UNIVERSAL ADVERSARIAL PERTURBATIONS USING ALTERNATING LOSS FUNCTIONS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS OF
THE MIDDLE EAST TECHNICAL UNIVERSITY
BY

DENIZ ŞEN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE
OF
MASTER OF SCIENCE
IN
MULTIMEDIA INFORMATICS, DEPARTMENT OF MODELLING AND SIMULATION

AUGUST 2022

Universal Adversarial Perturbations Using Alternating Loss Functions

submitted by **DENIZ ŞEN** in partial fulfillment of the requirements for the degree of **Master of Science in Multimedia Informatics, Modelling and Simulation Department, Middle East Technical University** by,

Prof. Dr. Deniz Zeyrek Bozşahin Dean, **Graduate School of Informatics**	______________
Assoc. Prof. Dr. Elif Sürer Head of Department, **Modelling and Simulation**	______________
Prof. Dr. Alptekin Temizel Supervisor, **Modelling and Simulation**	______________

Examining Committee Members:

Assoc. Prof. Dr. Elif Sürer Modelling and Simulation Department, METU	______________
Prof. Dr. Alptekin Temizel Modelling and Simulation Department, METU	______________
Assist. Prof. Dr. Ayşegül Dündar Computer Science Department, Bilkent University	______________

Date: 23.08.2022

ABSTRACT

UNIVERSAL ADVERSARIAL PERTURBATIONS USING ALTERNATING LOSS FUNCTIONS

Şen, Deniz
M.S., Multimedia Informatics, Department of Modelling and Simulation
Supervisor: Prof. Dr. Alptekin Temizel

August 2022, 49 pages

Deep learning models have been the main choice for image classification, however, recently it has been shown that even the most successful models are vulnerable to adversarial attacks. Unlike image-dependent attacks, universal adversarial perturbations can generate an adversarial example when added to any image. These perturbations are usually generated to fool the whole dataset and most successful attacks can reach 100% fooling rate, however they cannot be controlled to stabilize around a desired fooling rate. This thesis proposes 3 algorithms (Batch Alternating Loss, Epoch-Batch Alternating Loss, Progressive Alternating Loss) that utilize alternating loss scheme where the loss function is selected at each iteration to be either adversarial or norm loss based on some condition. Progressive Alternating Loss has been the best performing attack in terms of the fooling rate stabilization and L_p norm. Furthermore, training-time spatial filtering was applied to each of these proposed attacks to reduce the artefact-like perturbations which naturally form around the center, which was shown to be successful for L_2 attacks.

Keywords: adversarial attack, universal adversarial perturbations, alternating loss

ÖZ

DÖNÜŞÜMLÜ KAYIP FONKSİYONLARININ KULLANIMI İLE EVRENSEL ÇEKİŞMELİ BOZULMALAR

Şen, Deniz
Yüksek Lisans, Çokluortam Bilişimi, Modelleme ve Simülasyon Bölümü
Tez Yöneticisi: Prof. Dr. Alptekin Temizel

Ağustos 2022, 49 sayfa

Derin öğrenme modelleri imge sınıflandırma için ana seçim olmuşlardır, ancak son zamanlarda en başarılı modellerin bile çekişmeli saldırılara karşı savunmasız olduğu gösterilmiştir. İmgeye özel saldırıların aksine, evrensel çekişmeli bozulmalar herhangi bir imgeye eklendiği zaman bir çekişmeli örnek üretebilir. Bu bozulmalar genelde bütün bir veri setini yanıltacak şekilde üretilir ve çoğu başarılı saldırı 100% yanıltma oranına kontrolsüzce ulaşabilir, fakat istenilen bir yanıltma oranında stabilize edilemez. Bu tez dönüşümlü kayıp fonksiyonu yöntemini kullanan 3 algoritma (Yığına Bağlı Dönüşümlü Kayıp, Dönem-Yığına Bağlı Dönüşümlü Kayıp, İlerleyen Dönüşümlü Kayıp) önermektedir; dönüşümlü kayıp fonksiyonları her yinelemede belirli bir koşula göre çekişmeli ya da norm kayıp fonksiyonlarından biri olarak seçilmektedir. İlerleyen Dönüşümlü Kayıp yönteminin yanıltma oranları ve L_p normlarına göre en başarılı saldırı olduğu görülmüştür. Ayrıca, önerilen her saldırıya uzamsal filtreleme uygulanmış, böylece doğal şekilde imgenin orta kısmında oluşan yapay görünümlü bozulmalar azaltılmıştır. Bu uygulamanın L_2 saldırılarında başarılı olduğu görülmüştür.

Anahtar Kelimeler: çekişmeli saldırı, evrensel çekişmeli bozulmalar, dönüşümlü kayıp

This thesis is dedicated to everyone I love and that has been staying by me throughout the procedure.

ACKNOWLEDGMENTS

I want to give the biggest credit to my advisor, Prof. Dr. Alptekin Temizel for guiding me through the toughest times and sharing his wisdom with me. His motivation has been the key for me to complete and produce an academically valuable work.

I also want to thank my jury members for their valuable recommendations and comments.

I honestly think that METU Deep Learning and Computer Vision Lab meetings have had a remarkable impact on the quality of the work I have produced. For that, I want to thank Dr. Cihan Öngün and Oğuz Hanoğlu for their flawless organization of the meetings, and I also want to thank each member of the lab for their contributions to the meetings whether with their presentations, experiences, showcases, or comments.

My family has been my biggest supporter, and for that, I cannot express how grateful I am. I thank my father Erdem Şen, my mother Bedia Senem Şen and my sister Irmak Şen for all the help they provided for me to complete the biggest achievement of my life. I also thank my friend Alperen Inci for staying by me through the whole time, I honestly think that I would not complete my thesis without his companionship.

