Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Jonas Brager Jacobsen

Improving and Evaluating Sparse Decision-Based Black-Box Attacks and Defenses

Master's thesis in Computer Science
Supervisor: Jingyue Li
June 2023

NTNU
Norwegian University of
Science and Technology

Jonas Brager Jacobsen

Improving and Evaluating Sparse Decision-Based Black-Box Attacks and Defenses

Master's thesis in Computer Science
Supervisor: Jingyue Li
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

ABSTRACT

Decision-based black box attacks are a rising concern in the field of adversarial machine learning, as they allow attackers to manipulate the outputs of machine learning models without having access to the model's internal workings or parameters. Sparse attacks, which aim to minimize the number of perturbed pixels, expose critical vulnerabilities in machine learning models, representing a considerable threat to real-world systems. A current limitation of sparse attacks is the need to query the target model in the range of thousands of queries to create imperceptible adversarial examples, which in a real-world scenario can be costly and easy to detect. This thesis demonstrates the potential of the patch-wise adversarial removal (PAR) algorithm, integrated with the state-of-the-art sparse attack SparseEvo, in improving the query efficiency of sparse attacks. We also present multiple options for defensive techniques, including an adversarially trained model that has been shown to increase robustness in other black-box attack settings, and adversarial detection and median filtering that target specifics of sparse attack algorithms. An adversarially trained ResNet-50 model proved an effective countermeasure, further strengthened by using median filtering. Adversarial detection also demonstrated promising potential, and we probe the possibility of further enhancements to the attacks with a new version of the PAR algorithm that blurs the adversarial example together with the original unperturbed input, making it harder to detect. Results show that the f1-score of the trained detector drops from 0.97 to 0.89 with the new version of the PAR algorithm. The study highlights the importance of continued research into the optimization, defenses, and potential severity of sparse attacks, a crucial step toward ensuring the safety of deployed systems.

PREFACE

The supervision of my thesis was provided by Professor Jingyue Li at the Institute of Computer Science and Informatics at NTNU. I would like to express my gratitude for his invaluable guidance, mentorship, and support throughout the course of the past year.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

List of all abbreviations in alphabetic order:

- **ASR** Attack Success Rate
- **BA** Boundary Attack
- **CIFAR** Canadian Institute for Advanced Research
- **CNN** Convolutional Neural Network
- **DNN** Deep Neural Network
- **ES** Evolution Strategy
- **FLOPS** Floating Point Operations
- **GAN** Generative Adversarial Network
- **HSJA** HopSkipJumpAttack
- **MNIST** Modified National Institute of Standards and Technology database
- **PAR** Patch-wise Adversarial Removal
- **PGD** Projected Gradient Descent
- **ResNet** Residual Network
- **SOTA** State-of-the-art
- **ViT** Vision Transformer

CHAPTER

ONE

INTRODUCTION

Deep Neural Networks (DNNs) have been increasingly popular due to an increase in available computational power, and the networks' abilities to make accurate predictions in tasks such as image classification. As a result, these trained models have found widespread application in real-world domains such as diagnosing diseases [1, 2], autonomous driving [3, 4], security and surveillance [5], content moderation [6] and many other areas.

Failures in the classification models used in several of these systems can have severe consequences. Despite their performance, it has been demonstrated that these DNNs are vulnerable to maliciously crafted perturbations [7, 8]. This can have significant repercussions in applications such as autonomous cars that rely on these models to navigate and read traffic signs. Because of this, continued research into adversarial attacks and defenses is important to evaluate and improve the robustness of DNNs used in critical systems.

Adversarial attacks are typically divided into two categories: white-box and black-box attacks. A white-box attack assumes access to internal information of the target model, such as the underlying architecture and the model gradients. In deployed real-world applications of DNNs, this information is usually not available to a potential attacker. In the black-box scenario, the adversarial input is crafted only from the output predictions of the model. In this environment, some attackers exploit the transferability of DNNs [9, 10, 11] by training a surrogate model and then transferring these attacks to the target model. However, training a surrogate is not always feasible or practical, limiting the available information to the output labels of the target model. Attacks that are restricted to only the output labels are called decision-based attacks.

Decision-based attacks are evaluated by their similarity to benign inputs, measured by the size of the added perturbations. Dense attacks, attacks where this similarity is measured by the L_2 or L_∞ norm, have been explored in a variety of studies [12, 13, 14, 15]. Sparse attacks, where the L_0 norm measures the distortions, are limited to only a handful of studies [16, 17]. This limits our understanding of the threat of sparse attacks, and if current defensive techniques are as effective against sparse attacks. This warrants a need for additional research in improving

end evaluating the robustness of DNNs against sparse attacks. In this thesis, we investigate how we can create more query-efficient sparse attacks, and evaluate several defensive techniques. Finally, we propose a modified initialization algorithm to create an attack that is harder to defend against.

This thesis is divided into eleven chapters. In *introduction* and *related work*, we present the reader insight into the background and motivation for our research. In the chapter *methodology*, we present our motivation and research questions that we aim to answer through experiments in chapters 4 to 9. We discuss our results in *discussion* and suggest some directions for future work. In *conclusion* we summarize our findings.

CHAPTER

TWO

RELATED WORK

2.1 Attacks targeting DNNs

In a decision-based black-box attack, the adversary has access to the top predicted labels of the target network [18]. All the attacks we present in this chapter focus on the top 1 label output, the target model's highest predicted class, as this usually is the only available output of commercially available classifiers.

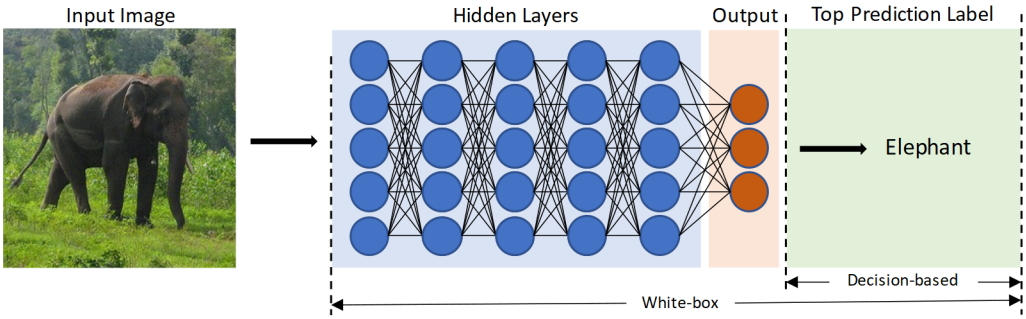


Figure 2.1.1: In a white-box attack the attacker typically has access to the model weights and gradients of all layers. In a decision-based attack, the attacker is restricted to only the output label.

We separate targeted attacks from untargeted attacks [19, 20]. In the untargeted setting, the attacker aims to deceive the target classifier into predicting any label other than the original label. In the targeted setting, the attacker selects a target label and creates perturbations to an original image such that it is classified as the target label. For example, an attacker may have an image of a dog and want the target model to classify it as a cat. The attack is considered successful if the image is classified as a cat, and not as a dog or any other class e.g. a horse. Some attacks focus on either setting, while many algorithms can be used for both targeted and untargeted attacks.

In the following sections, we look at some recent dense- and sparse attacks. There are mainly two areas in which they have improved the previous SOTA: the attack

success rate (ASR), and their query efficiency. The ASR determines the number of attacks that are successful under a given distortion threshold. The query efficiency metric measures the number of queries to the target model used to generate a successful attack below a predefined threshold, or how low perturbations are after a set amount of queries.

2.1.1 Dense Attacks

In the dense attacks, adversarial perturbations are measured by the L_2 or the L_∞ distance.

The L_2 distance is also known as Euclidean distance. It is calculated as the distance between two vectors in an n-dimensional space:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2} \quad (2.1)$$

The L_∞ distance is given by the largest distance between the original and the adversarial image:

$$d(p,q) = ||p - q||_\infty \quad (2.2)$$

In 2017, the Boundary Attack (BA) [21] emerged as the first decision-based attack that got significant attention. Prior to this, the majority of publications predominantly focused on white box attacks or transfer-based black box attacks. Chen et al. published HopSkipJumpAttack (HSJA) in 2020, inspired by the Boundary Attack, but highlighting the need to optimize for query efficiency. They achieved this with a novel estimation of the gradient direction at the decision boundary, drawing inspiration from zeroth-order optimization [18]. After the release of HSJA, several other dense attacks have been proposed to create even more query-efficient attacks. We give a quick overview of these attacks, most of them compare their results directly to HSJA.

GeoDa [22], Triangle Attack [23], and SurFree [24] are three geometry-based attacks that take advantage of the observation that the decision border of a DNN usually has a small curvature when in close proximity to a data point. Meho et al. [24], the creators of SurFree, found that they could create a more efficient attack on tight query budgets when disallowing any queries to estimate the gradient, and instead investigate more directions for descent at the decision border. Query-efficient boundary-based attack [25] and NonLinear-BA [26] are two attacks that have improved by exploiting prior knowledge about how information is structured in images to reduce the search domain.

2.1.1.1 PAR

Patch-wise Adversarial Removal (PAR) is a dense attack created by Shi et al. [27]. They investigate how different regions of an image are affected by noise. PAR operates by deviding the image into patches, and gradually removing noise

while maintaining the adversarial nature of the image. This reduction process starts with larger patches and iteratively progresses to smaller patches, prioritizing patches with higher noise magnitude. The modified image is queried against the target model. If the image remains misclassified, the corresponding patch is discarded. This process is repeated for all the patches. Once all patches have been processed, the remaining ones are halved in size, and the loop is repeated. The final result is an adversarial example that retains only the "essential" noise required to deceive the classifier.

The PAR algorithm starts by first initializing a noise sensitivity mask M_S and a noise magnitude mask M_N. The masks are set to the same size as the image but without the color channels. The full algorithm from the PAR-paper is given in algorithm 1 [27].

Algorithm 1 Patch-wise Adversarial Removal

Input: Target model $F(x)$, noise magnitude limit τ , original image x and label y
Max querying number T , initial variance of gaussian distribution var
Identity matrix I of the same dimension as x, initial and minimum patch size PS_0 and PS_{min}
Output: Adversarial example x^* with compressed noise
while $F(x^{init}) = y$ **do**
  $\xi^{\text{init}} \sim \mathcal{N}\left(0, \text{var}\ ^2 I\right), \quad x^{\text{init}} \leftarrow \text{Clip}_{x,\tau}\left\{x + \xi_0^{\text{Gau}}\right\}, \quad \text{var} \leftarrow \text{var} * 2, \quad T \leftarrow T - 1$
end while
Initialize M_N and M_S, $PS \leftarrow PS_{0,x^*} \leftarrow x^{init}$
while $T > 0$ **do**
  $M_Q \leftarrow M_N \odot M_S$
  if $\sum M_Q = 0$ **then**
    $PS \leftarrow PS/2$, initialize M_N and M_S
  end if
  if $PS \leq PS_{min}$ **then**
    break
  end if
  $row^*, col^* \leftarrow argmax(M_Q)$
  $z^{\text{query}} \leftarrow x^* - x, \quad z^{\text{query}}_{\text{row}\ *PS+1:(\ \text{row}\ *+1)*PS,\text{col}^*PS+1:(\text{col}^*+1)*PS} \leftarrow 0$
  $x^{\text{query}} \leftarrow \text{Clip}_{x,\tau}\{z^{\text{query}} + x\}$

  if $F(x^{query}) \neq y$ **then**
    $x^* \leftarrow x^{query}$, update M_N
  else
    $M_{Srow^*,col^*} \leftarrow 0$
  end if
  $T \leftarrow T - 1$
end while

Shi et al. [27] evaluate the algorithm by comparing it to other SOTA dense attacks on three datasets, including ImageNet. The attacks are measured by the final L_2

distances on query budgets. PAR performs better than several algorithms but is beaten by the newest of the decision-based attacks SurFree [24] and CAB [28] for some experiments. However, Shi et al. found that PAR can be used in combination with the other attacks, which improved every single attack tested, on all datasets and against every model in their experiments.

2.1.2 Sparse Attacks

Sparse attacks are measured by the L_0 distance, which is given by the number of changed elements. In the context of images, this translates to the number of changed pixels in an adversarial example. Unlike the dense attacks, measuring by the L_0-norm leads to an NP-hard problem where a global minimum cannot be guaranteed to be found [29, 16, 30].

Most related work measuring by the L_0 distance are not decision-based attacks, relying on other information about the target model than just the output label. To the best of our knowledge, there exist only two previous decision-based sparse attacks, PointWise [17] and SparseEvo [16]. In the following sections, we present these two attacks in more detail.

2.1.2.1 PointWise

Schott et al. used an algorithm they call Pointwise in their paper on testing the robustness of CNNs [17]. They employ a simple search algorithm to minimize the L_0 distance, making it the first available sparse decision-based attack we have found.

PointWise works by adding salt-and-pepper noise (see Chapter 4.4) until the image is misclassified by the target model. Following this, every single perturbed pixel is tested one by one, resetting its value to the original colors, and checking if the image is misclassified. If the adversarial example still fools the classifier, the pixel is removed. The attack is complete when every perturbed pixel is checked. In a general sense, PointWise can be seen as a very simple version of PAR, but without grouping the perturbed pixels into patches and without any prioritization of which regions should be explored before others.

In their experiments, Schott et al. [17] find that the accuracy of an unspecified CNN trained on the MNIST dataset is reduced from 99.1% to 19.9% when attacked. The primary objective of PointWise seems to test the resilience of CNNs, not necessarily to create an efficient sparse attack.

2.1.2.2 SparseEvo

SparseEvo is a decision-based sparse attack that uses a modified evolution strategy-based (ES) algorithm to create query-efficient attacks [16]. SparseEvo works as a black-box optimization algorithm, which makes it well-suited for addressing the NP-hard problem inherent to sparse attacks.

Exploring a large image with three color channels is both expensive and inefficient. To address this, Vo et al. disregard colors and treat the pixels as image coordinates that are either perturbed or original. This reduction transforms the search domain from $\mathbb{R}^{C \times W \times H}$ to $0, 1^{WxH}$. They modify the evolution algorithm to evolve binary representation of images and find this simplification to be very efficient at creating sparse attacks, significantly outperforming the previous SOTA PointWise.

The evolution algorithm works by initializing a population, and then through recombination and mutation, evolving new candidates closer to a desired solution. In the SparseEvo algorithm, this is done with binary vectors where each bit represents a pixel in the adversarial example. For each pixel, if the bit is set to one, the pixel is gathered from some noise. If the bit is set to zero, the pixel is the same as in the original starting image.

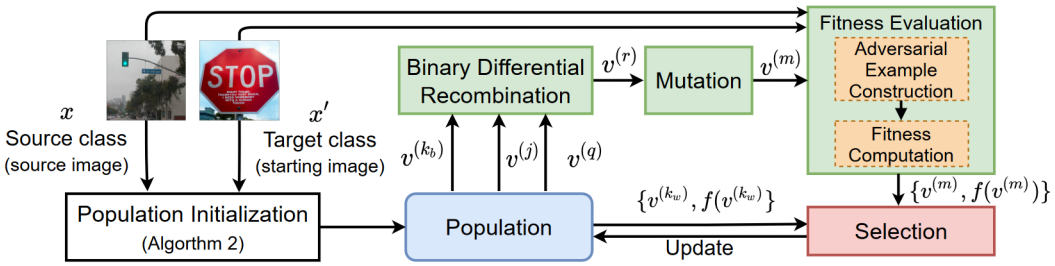


Figure 2.1.2: Overview of the algorithm from the SparseEvo paper [16].

In the following sections, we will explain the individual parts of the binary evolution process, and then give an overview of the total algorithm as presented in [16].

