Tampere University

Joni Pesonen

DIMENSIONALITY REDUCTION IN GENERATING DECISION-BASED ADVERSARIAL EXAMPLES

Master of Science Thesis
Faculty of Information Technology and Communication Sciences
Examiners: Prof. Tapio Elomaa
Prof. Martti Juhola
November 2022

ABSTRACT

Joni Pesonen: Dimensionality reduction in generating decision-based adversarial examples
Master of Science Thesis
Tampere University
Master's Programme in Computing Sciences
November 2022

As the ever-increasing popularity of machine learning continues to rise, its reliability is of utmost importance. Despite the enormous progress and its excellent performance, it has been discovered that machine learning is vulnerable to so-called adversarial examples. While they are based on a naturally occurring machine learning phenomenon, they can be exploited by malicious attackers for their own purposes, as it is possible to generate them. In the decision-based attack category, the key to the more efficient generation of adversarial examples has been dimensionality reduction. However, the effect different dimensionality reduction methods and their magnitudes have in this generation process has been poorly researched.

The thesis provides a diverse view of the phenomenon and discusses the often exaggerated security concerns raised by adversarial examples before focusing more specifically on the decision-based attack scenario. We explore three main research questions. First, we explore the current state and challenges of decision-based attacks in the scientific literature. Secondly, we present a few dimensionality reduction methods that could be used for our purposes. Lastly, tests are implemented to quantify the differences the dimensionality reduction methods with their differently sized subspaces have on the adversarial example generation.

The study showcases these differences and identifies the turning point after which the dimensionality reduction starts to be detrimental. We provide novel ways to perform dimensionality reduction and discuss the advantages and disadvantages of the methods. We demonstrate that there is room for improvement in generating decision-based adversarial examples by utilizing more extensive dimensionality reduction than is customary in the scientific literature.

Keywords: machine learning, adversarial examples, decision-based attack, dimensionality reduction, artificial intelligence

TIIVISTELMÄ

Joni Pesonen: Dimensioiden vähentäminen päätöspohjaisten peukaloitujen syötteiden generoimisessa
Pro gradu -tutkielma
Tampereen yliopisto
Master's Programme in Computing Sciences
Marraskuu 2022

Koneoppimisen laajamittaisen suosion kasvaessa sen toimintavarmuus tilanteesta riippumatta on äärimmäisen tärkeää. Valtavista edistysaskelista ja erinomaisista suorituksista huolimatta koneoppimisen on havaittu olevan haavoittuvainen niin kutsuttuja peukaloituja syötteitä kohtaan. Vaikka ilmiö pohjautuukin luonnolliseen koneoppimisen ilmiöön, voi pahansuopa hyökkääjä hyväksikäyttää kyseistä ilmiötä generoimalla peukaloituja syötteitä. Avain tehokkaampaan päätöspohjaisten hyökkäysten generoimiseen on ollut dimensioiden vähentäminen. Eri dimensioiden vähentämiseen keskittyvien menetelmien ja niiden suuruuksien vaikutusta tähän generoimiseen ei ole kuitenkaan tutkittu kovin syvällisesti.

Tässä työssä tarjoamme monipuolisen yleiskatsauksen peukaloiduista syötteistä ja käymme läpi niiden usein liioiteltuja turvallisuushuolia, minkä jälkeen keskitymme tarkemmin päätöspohjaisiin hyökkäyksiin. Tutkimme kolmea tutkimuskysymystä: ensiksi tutkimme tieteellisessä kirjallisuudessa esiintyvien päätöspohjaisten hyökkäysten nykytilaa sekä niiden kohtaamia haasteita. Sen jälkeen esittelemme muutamia tähän käyttötarkoitukseen soveltuvia menetelmiä dimensioiden vähentämiseen. Viimeisenä suoritamme kokeita mitataksemme, kuinka eri dimensioiden vähentämismenetelmät ja niiden erikokoiset aliavaruudet vaikuttavat peukaloitujen syötteiden generoimiseen.

Tutkielma esittelee menetelmien väliset erot ja havainnollistaa käännekohdan, jonka jälkeen dimensioiden vähentäminen alkaa haitata generoimista. Tuomme esiin uudenlaisia tapoja dimensioiden vähentämiseen ja teemme yhteenvedon eri menetelmien hyödyistä ja haitoista. Lisäksi osoitamme, kuinka päätöspohjaisten peukaloitujen syötteiden generoimisessa on parantamisen varaa hyödyntämällä aiempaa kirjallisuutta laajamittaisempaa dimensioiden vähentämistä.

Avainsanat: koneoppiminen, peukaloidut syötteet, päätöspohjainen hyökkäys, dimensioiden vähentäminen, tekoäly

PREFACE

This thesis was the final part of my master's degree studies at Tampere University. First, I would like to thank my thesis supervisor Tapio Elomaa for his support and feedback along the way. I would also like to thank the university and its staff for all these years. And last but not least, I would like to thank my close ones and friends who showed interest in the process and kept asking about my progress, providing further motivation.

Tampere, 4th November 2022

Joni Pesonen

CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

AE	adversarial example
ASR	attack success rate
AUC	area under the curve
CNN	convolutional neural network
DB	decision boundary
DBA	decision-based attack
DCT	discrete cosine transform
DFT	discrete Fourier transform
DR	dimensionality reduction
ICA	independent component analysis
ML	machine learning
NN	neural network
PCA	principal component analysis
$\mathbb{R}$	real numbers
RNN	residual neural network
$\boldsymbol{V}$	matrix
$\boldsymbol{v}$	vector
v	scalar

1. INTRODUCTION

The ever-increasing popularity of machine learning has led to its ubiquity in a variety of systems, some of which are critical in their operational reliability. The main contributor to this surge in popularity has been machine learning's excellent performance even on challenging tasks, which in turn has partly been enabled by the advances in computational power. Machine learning has been a revolutionizing factor in many different fields with numerous real-world applications, such as computer vision, healthcare, and predictive analytics.

As machine learning systems are deployed more frequently in environments where safety and security are critical, concerns regarding the reliability of machine learning systems are warranted. Despite the enormous progress made, it has been discovered that machine learning is vulnerable to so-called adversarial examples [46]. Oftentimes, these adversarial examples are made from deliberately perturbed normal inputs to obtain an erroneous classification from the machine learning system. These perturbations are usually imperceivable for a human observer, yet they can consistently cause the machine learning system to err in classification. However, as we discuss later in Section 3.2, the input can be adversarial even without manipulation due to the nature of this phenomenon.

The adversarial examples as a phenomenon gained traction after Szegedy *et al.* [46] published their paper in 2013, where they studied them on neural networks. The number of publications in this research direction has witnessed exponential growth ever since then [5]. However, the concept of adversarial examples can be traced all the way back to 2004, with Dalvi *et al.* [12] examining linear classifiers detecting email spam as the first demonstration on this subject.

The majority of the scientific literature on adversarial examples focuses on neural network classifiers, presumably due to their omnipresence and high performance in real-life scenarios. Regardless, adversarial examples have been shown to affect other types of classifiers as well, such as decision trees, support vector machines, and k-nearest neighbor classifiers [35]. Thus, adversarial examples are not an unwanted property of neural networks alone but are inherent to machine learning itself. Despite their niche status in real-life scenarios as of yet, there has been an abundant amount of research on this topic. However, the existence of this phenomenon is puzzling; regardless of this research, it is

still unclear exactly why these errors arise.

Due to the breadth of this subject, adversarial examples can be divided further into subcategories. In this thesis, we focus on the subcategory called decision-based attacks. Previous publications in this category poorly quantify the differences between different dimensionality reduction methods in their adversarial example generation algorithms. The magnitude of this dimensionality reduction and its effect on the process is also seldom explored. Quite often, the authors only state the selected dimensionality reduction method and its magnitude. Only sometimes, this choice is ratiocinated with a passing reference to their experiments, and even more rarely, any actual results of these experiments are reported. Therefore, this thesis aims to quantify better the difference between different dimensionality reduction methods and their magnitudes. The research questions in this thesis are:

1. What are the current state and challenges of decision-based attacks in the scientific literature?
2. What are some of the different dimensionality reduction methods that could be possibly utilized in attack generation?
3. What are the differences between the performances of different dimensionality reduction methods, and how much can the dimensions be reduced?

This thesis consists of six chapters. First, we offer background on machine learning and the chosen dimensionality reduction methods in Chapter 2. Then, the scientific literature is reviewed to understand the current state of the adversarial examples themselves in Chapter 3. We provide a diverse view of the phenomenon and discuss the often exaggerated security concerns there. In addition, we further motivate this thesis before focusing on the decision-based scenario in more detail. In Chapter 4, the chosen algorithm and the implementation details are discussed. The selected methods are applied, and the results are analyzed in Chapter 5 to offer a more quantitative look into the effect of dimensionality reduction. Closing thoughts and conclusions are provided in Chapter 6. In the end, we hope the reader is left with a solid understanding of this multifaceted phenomenon and satisfying experimental results.

2. BACKGROUND ON MACHINE LEARNING

Machine learning (ML) is a field of artificial intelligence aiming to create a system capable of performing tasks autonomously. In general, the purpose of ML algorithms is to "learn" from given training data. Learning is a process in which the algorithm learns to utilize given data to improve its performance in the given task. It has the advantage of not having to explicitly specify decision rules and operational instructions for every possible scenario, which is often infeasible in real-life problems. When properly implemented, the system can generalize to unseen cases and, compared to human performance, enables the utilization of the information embedded in much larger datasets. The focus of this thesis is on supervised tasks; for that reason, ML is inspected from that viewpoint only. We mostly follow the book by Russell and Norvig [41] up until Section 2.2.

Supervised learning is an ML paradigm where the training data provided to the ML system is labeled. In labeled data, each input vector is associated with a label. The objective of supervised learning is to learn an approximation of the true underlying function of a given phenomenon since finding the exact true function is almost always impracticable. As the task's difficulty rises, so does the challenge of finding this exact true function.

The goal of supervised learning is to learn a function $f : X \to Y$ that maps input values from input space $X \in \mathbb{R}^d$ to output values in output space $Y \in \mathbb{R}^c$. The learning takes place by providing the algorithm a training set of n example input-output pairs $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)$. Each $\boldsymbol{x}_i$ is a d-dimensional vector with d *features* (or *variables*) in it, and y_i is the *class* (or *label*) of that input vector. In single-label classification tasks[1] $y_i \in \{1, 2, \ldots, c\}$, or if y is one-hot encoded, y_i is a vector of size c that is all zeros except for the class that it has returned as its output, which is indicated with 1. One-hot encoding is common in multi-class settings ($c > 2$) for the algorithm to avoid creating a fictitious ordinal relationship in the data when the class is a nominal variable.

As each y_i is generated by the unknown true function[2] $y = f(\boldsymbol{x})$, the goal is to find a function $\hat{f}$ that approximates the true function f well enough. As the training data cannot practically ever contain all the possible input vectors, it cannot completely model

[1] In regression tasks, y_i would be real-valued, but this thesis focuses on classification tasks. Multi-label classification is also not the focus of this thesis.

[2] The function f can also be stochastic, and thus y would not be strictly a function of $\boldsymbol{x}$, but it is out of the scope of this thesis.

the target phenomenon. The function $\hat{f}$ is called a *hypothesis*, and the learning process is a search through the hypothesis space H to find a hypothesis that performs well. In order to evaluate how well the function $\hat{f}$ performs, i.e., fits the actual function f, a *loss function* $L : Y \times Y \to \mathbb{R}^+$ is defined. For example, the loss of predicting the value $\hat{y}$ for $(\boldsymbol{x}_i, y_i)$ is $L(y_i, \hat{y})$. The goal is to minimize the output of this loss function over the given dataset. Therefore, a loss function must be chosen to quantify this amount of loss between the predicted and the actual output. Common choices of loss functions are, for example, cross-entropy and log-loss. Because the true function f cannot be directly observed, the fit is measured between the outputs of the hypothesis function $\hat{f}$ and the data used in the learning process, with 0 meaning a perfect fit.

With enough training, the loss on the training examples will eventually reach 0. However, it is not an unbiased evaluation of the model's performance. In fact, the model has then overfitted itself to the training examples, following them too closely with no guarantees of how it will generalize to examples outside of them. Therefore, additional validation and test datasets are needed. They are separate sets of examples previously unseen by the learning algorithm and independent of the training set.

These three datasets are usually partitioned from all the data available for the model creation process, with varying strategies and partition sizes depending on the circumstances. The validation set is used during development to tune the model's hyperparameters and to select the best model. The test set provides the final performance evaluation of the chosen model. Sometimes, the validation phase is omitted. Confusingly enough, the terms validation set and test set are at times used with reversed meanings. Nevertheless, the last stage should provide the final evaluation of the selected model.

As discussed above, the training data cannot practically ever contain all the possible input vectors, as the input space consists of all the possible combinations of values. It means that usually, a large portion of the input space can be considered nonsense that cannot be classified meaningfully. Instead, the data lies on a *manifold*, which represents the region the data occupies in the input space. Each class has its own manifold; combined, they form the data manifold.

The task in classification is to find a *decision boundary* (DB) that separates these different classes well enough. Incorrect modeling of these manifolds enables the existence of *adversarial examples* (AEs). Usually, the manifold is representable with fewer dimensions than it occupies in the input space. For example, a sheet of paper crumbled and twisted into a ball is in three dimensions, but it can be perfectly represented with two. However, finding this lower-dimensional representation is not a trivial task.

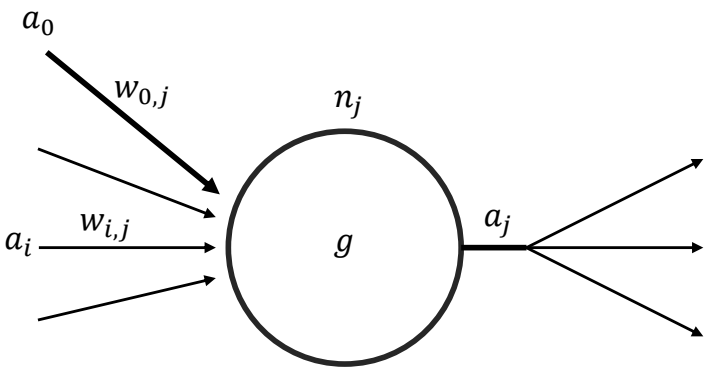

Figure 2.1. *A neuron n_j takes a weighted sum of all the inputs, applies an activation function g to it, and sends its output $a_j = g(\sum_{i=0}^{n} w_{i,j}a_i)$ to the next layer.*

2.1 Neural networks

Artificial neural networks, or simply just *neural networks* (NNs), are computing systems that draw their inspiration from the biological NNs found inside animal brains. However, they and their structure soon deviated from their true real-life precursors. They have found the most use in complex applications of ML, such as image classification and speech recognition, where they are used in a supervised manner. NNs can achieve competitive performance with humans in complex, albeit very narrow tasks, such as traffic sign recognition [11]. While the history of NNs goes all the way back to 1943 with McCulloch and Pitts [30], only recent advances in computational capacity with the old repurposed *backpropagation* algorithm have unlocked their current potential.

Practically all state-of-the-art image classification models are either based on or utilize NNs [31] due to their high performance and theoretical capability to learn any function. Because we also use an NN classifier in the testing phase of this thesis, this section serves as an introduction to them, even though AEs affect other types of ML as well [35]. NNs can also be used with alternative approaches in addition to the supervised approach, but since the focus of this thesis is on supervised tasks, we will inspect NNs from that viewpoint only.

An NN consists of units called *neurons* or *nodes*, depicted in Figure 2.1. These neurons are then connected to other neurons; these *connections* can also be called *synapses* or *links*. The purpose of a connection is to propagate the neuron's *activation* (or *output*) a_i from neuron n_i to neuron n_j. Each connection has its own weight $w_{i,j}$, which determines the strength and sign of that particular connection. In other words, the weight of a connection determines the impact the output of a neuron n_i has on neuron n_j. Usually, the weights are assigned random numerical values at initialization. Adjusting the weights to increase or decrease the impact of a given connection is an essential part of the learning phase of NNs.

The inputs to a neuron come from the data itself, or they can be outputs of other neurons. The output of a neuron is computed by taking a weighted sum of all the inputs to that particular neuron. Each connection to the given neuron carries one input signal with the weight of that particular connection. This includes the *bias term* a_0 and its weight $w_{0,j}$. Then, an *activation function* g is applied to the result. The final output of a neuron n_j is then represented as $a_j = g(\sum_{i=0}^{n} w_{i,j} a_i)$. Thus, each neuron can have multiple inputs, but they produce a single output. This single output can be sent to multiple other neurons, however. The initial inputs to the network are the data itself, such as images or text, and the ultimate outputs of the network complete the task, such as image classification.