Finally, I want to express my honest gratitude to my TÜBITAK 1001 project team, Prof. Dr. Alptekin Temizel, Oğuz Hanoğlu, Berat Tuna Karlı, and Ayberk Aydın. Working with an excellent team whose main focus was on adversarial learning helped me learn and understand the field, and ultimately helped me complete my thesis work.

This work has been funded by The Scientific and Technological Research Council of Turkey, ARDEB 1001 Research Projects Programme project no: 120E093

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

UAP	Universal Adversarial Perturbation
AL	Alternating Loss
B-AL	Batch Alternating Loss
EB-AL	Epoch-Batch Alternating Loss
P-AL	Progressive Alternating Loss
FB-AL	Filtered Batch Alternating Loss
FEB-AL	Filtered Epoch-Batch Alternating Loss
FP-AL	Filtered Progressive Alternating Loss
GPU	Graphics Processing Unit
AI	Artificial Intelligence
CNN	Convolutional Neural Networks

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

Deep learning models have become the norm regarding tasks such as image classification. These models learn the patterns on high volumes of data in the forms of non-linear mathematical functions by estimating the parameter set that best represents the training data distribution. In the specific case of image classification, the models learn the decision boundaries that best separate different classes of samples from each other in the formulated feature space; this means that if a sample is very close to a decision boundary, to the perception of the neural networks, the class of the sample can be highly ambiguous. It was shown that sampling from these "ambiguous" areas of the feature space does not directly translate to human perception; therefore, such an image may confuse a model while not affecting the human visual perception. Adversarial learning aims to capitalize on this phenomenon by developing algorithms that craft special perturbations which, when added to a benign image, do not affect the human perception yet force deep learning models to misclassify even the simplest images. These additions are called *adversarial perturbations*, and their combination with benign images are called *adversarial examples*. This subject poses a crucial problem for deploying AI models in real life. For instance, if an autonomous driving AI model is embedded into a vehicle without considering the existence of adversarial examples, an attacker might perturb the real-life traffic signs to mislead the vehicles, which could potentially cause catastrophic accidents to occur [1]. Despite the existence of several adversarial defense mechanisms that were proven to be effective, creating new attacks that can overcome these defense methods is equally crucial for the further improvement of these defenses, which will lead to more secure deployment of AI models in the real life [2, 3, 4, 5, 6, 7].

Conventional adversarial example generation methods are image-dependent; therefore, to generate an adversarial dataset, each image must go through the same algorithm, which is a costly operation, especially if the algorithm involves an iterative optimization process (most successful adversarial attacks are iterative). However, it has been shown that image-independent perturbations also exist. These universal adversarial perturbations (UAP), when added to *any* benign image, make it adversarial [8].

Despite being adversarial perturbations, UAPs have distinct properties compared to image-dependent adversarial perturbations; for instance, while image-dependent adversarial perturbations visually appear like noise, UAPs contain image-like features about the target class. These features are strong enough to dominate the actual image features relative to the deep network's features space; as a result, the network effectively perceives the UAP as the actual image and the benign image as noise [9]. This fundamental difference brings out a new problem that is not valid for image-dependent attacks; controlling the fooling rate over a dataset. To fool a predefined fraction of a dataset with standard adversarial attacks, only a part of the dataset could be attacked. On the other hand, since -strong- UAPs are formed by image-like features, randomly applying them to any image is likely to produce the same outcome as any other image; thus, reaching a 100% fooling rate is quick and uncontrollable.

1.2 Proposed Methods and Models

This thesis adapts the concept of *Alternating Loss* (AL) into the UAP training procedure by proposing 3 different algorithms:

- Batch Alternating Loss Training(B-AL)
- Epoch-Batch Alternating Loss Training(EB-AL)
- Progressive Alternating Loss Training(P-AL)

B-AL training uses the current fooling rate to approximate the performance of the UAP over the whole dataset. On the other side, EB-AL strictly backpropagates the adversarial loss function until the desired fooling rate is achieved over an epoch of training, regardless of the individual performance over a batch. P-AL takes advantage of the fooling rate performance of the current epoch until a point of optimization; this way, the fooling rate can be stabilized around the desired level.

Following the algorithms, it is experimentally and visually shown that minimization-based UAPs tend to generate image-like features around the corners while having visually incoherent and adversarially weak noise around the center of the image. Specific filtering operations are integrated into each proposed training scheme to make use of this phenomenon to reduce the overall noise. These filtering operations effectively eliminate unwanted noise.

1.3 Contributions and Novelties

The main findings of this thesis were originally published in a paper in AAAI-22 Workshop on Adversarial Machine Learning and Beyond [10]. This thesis aims to elaborate further on the subject. The contributions of this work are as the following:

- Parametrization of the fooling rate for UAP training
- Integration of AL scheme into UAP training
- Proposition of application of spatial filters to enhance the quantitative and visual performance of the UAPs

1.4 The Outline of the Thesis

Chapter 2 starts by presenting a taxonomy of the types of adversarial attacks and proceeds with brief introductions of the state-of-the-art image-dependent and universal adversarial attacks. Chapter 3 presents the detailed descriptions of the proposed algorithms. Then the filtering operation is explained, along with the common filters' types and descriptions. Next, Chapter 4 begins with the details of the experimental setup, followed by the results of unfiltered and filtered attacks. Also, a study about the occurrence of the perturbations around the edges of the UAPs is given in this section. Chapter 5 compiles the results from the previous section and their discussions. Finally, Chapter 6 gives an overview of the thesis and potential future work.

CHAPTER 2

LITERATURE SURVEY

In this section, widely used adversarial attacks are presented in a brief and informative manner. The literature review starts by giving a taxonomy of the adversarial attack algorithms, based on several properties. The proposed algorithms are also fit into the given taxonomy. Then, the explanations of the most popular image-depend and universal adversarial attacks are presented briefly.