Recombination and Mutation. Vo et al. found that recombination worked best in their empirical results by combining the best $v^{(\mathrm{k_b})}$, and two other random $v^{(\mathrm{j})}, v^{(\mathrm{q})}$, candidates. The two random candidates form a new temporary recombination by uniform crossover, where each bit is chosen from each candidate with equal probability. The best candidate is then altered by setting all its 1-bits to 0 if they are set to 0 in the offspring from the other two candidates. This creates the new recombined candidate $v^{(\mathrm{r})}$. Equation 2.3 shows the recombination process to generate the new candidate.

$$v^{(\mathrm{r})} \leftarrow v^{(\mathrm{k_b})} \odot \text{ UniformCrossover } \left(v^{(\mathrm{j})}, v^{(\mathrm{q})}\right) \tag{2.3}$$

$v^{(\mathrm{r})}$ is then mutated by setting a random fraction μ of all 1-bits to 0.

Fitness Computation and Selection. The candidates are sorted and evaluated by giving them a fitness score set to their L_2 distance from the starting image if they are adversarial, and ∞ if they are not. If the offspring created by recombination and mutation is better than the worst candidate in the previous iteration,

it takes its place in the population, and the old candidate is discarded.

Algorithm. Algorithm 2 gives an overview of the entire SparseEvo evolution process. The algorithm is almost identical to the listed algorithm in the SparseEvo source paper [16].

Algorithm 2 SparseEvo

Input: Source image x, starting image x', source label y, target label y^*, model f, population size p, initialization rate α mutation rate μ, query limit T
$t \leftarrow 0; \mathbb{V}, G \leftarrow InitialisePopulation(x, x', f, p, \alpha)$
$k_w \leftarrow argmax(G), k_b \leftarrow argmin(G)$
for $t = 1, 2, ..., T$ **do**
  Uniformly select $v^{(\mathrm{j})}, v^{(\mathrm{q})} \in \mathbb{V} \backslash v_{k_b}$ at random
  Yield $v^{(\mathrm{r})}$ using equation 2.3 and $v_{k_b}, v^{(\mathrm{j})}, v^{(\mathrm{q})}$
  Yield $v^{(\mathrm{m})}$ by uniformly altering a fraction μ of all 1-bits of $v^{(\mathrm{r})}$ at random
  Construct $\tilde{x}$ with x, x' and $v^{(\mathrm{m})}$
  Calculate $g(\tilde{x})$ to compute fitnesses
  if $g(\tilde{x}_0) < G_{k_w}$ **then**
    $G_{k_w} \leftarrow g(\tilde{x})$
    $v_{k_w} \leftarrow v(\mathrm{m})$
  end if
  $k_w \leftarrow argmax(G), k_b \leftarrow argmin(G)$
end for
Construct $\tilde{x}$ with x, x' and $v^{(\mathrm{k_b})}$
return $\tilde{x}$

Vo et al. evaluate SparseEvo on the two datasets CIFAR10 and ImageNet. On the ImageNet dataset, PointWise struggles to produce efficient attacks with a 20k query budget, whereas SparseEvo generates very sparse adversarial examples. SparseEvo is also compared to the white-box attack PGD_0 [31]. SparseEvo matches the ideal white-box attack in attack success rate on limited query budgets of 200 and 500 queries, especially as the sparsity threshold nears $L_\infty = 0.1$. While better than the previous state-of-the-art PontWise, the SparseEvo algorithm suffers from low query efficiency and slow convergence in the earlier stage of the evolution process.

2.2 Defenses

There exists an extensive set of published studies on defending against adversarial attacks. Most focus on defenses against white-box attacks, but several of the defenses are also applicable to black-box attacks. Based on the survey papers [32, 19, 33, 20], we can group defensive techniques into the following categories:

- Restricted Model Access
- Input Transformation

- Adversarial Training
- Adversarial Detection

2.2.1 Restricted Model Access

Restricted model access includes masking or obfuscation of model gradients and restricted query access. Other than the obvious method of restricting the available information of the target classifier, there have been a set of other proposed techniques. Some techniques that aim to obfuscate the gradients include randomization of gradients [34], gradient regularization [35, 36], adding masking layers that mask features sensitive to adversarial manipulation before the final classification [37], and feature squeezing [38].

Hinton et al. propose defensive distillation where a smaller network is trained on the softened outputs of the original model [39]. The works by [40, 41] demonstrate how this technique can be used to train a model that is more robust to the small perturbations introduced by adversarial attacks. Chen et al. tested their dense attack HSJA against a defensive distilled model on the MNIST dataset. They found that their algorithm performed on par with the white-box attack C&W [41] given enough queries (10k - 50k) [18].

In their survey paper, Xu et al. argue that gradient masking/obfuscation methods are not safe, as they can only confuse or mislead adversaries, not eliminate the existence of adversarial examples [20]. This is backed by the works of Athalye et al. who find that defenses relying on these techniques can be circumvented and instead suggests adversarial training as a more robust defense [42].

2.2.2 Input Transformation

Input transformation is a defensive method that lies between the input and the model, modifying the input image to prevent or decrease the effect of adversarial attacks.

Dziugaite et al. [43] and Das et al. [44] have found JPEG compression to be an effective way to increase robustness against some white-box attacks. Xie et al. used random resizing of images to increase robustness [45]. Others try to reduce the precision of the input data to minimize the effect of small perturbation, with techniques like median blur [46] or median filtering [38]. In a competition organized by Google Brain aimed to accelerate research on adversarial examples and robustness of classifiers, several of the best-performing submissions used median filtering as part of their defensive techniques [47].

Samangouei et al. propose a more complex approach to input transformation called Defense-GAN [48]. This method uses a generative adversarial network (GAN) [49] that learns the data distribution of the clean dataset, and by mapping the input back to this distribution as a reconstructed input, becomes more resistant to adversarial perturbations.

2.2.3 Adversarial Training

Adversarial training is a defensive method where adversarial examples are introduced to the training data to create models that learn robustness to perturbations. It is perhaps seen as the most robust defensive technique available [42, 50].

Szegedy et al. introduced adversarial examples to their training data under a new label to enhance the resilience of their model [7]. Goodfellow et al. improved the accuracy of their model from 18% to 89% on perturbed MNIST images using an adversarially trained model [8]. These models were trained on adversarial examples generated by non-iterative attacks. To address this limitation, Madry et al. suggest training models using the iterative attack PGD [32], showing state-of-the-art robustness on MNIST and CIFAR-10 datasets [51].

Vo et al. [16] tested their sparse attack, SparseEvo, against an adversarially trained ResNet-18 on the CIFAR10 dataset. They found that their algorithm obtains a comparable performance to the white-box attack PGD on query budgets under 500. They did not include any results from datasets with larger images.

2.2.4 Adversarial Detection

Adversarial detection is another effective way of protecting classifiers. In this defensive method, the input is firstly considered as either benign or adversarial. If the input is considered adversarial, the model can refuse to predict the class, or give some random output to mislead the attacker.

Grosse et al. included an extra label for adversarial examples when training the classifier. In another experiment, they use statistical analysis to check if two datasets are drawn from the same distribution in order to detect adversarial examples [52]. Gong et al. train a separate binary classifier to filter adversarial examples before they reach the main classifier [53]. Metzen et al. also used an external detection classifier but gives it the input of the hidden layers of the main classifier instead of the input image [54]. Other addon-networks attempt to detect adversarial examples with a trained GAN [55, 56], or complete frameworks like MagNet [57] and SafetyNet [58].

Some researchers have changed the input or the model parameters slightly to check the consistency of the model's predictions [38, 59, 46]. The idea is that the model will predict benign examples consistently, but not adversarial.

CHAPTER

THREE

METHODOLOGY

In this chapter, our methodology will be explained along with our motivation for our experiments. We propose some research questions based on this discussion.

3.1 Motivation

As discussed in Chapter 2.1.2, sparse attacks present a difficult optimization problem, and there is a lack of research that focuses on sparse attacks. This has led to sparse attacks, at their current stage, performing worse and requiring more queries than the SOTA dense attacks. While more research on sparse attacks may not necessarily lead to attacks that can outperform the dense attacks, their separation from the focus on the decision border and unique ways of generating adversarial solutions may give the family of attacks an advantage against defenses that focus on the techniques currently used for generating dense attacks. We believe this warrants further investigation into developing query-efficient sparse attacks and evaluation of their response to different defensive techniques.

A current constraint for sparse decision-based attack methods is the need to query the target model with many queries to create sparse attacks. Previous state-of-the-art methods Pointwise [17] and SparseEvo [16] need thousands of queries to create sparse targeted attacks that are hard to detect with the human eye. This is especially true in the targeted setting where SparseEvo requires around ten thousand queries to create images that are visually indistinguishable from the starting image [16].

3.2 Research Questions

Based on the research motivation, we decide to answer the following three research questions:

- **RQ1:** How can we create more query-efficient and sparse decision-based black box attacks?

- **RQ2:** How effective are existing defensive techniques against sparse decision-based attacks for mitigating or detecting the threat of the attacks?
- **RQ3:** Given the attacks can be defended by the defensive methods properly, are there other possibilities to enhance the sparse decision-based attacks further?

In the next chapters, we aim to answer these research questions through experiments.

CHAPTER

FOUR

DESIGN AND IMPLEMENTATION TO ANSWER RQ1: QUERY-EFFICIENT ATTACKS

In this chapter, we propose an answer to our first research question: how can we create more query-efficient and sparse decision-based attacks? Based on the results by Shi et al. [27], we believe exploring their greedy initialization technique PAR, can significantly reduce the number of queries needed to create sparse attacks on a query budget. Shi et al. only test their algorithm in the dense attack setting. As opposed to dense attacks, when creating sparse attacks the distortions are measured directly by the number of adversarial pixels we can remove. PAR is a noise removal algorithm, and because of this, we believe PAR is even more suited to improve attacks in the sparse setting.

We start by presenting PAR as an initialization technique and then propose increasing the dimensionality of the problem to improve sparsity. Following this, we define experiments and improvements for targeted- and untargeted attacks. We want the attack to be applicable to both targeted and untargeted settings as they provide distinct perspectives on the vulnerabilities of image classifiers. Targeted attacks will tell us how an attacker with specific goals or targets can exploit our classifier. Untargeted attacks give an indication of the system's overall resilience against manipulation. Finally, we explain the experiment setup and the evaluation criteria.

4.1 Initialization by PAR

We use the PAR algorithm by Shi et al.[27] presented in Chapter 2.1.1.1 in order to quickly decrease the grid of adversarial pixels to reduce in the subsequent binary evolution method used in the SparseEvo algorithm [16]. This can be seen as a greedy initialization as all patches considered, in a coarse to fine search, that can be removed while the image is still adversarial are removed and the pixels in those patches are never considered again. This might quickly decrease the number of adversarial pixels fast, and thus the sparsity, but might discard solutions that would be closer to an optimal distribution of adversarial pixels. Figure 4.1.1 gives two random examples in the targeted setting from the ImageNet dataset. The

algorithm removes patches from a starting adversarial image, and in the process reveals the original image behind it. The top row of each example shows the created adversarial example. The bottom row is a visualization of the patches, black squares are areas that are seen as unnecessary to keep the image adversarial by the algorithm.

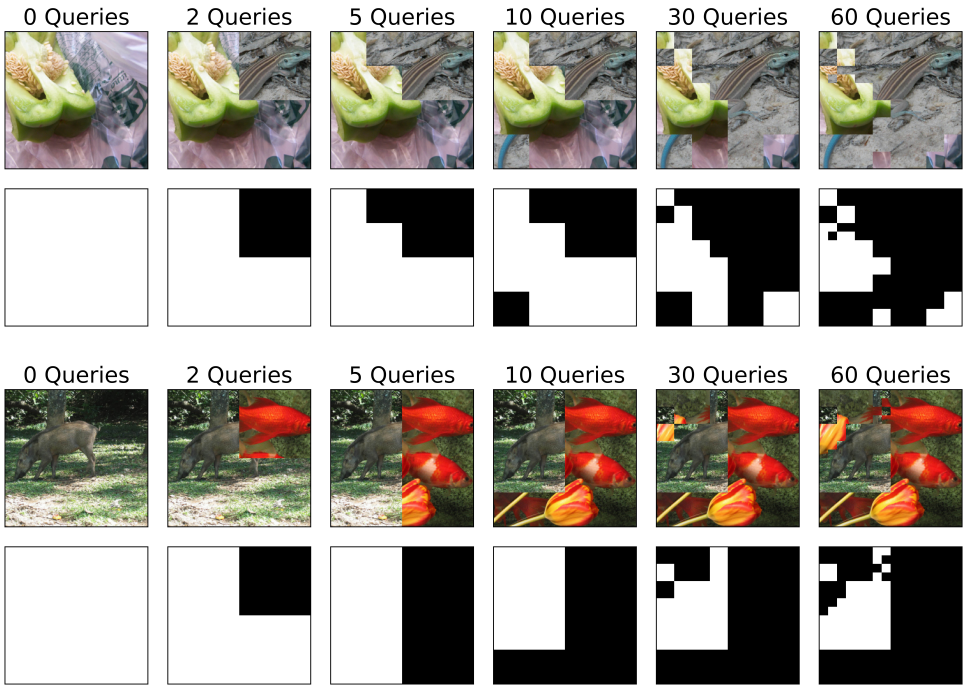


Figure 4.1.1: Two random examples of the PAR algorithm [27] on the ImageNet dataset in the targeted setting. The top row of each example shows the created adversarial example. The bottom row is a visualization of the patches, black squares are areas that are seen as unnecessary to keep the image adversarial by the algorithm. The algorithm starts with larger patches and decreases the patch size during later queries.

4.2 Increased Dimensionality

In SparseEvo [16], Vo et al. ignore the color intensity of pixels entirely to reduce the search domain to either on/off (perturbed/original) for an individual pixel. We hypothesize that if we can utilize PAR to sufficiently reduce the original image to a smaller subgrid of adversarial pixels, we could open for evolving a sparse image using the images' colors in three dimensions instead of the binary restrictions in the original SparseEvo algorithm, and in turn, find even better solutions.

Disregarding the binary optimizations made by Vo et al. we are no longer limited to just the ES algorithm. We evaluate the performance of ES along with other more recent gradient-free optimization algorithms:

- sep-CMA-ES (Separable Covariance Matrix Adaptation Evolution Strategy), a variant of a newer ES algorithm, CMA-ES. The distribution in CMA-ES

contains a mean vector and a covariance matrix. In sep-CMA-ES, the objective function is decomposed into independent or weakly dependent components, allowing for enhanced optimization speed and scalability to higher dimensions (like color images in the ImageNet dataset) [60].

- Nevergrad, a powerful framework by Facebook AI that contains a variety of algorithms for black-box optimization and hyperparameter tuning [61].
- LaMCTS, a lightweight variant of the Monte Carlo Tree Search (MCTS) algorithm. It is a fast and efficient search algorithm that employs a combination of random simulations and tree exploration to find optimal solutions in large black-box optimization problems [62].

4.3 Targeted Attacks

Figure 4.3.1 shows the targeted attack pipeline using PAR as the initialization method. Starting from the original image, a random image from the target class is used as an overlay. After that, we remove patches in iterations while still keeping the image adversarial. When the initialization algorithm is unable to remove any more patches, we continue with binary ES as in SparseEvo on the reduced subgrid. All three images to the right of the original image of a hyena are classified as a honeycomb by the target model.