The activation function determines the output of a neuron; for example, it can impose a threshold the output value must exceed for the neuron to send its signal forward. The activation function can either be, e.g., a hard threshold or a logistic function. If the activation function is nonlinear, it enables the NN to represent a nonlinear function. It is an important property to have, as many real-life classification problems are not linearly separable.

An NN is composed of different layers. The first layer is the input layer, where the signal is fed to the NN, and the last layer is the output layer, where the result is given; in between are the hidden layers. These hidden layers may each perform different kinds of operations on their inputs. If there are multiple hidden layers, the network is called *deep neural network*, and the learning process is called *deep learning*. Two layers can be connected in different ways. In fully connected layers, every neuron in one layer connects to every neuron in the subsequent layer. In pooling layers, a group of neurons in one layer connect to a single neuron in the subsequent layer. NNs can also be categorized by the direction in which their connections flow. If they only go to the following layer, the network is called a *feedforward network*; alternatively, a *recurrent network* allows connections between neurons in the same or previous layers.

When designing the network architecture, *hyperparameters* must be set. They are things such as the number of hidden layers, the number of neurons in each layer, the learning rate, and the connectedness of each layer. There are multiple different types of NNs, such as *convolutional neural networks* (CNNs) and *residual neural networks* (RNNs). Despite this, they always consist of these same building blocks of neurons, connections, weights, biases, and activation functions. Also, ultimately the signal always traverses from the input layer to the output layer, possibly going through the layers multiple times in the process.

More neurons and connections allow for more complex function representations. But if the network becomes too large, the weights between the neurons must be kept small. It prevents the network from overfitting the training data, preserving its generalization properties. However, this causes the activation values to stay in the linear region of the activation function. It means the network behaves like a linear function representable with far fewer parameters. Hence, a larger network is not always automatically a better one.

2.1.1 Training neural networks

For any ML model to complete its task, it must be trained first. The training consists of adjusting the weights and optional neuron thresholds of the network. The objective is to improve the accuracy, i.e., to minimize the observed errors. The error vector describes the error between the true values and the network outputs in the training phase. The training continues as long as the error decreases; the learning algorithms usually have an early stop mechanism if the error does not decrease meaningfully. Should the error rate be too high after the training has finished, the network must be redesigned or discarded.

However, this error describes the situation only at the output layer. NN consists of multiple layers with multiple neurons and weights between them that need to be adjusted to minimize the error at the output layer. The most common algorithm for this adjusting, or training, is called backpropagation. Let Δ_k be the measured error at neuron n_k at the output layer. The idea behind backpropagation is that a neuron n_j in the preceding hidden layer can be thought to be contributing to some fraction of the error Δ_k. Thus, the Δ_k values are divided between the connections to the neuron n_k according to the connections' weights and then propagated back to provide the Δ_j values for the preceding hidden layer. Propagating the Δ values back to the previous layer and updating the weights between the two layers starts from the output layer. It is repeated for each layer in the network, all the way to the earliest hidden layer.

2.2 Dimensionality reduction

In *dimensionality reduction* (DR), data is transformed from a higher-dimensional variable space $\mathbb{R}^d$ into a lower-dimensional variable space $\mathbb{R}^m$, where $0 < m < d$. The objective is that the lower-dimensional representation of the data loses as little meaningful information as possible. Ideally, this lower-dimensional representation is close to the *intrinsic dimension* of the dataset, i.e., the number of variables needed for a minimal representation of the original dataset.

DR is vital as d grows. The increasing dimensions make it more difficult and sometimes completely intractable to analyze the data or to search for a solution to a given task. For example, a square RGB image of only 224 pixels already has $3 \times 224 \times 224 = 150\,528$ dimensions in it. This makes searching for the adversarial directions more difficult as the number of dimensions increases rapidly. Very rarely, all the dimensions are needed to represent the data if the d is large, for high-dimensional data matrices are often sparse due to the curse of dimensionality.

DR has been the key in most of the recent and more efficient attacks in our selected category. The generation of perturbations is limited to a lower-dimensional subspace, after which they are projected back into the original space. It allows the attacker to find

AEs faster and more efficiently due to the smaller search space. On the downside, this limits the attacker's degrees of freedom and can sometimes result in larger perturbations in scenarios with unlimited queries.

DR as part of the attacks is further discussed in Subsection 3.5.2. In the following subsections, we will present the DR methods used in this thesis. Their formal definitions are left out for conciseness. Further details surrounding their implementation are provided in Section 4.2.

2.2.1 Discrete cosine transform

Discrete cosine transform (DCT) is a signal transformation technique that expresses a given signal in a sum of cosine functions of varying frequencies. There are different variants of DCT; we will use the most common variants, DCT-II and DCT-III, for the transformation and its inverse, respectively. DCT is closely related to Fourier transforms and similar to the *discrete Fourier transform* (DFT). The difference is that DCT uses only cosine functions instead of both sine and cosine functions used by DFT. Thus, DCT is represented with only real numbers, whereas DFT requires a complex number representation.

DCT is equivalent to DFT of roughly twice the length, which is why DFT is not compared in this thesis at all. There are also many other reasons why DCT triumphs over DFT in compression tasks. The boundary conditions of DCT allow for better compression over DFT and better computational convenience. These conditions smooth the function being represented by reducing discontinuities, which means better compression as fewer sinusoids are needed to represent that function. Using cosine over sine functions is also more efficient; fewer cosine functions are needed to approximate a typical signal. Hence, DCT is preferred over DFT in these circumstances.

DCT is not a DR technique per se, but it can be used to restrict the perturbations only to the low-frequency components of the image. The more pixels it takes to undergo a change, the lower the frequency is, and vice versa. In images, low-frequency components refer to regions where the pixel intensity changes slowly, such as a blue sky. The high-frequency components refer to the fine detail in the image.

The choice to focus only on the low-frequency components draws inspiration from image compression and especially the JPEG codec, which also utilizes DCT to compress the image. The low end of the frequency spectrum is more crucial in defining the image and its contents, while high frequencies are more associated with noise. Guo *et al.* [21] demonstrated the efficacy of restricting the perturbation to the low end of the frequency spectrum. This prompted the subsequent papers which utilize DCT to do the same since then.

2.2.2 Interpolation, maxpooling and minpooling

Interpolation is an estimation method for finding new intermediate data points within the range of known data points. In other words, interpolation estimates a value between previously measured data points. There are many different types of interpolation, such as nearest neighbor, linear, and cubic interpolations. In this thesis, we use bicubic interpolation to scale the image both up and down. Despite being the slowest option available, it is the smoothest applicable option and usually results in the best quality in image processing applications.

Maxpooling is a standard method used in the hidden layers of CNNs. Maxpooling slides a $k \times k$ window called *kernel* over the image and retains only the highest value within that window. Here, the stride is equal to the kernel's size, meaning that each pixel in the image is visited exactly once. The difference between interpolation and maxpooling is that interpolation does not use the highest value. While maxpooling is used very differently in CNNs, it can be applied here to see whether it is easier to craft AEs with the highest value rather than the value provided by interpolation. For the same reason, we also experiment with *minpooling*, which works exactly the same way as maxpooling, except that it retains the smallest value within the window. As with interpolation, the upscaling is done via bicubic interpolation.

2.2.3 Principal component analysis

Principal component analysis (PCA) aims to extract the most important information in a given $n \times d$ data matrix by computing new vectors called *principal components*. They are linear combinations of the original variables that explain the most variance in the given data in decreasing order. Therefore, the first principal component is the linear combination of the original variables explaining the most variance. The second principal component then has the constraint of having to be orthogonal to the first principal component, and it has to explain the second most variance. This process is then repeated for the rest of the principal components. Each is orthogonal to the others and explains the most variance in decreasing order, resulting in a total of $\min(n, d)$ principal components. PCA can be utilized in DR by keeping fewer principal components than there are in total.

Before PCA is applied, the original variables (i.e., columns) are centered so that the mean of each variable is 0. Their scales are also unified to the same scale by, e.g., standardizing the data. This is done because PCA finds principal components that explain the most variance. If the scales of the variables are different by orders of magnitude, the higher values will dominate the variance. Thus, the variables with high values will form the principal components, with little or no contribution from the other variables. A unified scale ensures that all the variables will be taken into account, as the variables with a

smaller scale might be more useful than the variables with a higher scale.

2.2.4 Independent component analysis

Independent component analysis (ICA) attempts to decompose a multivariate signal into its independent components. The original signal x is assumed to be consisting of independent source signals s that are weighted by the mixing weights a. Thus, the assumption is that x is a linear mixture between these two. ICA also assumes that the underlying sources are statistically independent of each other and that their distributions, even though unknown, are non-Gaussian. The goal is to perform a linear transformation of the original data into these maximally independent components.

Whereas PCA finds axes of most variance and projects the data onto them, ICA assumes that the original signal consists of multiple independent components with varying weights and tries to separate them. While ICA is not a DR method in and of itself, we will test its impact in finding the directions for the perturbations out of curiosity.

3. ADVERSARIAL EXAMPLES AND RELATED WORK

Now that all the background information is available to the reader, the AEs themselves are introduced in this chapter. More precisely, they are presented in Section 3.1. We discuss the security concerns and distinguishability in Section 3.2, and a concept called threat models is examined in Section 3.3. We explore a collection of hypotheses for the existence of AEs in Section 3.4 and present a selection of the history of our selected attack category in Section 3.5.

3.1 Adversarial examples

AEs are inputs that have been deliberately manipulated to obtain an erroneous classification result from an ML system. They are usually constructed by adding *perturbation* (or *distortion*) to unmanipulated inputs called *clean inputs* or *original inputs*. A demonstration of a minimized AE is shown in Figure 3.1. On the left is the original image, in the middle is the added perturbation and on the right is the AE. For humans, the difference is indistinguishable due to the minimality constraint. Despite the apparent resemblance between the perturbation and random noise, the difference is that the perturbation is manufactured, which is why their statistical properties differ. For this reason, the AEs affect ML models differently than random noise. Feeding an AE to an ML system to get a misclassification is called an *attack*; its alternative names include *evasion* and *dodging attack*.

Per Szegedy *et al.* [46], AEs are defined as an optimization problem:

$$\begin{aligned} \min_{\boldsymbol{x'}} \quad & ||\boldsymbol{x'} - \boldsymbol{x}||_p, \\ s.t. \quad & f(\boldsymbol{x'}) = l', \\ & f(\boldsymbol{x}) = l, \\ & l \neq l', \\ & \boldsymbol{x'} \in [0,1]^d, \end{aligned} \tag{3.1}$$

where $\boldsymbol{x}$ and l are original input and its label, $\boldsymbol{x'}$ and l' are AE and its label, $||\cdot||_p$ is the distance between samples with a chosen distance metric (here, l_p-norm), and $\boldsymbol{x'} - \boldsymbol{x} = \boldsymbol{\epsilon}$ is the perturbation. The input is represented with values in a range from 0 to 1. The objective in this often referenced definition is to minimize the perturbation.

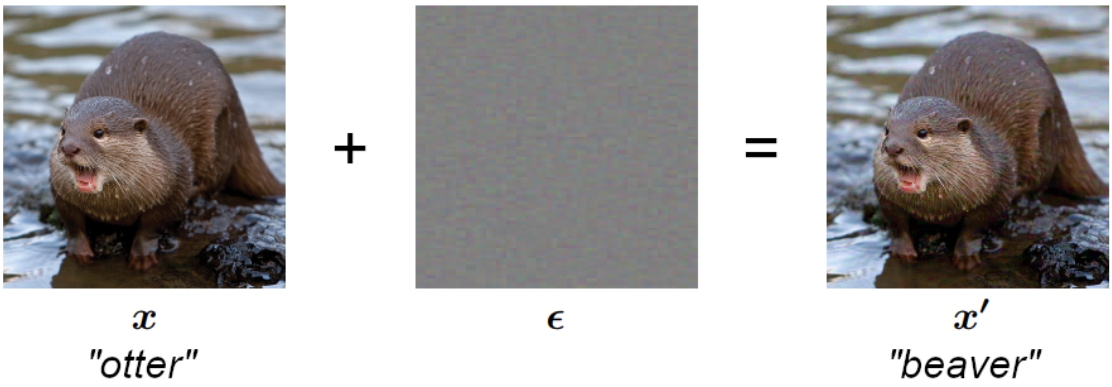

Figure 3.1. *A demonstration of a minimized AE. By adding a small perturbation to the original image, it can be seen how a tiny modification can cause a seemingly irrational change in classification. Original image is from the ImageNet [40] dataset.*

3.2 Distinguishability and security concerns

In this section, we will discuss AEs from a security standpoint and explore their motivations. A more interested reader is referred to the thorough paper by Gilmer *et al.* [18], as we concisely examine this topic. Firstly, despite our demonstration in Figure 3.1, the security concerns raised by AEs are often exaggerated from an economical point of view. Crafting AEs requires extensive ML know-how, and there are usually other simpler alternatives. Perhaps the most famous example of real-life AEs is by Eykholt *et al.* [14], where they slightly modify a stop sign to fool autonomous vehicles. While these types of scenarios certainly are to be taken seriously, in this case, it usually is easier for the attacker to merely cover or displace the sign altogether to achieve the same effect.

Secondly, it has been argued that the minimality constraint is often overemphasized, and the size of the perturbation does not matter [18, 44]. Hence, the chase for imperceptible perturbations should be motivated better, as the ML system does not have the concept of "indistinguishable" and "distinguishable" perturbations. Thus, it will give an erroneous classification when facing an adequate perturbation, regardless of its magnitude. This concept of distinguishability only applies to the human viewpoint, and the difference between human and machine perception has indeed been highlighted [18, 44] (see also Section 3.4). Ergo, with this argument, the perturbation size should not matter should the machine operate effectively autonomously.

If this restriction for minimizing the perturbation is then removed, any input that aims to force a mistake from the classifier qualifies as an AE. This includes inputs humans would deem as nonsense. Furthermore, it has been questioned whether the currently used similarity measurement methods suit the human perceptual system sufficiently [50, 45].

Nevertheless, this phenomenon does exist, despite its niche status in real-life scenarios, as of yet. It could prove to be more problematic in the future as the popularity of ML

increases. For example, sensor wear and minor statistical anomalies are possible in real-life scenarios, yet they should not affect the model the way AEs do. Indeed, as the AEs exploit the natural imperfections of the model, there might not be an attacker in the first place. Technically, an unmanipulated but misclassified input acts the same way as an AE. However, the attacker's presence is not ruled out; they can exploit this naturally occurring phenomenon.

In addition, it must be noted that the attacker does benefit from an indistinguishable attack. They might be able to utilize it for longer before being detected, even though this rarely is a strict requirement. The black-box decision-based scenario provides a realistic experiment to see how easy it would currently be to fool an unknown classifier in this way. The query count and other parameters can be limited to allow perturbations of any size, while indistinguishable ones are achievable with a larger query budget. Thus, indistinguishability is not a strict requirement but rather an option.

Despite all this, most of the research focuses on this formulation of minimizing the perturbation, as defined in Equation 3.1. Indeed, while Szegedy *et al.* [46] revitalized this research direction with their minimized optimization problem, even Dalvi *et al.* [12] in the first paper on this subject focused on minimizing the cost, i.e., the perturbation, of their AEs. Instead of blindly pursuing the imperceptible perturbations, this thesis aims to provide research into a more efficient generation of AEs. Therefore, our goal is to provide an ML contribution with hopefully generalizable results rather than to raise the alarm with security concerns.

3.3 Threat models

Threat models is a term used to describe the attacker's capabilities. As the attacks can be categorized in various ways, this section is meant to provide a taxonomy of the AEs for the reader. While defending against AEs is also one aspect of this phenomenon, it is omitted for brevity since it is not the focus of this thesis.

3.3.1 Attacker's knowledge of the target model

White-box attacks are attacks where the attacker knows the target model thoroughly. This information includes, for example, the model's parameters, training data used, weight matrices, and the model's architecture. While this allows for a very precise and fine-tuning approach, it is not a relatively realistic scenario since the information about the target model is rarely accessible to the attacker.

Sometimes in the scientific literature, the term *grey-box attack* is used to describe a situation where some, but not all, knowledge of the target model is available to the attacker. These can be, for example, some of the training data used, the model's architecture, or

some of its defenses. In any case, the information available about the target model is partial and not complete.

