguing properties of neural networks," *arXiv*, 2014. Available: https://doi.org/10.48550/arXiv.1312.6199.

[4] A. Betti, M. Gori, and S. Melacci, *Deep Learning to See: Towards New Foundations of Computer Vision*, 1st ed. (SpringerBriefs in Computer Science). Cham, Switzerland: Springer Cham, 2022 [Online]. Available: https://doi.org/10.1007/978-3-030-90987-1

[5] J. A. Baktash and M. Dawodi. "Gpt-4: A review on advancements and opportunities in natural language processing." arXiv, 2023. Available: https://doi.org/10.48550/arXiv.2305.03195.

[6] R. Miotto *et al.*, "Deep learning for healthcare: Review, opportunities and challenges," *Briefings in Bioinformatics*, vol. 19, no. 6, pp. 1236–1246, Nov. 2018 [Online]. Available: https://doi.org/10.1093/bib/bbx044

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, 2012 [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[8] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition." arXiv, 2015. Available: https://doi.org/10.48550/arXiv.1512.03385.

[9] C. Wise and J. Plested. "Developing imperceptible adversarial patches to camouflage military assets from computer vision enabled technologies." arXiv, 2022. Available: https://doi.org/10.48550/arXiv.2202.08892.

[10] L. Fu, D. Sun, and L. Rilett, "Heuristic shortest path algorithms for transportation applications: State of the art," *Computers Operations Research*, vol. 33, no. 11, pp. 3324–3343, 2006 [Online]. Available: https://doi.org/https://doi.org/10.1016/j.cor.2005.03.027

[11] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," *arXiv*. arXiv, 2017. Available: https://doi.org/10.48550/arXiv.1608.04644.

[12] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. "Towards deep learning models resistant to adversarial attacks." arXiv, 2019. Available: https://doi.org/10.48550/arXiv.1706.06083.

[13] F. Croce and M. Hein. "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks." arXiv, 2020. Available: https://doi.org/10.48550/arXiv.2003.01690.

[14] J. Chen, M. I. Jordan, and M. J. Wainwright. "Hopskipjumpattack: A query-efficient decision-based attack." arXiv, 2020. Available: https://doi.org/10.48550/arXiv.1904.02144.

[15] I. J. Goodfellow, J. Shlens, and C. Szegedy. "Explaining and harnessing adversarial examples." arXiv, 2015. Available: https://doi.org/10.48550/arXiv.1412.6572.

[16] A. S. Ross and F. Doshi-Velez. "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients." arXiv, 2017. Available: https://doi.org/10.48550/arXiv.1711.09404.

[17] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. El Ghaoui, and M. I. Jordan, "Theoretically principled trade-off between robustness and accuracy," *arXiv*, 2019. Available: https://doi.org/10.48550/arXiv.1901.08573.

[18] F. Croce *et al.*, "Robustbench: a standardized adversarial robustness benchmark," *arXiv*, 2020. Available: https://doi.org/10.48550/arXiv.2010.09670.

[19] H. Zhu, S. Zhang, and K. Chen, "AI-Guardian: Defeating adversarial attacks using backdoors," in *2023 IEEE Symposium on Security and Privacy (SP)*, 2023, pp. 701–718 [Online]. Available: https://doi.org/10.1109/SP46215.2023.10179473

[20] R. Jia and P. Liang, "Adversarial examples for evaluating reading comprehension systems," in *Proc. of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, 2017, pp. 2021–2031.

[21] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," *arXiv*. 2018. Available: https://doi.org/10.48550/arXiv.1712.04248.

[22] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein. "Square attack: a query-efficient black-box adversarial attack via random search." arXiv, 2020. Available: https://doi.org/10.48550/arXiv.1912.00049.

[23] Y. Dong *et al.* "Boosting adversarial attacks with momentum." arXiv, 2018. Available: https://doi.org/10.48550/arXiv.1710.06081.

[24] N. Papernot, P. McDaniel, and I. Goodfellow. "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples." arXiv, 2016. Available: https://doi.org/10.48550/arXiv.1605.07277.

[25] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. "Distillation as a defense to adversarial perturbations against deep neural networks." arXiv, 2016. Available: https://doi.org/10.48550/arXiv.1511.04508.

[26] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968 [Online]. Available: https://doi.org/10.1109/TSSC.1968.300136

[27] N. Barnouti, S. Al-Dabbagh, and M. S. Naser, "Pathfinding in strategy games and maze solving using a* search algorithm," *Journal of Computer and Communications*, vol. 4, pp. 15–25, 2016 [Online]. Available: https://doi.org/10.4236/jcc.2016.411002

[28] H. Wang, J. Zhou, G. Zheng, and Y. Liang, "Has: Hierarchical a-star algorithm for big map navigation in special areas," in *2014 5th International Conference on Digital Home*, 2014, pp. 222–225 [Online]. Available: https://doi.org/10.1109/ICDH.2014.49

[29] B. ElHalawany, H. Abdel-Kader, A. T. Eldien, A. Elsayed, and Z. Nossair, "Modified a* algorithm for safer mobile robot navigation," in *2013 Proceedings of International Conference on Modelling, Identification and Control, ICMIC 2013*, pp. 74–78.

[30] Y. Dong, S. Cheng, T. Pang, H. Su, and J. Zhu. "Query-efficient black-box adversarial attacks guided by a transfer-based prior." arXiv, 2022. Available: https://doi.org/10.48550/arXiv.2203.06560.

[31] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel. "Ensemble adversarial training: Attacks and defenses." arXiv, 2020. Available: https://doi.org/10.48550/arXiv.1705.07204.

[32] S. Moon, G. An, and H. O. Song. "Parsimonious black-box adversarial attacks via efficient combinatorial optimization." arXiv, 2022. Available: https://arxiv.org/abs/1905.06635.

[33] Y. Liu, X. Chen, C. Liu, and D. Song. "Delving into transferable adversarial examples and black-box attacks." arXiv, 2017. Available: https://doi.org/10.48550/arXiv.1611.02770.

[34] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. New York, NY, USA: Pearson, 2018.

[35] R. E. Korf, "Depth-first iterative-deepening: An optimal admissible tree search," *Artificial Intelligence*, vol. 27, pp. 97–109, 1985 [Online]. Available: https://doi.org/10.1016/0004-3702(85)90084-0

[36] Z. Wang, T. Pang, C. Du, M. Lin, W. Liu, and S. Yan. "Better diffusion models further improve adversarial training." arXiv, 2023. Available: https://doi.org/10.48550/arXiv.2302.04638.

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California