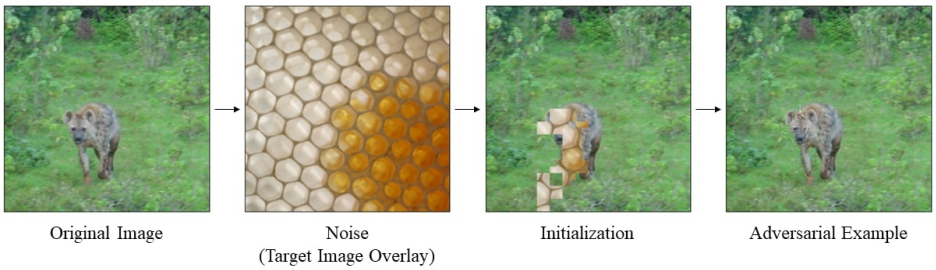
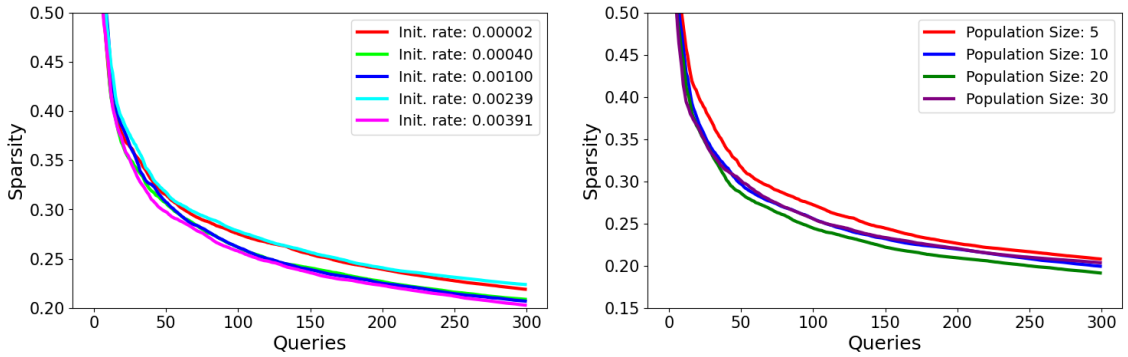


Figure 4.3.1: Targeted attack pipeline, using PAR as the initialization method to reduce the noise overlay.

We found it necessary to use different hyperparameters than in SparseEvo to create viable targeted attacks. As PAR reduced the problem to a smaller subgrid, the resulting areas were a lot more sensitive to the removal of pixels than when using the entire starting image. This meant that the old mutation rate led mostly to examples that were not adversarial. We did some experiments to find mutation rates that would lead to fast and efficient convergence after PAR and found that limiting the mutation rate to one pixel for the first 300 queries, and three pixels for the following 200 queries gave great results. After the first 500 queries, we continued with a mutation rate of 0.001 as in the original algorithm. We also tested different values for population size and initialization rate but found that they had a small impact on the performance of the attack. In Figure 4.3.2 we give a comparison of various tested values for initialization rate and population sizes. After 300 queries there is little difference in sparsity for the tested hyperparameters, and the differences might be attributed to the small test size of 50 images

per hyperparameter setting. In our experiments, we used an initialization rate of 0.0004 and a population size of 10. Our hyperparameter settings are summarized in Table 4.3.1.



(a) **(b)**

Figure 4.3.2: To the left: Testing different values for initialization rate in the SparseEvo+PAR targeted attack. The initialization rate is the number of pixels that will be randomly set to 0 when initializing a population candidate. To the right: Comparing the performance of different population sizes for the same attack. The lines are given as the average across 50 runs with random images.

Parameters	Value
Population size (p)	10
Initialization rate (α)	0.0004
Mutation rate (μ)	$\begin{cases} \frac{1}{\text{num\_pixels}} & \text{if t} < 200 \\ \frac{3}{\text{num\_pixels}} & \text{if } 200 \leq t < 500 \\ 0.001, & \text{otherwise} \end{cases}$

Table 4.3.1: Hyperparameters for the experiments in the targeted setting.

4.4 Untargeted Attacks

For the untargeted attacks, some initial noise needs to be added to the image to make it adversarial. PointWise was developed using salt-and-pepper noise [17]. Vo et al. tested both Gaussian and salt-and-pepper noise in their untargeted version of the SparseEvo algorithm and found that the salt-and-pepper noise was the most effective [16].

Gaussian noise is added by sampling from a normal distribution and adding the values to the original image. The standard deviation can be modified to alter the spread or intensity of the noise.

Salt and pepper noise is a black-and-white type of noise that sometimes appear in digital images [63]. It is a form of impulse interference that happens during the

taking and processing of a digital image, typically over polluting the pixels (white pixels/salt) or introducing zero values (black pixels/pepper) [64]. This noise is efficient at fooling classifiers and can occur naturally in input images, which makes it a relevant source of noise for our studies. We can artificially create this noise by setting random pixels of our input image to 0 or 1 until the image is adversarial.

We test both Gaussian and salt-and-pepper noise in our new attack. Figure 4.3.1 shows the untargeted attack pipeline. Starting from the original image, random noise is added as an overlay. After that, we do as in the targeted attack, remove patches during initialization, and continue from a smaller subgrid with binary ES. All three images to the right of the original image are misclassified by the target model.

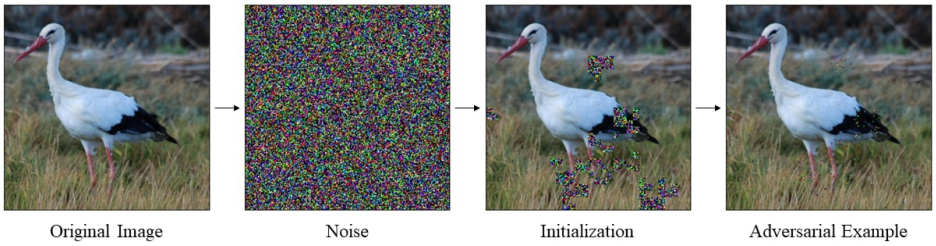


Figure 4.4.1: Untargeted attack pipeline, using PAR as the initialization method to reduce the noise overlay.

Unlike the targeted setting, the hyperparameters used in SparseEvo worked just as well in the untargeted setting when introducing PAR. These are given in Table 4.4.1.

Parameters	Value
Population size (p)	10
Initialization rate (α)	0.004
Mutation rate (μ)	0.004

Table 4.4.1: Hyperparameters for the experiments in the untargeted setting.

4.5 Target Image Selection

In the targeted attacks, including SparseEvo, selecting an overlay image as the starting noise in the targeted setting is done by selecting a random image from the target class. However, there might be a benefit in analyzing the target images available first, before creating the adversarial example. As a final experiment, we analyze the relationship between the initial L2 distance of the original image and the overlay image, and the L0 distance after initialization by PAR. An analysis of the initial L2 distances does not require any model queries, and our hypothesis is that selecting target images based on this distance can lead to better attacks.

4.6 Experiment Setup and Evaluation

We use the renowned computer vision dataset ImageNet to evaluate our experiments [65]. This was chosen as we believe larger images to be closer to real-world classification applications. Other common computer vision datasets such as MNIST [66], CIFAR-10 and CIFAR-100 [67], contain too small images to give an accurate indication of the attack's efficiency in images with some detail. A potential downside to ImageNet is a large amount of label classes, and label classes that are similar to each other that are meant to teach the model fine-grained classification. Especially in the untargeted setting, this makes it very easy to obfuscate small parts of an image, and have the model guessing wildly between a set of similar images. As an example, there are 118 classes (11.8%) of different dog breeds in the dataset

As the target classifier, we use the state-of-the-art ResNet architecture [68]. Specifically, we use a version created for ImageNet called ResNet-50 that has a 76.15% accuracy in predicting the label of unperturbed images in the dataset. The model is provided for download by torchvision [69]. The full architecture is shown in Appendix A. In total $3.8 * 10^9$ floating point operations (FLOPS) are needed to perform a single forward pass through this network.

The model takes inputs of size 224 x 224 x 3. To achieve this and ensure the images are in the same format as during the training of ResNet-50, every image in the dataset is preprocessed before being passed to the network. This is done by [70]:

- Resizing image to 256 (smallest edge will be 256) using bilinear interpolation mode
- Center crop to size 224
- Rescale values to [0.0, 1.0]
- Normalize values using mean=[0.485, 0.456, 0.406] and std=[0.229, 0.224, 0.225]

We evaluate PAR as an initialization technique by measuring how much of the original adversarial noise can be removed while keeping the image adversarial and how many queries are needed to do so. This is tested for both the untargeted and the targeted setting. We then evaluate the combined SparseEvo+PAR attack measuring the ASR and query efficiency at selected thresholds. We compare these results to the original SparseEvo algorithm for both targeted and untargeted attacks.

CHAPTER

FIVE

RQ1: RESULTS

5.1 Initialization by PAR

In the SparseEvo algorithm, once a bit is flipped from 1 to 0, it is never flipped back for that offspring. In our experiments, we tested allowing the evolution process of the method access to the whole image, not just the reduced grid after PAR initialization. To utilize this space, it would require the algorithm to be able to flip pixels from the original pixel, back to the adversarial pixel. Just like the authors of SparseEvo [16], we found that allowing flipping off-pixels back to on-pixels reduces the performance of the overall method, even when starting from a subsection of the whole image. This means that for all following experiments, adversarial distortions are limited to only the reduced grid after initialization, and pixels outside this area are untouched by the evolution process.

Figure 5.1.1 gives an illustration of PAR working to reduce the noise in a targeted setting. The first two columns are the original image and the target image. The next columns are the results after a selected number of iterations of PAR. Under each image is the predicted label by the target model, and a percentage indicating how much of the original image is present in the current image. Figure 5.1.2 shows the same in the untargeted setting. The second column in this figure is the images after salt-and-pepper initialization.

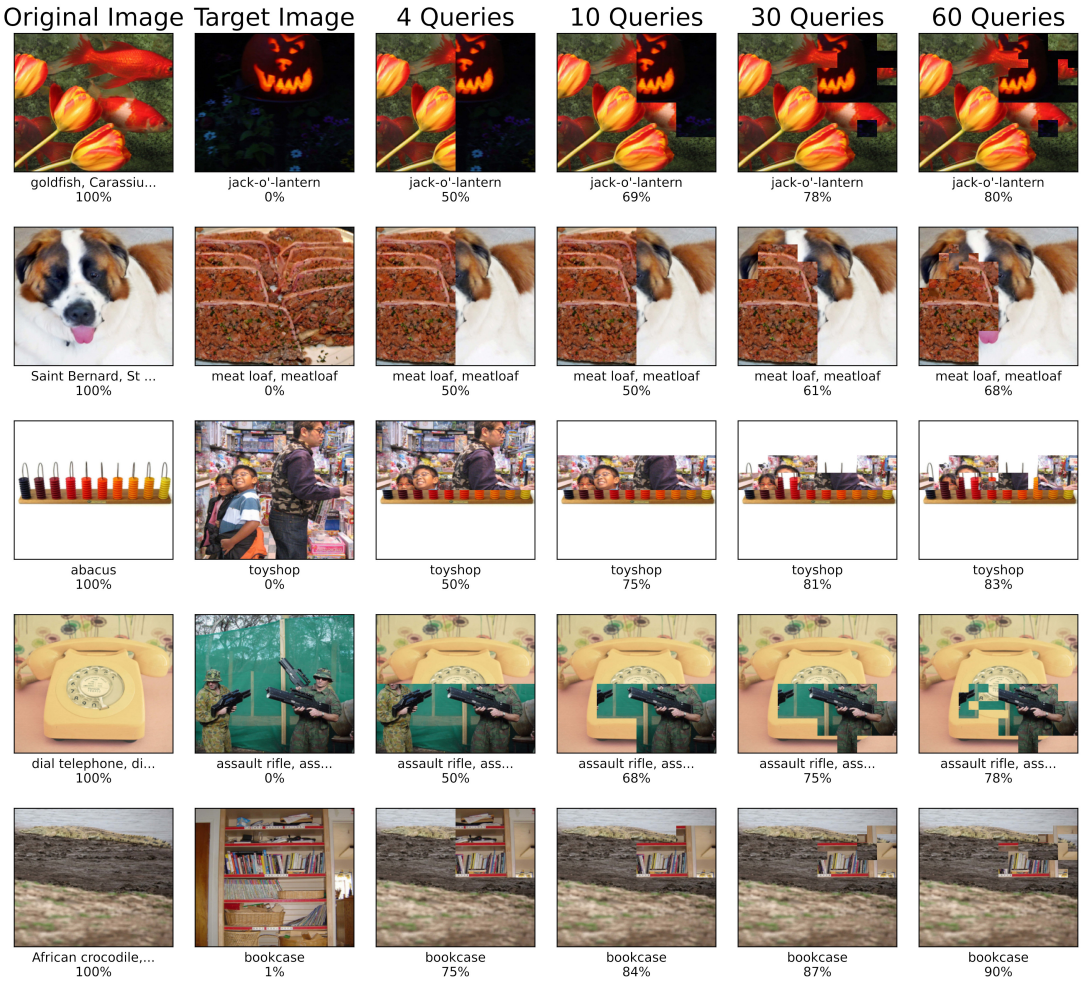


Figure 5.1.1: Examples of how the PAR algorithm reduces the noise in the targeted setting. Under each image is the predicted label and a percentage of how much of the original image is present.

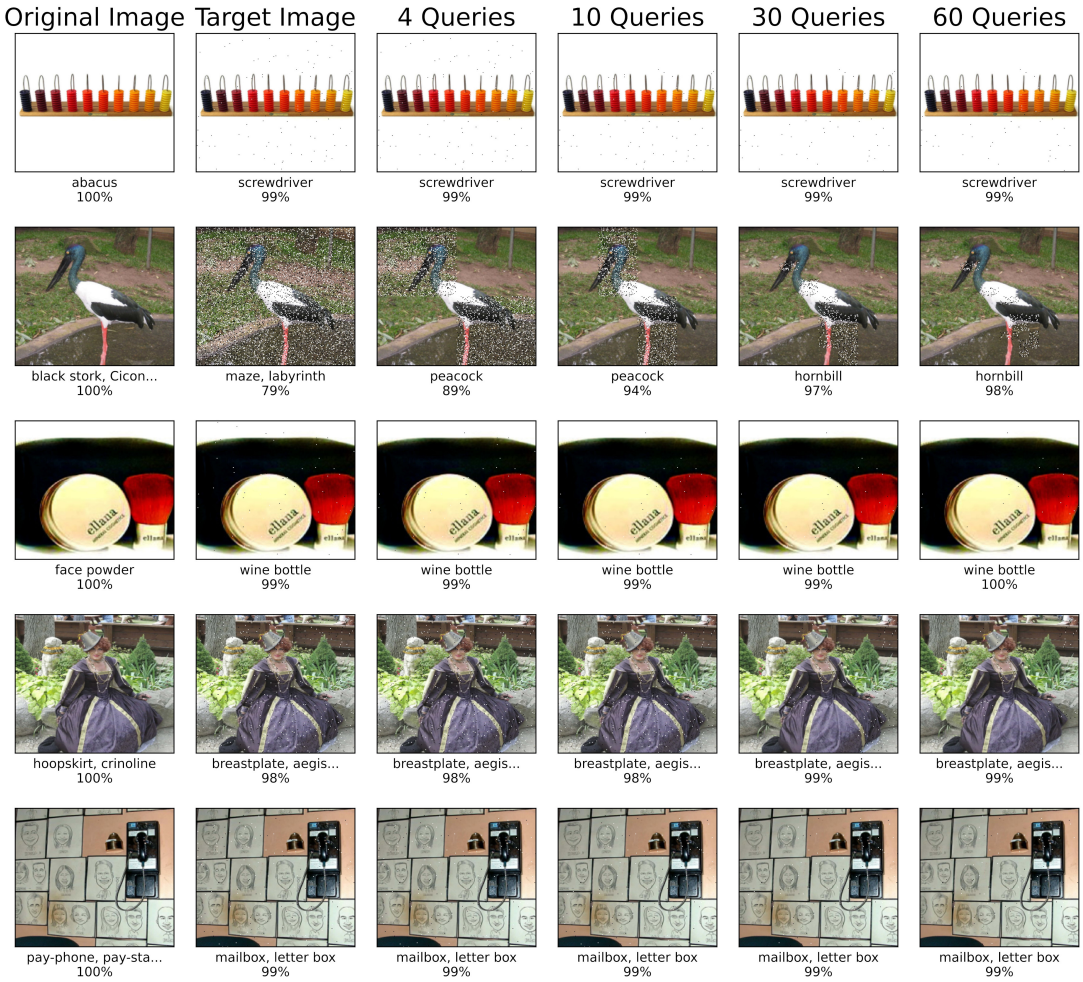


Figure 5.1.2: Examples of how the PAR algorithm reduces the noise in the untargeted setting. Under each image is the predicted label and a percentage of how much of the original image is present.