In *black-box attacks*, the attacker has very limited or no information at all about the target model. This makes it significantly harder to craft AEs with small perturbations, especially with images, as their dimensionality rapidly grows the larger the images become. Black-box attacks are usually further subcategorized into:

i) *Transfer-based attacks*, in which a substitute model for the target model is trained. As the attacker has white-box knowledge of the substitute model, they can utilize more efficient AE generation algorithms. The attacker then tries to fool the target model by feeding it an AE generated on this substitute model, trying to exploit an adversarial property called *transferability* [46, 35]. According to it, an AE generated on one model has a considerable probability of fooling another model, given that they are trained to perform the same task. However, training a substitute model is usually expensive resource-wise.

ii) *Score-based attacks*, in which the attacker has access to the logit output or the class probabilities of the target model. In other words, the attacker might get access to the top-k classes and their probabilities for a given input.

iii) *Decision-based attacks* (DBAs) (also known as *hard-label attacks* or *query-based attacks*), in which the attacker only receives the most probable class predicted by the ML system for a given input. In other words, the attacker only receives the top-1 class information with no probabilities whatsoever, making this a stricter scenario than the score-based attacks. DBA is considered the most challenging and most realistic of these scenarios, as it depends on nothing else but the most probable class information.

iv) *Non-traditional attacks*. Whereas most attacks try to minimize the perturbation either regarding the l_2-norm or l_∞-norm, non-traditional attacks abandon these norm-based constraints. The approaches in this category include, for example, recoloring the image or replacing a certain area of the image with an adversarial patch.

Black-box attacks, and more specifically DBAs, are considered to be the most realistic scenario in the sense that the attacker only has little or no information about the target model. It is usually the case in a real-life scenario, and the algorithm selected for this thesis is indeed from the DBA category.

3.3.2 The targeting method

The attacks can also be categorized by their specificity:

i) *Untargeted attacks* are counted as successful if they manage to change the classi-

fication of an input to any other class than its original class, i.e. $f(\boldsymbol{x}) \neq f(\boldsymbol{x}')$.

ii) *Targeted attacks* are more specific in that for them to be successful, they have to change the classification of an input to a specific pre-defined target class t, i.e., $f(\boldsymbol{x}') = y_t$.

A targeted attack is more specific in its criterion, as there is a single target class instead of $c - 1$ target classes. This makes the targeted attack the more challenging scenario since this single target class resides in a smaller region in the input space than the $c - 1$ classes. Depending on the algorithm, it can be either untargeted or targeted. They may also provide an option to choose either via an attacker-controlled parameter. Note that in binary classification there is no difference between these two scenarios, as they reduce to the same task. In this thesis, we examine only the untargeted scenario, as the authors of the selected algorithm [29] have not implemented the targeted version at the time of this thesis.

3.3.3 Distance metrics

A distance metric is used to measure the distance between two elements in a metric space; here, the elements are vectors. In this use case, they are utilized to measure the magnitude of the perturbation by comparing different AEs to the original image. By examining this magnitude, it is possible to see if the AEs generated by the algorithm are moving closer or further away from the original image.

Different papers use different metrics, making it sometimes hard to get an objective view of their performance when comparing them to each other. This is because different distance metrics produce different results. The goal of this subsection is to serve as a guide to the distance metrics used in the DBA literature.

The most common way of measuring this distance are norms, which map vectors to non-negative scalar values. Norms can also be used to measure the size of a vector. Formally, the l_p-norm is defined as:

$$||\boldsymbol{x}||_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}.$$

The distance between two vectors is acquired by taking the norm of their difference: $||\boldsymbol{x} - \boldsymbol{x}'||_p$. Usually, the perturbation is optimized only with one pre-selected norm. The most used norms in the scientific literature are:

1. l_0-norm: even though this is not technically a norm, this term is used in the literature to measure how many values were changed between the two vectors, i.e. $x_i \neq x'_i$. For example, in images, the l_0-norm is used to measure how many pixels[1] were

[1] In RGB images, every pixel consists of 3 color channels, and a pixel changes if at least one of the color channel values is changed.

changed, disregarding the information about how large the changes were.

2. l_2-norm: measures the Euclidean distance between the vectors. Even if there are many changes between the corresponding values, Euclidean distance can remain small, if the changes themselves are small.
3. l_∞-norm: measures the maximum change to any value:

$$||\boldsymbol{x} - \boldsymbol{x}'||_\infty = \max(|x_1 - x'_1|, \ldots, |x_n - x'_n|).$$

This means that only the highest change in any of the values is measured, and the information about how many of the values were changed is disregarded.

For example, Li *et al.* [24] used an alternative to l_p-norms by using mean squared error to measure the distance between two images:

$$MSE = \frac{1}{w \cdot h \cdot c} \sum_{i=1}^{w} \sum_{j=1}^{h} \sum_{k=1}^{c} (x[i,j,k] - x'[i,j,k])^2.$$

This approach was also adopted by Wang *et al.* [49], where they utilized root mean squared error: $RMSE = \sqrt{MSE}$.

However, as l_p-norms poorly measure how humans perceive differences [50, 45], it has been questioned whether other alternatives should be used in signal processing applications. Using a metric that mimics the human perception system better would also better motivate the chase for imperceptible perturbations. Since MSE and RMSE are very closely related to l_p-norms, they do not hold an advantage in this matter. Nevertheless, the scientific literature primarily uses l_p-norms, so this thesis will also adopt the l_2-norm as the distance metric. In the end, there is no consensus on what distance metric should be used, so it is at the authors' discretion to choose one.

3.4 On the existence of adversarial examples

AEs are a fascinating phenomenon haunting the research community. Countless hypotheses have been presented in the scientific literature, but many of them either fail to generalize or are in conflict with each other. In this section, we explore some of the explanations presented in the literature for the existence of this peculiar phenomenon.

This is not a complete overview for the sake of brevity; a more interested reader is referred to excellent papers such as the ones by Akhtar *et al.* [1], Balda *et al.* [2], and Serban *et al.* [44] for a more thorough overview. Despite the traces of adversarial ML going as far back as 2004 with Dalvi *et al.* [12], the reader is referred to the paper by Serban *et al.* [44] for these early phases. Again for brevity, we restrict ourselves to the literature from the year 2013 and onwards when this phenomenon resurged, as most of the literature does. Even

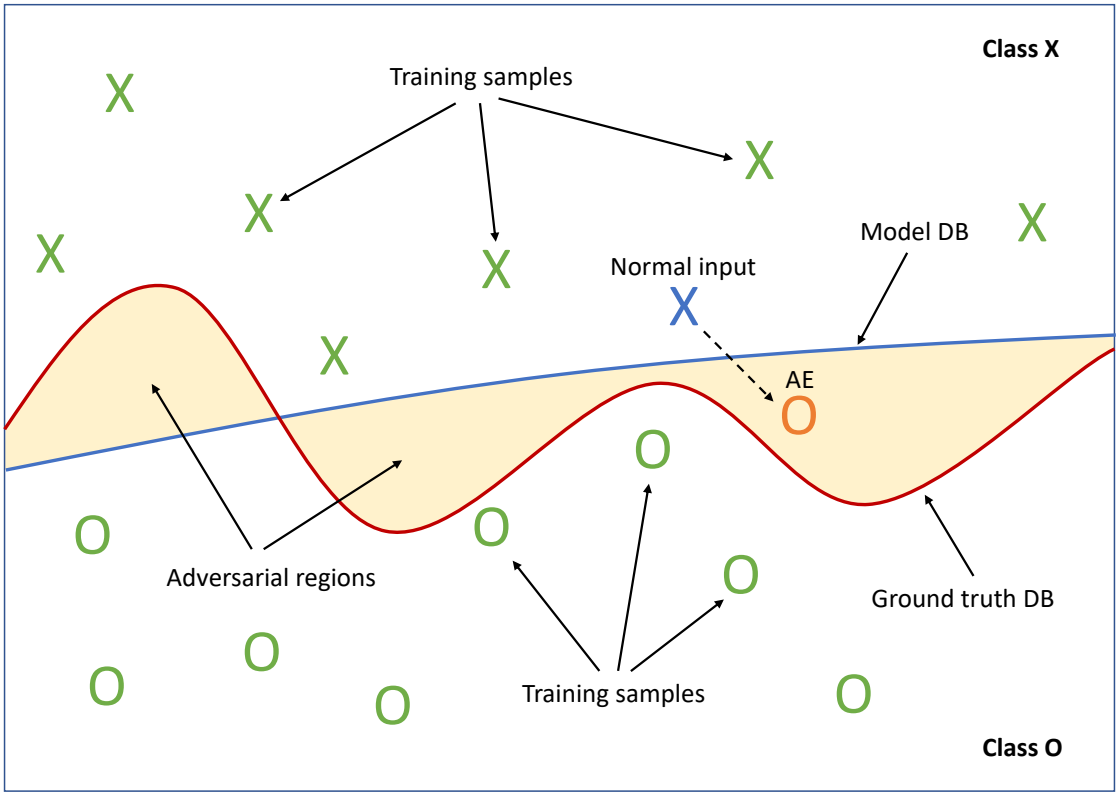

Figure 3.2. *Visualization of the AEs in the input space. Two classes, X and O, are separated by the curved red DB. Due to the model's approximation of this DB (depicted with the slightly curved blue line), the model misclassifies the slightly modified normal input.*

though this phenomenon has been mostly studied on NNs, the adversarial vulnerability exists independent of the ML model in question [35]. Indeed, there are papers specifically for, e.g., decision-tree AEs.

While the exact reasons for the existence of AEs are unknown — as in, what is the true culprit and how to fix it — the basic idea is as follows. As noted above in Chapter 2, in supervised ML, the goal is to find a good enough approximation for the ground truth function describing the given phenomenon. It is because finding the exact ground truth function is almost always impossible. However, it is this property of ML enabling the existence of AEs, since the approximated DB does not exactly match the ground truth DB. In our image classification task, the ground truth can be thought of as representing human perception.

This mismatch leads to regions in the input space where the attacker can "push" an otherwise correctly classified sample over one of the boundaries without crossing the other. Thus, the ML model makes an erroneous classification from a human perceptual standpoint. We illustrate this in Figure 3.2. The two classes, X and O, are separated by the red ground truth DB, and the model has approximated it with the slightly curved blue DB. The attacker can nudge the normal input into the adversarial region, prompting

a misclassification. In addition, as discussed in Section 3.2, the input can be adversarial even without an adversary. If the model encounters an unmanipulated input in place of the orange O in Figure 3.2, the true class should be X, but the model misclassifies it as O nonetheless.

Note that this is a perfectly valid model; the classifier has correctly classified every input it has encountered. However, the training data cannot feasibly cover the entire input space, and thus the model's DB is practically always an approximation. As the phenomenon becomes more complex, it is increasingly difficult to model the real DB sufficiently.

Then the question is, what causes this mismatch between the ground truth and learned DBs to be large enough to be exploited? Despite the substantial research, it is still not clear. However, there are hypotheses about them, which we will now explore.

3.4.1 Initial hypothesis and linearity of the model

In the initial paper that caused the resurgence of research into AEs, Szegedy *et al.* [46] theorize that the AEs are inputs of extremely low probability. They claim that while input transformations are used during training in many models to improve performance, they are highly correlated, unlike AEs. The authors continue by stating that these low-probability areas are hard to reach efficiently by this kind of sampling and that they form "adversarial pockets" in the data manifold. Furthermore, they speculate that the discontinuities of the model contribute to this phenomenon's existence.

Gu and Rigazio [20] followed by examining these adversarial regions, which they found to be relatively large and locally continuous. They claim the training procedure and objective function to be the culprits and call for changes in the training procedure. They suggest the model should learn flat and invariant regions around the training data to combat this problem.

Goodfellow *et al.* [19] first presented the linearity hypothesis on NNs. They argue that even though NNs are nonlinear in nature, they exhibit linear behavior. They demonstrate this experimentally by generating AEs efficiently using a first-order approximation of NNs around a given instance. Goodfellow *et al.* continue by stating that the input space is not full of small adversarial "pockets"; instead, they found the adversarial space to be large and high-dimensional. They also add that the direction of perturbance is more important than the specific point in the input space and that models trained to model the input distribution are not resistant to AEs.

Despite its results, the linearity hypothesis was at least partly challenged, as nonlinear ML models are equally vulnerable to AEs. Furthermore, the general nonlinearity of NNs also makes them unable to be replaced by a linear classifier completely. The linearity hypothesis was also refuted with AEs which cannot be explained by it [27, 42]. Luo *et*

al. [27] refined the linearity hypothesis further by proposing that the DBs can be approximated well by a linear boundary around the objects recognized by the model. In other words, they are *locally* linear and can exhibit nonlinear behavior around instances not recognized by the model.

Fawzi *et al.* [15, 16] continued and examined the effect of the curvature of the DB, providing complementary results for the local linearity hypothesis. They show that the robustness of the classifier against AEs can be increased by introducing a bit of curvature to the DB. Thus, they assume that limiting the curvature of DBs increases sensitivity to AEs. They also show that the AEs can exist if the boundaries are flat along most directions and highly curved only along a few.

While the (local) linearity hypothesis has been challenged, it is utilized effectively in some AE generation methods. This means that while the (local) linearity cannot explain the phenomenon by itself, it most likely is an important contributing factor in the existence of AEs.

3.4.2 Feature selection

Tsipras *et al.* [48] argue that there exists a trade-off between standard accuracy and adversarial robustness, which arises from standard and robust classifiers using different sets of features in classification. Ilyas *et al.* [22] make a case for feature selection and divide the features into robust and non-robust ones. They suggest that perturbing these non-robust features could explain the existence of AEs.

Ilyas *et al.* [22] continue that while robust features are more resistant to adversarial perturbations, non-robust ones are essential for generalization despite their brittleness to perturbations. These features are inherent to the data distribution, meaning different classifiers trained on that distribution likely utilize robust and non-robust features in a similar way. Since the perturbations target the non-robust ones, which are similar between different classifiers, it would also explain the transferability of AEs.

Ilyas *et al.* [22] also highlight the difference between human and machine perceptions by disentangling the features. They show that non-robust features are valuable in boosting model performance, but robust features align more with human perception. According to the authors, the training process requires "explicit human priors" encoded into it to have robust and human-interpretable models that adhere to how humans perceive similarity.

This is in line with earlier findings by Jetley *et al.* [23], where they assert that the directions NNs utilize to achieve better performance are the same that causes their vulnerability to AEs. However, AEs can exist even if there are no vulnerable directions to exploit in the data distribution [33]. This further hints at the existence of both on-distribution and off-distribution AEs.

3.4.3 Model geometry and other explanations

The evolutionary stalling hypothesis by Rozsa *et al.* [39] suggests that most of the training data lie close to a DB, and therefore small perturbations are enough to change the classification. In a similar vein to Gu and Rigazio [20], they argue that this stems from the inability of the training process to create flat regions around the training samples. They suggest that as the network weights are adjusted using the gradient of loss during training, the contribution of already correctly classified inputs to the loss diminishes.

Tanay and Griffin [47] put forward boundary tilting as their hypothesis. Instead of the higher dimensional space, they assume that the data lies on a low-dimensional manifold. They argue for the existence of many classifiers with similar accuracy, with the difference being that the boundary is tilted w.r.t. the ground truth boundary. This results in easier AE generation since the tilted boundary lies closer to the data subspace than the ground truth boundary, which enables smaller perturbations to fool the classifier.

Universal perturbations — where a single perturbation can be applied to any input — were shown to be possible by Moosavi-Dezfooli *et al.* [32]. They hypothesize that the universal perturbations exploit the geometric correlations between different parts of the DBs. A prerequisite for the existence of universal perturbations for flat and even for curved DBs seems to be a shared subspace along which the DB is positively curved, at least for most directions.

Gilmer *et al.* [17] studied an idealized dataset and hypothesize that most of the correctly classified points are close to an incorrectly classified input. This means that the points lie close to a DB in a similar vein Rozsa *et al.* [39] presented in evolutionary stalling. Gilmer *et al.* argue that the AEs exist due to the high-dimensional geometry of data manifolds whenever the classifier has non-zero error rates. They also suggest that the key to defending against AEs is to reduce test error significantly. In other words, the model generalization should be improved. They managed to remove AEs on their idealized dataset, given enough data and a proper model, but the authors call for more research on real-world datasets.

The off-manifold hypothesis argues that the AEs lie off the data manifold rather than on it. This assumption means that the AEs would follow a different distribution when compared to the clean data distribution, which could lead to easy detection methods. However, Gilmer *et al.* [17] refute this hypothesis by arguing that the opposite setting of the typical adversarial scenario is also true. Because a typical incorrectly classified point can be changed into a correctly classified one with a small perturbation, there exists no identifying character for AEs for easy detection. This argument is further reinforced by Carlini and Wagner [6]. Nevertheless, there are arguments for both on-manifold [22] and off-manifold [33] AEs, with evidence on both sides hinting that both are possible.

Deniz *et al.* [13] also call for changes to the training process. Their hypothesis points to the bias-variance (also known as fitting-generalization) dilemma incorporated in ML algorithms as the root cause. It states that there is an inevitable trade-off between generalization and fitting. They question whether this has to be so, as such a trade-off may be fundamentally different from any such trade-off potentially used by the human perceptual system.

Deniz *et al.* [13] continue by arguing that the algorithm must have both overfitting and good generalization simultaneously, whereas the current algorithms are biased towards the latter. They clarify their definition of overfitting to be good fitting to the training samples. According to them, model flexibility should be improved by fitting better without sacrificing generalization, as they show that better fitting increases robustness towards AEs. They add that as the goal is to find a boundary that best separates two sets of samples, the best way to do it might not be to have the best separation between each individual sample and other samples.