2.1 Types of Adversarial Attacks

There are several ways that adversarial attacks can be categorized, such as targeted and non-targeted attacks, where the objective of the adversarial attack is respectively either forcing the model to predict the *target* class or to simply make a wrong prediction for any input sample.

An alternative way to categorize the adversarial attacks is according to their access to the target model. White-box attacks assume complete knowledge of the model parameters such as the architecture, weights, gradients, and training procedure; on the other hand, black-box attacks are performed without such knowledge. Thanks to the high transferability of the adversarial examples, most of the time, the black-box attacks are performed by attacking off-the-shelf models in a white-box setting and feeding the obtained adversarial examples to an unknown target model to achieve misclassification. In some cases, the attacker may only know the architecture of the target model; these attacks are classified as gray-box; however, these attacks are also mostly applied in the same way as black-box attacks, disregarding some exceptions [11].

Furthermore, adversarial attacks can be single-step or iterative, where the algorithm makes either a single or multiple updates over the learned perturbation vector. Despite being more computationally efficient, single-step attacks are usually less successful than iterative attacks.

Adversarial attacks can also either be norm-bounded or minimization based. Norm bounded attacks usually limit the amount of change between the original sample and the adversarial example (the limit is usually defined as ϵ, and the norm function is mostly taken as L_p). In contrast, minimization attacks try to minimize the norm of the perturbation vector as much as possible while still fooling the network.

Finally, the adversarial perturbations may either be image-dependent (i.e., only effective for a specific image) or universal (i.e., can be applied to any image from a target dataset). Image-dependent attacks are usually less perceivable by the human eye while being very costly as they require each image to be attacked individually. Universal attacks, on the other hand, are more perceivable by humans while only needing to be trained once over a training set; then, the learned perturbation can be added to any target image to generate an adversarial example. This quality also makes UAPs more usable in the real world, while the image-dependent attacks are more suitable for offline cases, such as CAPTCHA [12].

The taxonomy of the adversarial attacks proposed in this thesis or presented in the next section is given in Table 1. As all the given attacks are white-box in terms of access to the target model, such a column was not included in the table; besides, each of these attacks can be turned into black-box by transferring the adversarial examples that were generated in white-box settings.

2.2 Image-dependent Adversarial Attacks

In the remainder of this thesis the following variables are used, descriptions of which are as the following:

- x: Benign image
- y: Class label of x
- v: Adversarial perturbation
- f: Target classifier
- t: Target class
- ϵ: Maximum amount of perturbation

Table 1: Taxonomy of the presented adversarial attacks

Attack Type	Image Dependency		Algorithm		Optimization		Objective	
	Dependant	Universal	Single Step	Iterative	Bounded	Minimization	Targeted	Non-targeted
L-BFGS [13]	•			•		•	•	•
FGSM [3]	•		•		•		•	•
BIM [2]	•			•	•		•	•
PGD [1]	•			•	•		•	•
DeepFool [14]	•			•		•	•	•
C&W [15]	•			•		•	•	•
PerC-C&W [16]	•			•		•	•	•
StAdv [17]	•			•		•	•	•
UAP [8]		•		•	•		•	•
NAG [18]		•		•	•			•
F-UAP [9]		•		•	•		•	•
HP-UAP [19]		•		•	•		•	•
FFF [20]		•		•	•			•
DTA [21]		•		•	•		•	•
(F)B-AL		•		•		•	•	
(F)EB-AL		•		•		•	•	
(F)P-AL		•		•		•	•	

2.2.1 L-BFGS

The concept of adversarial examples was proposed along with the first adversarial attack on a neural image classifier; the algorithm used a box-constrained optimization method as given in Equation 1 [13]. This optimization tries to optimize a min-max problem, where the first term signifies the L_p norm of the learned perturbation, and the second term is the adversarial loss. The first term inherently increases the adversarial loss (since reducing the norm weakens the misclassification power while making the perturbation less perceivable to humans), while the second term tries to minimize it. Hypothetically, when the optimization converges, an imperceptible and adversarially successful example should be obtained. The algorithm was named by the literature after the optimization algorithm that the attack used. This attack is image-dependent, white-box, iterative, and minimization; furthermore can be both targeted and untargeted depending on the formulation of the adversarial loss function (Equation 1 shows a targeted setting).

$$\text{Minimize } \; c|v| + \text{loss}_f(x+v,t) \; \text{ subject to } \; x+v \in [0,1]^m \tag{1}$$

2.2.2 Fast Gradient Sign Method

Fast Gradient Sign Method (FGSM) is a single-step adversarial attack. Despite its weak fooling rate performance due to its simplicity, it is still being widely used as a benchmark for new adversarial attacks [3]. The formal definition of the attack is given in Equation 2.

$$x_{adv} = x + \epsilon \text{sign}(\bigtriangledown_x J(x,y)) \tag{2}$$

FGSM resembles a step of gradient descent, except the step is taken in the same direction as the gradient to **maximize** the loss and successfully generates an adversarial example. This attack is image-dependent, norm bounded (because of ϵ scale), white-box, single-step, and targeted (although it is possible to make the attack targeted by taking a standard gradient descent step towards the target class).

2.2.3 Basic Iterative Method and Projected Gradient Descent

Basic iterative method (BIM) and projected gradient descent (PGD) algorithms convert FGSM into an iterative attack, achieving the same amount of perturbation ϵ, while taking smaller steps defined by α. BIM and PGD are defined in the equations 3 and 4 respectively [2, 1].

$$v^{i+1} = Clip_\epsilon\{v^i + \alpha \text{sign}(\bigtriangledown_{x+v^i} J(x+v^i, y))\} \tag{3}$$

$$v^{i+1} = Project_{\epsilon}\{v^i + \alpha \text{sign}(\bigtriangledown_{x+v^i} J(x+v^i, y))\} \tag{4}$$

The main idea of both attacks is to take fixed-sized steps on the loss curve until the total amount of perturbation exceeds the boundary (ϵ). When the boundary is reached, the norm of the perturbation is pulled back into a valid range by simply clipping the perturbation values (BIM) or projecting it into the nearest L_p boundary (PGD). BIM and PGD are iterative, norm bounded, white-box, and untargeted attacks, although they can be turned into targeted attacks by taking gradient descent steps towards the target.