Inialization in the targeted setting: To see how much the PAR initialization help reduce the perturbation size, we test on 1000 randomly drawn images from ImageNet. In the targeted setting, the initialization reduces the adversarial perturbation to 18.5% of the original image size on average. Figure 5.1.3 shows the relationship between the queries used in initialization and the resulting perturbation size. For most of the runs, the algorithm creates quite sparse adversarial examples in 200 to 400 queries. As can be seen in the figure, there is a linear relationship between queries used for initialization and the resulting L0 distance, with less sparse examples requiring fewer queries.

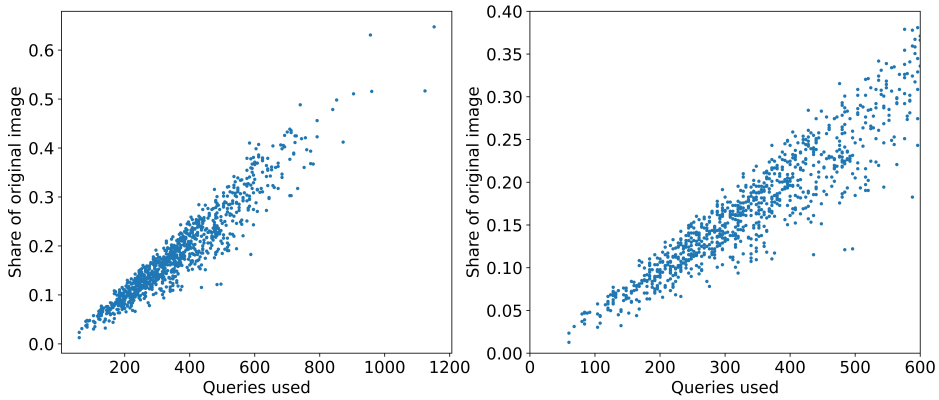


Figure 5.1.3: The reduced sizes after initializations using PAR, and the queries used to do so for 1000 different images in the targeted setting. The image to the right is a zoomed-in version of the left image.

Initalization in the untargeted setting: In the untargeted setting, the initialization includes salt-and-pepper to create an initial adversarial example, as in SparseEvo, before doing PAR. The PAR initialization reduces the original perturbation to around 0.05% on average. Figure 5.1.4 shows the relationship between the queries used and the final perturbation size in untargeted initialization. As in the targeted setting, the algorithm uses less than 400 queries for most runs, and there is a seemingly linear relationship between queries spent and the final L0 distance.

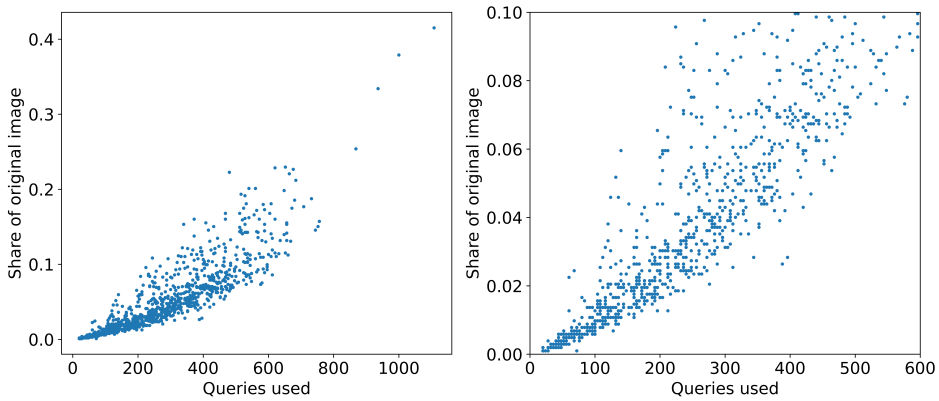


Figure 5.1.4: The reduced sizes after initializations using PAR, and the queries used to do so for 1000 different images in the untargeted setting. The image to the right is a zoomed-in version of the left image.

In Table 5.1.1 we summarize the final L0 distance and queries spent after initialization for targeted and untargeted attacks. In the untargeted setting, we have included results after using both salt-and-pepper and Gaussian noise to create the initial adversarial example before PAR. The distortions of the images in the targeted setting are about three times greater than in the untargeted setting. The

salt-and-pepper initialization led to better results in our experiments, and we continue using this method of creating initial adversarial examples for all following untargeted experiments.

Method	Avg. Adv. Pixels	Avg. L0	Median L0	Avg. Queries	Median Queries
PAR Targeted Attack	9282	0.185	0.168	365	348
PAR + S&P Untargeted Attack	2558	0.051	0.038	277	256
PAR + Gauss Untargeted Attack	2759	0.055	0.043	281	260

Table 5.1.1: Comparison of final L0-distances and queries used for initialization in the targeted and untargeted setting. The data is sampled from 1000 runs.

5.2 Testing increased dimensionality to improve sparsity

In Chapter 4.2, we hypothesized that with the reduced domain we could open for evolving sparse images using the images' colors in three dimensions instead of the binary restrictions posed by Vo et al. [16] This increase the dimensionality from $\{0,1\}^{WxH}$ to $\mathbb{R}^{3\times P}$, where $W \times H$ is the original image size and P is the number of perturbed pixels after initialization. We then tested several gradient-free optimization methods that are more recent and efficient than ES for a lot of different problems, including optimizations in problems of higher dimensions. Amongst the techniques tested were sep-CMA-ES [60], Nevergrad [61], and LaMCTS [62].

In our experiments, the increased dimensionality made the sparsity worse for all tested examples, even though the search grid was greatly reduced. Since we are measuring L0-distance, there is no gain to tweaking the color variance of the adversarial pixels if they do not contribute to higher sparsity overall. Since this was not the case, and because we adhere to strict query budgets, we found the restricted binary domain to be the preferred domain for further experiments. We continue the other experiments using the SparseEvo algorithm after PAR initialization.

5.3 Targeted attacks

In Chapter 5.1 we found that PAR can be used to remove about 80% of the adversarial perturbations in the targeted setting. For the next experiment, we combine the initialization with the SparseEvo algorithm to see if the combination can create a more query-efficient attack. In Figure 5.3.1, we see how adding the initialization greatly improves the performance of the method in the targeted setting. The improved method reduces the L0 distance very fast in the first few hundred queries

during the initialization, then converges steadily by reducing the reduced grid for the next thousand queries. The results are created as an average of 200 runs. The graph lines represent the average L0-distance for a given query. The line's shadow shows the standard deviation from the average, this goes for all similar graphs in the following chapters.

The SparseEvo method surpasses the version with PAR at around 8500/9000 queries on average. This is expected to happen at some point as the version without PAR will work on the entire image, while the new version with PAR is restricted to a smaller sub-image which will most likely discard the most optimal solutions that the other might find given enough time.

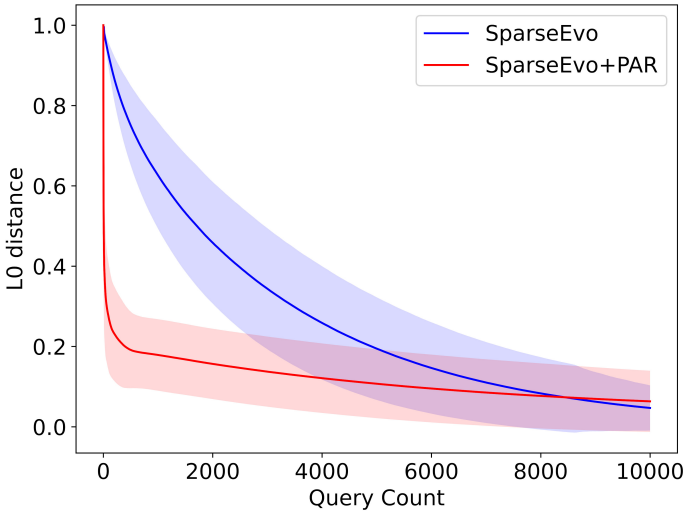


Figure 5.3.1: Comparison of the SparseEvo algorithm with and without PAR initialization. The lines mark the L0-distances at queries up to 10k in the targeted setting. The lines are given as the average of 200 runs.

Table 5.3.1 gives a comparison of the average L0 distance of the adversarial example produced by SparseEvo and SparseEvo with PAR, on different query budgets.

	q:50	q:250	q:500	q:1500	q:3000	q:5000	q:9000
SparseEvo	0.96	0.84	0.75	0.53	0.34	0.19	**0.06**
SparseEvo + PAR	**0.31**	**0.22**	**0.19**	**0.17**	**0.14**	**0.11**	0.07

Table 5.3.1: The average L0-distance after different query budgets with and without initialization in the targeted setting. Using PAR initialization improves the query efficiency up to around 9k queries.

Table 5.3.2 shows the attack success rate (ASR) at different sparsity thresholds. The methods are compared at four query budgets: 1k, 2k, 5k, and 10k queries. The added initialization greatly improves the ASR at lower query budgets. At a budget of 1k queries, the version without PAR has no successful attacks, while the version with PAR is successful in 89% of the samples with a sparsity threshold of

0.3. As previously shown, the plain SparseEvo method surpasses the ASR of the combined method at higher query budgets.

Method \Sparsity	**0.05**	**0.10**	**0.15**	**0.20**	**0.30**
SparseEvo - 1k	0	0	0	0	0
SparseEvo+PAR - 1k	5	20	41	65	89
SparseEvo - 2k	0	0	0	0	11
SparseEvo+PAR - 2k	10	31	52	72	92
SparseEvo - 5k	3	21	44	63	82
SparseEvo+PAR - 5k	32	58	74	85	96
SparseEvo - 10k	69	89	95	98	99
SparseEvo+PAR - 10k	64	78	86	91	98

Table 5.3.2: The ASR of SparseEvo with and without initialization using PAR. The algorithm with initialization has higher success rates at all sparsity thresholds on query budgets 1k, 2k, 5k and 10k.

5.4 Untargeted attacks

The PAR algorithm was able to greatly reduce perturbed noise in the untargeted setting. As with the targeted attacks, we see if the initialization method can be combined with SparseEvo to produce more query-efficient attacks in the untargeted setting. Figure 5.4.1 shows that initialization greatly improves the attack efficiency in the untargeted setting. Both versions with and without PAR first initialize noise using salt-and-pepper, which is typically done in about 50 queries as can be seen in the graph. The version with PAR then further reduces the size of the perturbation before the evolution part of the process. As with the targeted attack, it is expected that the plain SparseEvo algorithm will surpass the new method given enough queries. In our experiments, that does not happen on average during the first 500 queries, which is enough to produce very sparse adversarial attacks.

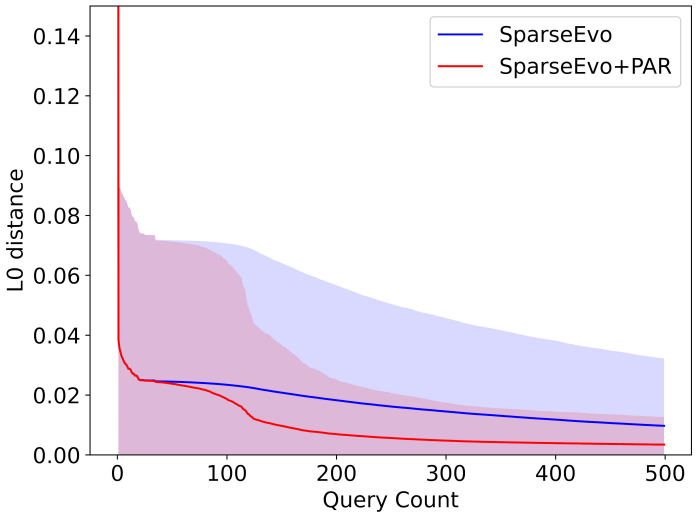


Figure 5.4.1: Performance of the SparseEvo algorithm with and without PAR initialization. The lines mark the L0-distances at queries up to 500 in the untargeted setting. The lines are given as the average of 200 runs.

Table 5.4.1 gives a comparison of the average L0-distance of the adversarial example produced by SparseEvo and the version with improved initialization, on different query budgets. The version with PAR is better on average for all queries up to 500 after the salt-and-pepper process.

	q:50	q:100	q:200	q:300	q:500
SparseEvo	**0.024**	0.023	0.018	0.015	0.009
New	**0.024**	**0.019**	**0.007**	**0.005**	**0.003**

Table 5.4.1: The average L0-distance after different query budgets with and without initialization in the untargeted setting. Using initialization improves the query efficiency for query budgets between 50 and 500.

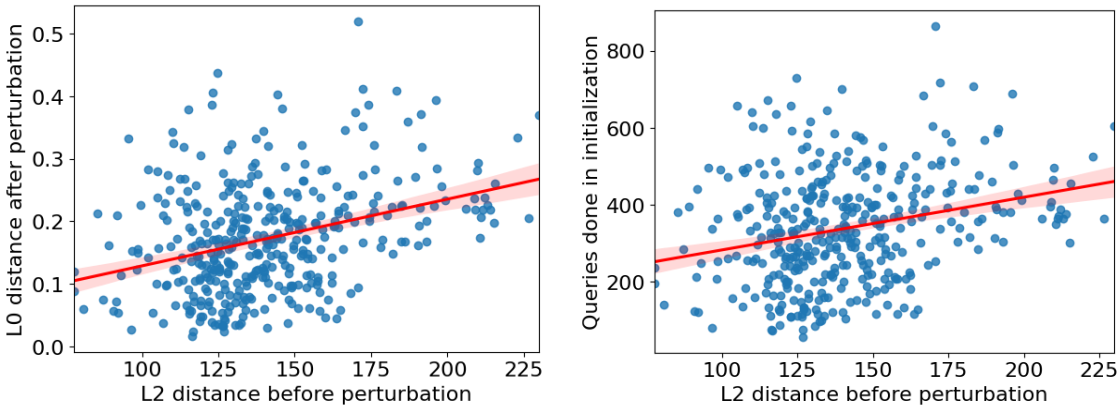
Table 5.4.2 shows the attack success rate (ASR) at different sparsity thresholds. The versions with and without PAR are compared at four query budgets: 100, 300, and 500 queries.

Method \Epsilon	0.001	0.002	0.005	0.010	0.020
SparseEvo - 100	28	34	49	65	76
SparseEvo+PAR - 100	35	48	63	75	83
SparseEvo - 300	40	46	58	73	83
SparseEvo+PAR - 300	48	62	80	91	96
SparseEvo - 500	47	55	68	81	88
SparseEvo+PAR - 500	53	69	86	95	97

Table 5.4.2: The ASR of SparseEvo with and without initialization using PAR. The algorithm with initialization has higher success rates at all sparsity thresholds on query budgets 100, 300, and 500.

5.5 Target Image Selection

We did an experiment to see if there is any relationship between the performance of the PAR initialization and the initial L2 distance of the original image and the overlay image used for targeted attacks. Figure 5.5.1 shows that, on average, there is a slight improvement in the queries needed and the resulting L0-distance after PAR when selecting an image from the target class with a lower L2-distance from the original image. Picking this image as the noise for targeted attacks can be a great way to improve the chance of a successful attack on query budgets, as the attacker does not have to query the target model to check the initial L2 distance.