Complementary results were provided by Pedraza *et al.* [37]; even though better fitting decreases performance on the test set, it increases the adversarial robustness. While there are studies that vice versa show that improved generalization results in better robustness [17], this highlights that both might indeed be needed for better robustness.

Schmidt *et al.* [43] propose that the available datasets are insufficient to train robust models. They demonstrate that the greater the model complexity is, the greater the sample complexity becomes for robust learning. They add that this difference in sample complexity for robust and "standard" learning can be significant.

In summary, it can be argued that the root cause of this phenomenon remains unsolved. It is unclear whether the AEs exist due to the limited number of samples in the training phase, generalization and fitting trade-off, or local anomalies in the model. Maybe the models lack expressive power, the points are too close to the DB, or the input dimension is too large to solve this problem efficiently. Or perhaps, the features ML systems learn to utilize differ too much from the equivalents used by human perception, and the models are fundamentally flawed by design. In any case, the phenomenon continues to exist, even though it is uncertain whether it is unavoidable.

3.5 History of decision-based attacks

In this section, a brief history of DBAs and the tracks of improvement in that category are presented. We confine ourselves solely to the DBA category in this thesis. The reasoning for this is that the selected algorithm is from the DBA category, and the advances in each of the other attack categories are just as numerous. As noted in Subsection 3.3.1, the only information DBAs receive from the target classifier for any given input is the most

probable class called top-1 class. Formally, the classifier is represented as $f : X^d \to Y^c$, the top-1 label is represented as $f(\boldsymbol{x}) := \arg\max_k f_k(\boldsymbol{x})$, where $f_k(\boldsymbol{x})$ is the predicted probability of class k, $1 \leq k \leq c$.

This section does not serve as a complete list of all the papers in the DBA literature. Instead, it represents a subjective selection of often referenced and influential publications, as the goal is to give the reader a general overview of overall trends. A list of papers mentioned in this and the following subsection can be found in Table 3.1. Even though they are not complete lists either, there exist more thorough surveys, such as the ones by Bhambri *et al.* [3] in 2020, Akhtar *et al.* [1] in 2021, and Mahmood *et al.* [28] in 2021. It must be noted that these surveys may use alternative names, such as hard-label attacks or query-based attacks, to describe DBA.

There have been even more recent papers in this category after the aforementioned surveys; some examples include `AHA` by Li *et al.* [25], a Bayesian optimization approach `BO-DBA` by Zhang and Yu [51], `Triangle Attack` by Wang *et al.* [49], and `f-attack` by Li *et al.* [26]. Despite the numerous newer candidates in the DBA category, the selection of the chosen algorithm `SurFree` [29] is motivated in Section 4.1.

3.5.1 Previous work

For a timeline of the attack algorithms, see Table 3.1. The first paper in the DBA category was by Brendel *et al.* called `Boundary Attack` [4] in 2017. The idea behind `Boundary Attack` is to find the DB between the original and an adversarial image. Then, a random walk is utilized along the DB to move the adversarial image closer to the original image. The query count for this inaugural paper was reported to be in the hundreds of thousands, even as high as one million queries on the ImageNet [40] dataset.

This was followed by `OPT attack` and the improved `Sign-OPT attack` by Cheng *et al.* [9, 10]. They re-formulate the attack as a real-valued optimization problem and use a zeroth-order optimization[2] algorithm to solve it. `HopSkipJumpAttack` by Chen *et al.* [7] was the first to implement *gradient estimation* at the DB. The gradient is the direction and the rate of the fastest increase; it can be used locally to estimate the geometry of DB. This idea was developed further in `QEBA` by Li *et al.* [24], where they estimate the gradient in various subspaces instead of the full input space. `GeoDA` by Rahmati *et al.* [38] is a geometry-based attack that approximates the DB locally with a hyperplane to find its normal vector. This normal vector can then be used to craft an AE.

Moving away from using any gradient estimation, `RayS` by Chen and Gu [8] re-formulates the problem of finding the direction to the closest DB as a discrete optimization problem.

[2] In zeroth-order optimization, the goal is to minimize the objective function when only the function value can be evaluated at chosen inputs.

Attack name	***Date***	***Author***
Boundary Attack	12 Dec 2017	Brendel *et al.* [4]
OPT	12 Jul 2018	Cheng *et al.* [9]
HopSkipJumpAttack	3 Apr 2019	Chen *et al.* [7]
Sign-OPT	24 Sep 2019	Cheng *et al.* [10]
GeoDA	13 Mar 2020	Rahmati *et al.* [38]
QEBA	28 May 2020	Li *et al.* [24]
RayS	23 Jun 2020	Chen and Gu [8]
SurFree	25 Nov 2020	Maho *et al.* [29]
BO-DBA	4 Jun 2021	Zhang and Yu [51]
AHA	13 Oct 2021	Li *et al.* [25]
Triangle Attack	13 Dec 2021	Wang *et al.* [49]
f-attack	3 Mar 2022	Li *et al.* [26]

Table 3.1. *Attacks mentioned, their original publication dates, and their authors.*

The selected algorithm `SurFree` by Maho *et al.* [29] exploits geometrical properties by assuming the DB to be locally linear. They then construct a circle on the DB with the original image and an adversarial point. After that, the optimal perturbation is found in the intersection of this circle and the DB.

As the efficiency of the attack algorithms increased, the query count was cut down to tens of thousands and even to a few thousand, with `SurFree` [29] claiming to be the first to explore DBA scenarios with less than one thousand queries. The tracks of improvement that enabled this increase in efficiency will be explored in the following subsection.

3.5.2 Improvements

There have been two major tracks of improvement in the recent works in this category: firstly, the gradient estimation step has been mostly omitted. A typical step in performing a DBA is to find the DB by finding a point in the vector space that gets labeled differently than the original input. The DB then lies somewhere between these two points and can be found, e.g., with a binary search. Then, some algorithms used a gradient estimation step to estimate the gradient of this DB and used that information to update the AE.

However, the issue with gradient estimation is that it requires multiple queries, as many small changes are made to the point lying on the boundary to estimate the boundary's gradient. It rapidly increases the number of queries the attack requires, making it very expensive from a query count viewpoint. Some algorithms [7, 38] try to mitigate this problem by dynamically modifying the queries spent for gradient estimation w.r.t. the iteration number. Even though the gradient estimation would more accurately give the new direction,

the impact this gradient estimation has in the direction search does not seemingly justify its cost. When fewer queries are spent for gradient estimation leading to a smaller decrease in perturbation, it still decreases at a higher rate since the size of the perturbation does not plateau. Thus, recent papers increasingly stray away from gradient estimation.

The second track of improvement in many recent papers [38, 29, 24, 49] is DR, which constrains the perturbation to a lower-dimensional subspace of the original input space. This makes the attack converge faster to a smaller perturbation due to the smaller search space. On the other hand, the perturbation might become larger than its non-restricted counterpart the higher the query count becomes, as the lower-dimensional subspace simultaneously lowers the attacker's degrees of freedom.

In other words, the DR vastly reduces the number of possible directions to search through. Eventually, it may result in a larger perturbation, if the query count is allowed to increase high enough since all the directions are not available for searching. However, faster attack convergence is more crucial in low query count settings, as the objective is to minimize the distance between the original and adversarial inputs as fast as possible. Our goal is to test if any particular choice of subspace and its size works better than others for finding the directions of perturbations.

4. IMPLEMENTATION OF THE TESTS

In this chapter, the details about the chosen algorithm and its motivation are presented. Furthermore, the experimental setup is specified. The results themselves are evaluated separately in Chapter 5, with further discussion in Chapter 6.

4.1 SurFree

SurFree (a fast surrogate-free black-box attack) is a recent DBA algorithm proposed by Maho *et al.* [29] in late 2020 to achieve small perturbations with minimal query counts. While it is not the newest algorithm in the DBA category, it can nonetheless be considered state-of-the-art. Many of the more recent entries in this category [25, 49, 26] compare their performance to SurFree while it retains competitive results.

The "surrogate-free" in the subtitle refers to the algorithm not using any gradient surrogate estimations nor a surrogate of the target model; the algorithm only utilizes the input itself in crafting the AE. Training a surrogate model is very expensive, as the target model is queried for non-adversarial classification to learn the surrogate for it. Often, this target model might not be conveniently available in practice, and the queries might not be free of charge. Some algorithms, on the other hand, such as QEBA [24] for its PCA version, need a large set of inputs for the perturbation sampling, which is an additional overhead for the algorithm.

There were varying reasons why some newer algorithm was not chosen for this thesis. Some of them were not chosen as they did not have a public source code when the algorithm for this thesis was selected, or their implementation did not allow for a convenient comparison of DR. Others only evaluated their results with algorithms not considered state-of-the-art nowadays, depriving us of a more objective look at their performance.

While some preceding algorithms attain comparable or better results than SurFree when the query count is allowed to grow high enough, it importantly achieves better results in scenarios with limited queries (e.g., one thousand queries). It also does not need any additional data besides the input to be perturbed. All in all, SurFree was chosen due to its good performance, public source code, independence of external data, and convenience in implementing this thesis.

4.1.1 The setting and notation

In this subsection, the basic idea of SurFree and its setting is presented, with a visualization available in Figure 4.1. For a more thorough explanation of the algorithm and its proofs, the reader is pointed to the original paper [29]. Note that the original notation is slightly altered here to avoid confusion with the notation used in this thesis.

To put it briefly, SurFree exploits geometric properties by assuming the DB to be locally linear. First, it finds a point on the DB between the original image and a random noisy adversarial point. Then, the algorithm constructs a circle on the DB with the original image and the point on the DB. The optimal perturbation is found in the intersection of this circle and the DB, and the algorithm iteratively moves closer toward it. We will now explore the algorithm in more detail.

While studying previous attacks [7, 38, 24] from the query budget viewpoint, Maho *et al.* noticed a presence of plateaus in the perturbation's size due to the queries used for the gradient estimation step. It is utilized in many DBA algorithms, and Subsection 3.5.2 offers more discussion on it. SurFree omits it completely and uses this saved query budget to investigate more directions instead.

The rationale behind this is that even though the size of the perturbation decreases more slowly without gradient estimation, it does so at a higher rate since it does not plateau. Instead of using the gradient descent approach, the approach of SurFree can be seen as a coordinate descent on a random basis. Maho *et al.* [29] do note that, in theory, the random coordinate descent is essentially the same as the worst-case rate of the gradient descent [34]. However, the saved query budget seems to outweigh the advantage of gradient estimation. As a result, Maho *et al.* call for more investigation into these conflicting arguments.

SurFree follows the optimization problem presented in Equation 3.1. Let the original correctly classified image be $\boldsymbol{x}_o$, and the outside region $\mathcal{O} := \{\boldsymbol{x} \in \mathbb{R}^d : f(\boldsymbol{x}) \neq f(\boldsymbol{x}_o)\}$. In other words, $\mathcal{O}$ represents all the images in the input space labeled differently than the original image and its label.

Maho *et al.* make the usual assumption that when knowing a point $\boldsymbol{y} \in \mathcal{O}$, it is possible to find a so-called *boundary point* $\boldsymbol{x}_b$ between $\boldsymbol{x}_o$ and $\boldsymbol{y}$ such that it lies on the DB, denoted by $\partial\mathcal{O}$. Due to the local linearity hypothesis presented in Subsection 3.4.1, it is also assumed that the boundary can be approximated by a hyperplane locally around the boundary point $\boldsymbol{x}_b$. They note that this might not always be the case, and the boundary might be convex. They offer optional methods such as interpolation and vector cycling to aid in this case. However, the original paper does not utilize these methods outside of its ablation studies, and thus they are ignored in this thesis as well.

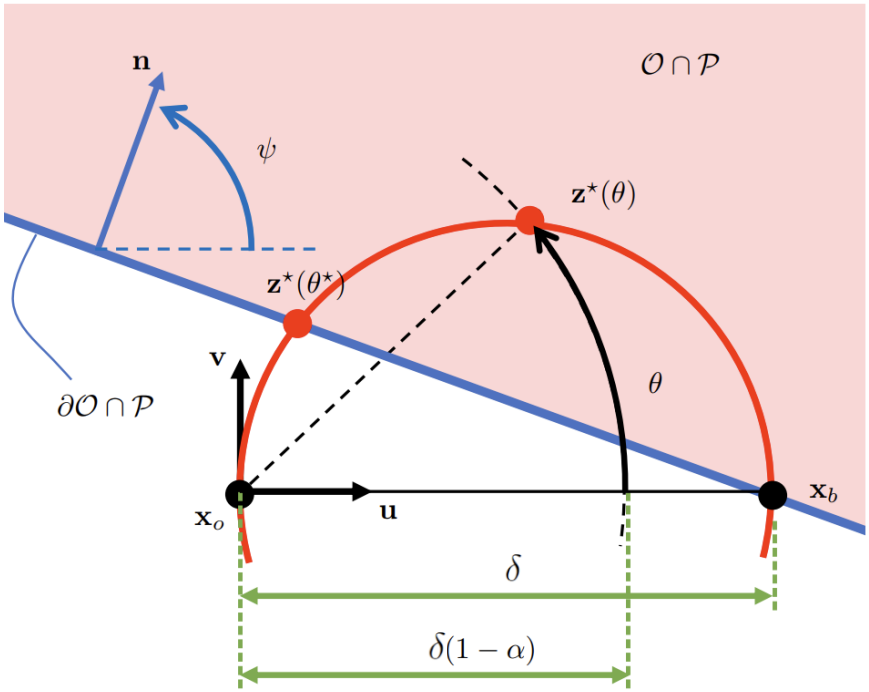

Figure 4.1. *The geometrical setting of* `SurFree` *visualized. Adapted from the original paper by Maho* et al. *[29].*

Let us define $\delta := ||\boldsymbol{x}_b - \boldsymbol{x}_o||$ and $\boldsymbol{u} := (\boldsymbol{x}_b - \boldsymbol{x}_o)/\delta$, $||\boldsymbol{u}|| = 1$. The search for a closer adversarial point is restricted to a random two-dimensional affine plane $\mathcal{P}$, which is spanned by vectors $\boldsymbol{u}$ and a random orthogonal direction $\boldsymbol{v} \in \mathbb{R}^d$, $||\boldsymbol{v}|| = 1$. This plane $\mathcal{P}$ also contains both points $\boldsymbol{x}_o$ and $\boldsymbol{x}_b$. It is assumed that the intersection $\partial\mathcal{O} \cap \mathcal{P}$ is a line that passes by $\boldsymbol{x}_b$ with normal vector $\boldsymbol{n} \in \mathcal{P}$ that is pointing in the direction of $\mathcal{O}$, $||\boldsymbol{n}|| = 1$. In polar coordinates, $\boldsymbol{n} := \cos(\psi)\boldsymbol{u} + \sin(\psi)\boldsymbol{v}$, with $\psi \in (-\pi/2, \pi/2)$.

The adversarial point in $\mathcal{P}$ is represented with polar coordinates $\boldsymbol{z}(\alpha, \theta)$; the point is at a distance of $\delta(1-\alpha)$ from $\boldsymbol{x}_o$ and makes an angle θ with $\boldsymbol{u}$:

$$\boldsymbol{z}(\alpha, \theta) = \delta(1-\alpha)(\cos(\theta)\boldsymbol{u} + \sin(\theta)\boldsymbol{v}) + \boldsymbol{x}_o, \tag{4.1}$$

with $\alpha \in [0, 1]$ and $\theta \in [-\pi, \pi]$. Hence, $\boldsymbol{z}(0, 0) = \boldsymbol{x}_b$ and $\boldsymbol{z}(1, \theta) = \boldsymbol{x}_o, \forall\theta$. Because the goal is to minimize the distance from $\boldsymbol{x}_o$, the optimal point $\boldsymbol{z}(\alpha, \theta) \in \partial\mathcal{O} \cap \mathcal{P}$ would be the projection of $\boldsymbol{x}_o$ onto this intersecting line $\partial\mathcal{O} \cap \mathcal{P}$, and it is obtained when $\theta = \psi$ and $\alpha = 1 - \cos(\psi)$. Because of this coupling between θ and α, the adversarial point can generally be represented as $\boldsymbol{z}^*(\theta) := \boldsymbol{z}(1 - \cos(\theta), \theta)$. By substituting $\boldsymbol{n}$ and $\alpha = 1 - \cos(\psi)$ into Equation 4.1, the optimal point can now be represented as $\boldsymbol{z}^*(\theta^*) = \delta\cos(\psi)\boldsymbol{n} + \boldsymbol{x}_o$. The problem for the attacker in finding this optimal point $\boldsymbol{z}^*(\theta^*)$ is that the angle ψ is unknown.