2.2.4 DeepFool

DeepFool estimates the geometric locations of the decision boundaries of a classifier and iteratively updates the perturbation vector until it reaches the closest decision boundary [14]. Given a binary classifier (for the sake of simplicity, although the attack can be performed against multi-class classifiers as well), the distance estimation and optimization are done via the closed-form formula given in Equation 5.

$$r_*(x_0) := argmin||r||_2 \text{ subject to sign}(f(x_0+r)) \neq \text{sign}(f(x_0)) = -\frac{f(x_0)}{||w||_2^2}w \tag{5}$$

Theoretically, the perturbation vector with the smallest L_p norm should be achieved since the resulting adversarial perturbation lies just on the other side of the decision boundary. This attack is strictly untargeted on paper since the main idea of the distance estimation is to find the closest decision boundary, i.e., the closest class that is not the original label of the image; however, as the decision boundaries are also non-linear, estimation of the distance to a target boundary can still be effective (this property is used in UAP [8] as well). Furthermore, DeepFool is an iterative, minimization, white-box, and image-dependent attack.

2.2.5 Carlini&Wagner

Carlini&Wagner (C&W) is another iterative minimization-based adversarial attack and is still being used as a benchmark for new adversarial attacks [15]. C&W reformulates the adversarial attack definition such that the non-linear $C(x+v) = t$ can be embedded into the optimization without being a box constraint. For that, a new adversarial loss function is proposed, which can be seen in Equation 6. The overall optimization is given in Equation 7.

$$F(m) = \max(\max_{i \neq t}(Z(m)_i) - Z(m)_t, 0) \tag{6}$$

$$\text{minimize}||\frac{1}{2}(\tanh(v)+1)-x||_p^p + c.F(\frac{1}{2}\tanh(v)+1) \qquad (7)$$

Equation 7 is first optimized using Adam optimizer for several iterations, then the scale factor c is adjusted in a binary search step; if the newly learned v is a successful attack, decrease the value of c, otherwise, increase it. In practice, v with the smallest c yields the best result in terms of $||L_p||$ norm and attack success. This attack is iterative, minimization, white-box, image-dependent and can be either targeted or untargeted depending on the adversarial loss.

2.2.6 Perceptual Color Distance Alternating Loss

Perceptual Color Distance C&W (PerC-C&W) aims to learn adversarial perturbations that cannot be perceived by human eye, by reformulating Equation 7 where instead of optimizing the traditional $||L_p||$ norm, the perceptual color distance CIEDE2000 [22] is optimized [16]. However, since C&W is a considerably slow algorithm (mainly because of the unparallelizable binary search procedure), "Perceptual Color Distance Alternating Loss (PerC-AL)" was also introduced. The adversarial perturbation is iteratively optimized using a loss function based on the attack success; if the current perturbation is successful, optimize the CIEDE2000 norm of the perturbation; else, optimize the classification loss. A similar approach was selected in this thesis work as well. This attack is white-box, iterative, image-dependent, minimization and can be either targeted or untargeted.

2.2.7 Spatially Transformed Adversarial Examples

An alternative approach to the adversarial example generation problem is proposed in that, rather than learning the perturbations in the pixel space, an algorithm learns the most optimal flow field that, when applied to the benign image, can fool the target network [17]. In this way, the distortions on the adversarial example originate from the already existing pixels in the image, and there is no extra additive noise, which makes them less perceptible. This approach was taken further by doing the optimization on the chrominance channels of the YUV transformation of the benign image [23]. In this way, the perturbations do not affect the shapes in the image, which makes the attack even less perceptible. These attacks are also image-dependent, minimization, iterative, white-box, and targeted.

2.3 Universal Adversarial Perturbations

The attacks described in the previous section were strictly image-dependent, and to generate an adversarial dataset, each algorithm must be applied to each sample individually. This section describes UAPs, which require a training set to either train the

UAP directly as if it is a parameter of the target model or a UAP generator model. Note that all of the mentioned attacks are inherently iterative during training but single-step during the application of the UAP into the benign image.

2.3.1 Universal Adversarial Perturbation

The problem of learning a single perturbation that, when applied to any image, can fool a deep network was first introduced and named "Universal Adversarial Perturbations" [8]. The UAP problem is defined as given in Equation 8. The aim is to learn a UAP v with a norm smaller than ϵ, that, in conjunction with any sample x from a dataset μ, will fool the classifier f with at least a probability of δ.

$$P_{x \sim \mu}(f(x+v) \neq f(x)) \geq \delta \text{ s.t. } \quad \|\nu\|_p \leq \epsilon \tag{8}$$

The work proposed a universal attack where a learned perturbation is iteratively applied to each sample of a dataset and put through DeepFool [14] algorithm if the current sample cannot fool the network. This way leads to a procedure that accumulates the smallest image-dependent perturbations over the whole dataset and obtains a single UAP; note that this procedure is very costly and does not guarantee an optimal result. UAP attack is a universal, norm bounded, white-box, iterative and can be either targeted or untargeted.