(a) **(b)**

Figure 5.5.1: The relationship between the L2-distance before PAR and the **(a)** L0 distance after perturbation, and **(b)** queries used for the perturbation. The graphs are created from 10 runs of PAR, where each run has a random original image and label, a random target label, and 50 potential target images from the target class to use as initial noise (500 total samples). The red line marks the regression line with a confidence interval of 95%.

We did not use this selection for any other experiments as it was not directly relevant to comparing SparseEvo with or without PAR, or to evaluate defensive techniques.

CHAPTER

SIX

DESIGN AND IMPLEMENTATION TO ANSWER RQ2: DEFENSES

To answer our second research question and evaluate how effective existing defensive techniques are against sparse attacks, we first discuss some relevant defensive techniques. There has been a magnitude of proposed defenses against adversarial attacks. As discussed in Chapter 2.2, we can divide them into four main categories:

- Restricted Model Access
- Input Transformation
- Adversarial Training
- Adversarial Detection

Restricted model access is not relevant to the black-box attack setting as we already assume the gradients to be unavailable and that we have unrestricted query access. However, the possibility of query access restrictions as a defensive method is a good argument for the importance of query-efficient attacks.

Input transformation includes techniques such as rotation/shifts, compression, clipping, and filtering. Clipping, resizing, and normalizing the input data is already part of the pipeline when using the ResNet-50 model in our experiments. Attacks centered around these properties, e.g. exploding values or unnatural image sizes, would therefore not work as they will be neutralized by the input pipeline.

In the next sections, we present some defensive techniques that we think are most promising based on the SparseEvo+PAR algorithms and previous evaluations of defenses presented in chapter 2.2.

6.1 Adversarial Training

The final sparse adversarial examples created by the attacks are often reduced to a very small set of scattered pixels that throw off the model. This is especially true in the untargeted setting. The first defensive technique we test is an

adversarially trained model, that might give increased robustness for these small, but effective, distortions. The adversarially trained model might not be able to completely stop the attack, but can hopefully reduce the query efficiency and the ASR of the attacks. Adversarial training is, as a general defense, seen as the most robust defensive technique available [42, 50].

Instead of the vanilla ResNet-50 model provided by torchvision [69], we use an adversarially trained ResNet-50 model provided by Engstrom et al. [71]. Their model has a 57.90% accuracy on clean images, down from the 76.13% accuracy of the model used in previous chapters, but shows impressive robustness to distortions created by the white-box attack PGD [51].

We compare the robustness of the adversarially trained model to the normally trained model on ASR and query efficiency for untargeted and targeted attacks produced by SparseEvo with improved initialization using PAR.

6.2 Median Filter

The second defense we consider is a type of input transformation using a median filter. In the untargeted setting, random noise is added to the input image to create an adversarial attack before reducing the noise as much as possible to create a sparse attack. The effect of this noise can be reduced by running the input through a median filter before it is labeled by the target model. This filter is especially effective against salt-and-pepper noise [72, 73, 74], which is the initialization used in SparseEvo, as they found it to be the most effective way of initializing noise in their experiments [16].

Figure 6.2.1 illustrates the effect of the median filter on an image distorted with salt-and-pepper noise. The image after being put through the filter is very similar to the original image.

Figure 6.2.1: To the left: An image from the ImageNet dataset distorted with salt and pepper noise. To the right: The same image after a median filter has been applied. The image to the right is visually less distorted to the human eye but slightly blurred.

Firstly, we test the effect this filter has on the model on clean images, to ensure this filtering will not considerably reduce the overall accuracy of the model. In the following experiments, we test the efficiency of the median filter input transformation in untargeted attacks by measuring the sparsity after 500 queries, and the ASR with sparsity thresholds. We compare the results to the earlier results of untargeted attacks without any input transformations.

6.3 Adversarial Training + Median Filter

Motivated by the results of the previous two experiments we introduce another experiment to see if the two defensive techniques can be combined to create even more effective robustness to sparse attacks without decreasing the overall accuracy of the model on clean images.

We first evaluate the adversarially trained model's performance on clean images when the images are run through a median filter. We then compare the ASR and query efficiency using these defenses to the results of the previous chapters.

6.4 Adversarial Detection

A downside to using PAR as an initialization technique is that the algorithm leaves very distinct and recognizable traces in the adversarial examples when querying the model during initialization.

As seen in the previous example Figure 4.1.1, during PAR, the image is divided between the original image and the target image with straight and sharp edges of squares. In this experiment, we see if this property can be exploited by training a model to detect if an image has been perturbed by PAR.

In Chapter 2.2, we discuss several methods of adversarial detection. Among the most promising techniques are frameworks SafetyNet [58] and Magnet [57], defense-GAN [55, 56] and specific homemade detection classifiers [53, 54]. We consider developing a separate GAN for adversarial detection too extensive for this scope. The two frameworks were developed and tested for much smaller datasets, so their efficiency and training time on the ImageNet dataset is uncertain [57, 58]. While it might be worthwhile to investigate, we instead opt to create a separate ResNet-50 model tailored to our experiments on the ImageNet dataset. We use a pre-trained ResNet-50 model, the same as in previous experiments, as a base for transfer learning. This is a well-suited choice as the architecture has been shown to be both fast and efficient in classification tasks on ImageNet. The new dataset for training the model consists of 2500 images total, divided into two classes: clean images and images distorted by PAR, with the same number of images for each class. The images distorted by PAR are created from random pairs of original and target images sampled from the dataset, and saved after 10, 30, and 50 queries of PAR, evenly distributed. For training and testing, we split the new dataset into 80% training data and 20% test data. 20% of the training data is set aside as a validation set to be used during training.

We evaluate the trained model on overall accuracy on our newly created dataset to see if it can reliably detect the adversarial images. To act as a robust defense, the model should have a high detection accuracy while maintaining a low false negative rate. To achieve this we can analyze the model's output predictions and experiment with different detection thresholds to find the best balance of true- and false negatives. We evaluate the model on its f1-score, a measure that combines precision and recall to determine the overall accuracy of a classification model. It ranges from 0 to 1, where 1 is the best possible score. Precision is the number of true positive results divided by the number of all positive results, including the false positives. Recall is the number of true positive results divided by the number of all results that should have been identified as positive.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{6.1}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{6.2}$$

The f1-score is then calculated as:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6.3}$$

A balance of 50/50 adversarial- and benign images is probably not a representative balance for a real-world scenario. Because of this, and since the f1-score heavily penalizes the false negative rate, we also evaluate the model's score on a distribution of 10k benign and 1k adversarial examples.

We will also consider a case where an attack is unsuccessful if one or more of the first 50 queries of initialization is prevented, the reasoning being that not all queries made during initialization need to be detected to render the attack powerless. While a few detected PAR queries might not mean that the PAR algorithm

will completely fail, it might be enough to effectively cripple the improvements it has shown as an initialization technique.

CHAPTER

SEVEN

RQ2: RESULTS

In this chapter, we present the results from our experiments to evaluate adversarial training, median filtering, and adversarial detection as defenses against the SparseEvo+PAR attack.

7.1 Adversarial Training

The adversarially trained ResNet-50 model got an accuracy of 53.55% in our experiments on clean images (ImageNet validation set, 50k images). The normally trained ResNet-50 provided by PyTorch has an accuracy of 76.13% [70]. Figure 7.1.1 shows the efficiency of SparseEvo and SparseEvo+PAR in the targeted setting against the normal ResNet-50 model and against the adversarially trained version. The adversarially trained model lowers the efficiency of SparseEvo with and without PAR. SparseEvo alone is unable to generate any adversarial images with sparsity lower than 0.6 against the adversarially trained model on a query budget of 2k queries. The versions with initialization are more similar, but the adversarial model lowers the sparsity by about 0.1 on average for the first 2k queries.

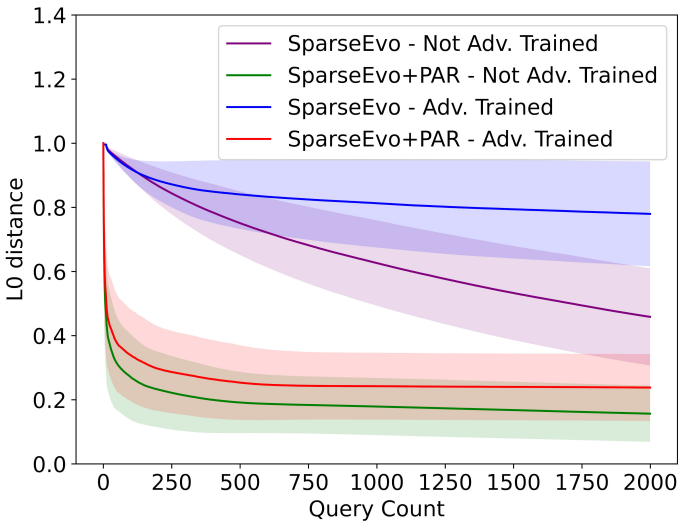


Figure 7.1.1: The SparseEvo algorithm with and without PAR initialization against an adversarially trained model and a normally trained model. The lines mark the L0-distances at queries up to 2k in the targeted setting. The lines are given as the average of 200 runs.

Figure 7.1.2 shows the efficiency of the two SparseEvo versions with and without PAR in the untargeted setting against the normal ResNet-50 model and the adversarially trained model. The adversarially trained model reduces the query efficiency of both versions. In the attack with PAR, it reduces the sparsity by about 0.02, and by around 0.035 in the attack without PAR. The salt-and-pepper initialization still manages to reach a pretty good sparsity of 0.05, and PAR is effective at increasing this sparsity in the next hundred queries. The evolution process, however, is very slow at improving the sparsity against the adversarially trained target model.

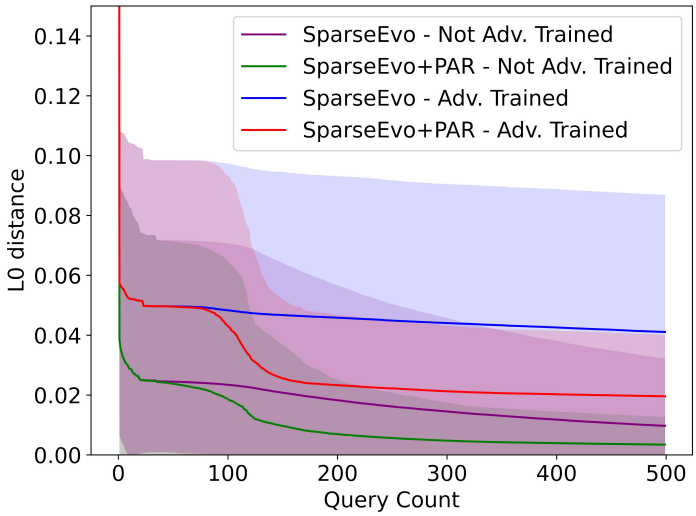


Figure 7.1.2: The SparseEvo algorithm with and without PAR initialization against an adversarially trained model and a normally trained model. The lines mark the L0-distances at queries up to 500 in the untargeted setting. The lines are given as the average of 200 runs.

Table 7.1.1 shows the ASR of the two methods against normal and adversarially trained models in the untargeted setting. The ASR is measured at different sparsity thresholds with a query budget of 500 queries. The ASR is heavily reduced for SparseEvo and SparseEvo with PAR. At a sparsity threshold of 0.001, the ASR is about 16 times less without PAR and 6 times less with PAR. At a sparsity threshold of 0.020 where the version with PAR was almost 97% effective against the normally trained model, the adversarially trained model reduce the success rate to 58%. The sparsity threshold needs to be increased to 2000 perturbed pixels on a query budget of 500 queries before the version with PAR is above 90% ASR. This shows how the adversarially trained defense reduces the query efficiency of the attacks, but the defense is not able to completely defeat the ASR at larger sparsity thresholds.

Method \Sparsity	**0.001**	**0.005**	**0.010**	**0.020**	**0.040**
SparseEvo - Not Adv. Trained	0.47	0.68	0.81	0.88	0.93
SparseEvo - Adv. Trained	0.03	0.14	0.25	0.39	0.63
SparseEvo+PAR - Not Adv. Trained	0.52	0.86	0.95	0.97	0.98
SparseEvo+PAR - Adv. Trained	0.08	0.24	0.39	0.58	0.91

Table 7.1.1: Comparison of the ASR of SparseEvo with and without initialization against the normal- and the adversarially trained model. The ASR is measured on a query budget of 500 queries in the untargeted setting.

7.2 Median Filter

In Figure 7.2.1 we show examples to illustrate how the sparsity is weakened by adding the median filter before the target model. For this example run, the L0 distance is 83.6 times bigger with the filter applied after salt-and-pepper is added, and 65.2 times bigger after evolution.

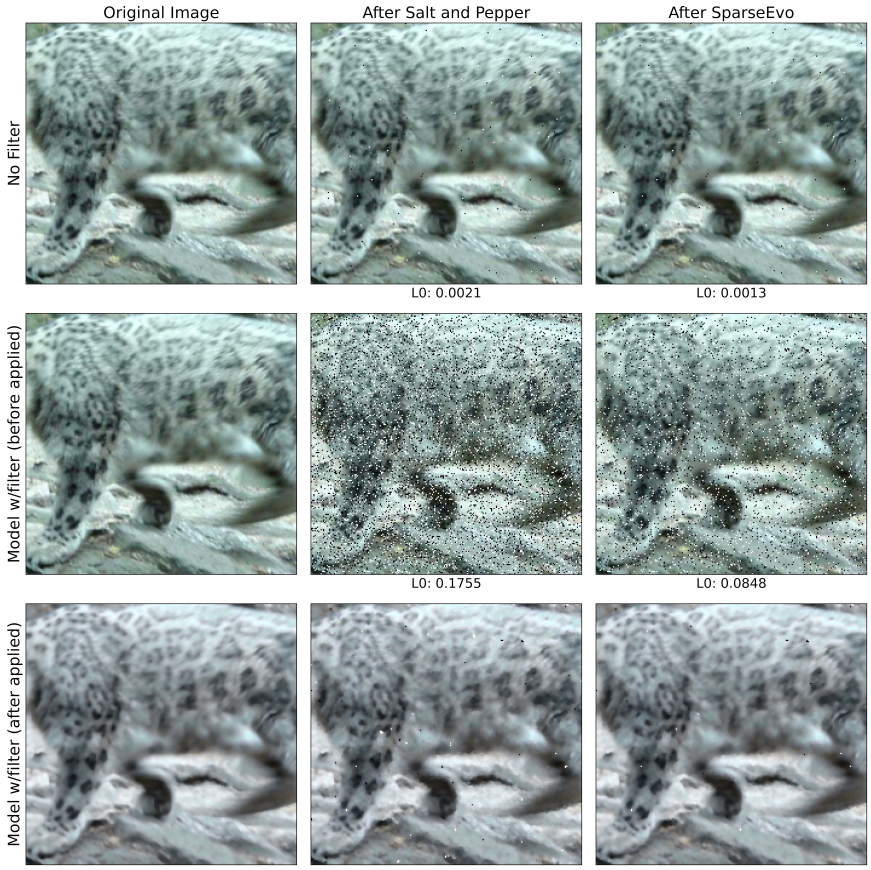


Figure 7.2.1: The SparseEvo process for a random example with and without a median filter applied. The first column shows the original image, the second column shows the perturbed image after salt and pepper initialization and the third column shows the final image after SparseEvo evolution with 500 queries. The first row shows the adversarial examples without a median filter applied, and the second row shows the examples when the model contains a filter. The third row contains the same images as in the second row but shows how the model receives the image after the filter is applied.