In order to find the optimal adversarial point, i.e., the projection of $\boldsymbol{x}_o$ to the boundary, `SurFree` iterates over orthonormal directions. Let $\boldsymbol{x}_{b,k}$, $\boldsymbol{u}_k$, and $\boldsymbol{v}_k$ be their respective vectors at iteration k. It is assumed that the boundary $\partial\mathcal{O}$ is an affine hyperplane passing

through $\boldsymbol{x}_{b,1}$ in $\mathbb{R}^d$ with normal vector $\boldsymbol{N}$. Consider a random basis of span$(\boldsymbol{x}_{b,1} - \boldsymbol{x}_o)^\perp$ composed of $d-1$ vectors $\{\boldsymbol{v}_i\}_{i=1}^{d-1}$. In a generalized form, the normal vector is decomposed in spherical coordinates:

$$\boldsymbol{N} = \sin(\psi_{d-1})\boldsymbol{v}_{d-1} + \cos(\psi_{d-1})\sin(\psi_{d-2})\boldsymbol{v}_{d-2} + \ldots + \cos(\psi_{d-1})\ldots\cos(\psi_2)\boldsymbol{n}_1,$$

with $\boldsymbol{n}_1 := \sin(\psi_1)\boldsymbol{v}_1 + \cos(\psi_1)\boldsymbol{u}_1$ being the l_2-normalized projection of $\boldsymbol{N}$ onto hyperplane P_1, which is spanned by vectors $\boldsymbol{v}_1$ and $\boldsymbol{u}_1 := (\boldsymbol{x}_{b,1} - \boldsymbol{x}_o)/\delta$. Then, $\boldsymbol{x}_{b,2} := \boldsymbol{z}^*(\theta^*) \in \mathcal{O} \cap P_1$ can be found as given in the definition of $\boldsymbol{z}^*(\theta^*)$ and $\boldsymbol{u}_2$ is defined as $\boldsymbol{u}_2 := (\boldsymbol{x}_{b,2} - \boldsymbol{x}_o)/\delta\cos(\psi_1) = \boldsymbol{n}_1$. After that, the problem in P_2, spanned by $\boldsymbol{v}_2$ and $\boldsymbol{u}_2$, is solved. Note that $\boldsymbol{N}^\top\boldsymbol{u}_2 \geq \boldsymbol{N}^\top\boldsymbol{u}_1$.

Iterating this process increases the scalar product between $\boldsymbol{N}$ and $(\boldsymbol{x}_{b,k} - \boldsymbol{x}_o) \propto \boldsymbol{u}_k$ given by:

$$\boldsymbol{N}^\top\boldsymbol{u}_k = \prod_{i=1}^{d-k}\cos(\psi_{d-i}).$$

In the end, this process converges to the adversarial point with minimal distortion: $\boldsymbol{x}_{b,d} \in \mathcal{O}$ and $\boldsymbol{x}_{b,d} - \boldsymbol{x}_o$ is colinear with $\boldsymbol{N}$. Thus, it points towards the optimal adversarial point, i.e., the projection of $\boldsymbol{x}_o$ to the hyperplane boundary. However, this would mean there would have to be as many iterations as there are dimensions. A clever strategy instead goes through the directions in decreasing order of their angles $(|\psi_k|)_k$, which means that the biggest distortion decreases first. This is unavailable for the attacker who does not take $\boldsymbol{N}$ into account and is unwilling to spend queries to estimate it.

4.1.2 The algorithm

`SurFree` consists of four stages: initialization, direction generation, sign search, and binary search of an angle. In this subsection, the basic ideas behind each of these four steps are presented. For the complete algorithm, the reader is referred to the original paper [29].

i) **Initialisation**. Here, the goal is to find an initial point $\boldsymbol{x}_{b,1}$ that is close to the boundary $\partial\mathcal{O}$. First, a point $\boldsymbol{y}_0 \in \mathcal{O}$ is generated. In the targeted version, $\boldsymbol{y}_0$ is an image belonging to the target class, and in the untargeted version, it is a noisy version of $\boldsymbol{x}_o$. Then, a binary search between $\boldsymbol{x}_o$ and $\boldsymbol{y}_0$ is utilized to find the initial point $\boldsymbol{x}_{b,1}$ lying close to the boundary.

ii) **New direction**. At iteration k, the point close to the boundary $\boldsymbol{x}_{b,k} \in \mathcal{O}$ defines the vector $\boldsymbol{u}_k \propto \boldsymbol{x}_{b,k} - \boldsymbol{x}_o, ||\boldsymbol{u}_k|| = 1$. A random direction $\boldsymbol{t}_k$ that is perceptually shaped as $\boldsymbol{x}_o$ is generated by transforming $\boldsymbol{x}_o$ into a lower-dimensional representation. Then, its coefficients are multiplied with a random sample from the uniform distribution over the values $\{-1, 0, 1\}$. After that, the amplitude function and in-

verse transformation are applied. This new direction is then made orthogonal to $\boldsymbol{u}_k$ and to (at most) the last L directions $V_{k-1} := \{\boldsymbol{v}_j\}_{j=\max(k-L,1)}^{k-1}$ with Gram-Schmidt procedure, producing the new directions $\boldsymbol{v}_k$.

iii) **Sign search**. Because the sign of θ depends on the sign of the unknown ψ (see original paper [29]), angles with alternating $+$ and $-$ signs are tested. The angles are tested in the order of decreasing amplitude: $\theta_{\max} \cdot \boldsymbol{\tau}$, $\boldsymbol{\tau} := (1, -1, (T-1)/T, -(T-1)/T, \ldots, 1/T, -1/T)$, with T being a user-controlled parameter. If an adversarial image is found, the search is stopped; otherwise, $\theta_{\max}$ is decreased, direction $\boldsymbol{v}$ is given up, and another direction is generated.

iv) **Angle binary search**. When an adversarial point is found by the sign search, the angle θ is refined by a binary search with at most l steps. The result is θ^*, and $\boldsymbol{z}^*(\theta^*) \in \mathcal{O}$ is the new boundary point $\boldsymbol{x}_{b,k+1}$.

4.2 Experimental setup

In this section, we specify the details of our implementation of the experiments. We follow the original paper's [29] setup to provide a fair comparison. As in the original paper, we choose ImageNet[1] [40] as our dataset, and a pre-trained ResNet18 available in the PyTorch environment[2] [36] as our classifier. 200 *correctly* classified images were randomly sampled from ImageNet's validation set, with their pixel values scaled between 0 and 1, and their size limited to $3 \times 224 \times 224$ pixels. The correct classification of the original images was ensured after these modifications.

The algorithm was given a budget of 5 000 queries. We consider it reasonable for the current state of the DBAs, as many of the more recent ones aim for results with less than 1 000 queries. With 5 000 queries, however, it is possible to examine better when the perturbation starts to plateau due to the limited search space in the smaller dimensions. It is also customary in the scientific literature to run tests with more than 1 000 queries and report the results.

The tests were run on a PC with AMD Ryzen 5 3600 CPU and NVIDIA GeForce RTX 2070 GPU, with CUDA[3] enabled whenever possible in the code. With CUDA enabled, it is possible to utilize GPU for computations, cutting down the runtimes. `SurFree`'s standalone version[4] was used with parameters from the original paper. It must be noted that `SurFree` has received updates to its GitHub after the original publication. Despite some parameters being changed as well, the changes are seemingly mostly updates for deprecated libraries, which enable the codebase to work in the first place. For this reason, we

[1] https://www.kaggle.com/competitions/imagenet-object-localization-challenge/data
[2] https://pytorch.org/
[3] https://developer.nvidia.com/cuda-toolkit
[4] https://github.com/t-maho/SurFree

use the latest version (`c9920f2`) with the old parameters from the original paper, including the $\tanh$ amplitude function.

We compare our results to `Triangle Attack`[5] (`5c468fc`) by Wang *et al.* [49], who claimed superior performance over `SurFree` in their paper. We did not choose `Triangle Attack` initially, as its source code was published when this thesis was already well underway. For fairness, we adopt original parameter values for `Triangle Attack` as well. By comparing both with their original parameter values, we hope to have a better comparability of our results.

The original paper of `SurFree` [29] tested $1/2$ (50%) and $1/4$ (25%) of the dimensions for DR but settled on using $1/2$ after their initial tests. With DCT, we test both the full-frame version and the block DCT approach. In block DCT, we divide the image into 8×8 blocks and then perform DCT on each of these blocks. It is the version used in the original paper. We denote the full frame version DCT-F and the block version DCT-B. Thus, DCT-B $1/2$ is equal to the performance of the original paper.

We keep halving the amount of DR with each method until the performance degrades. The halving was done out of convenience as then the DR evenly divides the fixed image size of $3 \times 224 \times 224$. This property is crucial for methods such as maxpooling, which otherwise would have to deal with the edges of the image as a special case. Indeed, we confirmed this by a quick test with $1/10$ (10%) without handling it as a special case, resulting in poor performance due to the uneven dividing of the image. The lowest amount of DR we will examine in this thesis is $1/256$ $(0{,}390625\%)$.

With minpooling, maxpooling, and interpolation, we set the upscaling[6] parameters to `mode='bicubic'` and `align_corners=True`. With interpolation, we used the same parameters to reduce the size of the image. We tested different combinations of the available functions and their parameter values, but they either did not have much of an impact or the performance worsened.

In addition, it is not possible to perform PCA or ICA on a single vector with $3 \times 224 \times 224 = 150\,528$ dimensions. Hence, with them, the image has been divided into 8×8 blocks as well; each block can then be represented as a 64-dimensional vector. As each image is composed of $2\,352$ blocks of this size, we can then form a $2\,352 \times 64$ matrix, on which it is possible to perform PCA and ICA.

For evaluation, we use the l_2-norm as defined in Subsection 3.3.3 to measure the average size of the perturbation. In addition, we use *attack success rate* (ASR) to determine the percentage of perturbations lower than a target ϵ at query budget K. We also measure the runtimes for each method. We will present the results in the following chapter.

[5] https://github.com/xiaosen-wang/TA

[6] For scaling up the image, PyTorch offered functions for interpolation and upsampling. Our small-scale experiments implied equal performance between the two, so we used upsampling.

5. EVALUATION OF THE RESULTS

In this chapter, we present the results of our experiments. First, we inspect how the size of the perturbation decreases w.r.t. the query count. Then, we examine the ASR and consider three different query budgets, $K \in \{200, 500, 1\,000\}$. As the objective in these attacks is to spend as few queries as possible, these relatively low query budgets allow us to examine the effect DR has in the crucial early steps. Lastly, we also present the averaged runtimes of each method. We discuss the results further in Section 6.1.

We use the term *no reduction* here to refer to the `SurFree` itself without any DR and *original paper* [29] to refer to `SurFree` with DCT-B $1/2$ (50%) as the DR method. As presented in Section 4.2, DCT-F refers to the full-frame DCT, while DCT-B refers to the block DCT. We also compare our results to the more recent `Triangle Attack` by Wang *et al.* [49] with its original parameters. We did not provide the same starting point for the algorithms nor for different DR methods, which introduces very slight randomness to our results. However, we believe this randomness does not meaningfully affect the results, as all the starting points are randomized. We discuss this topic further in Section 6.2.

5.1 Average size of the perturbation

We present the benchmark on the average size of the perturbation w.r.t. the query count with each method in Figure 5.1. Note that we omitted DCT-F $1/32$ $(3{,}125\%)$ for visual clarity; it had a similar performance to DCT-F $1/64$ $(1{,}5625\%)$. We offer a zoomed-in version in Figure 5.2 to better showcase the decrease in perturbation in the crucial early steps. Due to the limited space, interpolation's results and the zoomed-in version are provided separately in Figure 5.3. Note also that for Figures 5.1 - 5.4, we have limited the perturbation axis for better readability. Due to the random initialization step, the first queries are highly perturbed, and the figures would thus be hard to interpret without this limitation. We have provided the reasoning for the selected magnitudes of DR in Section 4.2; in short, they keep halving from the original $1/2$ until degradation in the performance is observed.

Generally speaking, DCTs, maxpooling, minpooling, and interpolation have very smooth curves, and they gain an advantage from a more extensive DR up to a certain point. Indeed, the dimensions cannot be infinitely reduced to achieve better and better results.

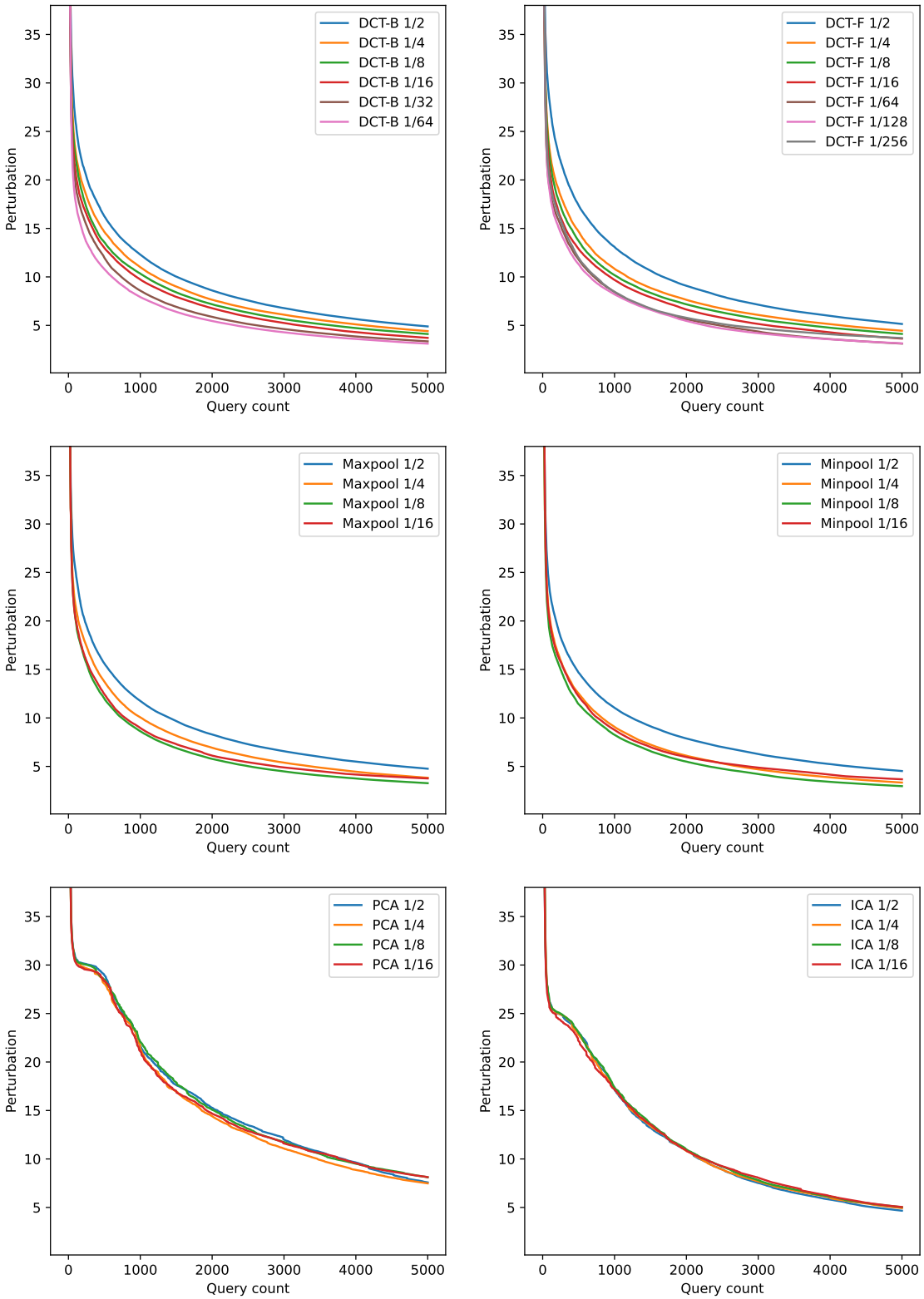

Figure 5.1. *The average size of the perturbation of each method vs. the number of queries. Note that we omit DCT-F 1/32 to avoid more clutter; it had near-identical results to DCT-F 1/64.*