2.3.2 Network for Adversary Generation

Network for adversary generation (NAG) is a generative model that aims to learn the distribution of UAPs. As opposed to a standard generative adversarial network (GAN) or autoencoder (AE) model, NAG only trains a generator, whose weights are updated based on the loss obtained by the target network [18, 24]. The target network is fed three batches of inputs each iteration; an adversarial batch, a benign batch, and a shuffled adversarial batch. The loss function used to train NAG is composed of 2 parts; a fooling loss and a diversity loss. The fooling loss is computed using the adversarial batch and the benign batch, in that the distance between the predictions made on these batches is wanted to be maximized to ensure misclassification. On the other side, the diversity loss aims to maximize the distance between the generated UAPs of the same batch to force the generator to create diverse UAPs. NAG generates norm-bounded attacks by adding a scaled tanh non-linearity at the end of the last deconvolution layer of the generator. As tanh is defined in the interval of [-1,1], when the function is scaled by ϵ, the generated UAP is guaranteed to be in the defined range. NAG is a universal, white-box, and norm-bounded attack.

2.3.3 Feature-UAP

Feature-UAP (F-UAP) trains the UAP vector as if it is a weight of the target network; however, the loss function becomes an adversarial one to maximize the fooling rate performance [9]. This algorithm is also efficient in that the UAP can be trained using mini-batches and Adam optimizer; therefore, the convergence is achieved relatively quickly. This work also investigates the existence and surprisingly good performance of the UAPs by correlating the logits of the networks with benign images and UAPs. They find that when the network is fed a benign image, it produces a high positive correlation; however, when it is coupled with a UAP, the correlation of the UAP with the logits becomes strong, while the correlation of the actual image becomes weak. In this way, it is concluded that UAPs fool the networks by showing themselves as the actual image while showing the real image as some noise, such as Gaussian. F-UAP is also a norm-bounded adversarial attack, where the UAP is clamped by some ϵ every iteration after the update.

2.3.4 High-pass UAP

The existence of the UAPs was investigated from a deep steganography standpoint, which is a method where some message is encoded and added to an arbitrary benign image, which is in result called a container [19]. The container is then put through some decoder to extract the hidden message out of the image. It was found that the encoder networks strictly learned to encode the messages with high frequencies, which also meant that the human eye could struggle to perceive the distortions coming from the message. High-pass UAP (HP-UAP) was developed using this information; the algorithm is very similar to F-UAP, except that after each iteration, the UAP is filtered by a differentiable high-pass filter. This operation not only eliminates the visible artifacts from the UAP but also increases the performance of the UAP.

2.3.5 Fast Feature Fool

Fast feature fool (FFF) is a different adversarial attack in that the algorithm does not require any image data to train a UAP [20]. Instead, the UAP is fed into the target network by itself and trained to maximize the activation (therefore the output of the non-linear activation function such as ReLU) of each layer. It is also stated that it was empirically found that only maximizing the convolution layer activations is sufficient. To achieve this maximization, the loss function given in Equation 9 is used, where K is the number of layers of the target network, and $\bar{l}_i(v)$ is the mean activation of i^{th} layer. Also, after the update of the UAP in each iteration, the L_∞ perturbation values are clipped to ϵ, which makes this attack norm bounded. The attack is also white-box. FFF underperforms when compared to other UAPs in terms of fooling rate; however, it is a proof of concept of the ability to learn UAPs in a data-free fashion.

$$Loss = -\log\left(\prod_{i=1}^{K} \bar{l}_i(v)\right) \text{ such that } ||v||_\infty < \epsilon \tag{9}$$

2.3.6 Double Targeted Attack

Double targeted attacks (DTA) solve a different problem than the standard UAP problem; the main goal is to learn a UAP that will only change the prediction of a single class [21]. This way, a potential attack will be less suspicious, as only a single class is forced to be misclassified. The formal definition of this problem is given in Equation 10, where the sample belonging to the class desired to be misclassified is x_t (referred to as *targeted source class*), and y_{sink} is the *sink class* which x_t should be turned into.

$$\widetilde{F}(x_t + v) = y_{sink} \text{ subject to } ||v||_p \leq \epsilon \tag{10}$$

The UAP is trained using a particular loss function that is constituted of two components; target and non-target losses. The target loss is also composed of two parts, where the first one aims to decrease the logit value of the targeted source class of the sample, and the second one is to maximize the logit value of the sink class. These functions are based on the adversarial loss function proposed in C&W attack6. The non-target loss is to maintain the standard accuracy of all classes but the targeted source class, which is achieved using cross-entropy loss. During each iteration of the mini-batch training phase, a batch of samples, half of which is from the targeted source class and the other half from the sink class, is used; thanks to the hybrid loss function, the UAP learns to fool the network via only the targeted source class.

CHAPTER 3

METHODOLOGY

The standard UAP learning problem is defined as given in Equation 8, whose constraint is the maximum L_p norm of the vector. This definition is given to maximize the fooling rate as long as the norm is smaller than some ϵ; therefore, it does not allow parameterization of the fooling rate. In this thesis, the UAP training problem is reformulated to be able to set a target fooling rate. Equation 11 presents the formal definition of the problem where δ is the target fooling rate.

$$\text{minimize}\, ||v||_p \quad \text{s.t.} \quad P_{x \sim \mu}(f(x+v) = t) \approx \delta \tag{11}$$

With this formulation, the optimization target is to find the smallest perturbation v which can fool the target network P with a probability of δ, in conjunction with a benign example x, sampled from some distribution μ.

Many solutions can be generated to the problem of achieving a desired fooling rate over a dataset. AL scheme can be used to adjust the learning procedure of the UAP; in that, altering the loss function based on the current performance of the UAP is a possible choice. The AL strategy was already used to generate image-dependent adversarial perturbations; however, it was not used for the generation of UAPs. The primary motivation behind using AL is that it is a greedy approach that aims to prioritize the weakness of the optimized entity in each step, and in this case, the UAP. Since, in this context, the **weakness** can be easily detected by only estimating the fooling rate performance of the UAP over the dataset, the altering conditions can be defined easily as well. In this section, the 3 UAP generation algorithms that take advantage of the AL scheme will be introduced by defining different loss-altering conditions. Each subsequent attack will address the shortcomings faced in the previous one; hence multiple attacks were introduced until reaching an optimal solution to the problem of stabilizing the fooling rate of the UAP. Furthermore, based on some empirical quantitative and visual results, the train-time filtering operations were developed, and their methodologies will be further explained in the next section. The proposed UAP generation algorithm is presented as a pipeline in Figure 1.