Overall, adding the median filtering drops the accuracy of the model on benign images from 76.13% to 71.59% (ImageNet validation set, 50k images). Figure 7.2.2 compares the models with and without the filter on a query budget of 500 queries in the untargeted setting. The filter significantly worsens the performance of the attack. These results show that median filtering is an effective measure to reduce the effect of salt-and-pepper distortions, at the cost of around a 5% drop in prediction accuracy.

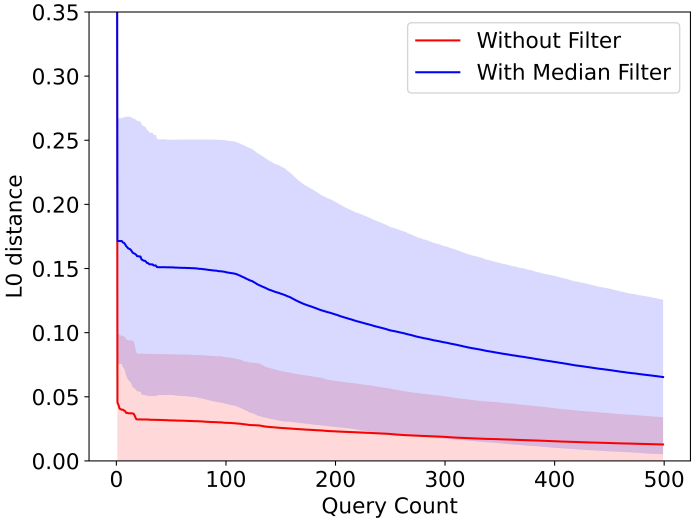
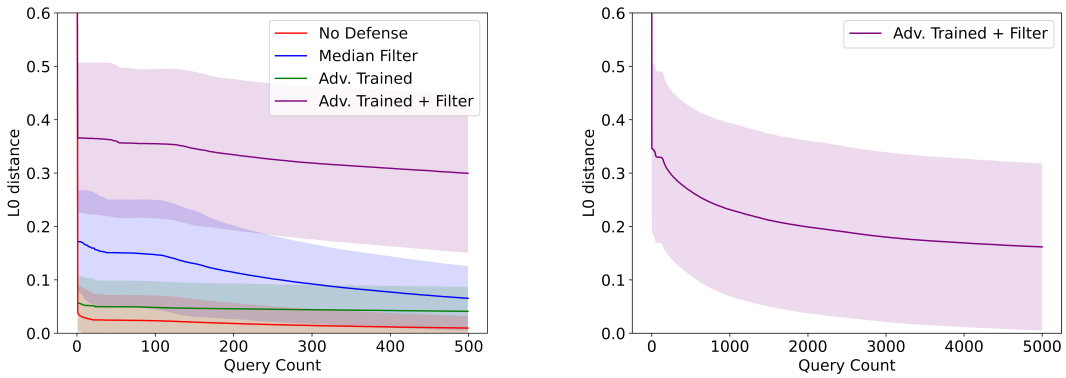


Figure 7.2.2: The SparseEvo+PAR algorithm with and without running the output through a median filter before the target model prediction. The lines mark the L0-distances at queries up to 500, given as the average of 200 runs.

7.3 Median Filter + Adversarial Training

In the next experiment, we combine the adversarially trained model with the median filter technique to further increase the robustness of the untargeted attack. The adversarially trained model with the median filter applied gives a 52.72% accuracy on clean images (ImageNet validation set, 50k images). This is very similar to the accuracy of the adversarially trained model without the filter, only 0.83% worse. In Figure 7.3.1a we compare the combined techniques of adversarial training and median filter to three other results from previous chapters: only median filter defense, only adversarial training defense, and a normal attack without any defenses. The graph shows a significant improvement in robustness for the combined defenses. Adding median filtering to the adversarially trained model increased the robustness of the model from around 0.03 sparsity to around 0.35 sparsity on a query budget of 500 queries. The robustness of the combined defenses is also persistent on larger query budgets, as shown in 7.3.1b. The average sparsity after 5000 queries is no better than about 0.2.



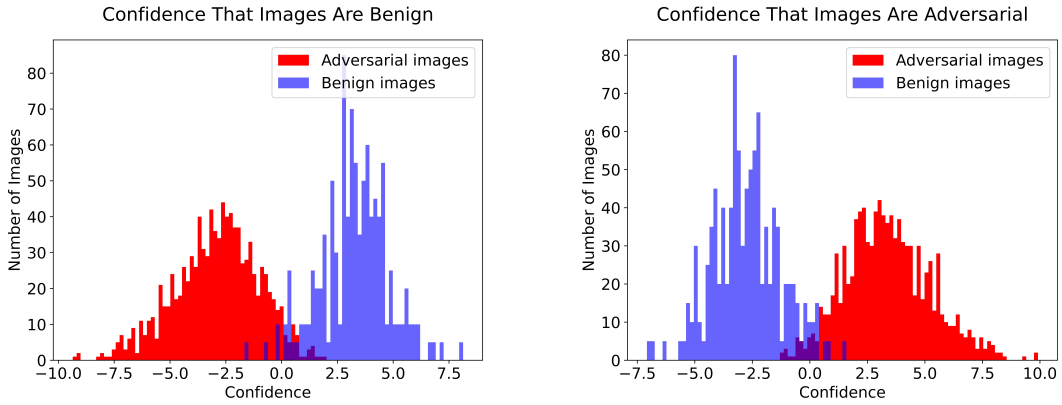
(a) **(b)**

Figure 7.3.1: **(a)**: The SparseEvo+PAR algorithm with and without running the output through a median filter before passing to an adversarially trained model and a normal model. **(b)**: The adversarially trained model + median filter defense, left to run for 5k queries. The lines mark the average L0-distances across 200 runs.

7.4 Adversarial detection

As a final defensive experiment, we trained a model to classify images as either benign or perturbed by PAR in the targeted setting. The trained model got an accuracy of 81.89% on our dataset after 50 epochs.

In Figure 7.4.1 we have analyzed the model's predictions of 1000 benign images, and 1000 adversarial images after every iteration of PAR up to 50 queries (about 50000 images total). For each run with PAR, the model is given up to 50 different images, and we then select the single highest confidence value that any of these images are adversarial to represent that run. We then compare this to the confidence predictions of the benign images. We do this assuming that we do not care if some adversarial examples are allowed through the model, as long as every single run of PAR is stopped somewhere along its 50 first queries.



(a) **(b)**

Figure 7.4.1: The output predictions of a model trained to detect images perturbed by the PAR algorithm. The figure shows the prediction confidences of 1000 adversarial- (red) and 1000 benign images (blue). **(a)**: The model's confidence that the images are benign. **(b)**: The model's confidence that the images are adversarial.

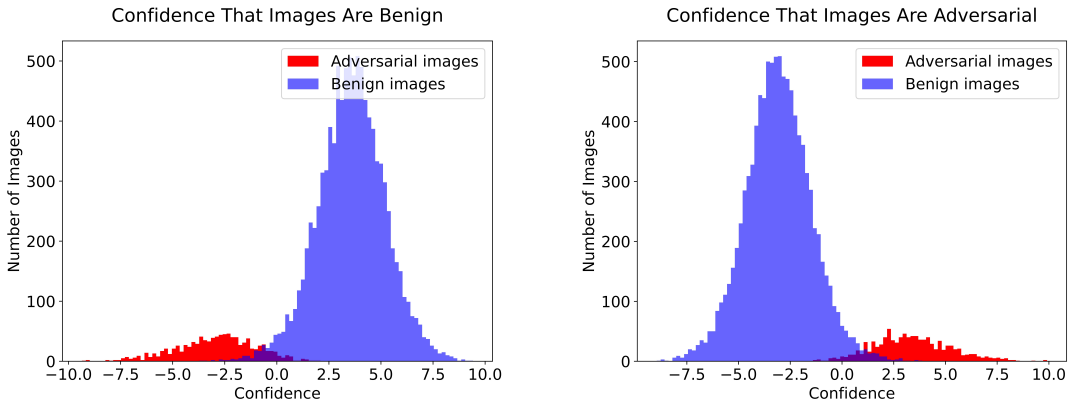
We can see that by selecting the "worst" offenders out of the 50 queries in the 1000 PAR-generated adversarial runs, there are not a lot of misclassified examples. For this dataset, we can adjust the threshold for what we consider an adversarial example to find a balance between false positives and false negatives. For instance, by saying all model predictions that are above zero for the adversarial confidence, we misclassify 4.0% (40) benign images and correctly identify 97.5% (975) PAR runs. If we adjust this threshold to 1.6 (cherrypicked value for example purposes), zero benign images are misclassified, and we identify 83.1% of the par runs. If we use a combination of both prediction confidences, for benign and adversarial, we can get even better predictions. By saying images with adversarial confidence above 1 and benign confidence below 0 are adversarial, only five benign images are misclassified, and 914 PAR runs are identified. We give an overview of prediction accuracies at selected thresholds along with f1-scores in Table 7.4.1.

The assumption that only detecting one of the adversarial examples in a single run of PAR is enough to stop the threat of the attack is not very strong. However, we observe that when one image is detected, usually many are detected in the same run. For instance, if we set the adversarial confidence threshold to 1, an average of 60.0% of the adversarial images created during the first 50 PAR queries are detected (69.23% median) in all the runs where at least one image was detected.

Adv. Thresh.	Benign Thresh.	True Positive	True Negative	Recall	Precision	F1 Score
0	-	960	975	0.97	0.96	0.97
1.6	-	1000	831	0.86	1.00	0.92
-	0	980	952	0.95	0.98	0.97
-	-0.5	990	908	0.91	0.99	0.95
-	-1.0	995	840	0.86	1.00	0.92
1	0	995	914	0.92	1.00	0.96
1.6	0	1000	831	0.86	1.00	0.92

Table 7.4.1: Overview of the f1-scores of the PAR detection model, when setting different confidence thresholds to determine if an image is adversarial or benign.

In a real-world scenario, the adversarial examples are most likely not 50% of the input a model receives. This distribution greatly affects the resulting f1-score. To give a more realistic example, we run the same numbers for 10k random benign examples from the ImageNet dataset, and the same PAR runs as before. It should be noted that 10% is still probably a very high estimate of adversarial input. Figure 7.4.2 shows the new distribution for adversarial and benign predictions done by the model, and Table 7.4.2 summarizes the performance of the classifier at different confidence thresholds. With the right thresholds, we achieve a very high f1-score of 0.99, however, it is important to note that the imbalance between benign and adversarial images favors setting a prediction threshold with a bias towards predicting an image as benign. With these thresholds around 10% of the PAR runs are not detected.



(a) **(b)**

Figure 7.4.2: The output predictions of a model trained to detect images perturbed by the PAR algorithm. The figure shows the prediction confidences of 10000 adversarial- (red) and 1000 benign images (blue). **(a)**: The model's confidence that the images are benign. **(b)**: The model's confidence that the images are adversarial.

Adv. Thresh.	Benign Thresh.	True Positive	True Negative	Recall	Precision	F1 Score
0.5	0	9837	945	0.99	0.98	0.99
1.0	0	9903	914	0.99	0.99	0.99
1.5	0	9944	851	0.99	0.99	0.99
2	0	9974	774	0.98	1.00	0.99

Table 7.4.2: Overview of the f1-scores of the PAR detection model, when setting different confidence thresholds to determine if an image is adversarial or benign.

CHAPTER

EIGHT

DESIGN AND IMPLEMENTATION TO ANSWER RQ3: ENHANCED ATTACK

In Chapter 7.4, the model trained to detect the adversarial examples generated by PAR was highly effective on our created dataset. Unlike the other defenses, that worsened the sparsity on query budgets, we consider the adversarially trained detector to nearly remove the threat of attacks using PAR. Motivated by these results, and to answer research question three, we investigate ways to enhance the SparseEvo+PAR attack method to mitigate the threat of adversarial detection.

We hypothesized that the distinct edges in the adversarial examples created by the PAR algorithm would be easy to detect for a trained classifier in Chapter 6.4. To make these edges less distinct, we propose a change to the PAR algorithm that blurs the adversarial patches together with the original image along the edges of the patches, to make the PAR perturbations harder to detect. We give an example of an adversarial example created by the modified PAR algorithm in Figure 8.0.1.

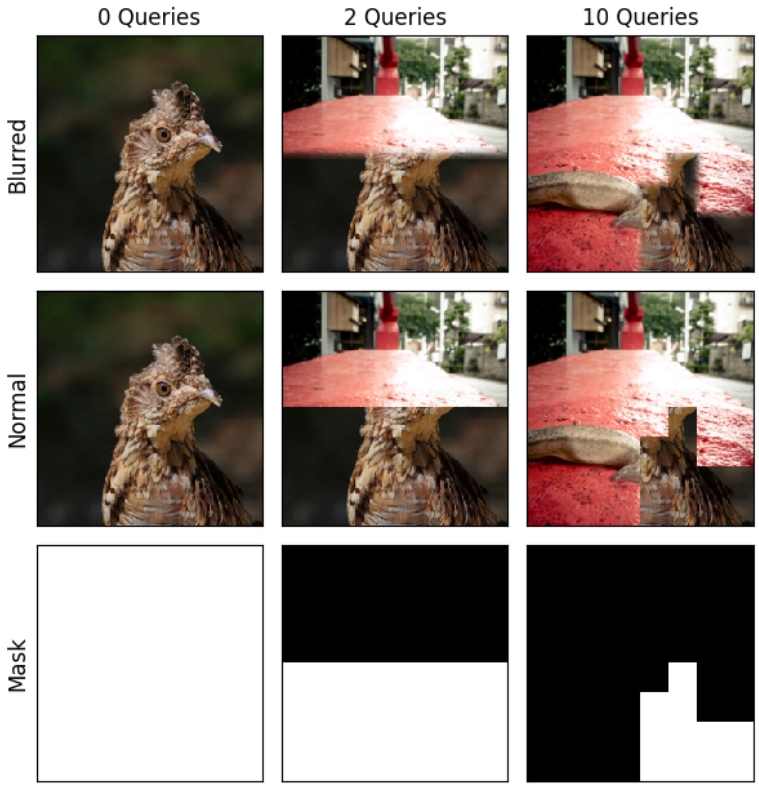


Figure 8.0.1: Example of an adversarial example created by the modified version of PAR that blurs the images together along the edges.

The modified PAR algorithm works similarly to before, querying the target model with patches to check if they are necessary to keep the image adversarial, and keeping track of a mask of the adversarial patches. In addition to this mask, the modified version of PAR also tracks all edges the mask share with the original image. Before passing the image to the target model, the target image and the original image are blurred together along these edges. The blurring is performed by gradually shifting from the original image to the target image along the edge of the mask, and thus the pixels in between will be a blend of the two images. We chose to blur six pixels on either side of the edge, as a balance between visually good-looking blends and keeping the distortions to a minimum.

We first evaluate the altered algorithm by checking how the blurring affects the distortion reduction compared to the normal PAR algorithm. Secondly, to test if the new algorithm is harder to detect by a trained model, we perform the same experiment setup and evaluation as done in Chapter 6.4, using the same base ResNet-50 model and dataset distribution.

CHAPTER

NINE

RQ3: RESULTS

Figure 9.0.1 shows how effective the blurred version of the PAR algorithm is at reducing the size of the overlay adversarial target image in the targeted setting. If we measure the final L0 distance by the mask (the patches necessary to keep the image adversarial) tracked by PAR, the images are reduced to 16.7% of the original on average. This is, however, not a fair measure as the edges on the mask are blurred when given to the model, and thus more pixels are distorted than just the mask. Counting the blurred edges, the images are 26.6% reduced on average.