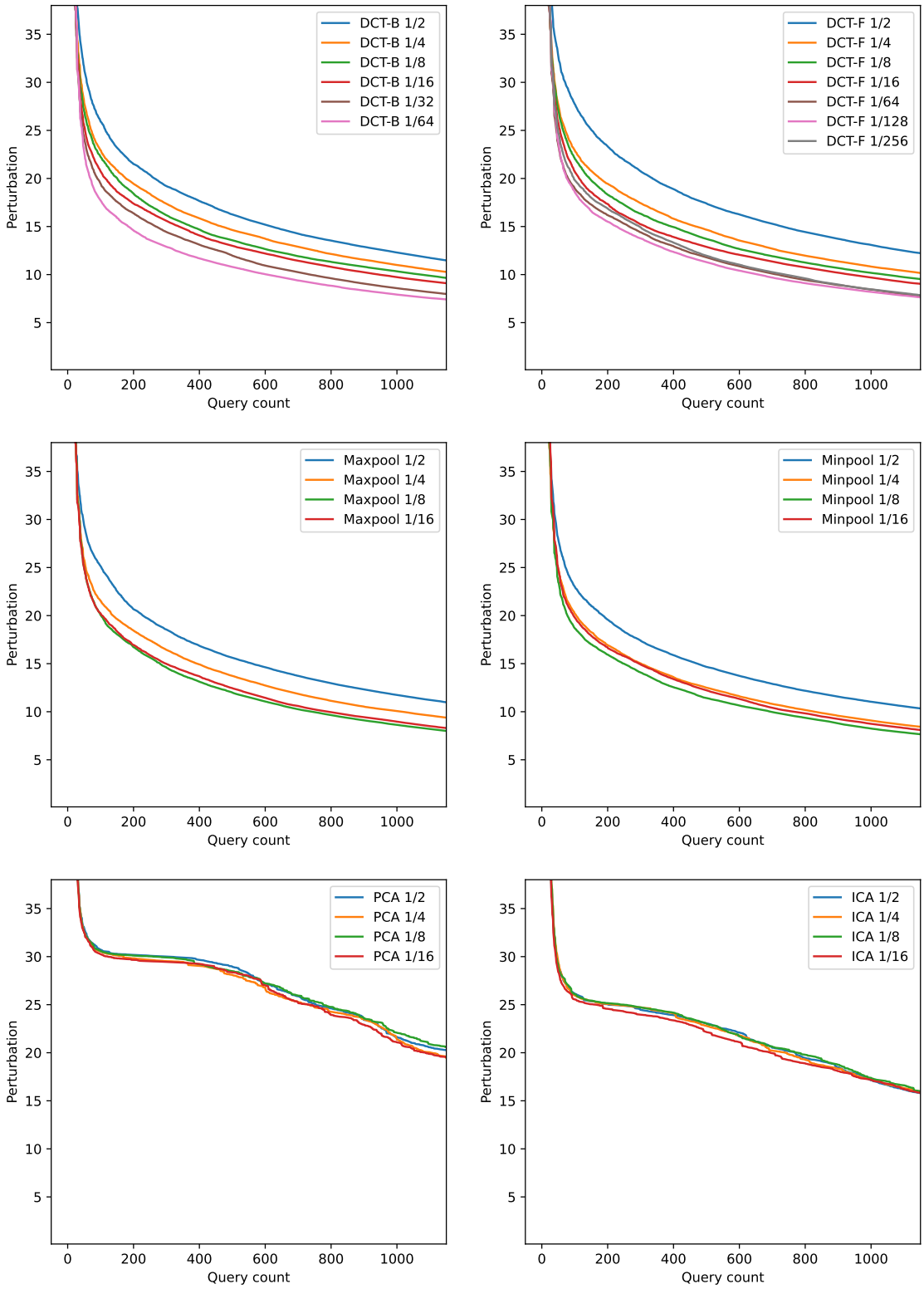

Figure 5.2. *The average size of the perturbation of each method vs. the number of queries. This is a zoomed-in version of Figure 5.1.*

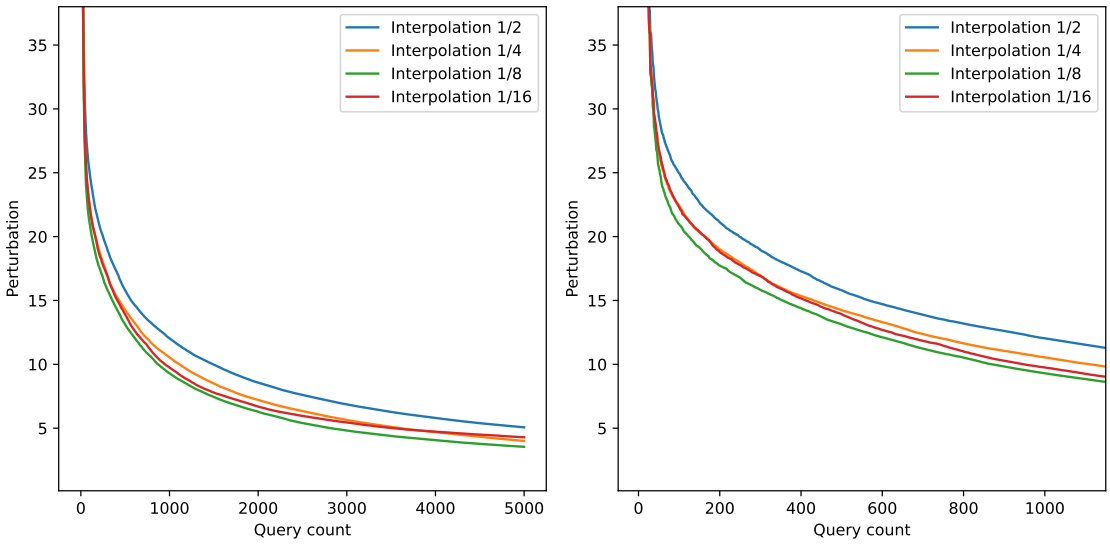

Figure 5.3. *The average size of the perturbation of interpolation vs. the number of queries, with a zoomed-in version on the right. Provided separately due to the lack of space in previous figures.*

DCT-B was an exception; we ran out of dimensions to reduce before we found its turning point. For both pooling methods and interpolation, the magic number where the reduction begins to be detrimental seems to be $1/8$ $(12{,}5\%)$, whereas DCT-F has the best performance with $1/128$ $(0{,}78125\%)$. The results showcase how aggressive DR improves the performance of these methods significantly.

As discussed in Subsection 3.5.2, it is observable for these methods that the reduced degrees of freedom — caused by overextensive DR — limit the attacker's capability to generate AEs. However, this seems to take effect after around 2 000 – 3 000 queries at the earliest, whereas the current scientific literature aims for results with lower query counts. Therefore, a dynamic amount of DR that allows using more dimensions as the query count increases might achieve better results with higher query budgets.

On the other hand, PCA and ICA have peculiar short plateau periods during the early queries before they transition into smoother curves as well. Due to their different nature, they do not seem to benefit from DR, despite ICA performing noticeably better. Their performance also leaves a lot to be desired, which indicates that it is not as straightforward to find suitable directions for the perturbations in the subspaces PCA and ICA capture.

Additionally, a key step in `SurFree`'s perturbation generation is to multiply the coefficients of the lower-dimensional representation randomly. With PCA and ICA, the coefficients correspond to the individual components. Usually, PCA explains most of the variance within the first few components, and an 8×8 block unlikely consists of many meaningful independent signals for ICA to use. This would explain why neither of those methods benefits from DR. Even though PCA and ICA have a promising start, the perturbation

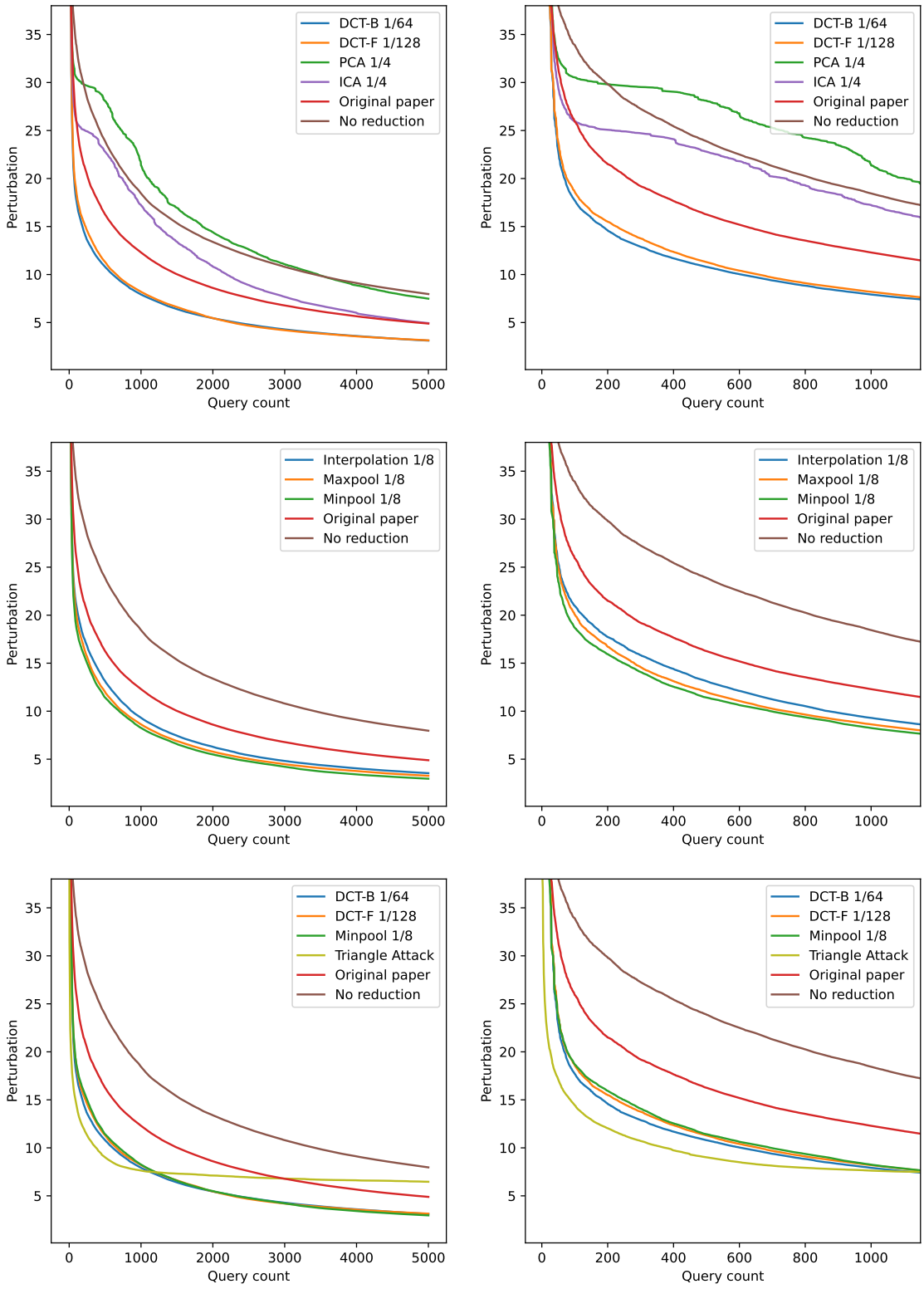

Figure 5.4. *The best performer of each method compared against the original paper* (DCT-B $1/2$) *and no reduction. They are split into the top two rows to avoid visual clutter in a single plot. On the right is a zoomed-in version of the plot on the left. The bottom two images are the best methods compared against* `Triangle Attack`.

decreases the fastest in the early queries with all the methods. We believe this is because the algorithm starts from a very noisy point, and any direction in the input space decreases the perturbation rapidly at the start, regardless of the method used.

For Figure 5.4, we chose the best performer of each method by visual inspection and compared their performance with each other. Because PCA and ICA performed similarly regardless of the magnitude of DR, we chose $1/4$ for their DR as it narrowly had the best results with PCA. On the right is a zoomed-in version of the plot on the left. To improve readability, we split the methods into two rows and selected our best ones for the final comparison on the bottom row. We added the original paper [29] (DCT-B $1/2$) and no reduction as baselines. In addition, we added a comparison between our best methods and the more recent `Triangle Attack` [49] on the bottom row.

Figure 5.4 highlights the significant advantage extensive DR offers to performance. In the top row, we observe an over 30% decrease with DCTs in the average size of the perturbation at 1 000 queries when compared to the original paper and about a 55% decrease when compared to no reduction at all. PCA and ICA perform poorly; PCA performs worse than no reduction until the very end, whereas ICA interestingly manages to catch up to the original paper's performance. In the middle row, interpolation, maxpooling, and minpooling achieve similar results. Minpooling exhibits the best performance by a narrow margin, achieving comparable results to both DCTs. Note that both pooling methods achieve slightly better results than interpolation.

Thus, we will compare DCTs and minpooling against `Triangle Attack` [49] in the bottom row of Figure 5.4. Its authors claimed superior performance over `SurFree` within 1 000 queries in their paper, and this claim holds even after our extensive DR. However, the gap in the average size of the perturbation between the two attack algorithms has become distinctly smaller. Furthermore, `Triangle Attack` peculiarly plateaus after these 1 000 queries, with `SurFree` surpassing its competitor after around 1 100 queries now instead of the original 3 000 queries. It is unclear whether this is an unavoidable limitation of `Triangle Attack`.

To summarize this section, DR holds a clear advantage in attack generation. While that is not a new discovery by any means, we demonstrate that the dimensions can be reduced significantly more than is customary before it starts to be detrimental. We highlight this by comparing our results to the results achieved with the original paper's DR. Minpooling performs better than maxpooling and interpolation with a narrow margin and achieves remarkably similar results with both DCTs, with DCT-B reaching the smallest average perturbation size ever so slightly within the first 1 000 queries. Furthermore, from the comparison against `Triangle Attack`, it is clear that while DR is a powerful augmentation for an attack algorithm, it does not replace an efficient attack algorithm by itself.

5.2 Attack success rate

In this section, we measure the performance of the different DR methods with ASR. Figures 5.5 - 5.7 depict the ASR on the three selected query budgets, $K \in \{200, 500, 1\,000\}$. We follow the `SurFree`'s original paper [29] by setting our highest target perturbation at 30 and lowest at 5, measured by l_2-norm. The figures present the ASR on large perturbations on the left and the ASR on small perturbations on the right. For conciseness, we selected the best performer of each method from Section 5.1.

We split the comparisons into different figures to avoid visual clutter in a single plot. Figure 5.5 displays the results of the first group of methods and Figure 5.6 of the second group. In Figure 5.7, we showcase the results of our best methods against `Triangle Attack`. All three figures include the same baselines: original paper (DCT-B $1/2$) and no reduction.

Figure 5.5 tells the same tale that was apparent in Section 5.1. Extensively reduced DCTs achieve the best ASR by a wide margin. Meanwhile, PCA struggles to compete with no reduction, and ICA's performance is between the original paper and no reduction in these low query scenarios. However, Figure 5.6 underlines the marginal advantage minpooling has over maxpooling and interpolation. As in Section 5.1, both pooling methods seem to achieve slightly better results than interpolation in general, which is an intriguing find.

Figure 5.7 reveals a slightly different story to the average size of the perturbation with its comparison of our best methods. Here, DCT-F and minpooling tend to perform better in the ASR with small perturbations. However, DCT-B does surpass them and achieves a somewhat better ASR with larger ones, where minpooling seems to lose its edge. `Triangle Attack` remains uncontested up until 1 000 queries.

We confirm these visual observations of Figure 5.7 by providing *Area Under the Curve* (AUC) values for the large perturbations ($l_2 <= 30$) in Table 5.1 and the small perturbations ($l_2 <= 5$) in Table 5.2. AUC — as its name suggests — computes the area under the curve, which means the higher the values, the better. `Triangle Attack` reigns supreme, except in small perturbations after 1 000 queries, where it is rivaled by DCT-F and minpooling. As we visually estimated, DCT-F and minpooling have the best performance with small perturbations, while DCT-B outperforms them with large ones. Minpooling succumbs to both DCTs in the long run ever so slightly. When compared to our baselines, these results further highlight how our DR methods alone have almost closed the gap between the original `SurFree` and `Triangle Attack`.

Considering DCT-B had the smallest average perturbation size by a narrow margin, these results suggest that while minpooling and DCT-F can generate more AEs with smaller perturbations, they might get "stuck" on some inputs. In other words, DCT-B finds directions to decrease the perturbation more often. On the other hand, minpooling and DCT-F are better at generating smaller perturbations, given that they do not get "lost" at the start.