Algorithm 1 Batch Alternating Loss Training(B-AL)

Input: Dataset μ, target class t, target fooling rate δ, epoch k, model f, norm p
Variables: Counter i, fooling rate fr, prediction out, adversarial loss function adv, loss L
Output: Universal adversarial perturbation v

```
1:  $v \leftarrow 0$
2:  $i \leftarrow 0$
3:  while $i < k$ do
4:      for $x \sim \mu$ do
5:          $out \leftarrow f(x + v)$
6:          $fr \leftarrow$ # of incorrect predictions / batch size
7:          if $fr < \delta$ then
8:              $L \leftarrow adv(out, t)$
9:          else
10:             $L \leftarrow ||v||_p$
11:         end if
12:         backpropagate $L$
13:         update $v$
14:         $i \leftarrow i + 1$
15:     end for
16: end while
17: return $v$
```

3.1 Batch Alternating Loss Training

The pseudocode of B-AL training can be seen in Algorithm 1. During B-AL training, the loss function is altered based on the fooling rate performance of the current batch. When the combination of the UAP and the current batch achieves the desired fooling rate, the loss function of that particular iteration is selected to be the L_p norm; otherwise, the loss function becomes adversarial. This method of integration is simple; however, it comes with several potential drawbacks. Firstly, it assumes that the batch size is large enough (at least 32) that the performance against a random sample from the dataset can be approximated as the fooling rate over the whole distribution, which can be costly in terms of GPU memory usage, since each image within a batch results in multiplication the number of tensors that needs to be held in memory. On the other hand, it is possible that the norm loss can bring the adversarial energy of the UAP down to a point where the fooling rate can never reach the desired level.

3.2 Epoch-Batch Alternating Loss Training

EB-AL (given in Algorithm 2) training is similar to B-AL training; however, instead of solely looking at the performance of each batch, the adversarial performance over the whole dataset is also considered. Unless the desired fooling rate was reached during the previous epoch, the norm could not be optimized; therefore, the loss function stays adversarial throughout the whole epoch. If the target fooling rate was achieved at the end of the previous epoch, then the loss function alteration condition becomes the same as B-AL. This method hypothetically solves the unsuccessful fooling rate convergence problem of B-AL training. However, it may cause the fooling rate to largely oscillate over and under the target level, which is undesirable regarding the defined problem. Besides, EB-AL also suffers from the high memory consumption problem since when the alteration condition becomes the same as B-AL, the batch size must be large enough.

3.3 Progressive Alternating Loss Training

P-AL (given in Algorithm 3) takes a different approach to the loss function alteration condition; in order to take the performance over the whole dataset into account, P-AL calculates the cumulative fooling rate of the current batch if it is higher than the target level, the norm loss is selected; otherwise, the adversarial loss is applied. This method solves the problem of large oscillation around the target fooling rate and reduces the GPU memory overhead. Also, as this algorithm utilizes the global cumulative fooling rate, the effect of the batch size becomes very minimal compared to the other two proposed attacks.

3.4 Masked Training

It was visually found that the image-like features tend to mitigate towards the edges of the UAPs. Also, relatively smaller magnitude perturbations were formed around the center of the UAP. However, these perturbations are mostly visually incoherent, which contradicts the idea that UAPs contain image-like features. Therefore their contribution to the fooling performance against the network is weaker than the ones forming around the edges of the UAP. Furthermore, as these perturbations are near the center of the image, it is likely that they modify the pixels of the actual object in the image since the objects are located in the middle of the frame with a higher probability. Hence, these perturbations are likely to draw more attention from a human observer while having a negligible impact on the adversarial performance of the UAP. Thus, to alleviate the emergence of these perturbations, different kinds of masks were incorporated into the training procedure, producing UAPs containing dominant perturbations around the edges. Four different types of filters were chosen for this masking operation.

3.4.1 Circular Filter

With a circular filter, the pixels that are farther from the center of the image more than by a value r, are set to 1, and the rest is set to 0 (Equation 12). Visualization of the filter with three different radius sizes is given in Figure 2. Note that Figure 2c does not show a perfect circle since the diameter of the circle (248) is higher than the length of the edges of the image (224).

$$f(x) = \begin{cases} 1, & \text{if } \sqrt{(\frac{w}{2} - x)^2 + (\frac{h}{2} - y)^2} \geq r \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

3.4.2 Gaussian Filter

Gaussian filter is a filter where the norm of a weight is valued between 0 and 1, inversely proportional to its distance from the center. However, in this case, each value is subtracted from 1 to strengthen the pixels that are farther from the center of the mask, as shown in Equation 13, where x and y are the positions of the pixels, and σ is the variance of the Gaussian distribution. Also, the visualization of the filter for 3 different σ values are shown in Figure 3.

$$G(x) = 1 - \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{13}$$

3.4.3 Butterworth Filter

High pass Butterworth filter [25] also contains smooth passage between passed and filtered frequencies, with the introduction of the variable n, which manipulates the clearness of the filter response; the transitions are faster than the Gaussian filter but slower and smoother than the circular filter. The formulation of the filter is given in Equation 14, where D_0 is the center coordinate of the image, (x, y) are the coordinates of the input pixel, and n is a parameter that affects the smoothness of the transition. The Butterworth filter is visualized in Figure 4 with different parameters.