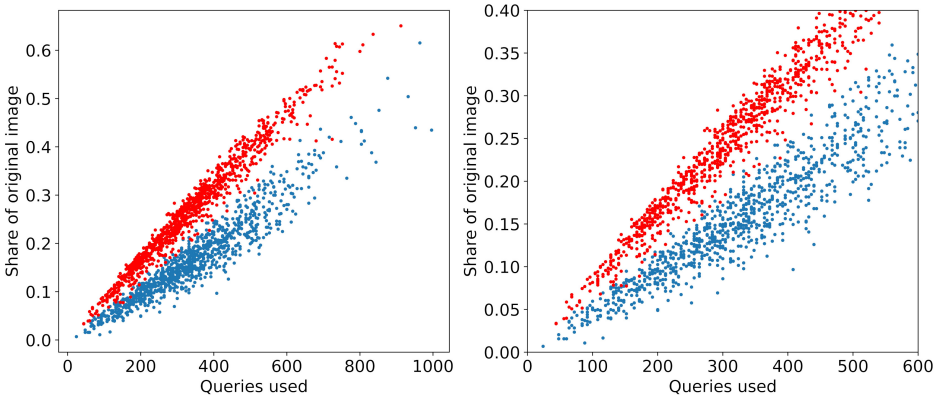
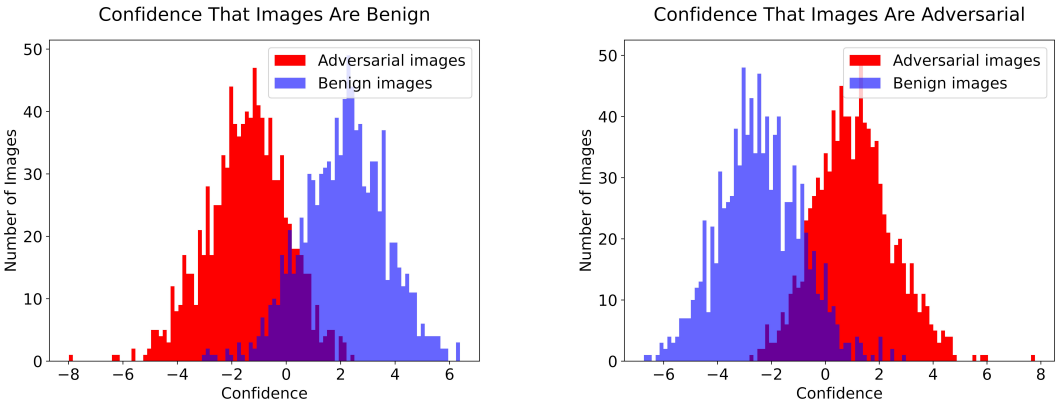


Figure 9.0.1: The reduced sizes after initializations using a blurred version of the PAR algorithm, and the queries used to do so for 1000 different images in the targeted setting. The red dots mark the number of perturbed pixels when counting the blurred edges between the adversarial patches. The blue dots mark the number of perturbed pixels when only counting the adversarial patches, disregarding the blurred edges. The image to the right is a zoomed-in version of the left image.

A new classifier is trained on the blurred PAR images to determine if an image is benign or adversarial, using the same setup as the adversarial detection model in Chapter 7.4. This model got an overall accuracy of 89.3% on the test set.

We run the same experiment as in Chapter 7.4 by combining the model's pre-

dictions of 1000 PAR runs, selecting the maximum confidence that a single image is adversarial across the 50 queries in the same run. We compare this to 1000 benign images. As we can see in Figure 9.0.2 there is more overlap in the prediction confidences of the model between the two classes than in Figure 7.4.1.



(a) **(b)**

Figure 9.0.2: The output predictions of a model trained to detect images perturbed by the blurred PAR algorithm. The figure shows the prediction confidences of 1000 adversarial- (red) and 1000 benign images (blue). **(a)**: The model's confidence that the images are benign. **(b)**: The model's confidence that the images are adversarial.

With the new detection model, there are no thresholds that can be set to significantly boost the f1-score of the model. The best configuration we found was setting the benign threshold at 0 and the adversarial threshold at -1. With this setting, we misclassify 8.3% (83) benign images and correctly identify 84.8% (848) PAR runs. This gives an f1-score of 88.64%, significantly worse than the f1-scores of 97% of the model that trained on non-blurred pictures.

Our results show that for this experiment, adversarial examples created from the blurred version of the PAR algorithm are harder to separate from benign images. It should be noted that this does not necessarily mean that it is true for all scenarios, a larger dataset, other classifier architectures, and larger training time may yield different results.

CHAPTER

TEN

DISCUSSION

The PAR algorithm has shown promising results when combined with state-of-the-art dense attacks. Our results in Chapter 5 demonstrated that we can use the PAR algorithm as an initialization method to improve the query efficiency of the best sparse attack SparseEvo. The introduction of PAR significantly improved the image sparsity and attack success rate of the attack on query budgets for both targeted and untargeted attacks. The implications of our findings are not just important for future research but also for the current state of the security of deployed systems. We have established that sparse attacks have the ability to create barely noticeable adversarial examples using less than a thousand queries. Improving the query efficiency of the attacks significantly complicates the challenge of distinguishing between benign inputs and attacks, which could have serious repercussions for safety-critical classification models.

In a minor experiment, we saw that analyzing the images used to create the initial adversarial example in the targeted setting can improve the query efficiency and the sparsity after PAR. Choosing the right target image is especially important in the current sparse attacks as the initial noise is never modified, only reduced. To the best of our knowledge, there exists no other research that has compared the target images to the original images to create more efficient attacks. A direction for future work includes further analysis of the relationship between the target image and the final perturbation size, especially in sparse attacks. There might also be benefits to an initial modification to the target image, such as mapping the original image's colors onto the target image.

An adversarially trained ResNet-50 model reduced the sparsity in the examples created by the attacks, and it seems to be an effective defensive option for reducing the attack success rate on query budgets. A drawback of this defense is the around 20% drop in accuracy on benign images, but the extra security this training provides might outweigh this downside. This is also the only defense we have tested that is general, meaning it is not specifically targeting any single technique used in SparseEvo or PAR. Given that it has been used effectively as a countermeasure in dense attacks and white-box attacks as well, this defense seems

to be the most promising to provide some general robustness to the classifier. Adding a median filter to the adversarially trained model, a defense targeting the salt-and-pepper perturbations, only decreased the accuracy on benign images by a small amount, and in turn, lowered the efficiency of the sparse attack in the untargeted setting considerably. The median filter as a single defense was not as effective as the adversarially trained model. There exist several newer and more advanced implementations based on the concept of median filters to more effectively remove salt-and-pepper distortions and produce less blurry images [72, 74, 75, 76]. These implementations might be an interesting direction for future work if salt-and-pepper continues to be used as an initialization method. Overall, the best defensive techniques we tested provide a significant boost to the robustness of the tested models but are not able to completely mitigate the threat of the attacks. It seems that if the attacker is allowed thousands of queries they will be able to create very sparse attacks, even with defensive countermeasures, and then it might be worth considering skipping the defenses entirely to improve the accuracy of the classifier. However, if decreasing the query efficiency of the attack from a few hundred queries to thousands is enough to deter or detect attackers, we can consider the defenses successful.

The results of the adversarial detection are promising and are the only tested defense that can be said to stop the attack. With the right prediction thresholds, the trained classifier is capable of detecting a high proportion of the adversarial examples created by the PAR algorithm, while keeping the false positive rate low. This suggests that adversarial detection can be a viable defensive option against techniques similar to PAR. However, the results are highly dependent on the balance between benign and adversarial examples, and the thresholds used for the limited dataset in our experiment may not necessarily transfer to other distributions. There exists a large number of attacks that do not use PAR in their algorithm, so it might be hard to justify creating a defensive layer that only targets this specific attack, especially if the false negative rate is large enough to trouble the end users of the classifiers. We also demonstrated that the attack algorithm can be modified to decrease the effectiveness of this defense, and possibly make the false negative rate of the detector too high. However, other or more powerful types of detectors may be able to detect these harder-to-detect versions and can be considered a possible path for future work.

As previously discussed, sparse attacks are harder to optimize than dense attacks. From a security standpoint, sparse attacks can be just as severe as dense attacks in theory. However, we have not seen any direct comparisons between the performance of SparseEvo and the best dense attacks, since we can't use the $L0$- $L2$-, or L_∞-metric interchangeably if we want a fair comparison. This also makes it hard to directly compare the efficiency of defenses. Future work should focus on comparing the query efficiency and robustness of sparse attacks compared to other decision-based black-box attacks, to evaluate if sparse attacks are worthwhile to investigate despite their harder optimization problem.

There are some threats to the validity of our thesis that should be discussed. Firstly, we only included one dataset and one target model architecture for our experiments. While we argue that they were the most fitting selections for our problem, this may limit the generalizability of our findings. To save time, we also had to limit the number of included examples for each experiment, which may also have affected the overall accuracy and generalizability. Secondly, there exist only two sparse attacks, and we only compared improvements and defenses to the SparseEvo algorithm. We excluded the other sparse attack PointWise from our experiments, as its performance in comparison to SparseEvo has been previously evaluated in the paper by Vo et al. [16] and we consider the techniques of PointWise to be included in the untargeted version of SparseEvo, only done more effectively. The limitation of only one sparse attack has made our defensive experiments more of an evaluation of SparseEvo as a single attack, rather than an evaluation of the field of sparse attacks.

CHAPTER

ELEVEN

CONCLUSION

The work presented in this thesis provides improvements and evaluations in the field of sparse decision-based black-box attacks. We demonstrated the potential of using the patch adversarial removal (PAR) algorithm as initialization to the state-of-the-art sparse attack SparseEvo. Their combination significantly improved the query efficiency and attack success rate for both targeted and untargeted attacks.

Moreover, we evaluated various defensive techniques against PAR and SparseEvo. An adversarially trained ResNet-50 model was as an effective countermeasure, despite a reduction in the model's prediction accuracy. In the untargeted setting, adding a median filter to the input proved to be an effective and robust addition to the adversarially trained model, almost without further reducing the prediction accuracy of benign images. In the targeted setting, adversarial detection demonstrated great promise in our study, with a trained classifier detecting a significant portion of perturbed images created by the PAR algorithm. In an attempt to make PAR harder to detect, we created a new version of the algorithm where the adversarial patches were blurred together with the original image. This resulted in adversarial examples that were significantly harder to detect but at the cost of a slight reduction in sparsity.

Our study underlines the potential security threat posed by sparse attacks. Given some hundred queries to a target classifier, an attacker can create visually unperturbed adversarial examples. We have demonstrated the effect of several defensive countermeasures, and how they can help reduce the query efficiency of sparse attacks.

Future work should focus on improving the proposed sparse attack enhancements, especially by exploring other methods for noise reduction. Input transformation and adversarial detection showed promise in our results, future work should work on confirming their efficiency on other sparse attack algorithms, datasets, and model architectures. We also believe research should focus on the comparative study of the query efficiency and robustness of sparse attacks to dense attacks, evaluating the practicality of pursuing sparse attacks despite their more complex optimization problem.

REFERENCES

[1] Sajad Ahmadian et al. "A novel deep neuroevolution-based image classification method to diagnose coronavirus disease (COVID-19)". In: *Computers in Biology and Medicine* 139 (2021), p. 104994. ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2021.104994. URL: https://www.sciencedirect.com/science/article/pii/S0010482521007885.

[2] Damilola A Okuboyejo, Oludayo O Olugbara, and Solomon A Odunaike. "Automating skin disease diagnosis using image classification". In: *proceedings of the world congress on engineering and computer science*. Vol. 2. 2013, pp. 850–854.

[3] Tolga Turay and Tanya Vladimirova. "Toward Performing Image Classification and Object Detection With Convolutional Neural Networks in Autonomous Driving Systems: A Survey". In: *IEEE Access* 10 (2022), pp. 14076–14119. DOI: 10.1109/ACCESS.2022.3147495.

[4] V. Shreyas et al. "Self-driving Cars: An Overview of Various Autonomous Driving Systems". In: *Advances in Data and Information Sciences*. Ed. by Mohan L. Kolhe et al. Singapore: Springer Singapore, 2020, pp. 361–371.

[5] Dimple Chawla and Munesh Chandra Trivedi. "A Comparative Study on Face Detection Techniques for Security Surveillance". In: *Advances in Computer and Computational Sciences*. Ed. by Sanjiv K. Bhatia et al. Singapore: Springer Singapore, 2018, pp. 531–541. ISBN: 978-981-10-3773-3.

[6] Fatih Cagatay Akyon and Alptekin Temizel. *Deep Architectures for Content Moderation and Movie Content Rating*. 2022. arXiv: 2212.04533 [cs.CV].

[7] Christian Szegedy et al. *Intriguing properties of neural networks*. 2014. arXiv: 1312.6199 [cs.CV].

[8] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2015. arXiv: 1412.6572 [stat.ML].

[9] Yanpei Liu et al. *Delving into Transferable Adversarial Examples and Black-box Attacks*. 2017. arXiv: 1611.02770 [cs.LG].

[10] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. *Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples*. 2016. arXiv: 1605.07277 [cs.CR].

[11] Nicolas Papernot et al. *Practical Black-Box Attacks against Machine Learning*. 2017. arXiv: 1602.02697 [cs.CR].

[12] Jianbo Chen, Michael I. Jordan, and Martin J. Wainwright. *HopSkipJumpAttack: A Query-Efficient Decision-Based Attack*. 2020. arXiv: 1904.02144 [cs.LG].

[13] Ali Rahmati et al. *GeoDA: a geometric framework for black-box adversarial attacks*. 2020. arXiv: 2003.06468 [cs.CV].

[14] Jinghui Chen and Quanquan Gu. *RayS: A Ray Searching Method for Hard-label Adversarial Attack*. 2020. arXiv: 2006.12792 [cs.LG].

[15] Huichen Li et al. *Nonlinear Projection Based Gradient Estimation for Query Efficient Blackbox Attacks*. 2021. arXiv: 2102.13184 [cs.LG].

[16] Viet Quoc Vo, Ehsan Abbasnejad, and Damith C. Ranasinghe. "Query Efficient Decision Based Sparse Attacks Against Black-Box Deep Learning Models". In: *CoRR* abs/2202.00091 (2022). arXiv: 2202.00091. URL: https://arxiv.org/abs/2202.00091.

[17] Lukas Schott et al. "Robust Perception through Analysis by Synthesis". In: *CoRR* abs/1805.09190 (2018). arXiv: 1805.09190. URL: http://arxiv.org/abs/1805.09190.

[18] Jianbo Chen and Michael I. Jordan. "Boundary Attack++: Query-Efficient Decision-Based Adversarial Attack". In: *CoRR* abs/1904.02144 (2019). arXiv: 1904.02144. URL: http://arxiv.org/abs/1904.02144.

[19] Guofu Li et al. *Security Matters: A Survey on Adversarial Machine Learning*. 2018. arXiv: 1810.07339 [cs.LG].

[20] Han Xu et al. *Adversarial Attacks and Defenses in Images, Graphs and Text: A Review*. 2019. arXiv: 1909.08072 [cs.LG].

[21] Wieland Brendel, Jonas Rauber, and Matthias Bethge. *Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models*. 2017. DOI: 10.48550/ARXIV.1712.04248. URL: https://arxiv.org/abs/1712.04248.

[22] Ali Rahmati et al. "GeoDA: a geometric framework for black-box adversarial attacks". In: *CoRR* abs/2003.06468 (2020). arXiv: 2003.06468. URL: https://arxiv.org/abs/2003.06468.

[23] Xiaosen Wang et al. "Triangle Attack: A Query-efficient Decision-based Adversarial Attack". In: *CoRR* abs/2112.06569 (2021). arXiv: 2112.06569. URL: https://arxiv.org/abs/2112.06569.