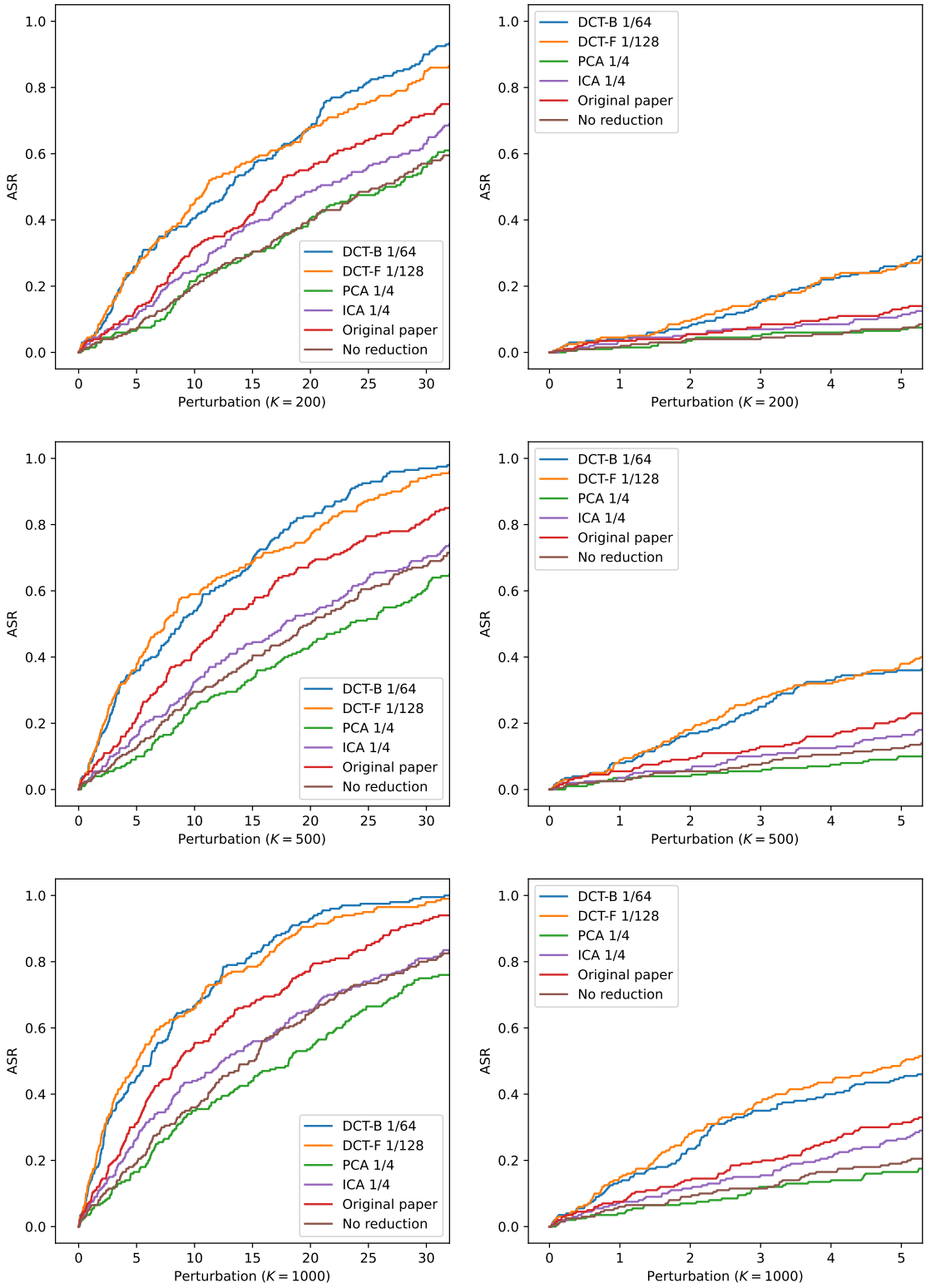

Figure 5.5. *ASR of the best performer of DCTs, PCA, and ICA with query budget* $K \in \{200, 500, 1000\}$ *queries compared against the original paper* (DCT-B $1/2$) *and no reduction. Plots on the right are zoomed-in versions.*

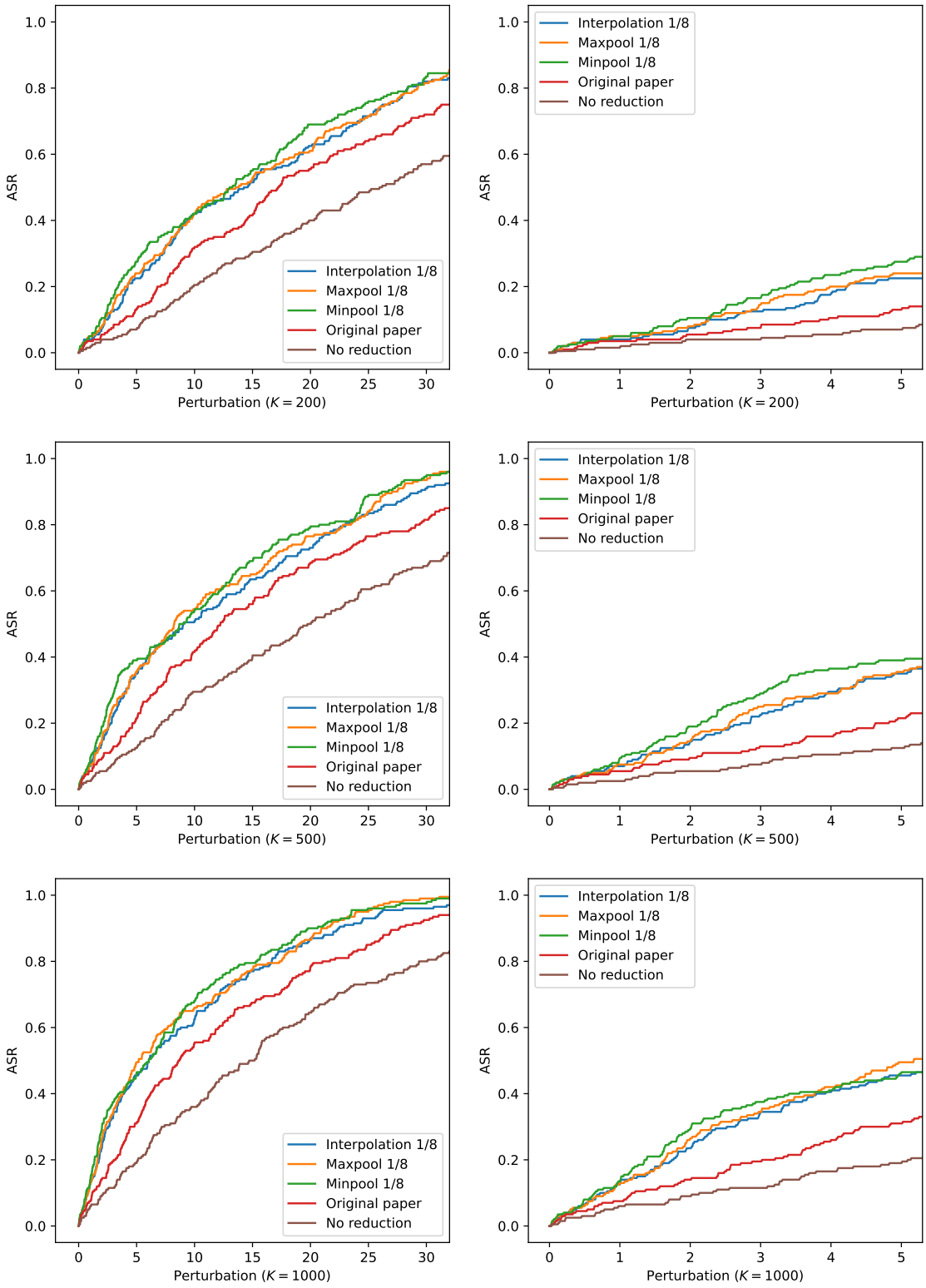

Figure 5.6. *ASR of the best performer of minpooling, maxpooling, and interpolation with query budget $K \in \{200, 500, 1000\}$ queries compared against the original paper* (DCT-B $1/2$) *and no reduction. Plots on the right are zoomed-in versions.*

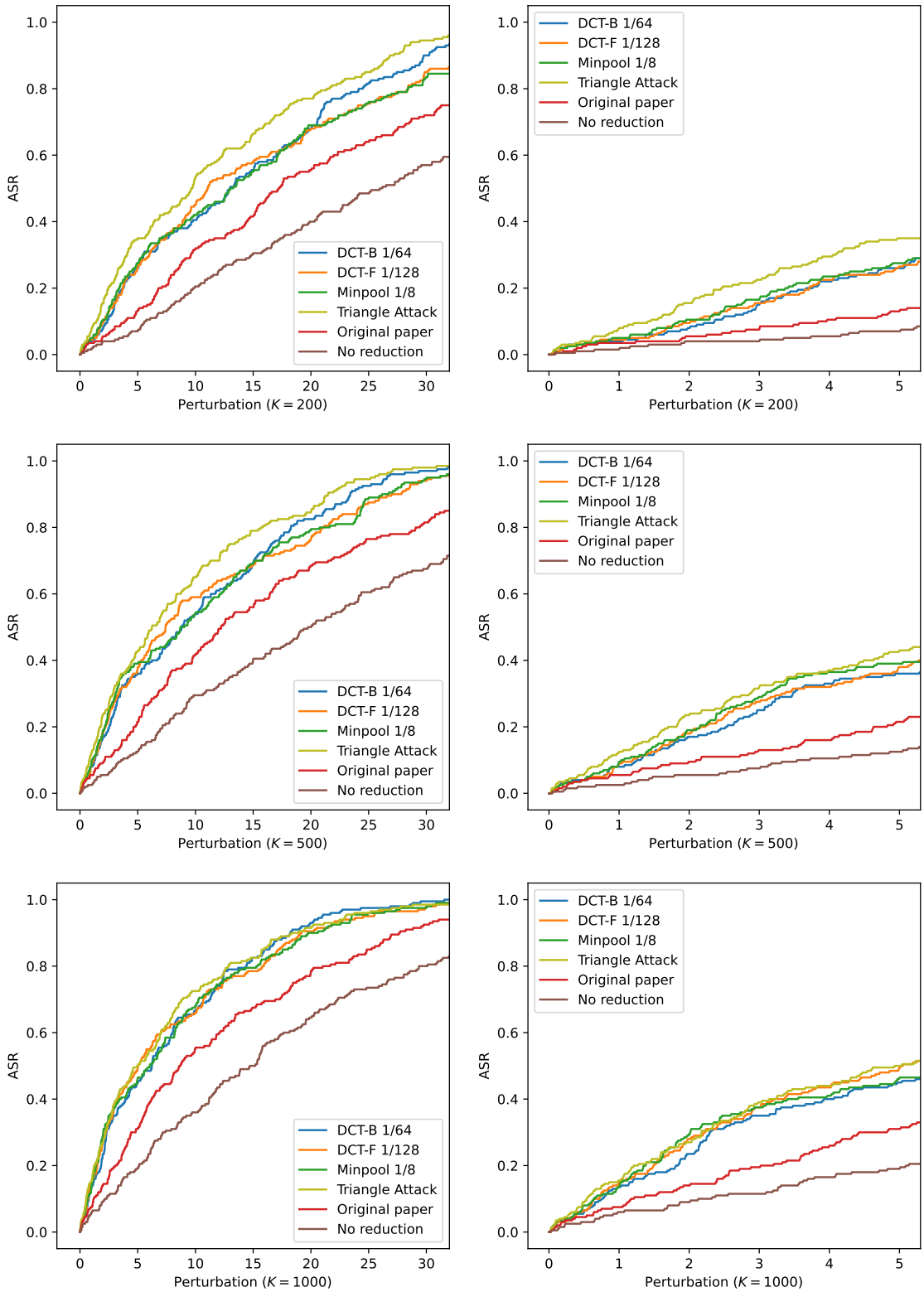

Figure 5.7. *ASR of our best performers compared against the original paper* (DCT-B $1/2$)*, no reduction and* `Triangle Attack` *with query budget* $K \in \{200, 500, 1000\}$ *queries. Plots on the right are zoomed-in versions.*

Method	$K = 200$	$K = 500$	$K = 1\ 000$
DCT-B $1/64$	*15,9*	*19,3*	*22,1*
DCT-F $1/128$	15,7	19,1	21,9
Minpool $1/8$	15,5	18,9	21,8
`Triangle Attack`	**18,3**	**21,1**	**22,5**
Original paper	12,2	15,3	18,3
No reduction	8,8	11,3	14,4

Table 5.1. *AUC of the large perturbations* $(l_2 <= 30)$ *on the left-hand side in Figure 5.7.*

Method	$K = 200$	$K = 500$	$K = 1\ 000$
DCT-B $1/64$	0,6	1,0	*1,4*
DCT-F $1/128$	0,6	*1,1*	**1,5**
Minpool $1/8$	*0,7*	*1,1*	**1,5**
`Triangle Attack`	**0,9**	**1,3**	**1,5**
Original paper	0,3	0,6	0,8
No reduction	0,2	0,3	0,5

Table 5.2. *AUC of the small perturbations* $(l_2 <= 5)$ *on the right-hand side in Figure 5.7.*

While `Triangle Attack` still prevails as the best performer, our methods are close contenders with significant improvement from the original paper's baseline of DCT-B $1/2$. As with the average perturbation size, our methods almost surpass `Triangle Attack` at $K = 1\ 000$ in ASR as well.

Evaluating the ASR further cements the notion of DR's usefulness in more efficient AE generation. As with the average size of the perturbation, minpooling and both DCTs achieve similar results, but they have slight differences this time. It opens up the question if it is possible or worth it to change DR methods on the fly to exploit the strengths of different ones.

5.3 Runtime

Lastly, we measured the execution time for each test run and then divided it by the number of queries to get an average time a single query takes. We present these runtimes in Table 5.3. We took the average of all of the test runs of the method in question, as the runtimes within methods did not vary much. Note that our objective here is not to measure runtimes with absolute precision but rather to give a rough estimation of the potential overhead of these methods. All the DR methods achieved a similar execution time of $5 - 8$ms per query, except for DCT-B, which took significantly longer with ~ 110ms per query. The minor increase in the runtimes of PCA and ICA is presumably explained by our slightly

Method	***Runtime per query (ms)***
DCT-B	111,0
DCT-F	5,9
Maxpool	5,8
Minpool	5,8
Interpolation	5,8
PCA	7,2
ICA	7,2
No reduction	5,5
`Triangle Attack`	6,6

Table 5.3. *Averaged runtimes of each method for a single query.*

inefficient implementation.

DCT-B was the choice for DR in the original paper [29], where no runtimes were reported. Hence, it is unclear whether the method has always been that slow or whether an update to the codebase has introduced a bug, making the runtime noticeably longer. To the best of our knowledge, the excessive runtime results from computing the DCT and its inverse block by block each iteration. Thus, the runtime would not be a bug; Li *et al.* [26] also believe this to be the cause for the long runtime of `SurFree` with its original parameters. If this is the case, there are few to no arguments supporting DCT-B judging by the runtimes alone, as all the other methods achieve comparable runtimes with each other.

In this chapter, we demonstrated our results by three different evaluation metrics. We measured the average size of the perturbation, the ASR, and the runtime of each method. We will summarize and discuss our results further in the following chapter.

6. DISCUSSION

In this chapter, we discuss our results and the previous chapters. The results are examined in Section 6.1. Threats to the validity of this thesis and avenues for future work are discussed in Sections 6.2 and 6.3, respectively. We conclude with final thoughts in Section 6.4.

We explored the current state and the challenges of decision-based attacks in Chapter 3.5, answering research question 1. In Section 2.2, we answered research question 2 by presenting a few different dimensionality methods. Finally, in Chapter 5, we demonstrated the differences between the presented methods and answered research question 3.

6.1 On the results

There exists some research on the effect DR has in the attack generation [24, 29, 49, 26], but many papers content themselves with only stating the used method and the magnitude of DR. Sometimes, they make a passing reference to internal experiments as a ratiocination. However, none of the existing papers examine the effects of DR as extensively as we did. Thus, we made a clear cut to the extant research by offering a more comprehensive and quantitative view of the effects of DR. Indeed, the most extensive DR we could find in the scientific literature was $1/16$ ($6{,}25\%$), whereas we went as far as $1/256$ ($0{,}390625\%$) before settling on $1/128$ ($0{,}78125\%$).

We also provided novel methods, such as minpooling, and experimented on using different ways to utilize already tested methods, such as PCA. `QEBA` [24] already experimented on PCA in their perturbation generation process, but due to the algorithm's nature, it first required 280 000 external images to generate the perturbations. Due to the differences between `QEBA` and `SurFree`, we managed to perform PCA on a single input without external data. While the performance of our PCA implementation was lackluster, we hope it either proves that PCA is not a suitable choice for DR in this use case or inspires future papers to test novel ways to use existing methods.

When it comes to the performance of our methods, DCT-F $1/128$, DCT-B $1/64$ ($1{,}5625\%$), and minpooling $1/8$ ($12{,}5\%$) achieved the best and near-identical performance with slight differences. DCT-B attained the smallest average perturbation size within the first 2 000

queries, whereas minpooling and DCT-F gained the highest ASR with small perturbations. We want to emphasize that the differences between our best methods with these two metrics were minor, and they achieved comparable performance to each other. The highest disparity between these methods was the runtime; DCT-B took approximately 19 times longer to run than the other two. It was left unclear whether the long runtime was due to the implementation or whether it was inherent to the block DCT approach used.

Another intriguing aspect we did not discuss in Chapter 5 is how efficiently the different methods utilize their coefficients in the perturbation generation. An essential step in `SurFree`'s perturbation generation is to multiply the coefficients of the lower-dimensional representation randomly. For example, with DCT-B, each block consists of 64 coefficients. Thus, a DR of $1/64$ allows `SurFree` to use only the lowest frequency coefficient (also known as the DC coefficient) to generate perturbations for each 8×8 block. DCT-F uses more of the higher-frequency coefficients with the same amount of DR.