$$B(x, y) = \frac{1}{1 + [D_0/D(x, y)]^{2n}} \tag{14}$$

3.4.4 Rectangular Filter

The rectangular filter is a filter where the pixels that are less distant from the edges by a value n_x (distance from left and right edges) and n_y (distance from top and bottom edges) are 1 and the rest is 0. The visualization and the equation of the filter are given in Figure 5 and Equation 15 respectively; x and y are the coordinates of the pixel, L_x and L_y are the horizontal and vertical lengths of the filter, n_x and n_y are the offsets from the edges where the filter value is 1(note that the origin is taken as the top left corner).

$$R(x, y) = \begin{cases} 0, & \text{if } n_x < x < L_x - n_x \text{ and } n_y < y < L_y - n_y \\ 1, & \text{otherwise} \end{cases} \tag{15}$$

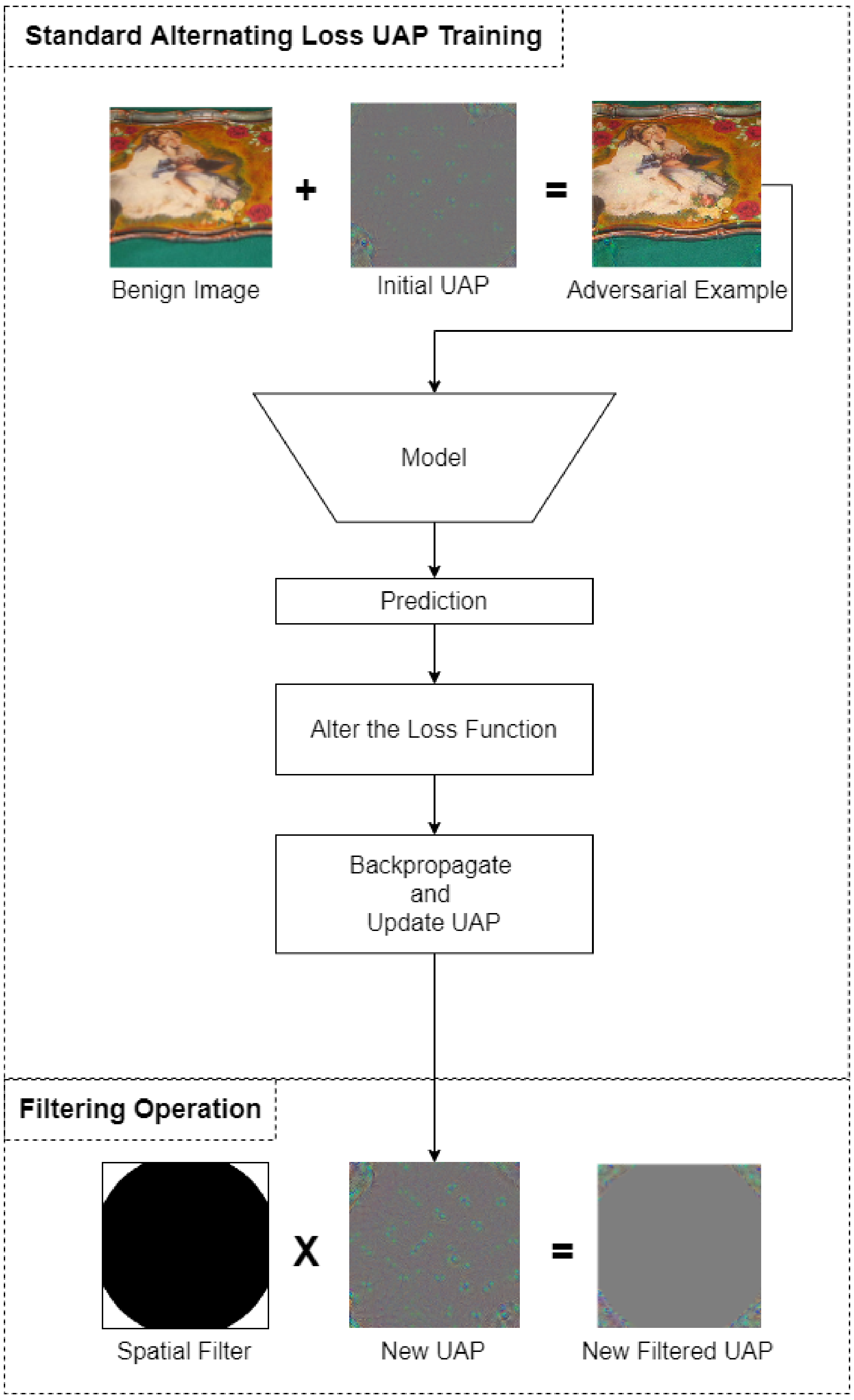


Figure 1: Alternating loss UAP training pipeline

Algorithm 2 Epoch-Batch Alternating Loss Training (EB-AL)

Input: Dataset μ, target class t, fooling rate δ, epoch k, model f, norm p
Variables: Counter i, fooling rate fr, prediction out, adversarial loss function adv, loss L, optimization mode m, number of correct predictions $correct$, image number counter $imcount$, fooling rate over the epoch $epochfr$
Output: Universal adversarial perturbation v

```
1: $v \leftarrow 0$
2: $i \leftarrow 0$
3: $m \leftarrow$ 'epoch'
4: while $i < k$ do
5:     $correct \leftarrow 0$
6:     $imcount \leftarrow 0$
7:     for $x \sim \mu$ do
8:         $out \leftarrow f(x + v)$
9:         $correct \leftarrow correct+$ # of correct predictions
10:        $imcount \leftarrow imcount+$ batch size
11:        $fr \leftarrow$ # of incorrect predictions / batch size
12:        if ($m$ == 'epoch') or
           ($m$ == 'batch' and $fr < \delta$) then
13:            $L \leftarrow adv(out, t)$
14:        else
15:            $L \leftarrow ||v||_p$
16:        end if
17:        backpropagate $L$
18:        update $v$
19:        $i \leftarrow i + 1$
20:    end for
21:    $epochfr \leftarrow 1 - correct/imcount$
22:    if $epochfr < \delta$ then
23:        $m \leftarrow$ 'epoch'
24:    else
25:        $m \leftarrow$ 'batch'
26:    end if
27: end while
28: return $v$
```