[24] Thibault Maho, Teddy Furon, and Erwan Le Merrer. "SurFree: a fast surrogate-free black-box attack". In: *CoRR* abs/2011.12807 (2020). arXiv: 2011.12807. URL: https://arxiv.org/abs/2011.12807.

[25] Huichen Li et al. "QEBA: Query-Efficient Boundary-Based Blackbox Attack". In: *CoRR* abs/2005.14137 (2020). arXiv: 2005.14137. URL: https://arxiv.org/abs/2005.14137.

[26] Huichen Li et al. "Nonlinear Projection Based Gradient Estimation for Query Efficient Blackbox Attacks". In: *CoRR* abs/2102.13184 (2021). arXiv: 2102.13184. URL: https://arxiv.org/abs/2102.13184.

[27] Yucheng Shi and Yahong Han. "Decision-based Black-box Attack Against Vision Transformers via Patch-wise Adversarial Removal". In: *CoRR* abs/2112.03492 (2021). arXiv: 2112.03492. URL: https://arxiv.org/abs/2112.03492.

[28] Yucheng Shi, Yahong Han, and Qi Tian. "Polishing Decision-Based Adversarial Noise With a Customized Sampling". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 1027–1035. DOI: 10.1109/CVPR42600.2020.00111.

[29] Apostolos Modas, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. "SparseFool: a few pixels make a big difference". In: *CoRR* abs/1811.02248 (2018). arXiv: 1811.02248. URL: http://arxiv.org/abs/1811.02248.

[30] Xiaoyi Dong et al. "GreedyFool: Distortion-Aware Sparse Adversarial Attack". In: *CoRR* abs/2010.13773 (2020). arXiv: 2010.13773. URL: https://arxiv.org/abs/2010.13773.

[31] Francesco Croce and Matthias Hein. "Sparse and Imperceivable Adversarial Attacks". In: *CoRR* abs/1909.05040 (2019). arXiv: 1909.05040. URL: http://arxiv.org/abs/1909.05040.

[32] Naveed Akhtar and Ajmal Mian. *Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey*. 2018. arXiv: 1801.00553 [cs.CV].

[33] Shilin Qiu et al. "Review of Artificial Intelligence Adversarial Attack and Defense Technologies". In: *Applied Sciences* 9.5 (2019). ISSN: 2076-3417. DOI: 10.3390/app9050909. URL: https://www.mdpi.com/2076-3417/9/5/909.

[34] Guneet S. Dhillon et al. *Stochastic Activation Pruning for Robust Adversarial Defense*. 2018. arXiv: 1803.01442 [cs.LG].

[35] Chunchuan Lyu, Kaizhu Huang, and Hai-Ning Liang. *A Unified Gradient Regularization Family for Adversarial Examples*. 2015. arXiv: 1511.06385 [cs.LG].

[36] Andrew Slavin Ross and Finale Doshi-Velez. *Improving the Adversarial Robustness and Interpretability of Deep Neural Networks by Regularizing their Input Gradients*. 2017. arXiv: 1711.09404 [cs.LG].

[37] Ji Gao et al. *DeepCloak: Masking Deep Neural Network Models for Robustness Against Adversarial Samples*. 2017. arXiv: 1702.06763 [cs.LG].

[38] Weilin Xu, David Evans, and Yanjun Qi. "Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks". In: *Proceedings 2018 Network and Distributed System Security Symposium*. Internet Society, 2018. DOI: 10.14722/ndss.2018.23198. URL: https://doi.org/10.14722%5C%2Fndss.2018.23198.

[39] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. *Distilling the Knowledge in a Neural Network*. 2015. arXiv: 1503.02531 [stat.ML].

[40] Nicolas Papernot et al. "Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks". In: *2016 IEEE Symposium on Security and Privacy (SP)*. 2016, pp. 582–597. DOI: 10.1109/SP.2016.41.

[41] Nicholas Carlini and David Wagner. *Towards Evaluating the Robustness of Neural Networks*. 2017. arXiv: 1608.04644 [cs.CR].

[42] Anish Athalye, Nicholas Carlini, and David Wagner. *Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples*. 2018. arXiv: 1802.00420 [cs.LG].

[43] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M. Roy. *A study of the effect of JPG compression on adversarial images*. 2016. arXiv: 1608.00853 [cs.CV].

[44] Nilaksh Das et al. *Keeping the Bad Guys Out: Protecting and Vaccinating Deep Learning with JPEG Compression*. 2017. arXiv: 1705.02900 [cs.CV].

[45] Cihang Xie et al. *Adversarial Examples for Semantic Segmentation and Object Detection*. 2017. arXiv: 1703.08603 [cs.CV].

[46] Xin Li and Fuxin Li. *Adversarial Examples Detection in Deep Networks with Convolutional Filter Statistics*. 2017. arXiv: 1612.07767 [cs.CV].

[47] Alexey Kurakin et al. "Adversarial Attacks and Defences Competition". In: *The NIPS '17 Competition: Building Intelligent Systems*. Ed. by Sergio Escalera and Markus Weimer. Cham: Springer International Publishing, 2018, pp. 195–231. ISBN: 978-3-319-94042-7.

[48] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. *Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models*. 2018. arXiv: 1805.06605 [cs.CV].

[49] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[50] Nicholas Carlini and David Wagner. *Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods*. 2017. arXiv: 1705.07263 [cs.LG].

[51] Aleksander Madry et al. *Towards Deep Learning Models Resistant to Adversarial Attacks*. 2019. arXiv: 1706.06083 [stat.ML].

[52] Kathrin Grosse et al. *On the (Statistical) Detection of Adversarial Examples*. 2017. arXiv: 1702.06280 [cs.CR].

[53] Zhitao Gong, Wenlu Wang, and Wei-Shinn Ku. *Adversarial and Clean Data Are Not Twins*. 2017. arXiv: 1704.04960 [cs.LG].

[54] Jan Hendrik Metzen et al. *On Detecting Adversarial Perturbations*. 2017. arXiv: 1702.04267 [stat.ML].

[55] Hyeungill Lee, Sungyeob Han, and Jungwoo Lee. *Generative Adversarial Trainer: Defense to Adversarial Perturbations with GAN*. 2017. arXiv: 1705.03387 [cs.LG].

[56] Shiwei Shen et al. *APE-GAN: Adversarial Perturbation Elimination with GAN*. 2017. arXiv: 1707.05474 [cs.CV].

[57] Dongyu Meng and Hao Chen. *MagNet: a Two-Pronged Defense against Adversarial Examples*. 2017. arXiv: 1705.09064 [cs.CR].

[58] Jiajun Lu, Theerasit Issaranon, and David Forsyth. *SafetyNet: Detecting and Rejecting Adversarial Examples Robustly*. 2017. arXiv: 1704.00103 [cs.CV].

[59] Reuben Feinman et al. *Detecting Adversarial Samples from Artifacts*. 2017. arXiv: 1703.00410 [stat.ML].

[60] Raymond Ros and Nikolaus Hansen. "A Simple Modification in CMA-ES Achieving Linear Time and Space Complexity". In: *Parallel Problem Solving from Nature – PPSN X*. Ed. by Günter Rudolph et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 296–305. ISBN: 978-3-540-87700-4.

[61] J. Rapin and O. Teytaud. *Nevergrad - A gradient-free optimization platform*. `https://GitHub.com/FacebookResearch/Nevergrad`. 2018.

[62] Linnan Wang, Rodrigo Fonseca, and Yuandong Tian. *Learning Search Space Partition for Black-box Optimization using Monte Carlo Tree Search*. 2022. arXiv: `2007.00708 [cs.LG]`.

[63] Dang N.H. Thanh et al. "Impulse denoising based on noise accumulation and harmonic analysis techniques". In: *Optik* 241 (2021), p. 166163. ISSN: 0030-4026. DOI: `https://doi.org/10.1016/j.ijleo.2020.166163`. URL: `https://www.sciencedirect.com/science/article/pii/S0030402620319690`.

[64] Melih Yildirim. "Analog circuit implementation based on median filter for salt and pepper noise reduction in image". In: *Analog Integrated Circuits and Signal Processing* 107 (Apr. 2021). DOI: `10.1007/s10470-021-01820-3`.

[65] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: `10.1109/CVPR.2009.5206848`.

[66] Li Deng. "The mnist database of handwritten digit images for machine learning research". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[67] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).

[68] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*. CVPR '16. Las Vegas, NV, USA: IEEE, June 2016, pp. 770–778. DOI: `10.1109/CVPR.2016.90`. URL: `http://ieeexplore.ieee.org/document/7780459`.

[69] Sébastien Marcel and Yann Rodriguez. "Torchvision the Machine-Vision Package of Torch". In: *Proceedings of the 18th ACM International Conference on Multimedia*. MM '10. Firenze, Italy: Association for Computing Machinery, 2010, pp. 1485–1488. ISBN: 9781605589336. DOI: `10.1145/1873951.1874254`. URL: `https://doi.org/10.1145/1873951.1874254`.

[70] *PyTorch, resnet50*. `https://pytorch.org/vision/main/models/generated/torchvision.models.resnet50.html`. Accessed: 2023-05-06.

[71] Logan Engstrom et al. *Robustness (Python Library)*. 2019. URL: `https://github.com/MadryLab/robustness`.

[72] Uğur Erkan, Levent Gökrem, and Serdar Enginoğlu. "Different applied median filter in salt and pepper noise". In: *Computers & Electrical Engineering* 70 (2018), pp. 789–798. ISSN: 0045-7906. DOI: `https://doi.org/10.1016/j.compeleceng.2018.01.019`. URL: `https://www.sciencedirect.com/science/article/pii/S0045790617320244`.

[73] Richard E Woods and Rafael C Gonzalez. *Digital image processing*. 2008.

[74] Uğur Erkan et al. “Adaptive frequency median filter for the salt and pepper denoising problem”. In: *IET Image Processing* 14.7 (2020), pp. 1291–1302. DOI: https://doi.org/10.1049/iet-ipr.2019.0398. eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/iet-ipr.2019.0398. URL: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-ipr.2019.0398.

[75] Kenny Kal Vin Toh and Nor Ashidi Mat Isa. “Noise Adaptive Fuzzy Switching Median Filter for Salt-and-Pepper Noise Reduction”. In: *IEEE Signal Processing Letters* 17.3 (2010), pp. 281–284. DOI: 10.1109/LSP.2009.2038769.

[76] S. Esakkirajan et al. “Removal of High Density Salt and Pepper Noise Through Modified Decision Based Unsymmetric Trimmed Median Filter”. In: *IEEE Signal Processing Letters* 18.5 (2011), pp. 287–290. DOI: 10.1109/LSP.2011.2122333.

APPENDICES

A - RESNET-50 ARCHITECTURE

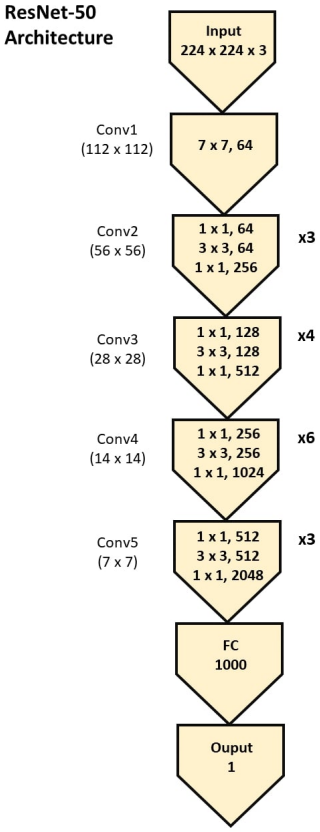
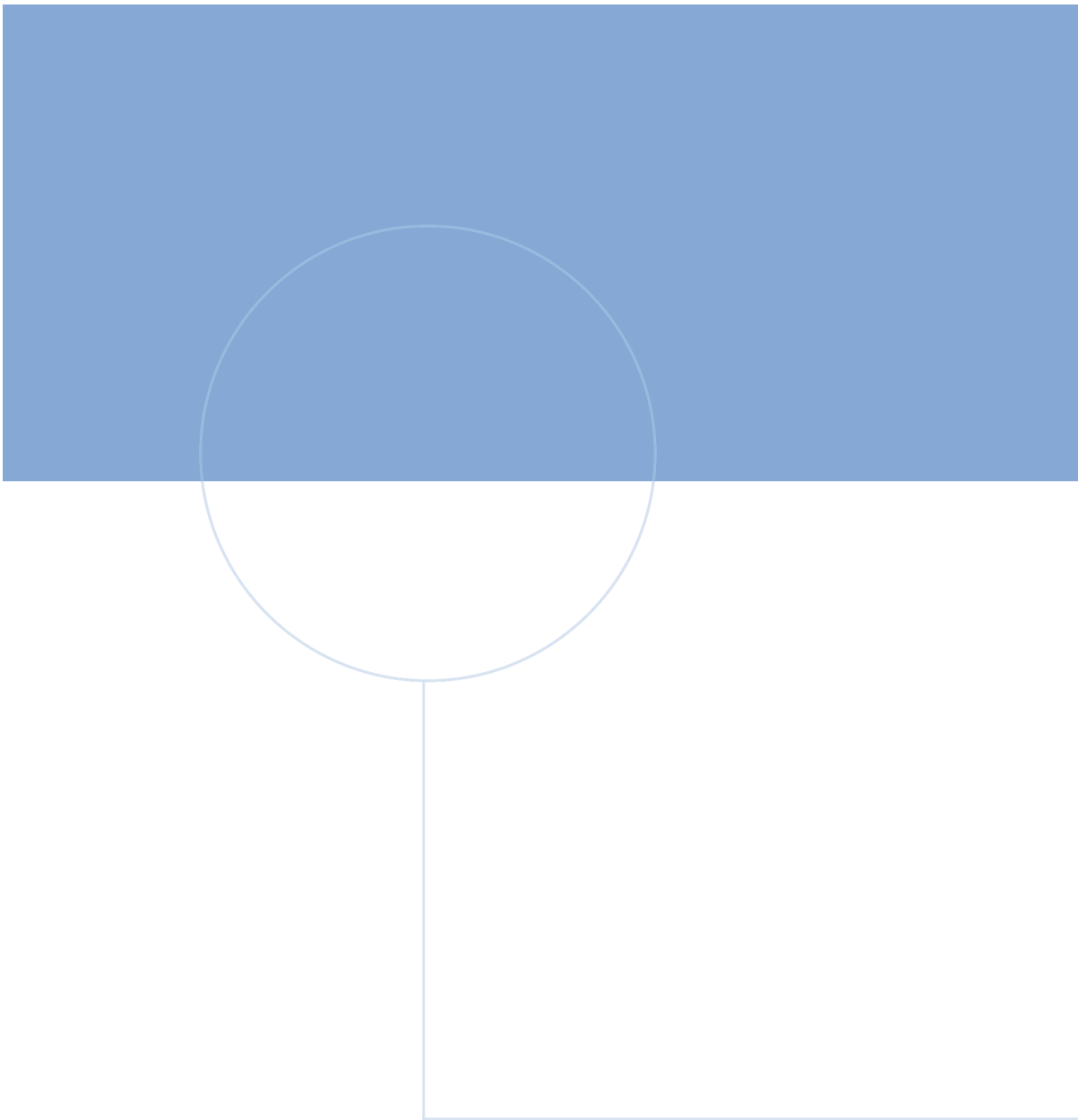


Figure .0.1: The ResNet-50 architecture's 50 layers. The layers are represented as nodes downwards from the input at the top to the output at the end. The convolution layers are named to the node's left, and the output size after the layer is listed in parentheses below the name. Some of the convolution nodes are repeated, indicated to the node's right. After the fully connected layer, we get 1000 outputs, one for each class in the ImageNet dataset. The output gives the label of the class with the highest probability.



NTNU

Norwegian University of
Science and Technology