The methods can be split into two groups by how many coefficients they use in total w.r.t. the amount of DR, as the number of coefficients used in the perturbation generation decreases asymmetrically. PCA, ICA, and both DCTs share the same amount of coefficients. With them, the number of coefficients increases rapidly with the number of dimensions. On the other hand, maxpooling, minpooling, and interpolation share the same amount of coefficients, and the number of coefficients increases moderately with the number of dimensions. They also seem to utilize the coefficients more efficiently throughout the number of dimensions available, as they achieve comparable performance with the first group even in higher dimensions despite having fewer coefficients.

It turns out that both DCT $1/64$ and minpooling $1/8$ use the same amount of coefficients in total to generate perturbations, while DCT-F $1/128$ uses half of their coefficients. Curiously, DCT-F $1/256$ and minpooling $1/16$ use the same amount of coefficients; they also exhibited worse performance for the first time at those points. This would suggest that the sweet spot for DR is near-identical between our methods when measured by the total number of coefficients, as even maxpooling and interpolation displayed degraded performance at $1/16$. This is a fascinating finding, hopefully shedding some light on how much information is actually needed in the generation process.

In the end, the results confirm that the DR offers a massive advantage. The dimensions can also be reduced surprisingly much before it starts to hinder the process, which is an intriguing discovery. Sometimes, these tests needed a full-sized test run to observe this effect, as small-scale tests might not be sufficient due to the intrinsic randomness within these attack algorithms. However, the results also highlight the importance of an efficient attack algorithm itself. Despite the considerable boost in performance when compared to the original `SurFree`, our implementation does not outperform `Triangle Attack` [49] within 1 000 queries, even with extensive DR.

6.2 Threats to validity

The first threat to the validity of this thesis is whether the results are replicable with other attack algorithms or even SurFree itself with the new parameters. In other words, the generalizability of our results is uncertain. Promisingly, many algorithms benefit from the DR already, even though none of the existing papers have tested as aggressive DR as we did in this thesis. Secondly, we only tested against one target model, which opens the question of whether these results generalize to other target models. However, we believe that the property of transferability [35] makes our results relatively model-agnostic.

In addition, there was a moderate difference between the average size of the perturbation between the first 100 images and the last 100 images of our dataset. Thus, it is unclear what is the dataset's full effect on the results and whether our test set of 200 correctly classified images is a representative set for performing the tests. Also, the attack algorithms and SurFree specifically include inherent randomness in them. It is uncertain how much, e.g., the average perturbation size varies between test runs. For what it is worth, we ran multiple tests a second time, obtaining near-identical results with the first runs. Alas, due to the computational cost and time a single test run takes, we could not verify our results by running all the tests multiple times to take the average between each run for each method. For the same reason, we did not confirm our results with the newest version of SurFree available.

As noted at the beginning of Chapter 5, we did not provide the same starting point for the different methods or algorithms. However, we believe it does not meaningfully affect the results, as these differences in initialization get averaged over the 200 test images. There is a risk that a singular starting point could distort the results instead. The test would only measure how effectively the algorithm finds AEs from that single point instead of general performance. Therefore, should this approach be used, the starting point should be randomized for each image exactly once and then used in all tests.

We also restricted ourselves solely to the decision-based attack scenario, leaving out other categories. This means that although the results might generalize within this category, it is uncertain whether they generalize between them. The DR methods used are also simplistic and leave room for more creativity.

The final threat to the validity is runtimes. The implementations are most likely not as efficient as they could be, and thus it is unsure how comparable the runtimes truly are. It is also uncertain how much the runtimes even matter. Especially the high runtime of block DCT raises questions. However, the objective with runtimes was to give a general idea of how long it takes to run each method, and we believe the runtimes we presented serve their purpose.

6.3 Future work

There are plenty of open questions left for future work. Firstly, we only examined untargeted attacks. Even though this is a common scenario in the DBA literature and it was the only readily available option in `SurFree`, it is unclear what is the effect of DR in targeted attacks. Secondly, due to the limited scope of this thesis, we did not exhaustively research the suitability of l_p-norms for the similarity measurement. Even though they are ubiquitous in the literature, their role as the standard has been questioned [50, 45].

As the results demonstrate, DCT-B and DCT-F achieved comparable performance bar the runtime. However, we ran out of dimensions to reduce with block DCT before we could observe a deterioration in the performance. Nevertheless, DCT can be used with larger blocks than 8×8. Thus, the dimensions could be reduced even more, which could help find the turning point for DCT-B as well. We suspect the best performance is encountered when using the same amount of coefficients as with DCT-F $1/128$.

Hybrid approaches might also be considered in the future. As observed in the comparison against `Triangle Attack` [49], some attack algorithms might perform better in the early stages than others but lose their lead rather quickly. Therefore, perhaps one could combine two attack algorithms together by starting with the rapidly decreasing one and switching to another later if the first one plateaus. This could apply to different DR methods as well. Indeed, we observed minpooling and DCT-F achieving slightly better ASRs with small perturbations, but supposedly got lost at the start of the process sometimes. Thus, DCT-B could be used as a kick starter for the generation process, ensuring that the generation process does not get "stuck" in the beginning. However, it is uncertain how much there actually is to gain with these hybrid approaches. We leave this question open for future research.

In addition, the inputs were also not equally challenging for each method; some inputs were more straightforward for some DR methods and more difficult for others. This is presumably explained by the somewhat different subspaces our methods capture. We also noticed by testing a single image multiple times that, sometimes, different DR methods tended to gravitate towards slightly different adversarial classes. Ergo, it is unclear whether the second method would efficiently find new directions for the perturbations with hybrid approaches, even with the first method helping it to get started.

6.4 Conclusion

In conclusion, we have explored the current scientific literature and background on AEs and demonstrated the efficiency of DR in generating decision-based AEs. The efficiency was demonstrated by implementing different DR methods with varying magnitudes on an existing algorithm called `SurFree` [29]. We confirm with our results the already prevalent

notion that DR is crucial in achieving noticeably more efficient AE generation. However, our results indicate that the dimensions can be reduced more extensively than is customary in the scientific literature for a significant gain in performance. We showcased block DCT, full-frame DCT, and minpooling to be the best methods for DR. Based on our results, we call for more aggressive DR, especially in the DBA algorithms. Furthermore, we question whether there is room for improvement by using more creative methods for DR.

We limited the thesis scope to the DBA category with a few different DR methods and l_2-norm to measure similarity. The scope thus leaves an opening for future work to explore the effects of DR on various adversarial scenarios with different methods and alternative similarity measurements.

REFERENCES

[1] Akhtar, Naveed et al. “Advances in Adversarial Attacks and Defenses in Computer Vision: A Survey”. In: *IEEE Access* 9 (2021), pp. 155161–155196.

[2] Balda, Emilio Rafael, Behboodi, Arash, and Mathar, Rudolf. “Adversarial Examples in Deep Neural Networks: An Overview”. In: *Deep Learning: Algorithms and Applications*. Studies in Computational Intelligence. Cham, Switzerland: Springer International Publishing, 2019, pp. 31–65.

[3] Bhambri, Siddhant et al. *A Survey of Black-Box Adversarial Attacks on Computer Vision Models*. 2019. URL: https://arxiv.org/abs/1912.01667.

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

[5] Carlini, Nicholas. *A Complete List of All (arXiv) Adversarial Example Papers*. 2019. URL: https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html (visited on October 8, 2022).

[6] Carlini, Nicholas and Wagner, David. “Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods”. In: *AISec 2017 - Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, co-located with CCS 2017*. AISec ’17. ACM, 2017, pp. 3–14.

[7] Chen, Jianbo, Jordan, Michael I, and Wainwright, Martin J. “HopSkipJumpAttack: A Query-Efficient Decision-Based Attack”. In: *2020 IEEE Symposium on Security and Privacy (SP 2020)*. IEEE Symposium on Security and Privacy. IEEE. Los Alamitos, CA, USA: IEEE, 2020, pp. 1277–1294.

[8] Chen, Jinghui and Gu, Quanquan. “RayS: A Ray Searching Method for Hard-Label Adversarial Attack”. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD ’20. Virtual Event, CA, USA: Association for Computing Machinery, 2020, pp. 1739–1747.

[9] Cheng, Minhao et al. “Query-Efficient Hard-Label Black-Box Attack: An Optimization-Based Approach”. In: *7th International Conference on Learning Representations, ICLR 2019*. 2019.

[10] Cheng, Minhao et al. *Sign-OPT: A Query-Efficient Hard-Label Adversarial Attack*. 2019. URL: https://arxiv.org/abs/1909.10773.

[11] Cireşan, Dan, Meier, Ueli, and Schmidhuber, Juergen. “Multi-Column Deep Neural Networks for Image Classification”. In: (2012). URL: https://arxiv.org/abs/1202.2745.

[12] Dalvi, Nilesh et al. “Adversarial Classification”. In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’04. Seattle, WA, USA: Association for Computing Machinery, 2004, pp. 99–108.

[13] Deniz, Oscar et al. “Robustness to Adversarial Examples Can Be Improved With Overfitting”. In: *International journal of machine learning and cybernetics* 11.4 (2020), pp. 935–944.

[14] Eykholt, Kevin et al. “Robust Physical-World Attacks on Deep Learning Visual Classification”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[15] Fawzi, Alhussein, Moosavi-Dezfooli, Seyed-Mohsen, and Frossard, Pascal. “Robustness of Classifiers: From Adversarial to Random Noise”. In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc., 2016.

[16] Fawzi, Alhussein, Moosavi-Dezfooli, Seyed-Mohsen, and Frossard, Pascal. “The Robustness of Deep Networks: A Geometrical Perspective”. In: *IEEE Signal Processing Magazine* 34.6 (2017), pp. 50–62.

[17] Gilmer, Justin et al. “Adversarial Spheres”. In: *6th International Conference on Learning Representations, ICLR 2018 - Workshop Track Proceedings*. 2018.

[18] Gilmer, Justin et al. *Motivating the Rules of the Game for Adversarial Example Research*. 2018. URL: https://arxiv.org/abs/1807.06732.

[19] Goodfellow, Ian J., Shlens, Jonathon, and Szegedy, Christian. *Explaining and Harnessing Adversarial Examples*. 2014. URL: https://arxiv.org/abs/1412.6572.

[20] Gu, Shixiang and Rigazio, Luca. “Towards Deep Neural Network Architectures Robust to Adversarial Examples”. In: *3rd International Conference on Learning Representations, ICLR 2015 - Workshop Track Proceedings*. 2015.

[21] Guo, Chuan, Frank, Jared S., and Weinberger, Kilian Q. *Low Frequency Adversarial Perturbation*. 2018. URL: https://arxiv.org/abs/1809.08758.

[22] Ilyas, Andrew et al. “Adversarial Examples Are Not Bugs, They Are Features”. In: *Advances in Neural Information Processing Systems 32 (NIPS 2019)*. Vol. 32. Advances in Neural Information Processing Systems. La Jolla, CA, USA: Neural Information Processing Systems (Nips), 2019.

[23] Jetley, Saumya, Lord, Nicholas A., and Torr, Philip H. S. “With Friends Like These, Who Needs Adversaries?” In: *Advances in Neural Information Processing Systems 31 (NIPS 2018)*. Vol. 31. Advances in Neural Information Processing Systems. La Jolla, CA, USA: Neural Information Processing Systems (Nips), 2018.

[24] Li, Huichen et al. “QEBA: Query-Efficient Boundary-Based Blackbox Attack”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Conference on Computer Vision and Pattern Recognition. IEEE. New York, NY, USA: IEEE, 2020, pp. 1218–1227.

[25] Li, Jie et al. “Aha! Adaptive History-Driven Attack for Decision-Based Black-Box Models”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 16148–16157.

[26] Li, Xiu-Chuan et al. “Decision-Based Adversarial Attack With Frequency Mixup”. In: *IEEE Transactions on Information Forensics and Security* 17 (2022), pp. 1038–1052.

[27] Luo, Yan et al. *Foveation-Based Mechanisms Alleviate Adversarial Examples*. 2015. URL: https://arxiv.org/abs/1511.06292.

[28] Mahmood, Kaleel et al. “Back in Black: A Comparative Evaluation of Recent State-Of-The-Art Black-Box Attacks”. In: *IEEE Access* 10 (2022), pp. 998–1019.

[29] Maho, Thibault, Furon, Teddy, and Merrer, Erwan Le. *SurFree: A Fast Surrogate-Free Black-Box Attack*. 2020. URL: https://arxiv.org/abs/2011.12807.

[30] McCulloch, Warren S. and Pitts, Walter. “A Logical Calculus of the Ideas Immanent in Nervous Activity”. In: *The Bulletin of Mathematical Biophysics* 5.4 (1943), pp. 115–133.

[31] Meta AI. *ImageNet Benchmark (Image Classification) | Papers With Code*. 2022. URL: https://paperswithcode.com/sota/image-classification-on-imagenet (visited on October 28, 2022).

[32] Moosavi-Dezfooli, Seyed-Mohsen et al. *Robustness of Classifiers to Universal Perturbations: A Geometric Perspective*. 2017. URL: https://arxiv.org/abs/1705.09554.

[33] Nakkiran, Preetum. “A Discussion of ’Adversarial Examples Are Not Bugs, They Are Features’: Adversarial Examples are Just Bugs, Too”. In: *Distill* (2019). URL: https://distill.pub/2019/advex-bugs-discussion/response-5.

[34] Nesterov, Yu. “Efficiency of Coordinate Descent Methods on Huge-Scale Optimization Problems”. In: *SIAM Journal on Optimization* 22.2 (2012), pp. 341–362.

[35] Papernot, Nicolas, McDaniel, Patrick, and Goodfellow, Ian. *Transferability in Machine Learning: from Phenomena to Black-Box Attacks Using Adversarial Samples*. 2016. URL: https://arxiv.org/abs/1605.07277.

[36] Paszke, Adam et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. URL: https://arxiv.org/abs/1912.01703.

[37] Pedraza, Anibal, Deniz, Oscar, and Bueno, Gloria. “On the Relationship Between Generalization and Robustness to Adversarial Examples”. In: *Symmetry (Basel)* 13.5 (2021), pp. 817–.

[38] Rahmati, Ali et al. “GeoDA: A Geometric Framework for Black-Box Adversarial Attacks”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2020, pp. 8443–8452.

[39] Rozsa, Andras, Gunther, Manuel, and Boult, Terrance E. *Towards Robust Deep Neural Networks with BANG*. 2016. URL: https://arxiv.org/abs/1612.00138.

[40] Russakovsky, Olga et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252.

[41] Russell, Stuart and Norvig, Peter. *Artificial Intelligence: A Modern Approach*. eng. 3rd edition. Prentice Hall series in artificial intelligence. Boston, MA, USA: Pearson, 2016.

[42] Sabour, Sara et al. “Adversarial Manipulation of Deep Representations”. In: *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*. 2016.

[43] Schmidt, Ludwig et al. “Adversarially Robust Generalization Requires More Data”. In: *Advances in Neural Information Processing Systems 31 (NIPS 2018)*. Vol. 31. Advances in Neural Information Processing Systems. La Jolla, CA, USA: Neural Information Processing Systems (Nips), 2018.

[44] Serban, Alexandru Constantin, Poll, Erik, and Visser, Joost. *Adversarial Examples - A Complete Characterisation of the Phenomenon*. 2018. URL: https://arxiv.org/abs/1810.01185.

[45] Sharif, Mahmood, Bauer, Lujo, and Reiter, Michael K. “On the Suitability of Lp-Norms for Creating and Preventing Adversarial Examples”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2018, pp. 1686–16868.

[46] Szegedy, Christian et al. *Intriguing Properties of Neural Networks*. 2013. URL: https://arxiv.org/abs/1312.6199.

[47] Tanay, Thomas and Griffin, Lewis. *A Boundary Tilting Persepective on the Phenomenon of Adversarial Examples*. 2016. URL: https://arxiv.org/abs/1608.07690.

[48] Tsipras, Dimitris et al. “Robustness May Be at Odds with Accuracy”. In: *7th International Conference on Learning Representations, ICLR 2019*. 2019.

[49] Wang, Xiaosen et al. *Triangle Attack: A Query-Efficient Decision-Based Adversarial Attack*. 2021. URL: https://arxiv.org/abs/2112.06569.

[50] Wang, Zhou and Bovik, Alan C. “Mean Squared Error: Love It or Leave It? A New Look at Signal Fidelity Measures”. In: *IEEE Signal Processing Magazine* 26.1 (2009), pp. 98–117.

[51] Zhang, Zhuosheng and Yu, Shucheng. *BO-DBA: Query-Efficient Decision-Based Adversarial Attacks via Bayesian Optimization*. 2021. URL: https://arxiv.org/abs/2106.02732.