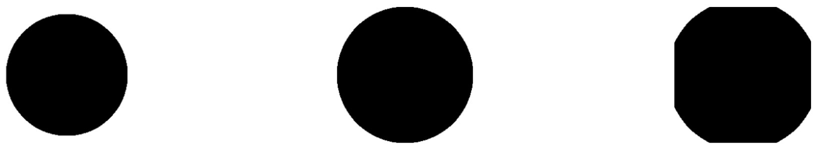
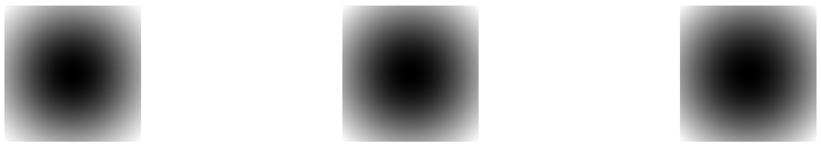

(a) $r = 100$ (b) $r = 112$ (c) $r = 124$

Figure 2: Circular filters with different r values

Algorithm 3 Filtered Progressive Alternating Loss Training (FP-AL)

Input: Dataset μ, target class t, fooling rate δ, epoch k, model f, norm p, mask radius D
Variables: Counter i, fooling rate fr, prediction out, adversarial loss function adv, loss L, number of correct predictions $correct$, image number counter $imcount$, circular filter $filter$
Output: Universal adversarial perturbation v

```
1:  $v \leftarrow 0$
2:  $i \leftarrow 0$
3:  $filter \leftarrow$ any of the filters
4:  **while** $i < k$ **do**
5:      $correct \leftarrow 0$
6:      $imcount \leftarrow 0$
7:      **for** $x \sim \mu$ **do**
8:          $out \leftarrow f(x + v)$
9:          $correct \leftarrow correct$ + # of correct predictions
10:         $imcount \leftarrow imcount$ + batch size
11:         $fr \leftarrow$ ($imcount-$ length of $correct$) / $imcount$
12:         **if** $fr < \delta$ **then**
13:             $L \leftarrow adv(out, t)$
14:         **else**
15:             $L \leftarrow ||v||_p$
16:         **end if**
17:         backpropagate $L$
18:         update $v$
19:         $v \leftarrow filter(v)$
20:         $i \leftarrow i + 1$
21:     **end for**
22: **end while**
23: **return** $v$
```



(a) $\sigma = 100$ (b) $\sigma = 112$ (c) $\sigma = 124$

Figure 3: Gaussian filters with different σ values

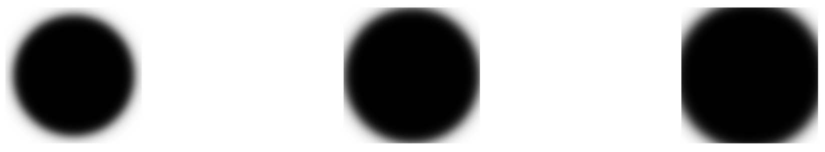

(a) $D_0 = 100$ (b) $D_0 = 112$ (c) $D_0 = 124$

Figure 4: Butterworth filters with $n = 30$ and different D_0 values

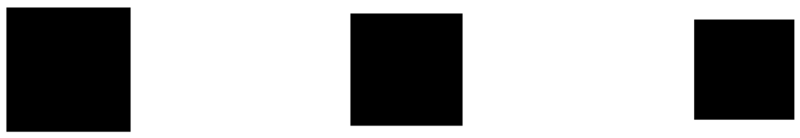

(a) $n_x = n_y = 100$ (b) $n_x = n_y = 112$ (c) $n_x = n_y = 124$

Figure 5: Rectangular filters with $L_x = L_y = 224$ and different n_x, n_y values

CHAPTER 4

RESULTS

This section gives the results of the experiments done using multiple parameters. It presents the settings where the experiments were done in detail, which includes the explanation of the target dataset, chosen attack parameters, comparison methods, and the computation environment.

4.1 Experimental Settings

The dataset used in this work is ImageNet, whose training and validation subsets contain around 1.2 million and 50000 images of 1000 classes [26]. However, since, in this case, the trained parameter set is not a model but a small noise vector, the training set was sampled in a way that only 10 images were randomly selected from each class, thus significantly shortening the training times of the UAPs. As a result, the UAPs were not trained on the whole ImageNet data; instead, they were trained using 10000 images from 1000 classes. Note that the validation set was not sampled, and the whole set was used for network evaluation. The same methodology is also adopted in the compared papers.

As the attacks are strictly targeted in this case, the target class was chosen to be "peacock", and the attacks were applied under L_2 and L_∞ settings. Note that the attacks are more suitable for optimization under L_2 boundaries since small updates on the noise vector are likely to be more effective under these settings. Also, the target fooling rate was set to be 95% throughout the experiments. It was empirically found that if a UAP can reach a 95% fooling rate, in the absence of regulations, it tends to reach 100% in several iterations, which is normally considered to be a successful attack; however, under the problem defined in this thesis, diverging from the target fooling rate is considered unsuccessful. It is highly possible that a UAP cannot reach 95% at all, as it is a relatively high fooling rate, and in such a case, the UAP is also considered unsuccessful. Therefore, 95% was found to be a reliable level that shows both the adversarial capacity and the controllability of a UAP generation method. Each UAP was trained for 20 epochs using Adam optimizer (with an initial learning rate of 0.01), with the sampled training set, and tested using the standard validation set. The target networks were selected to be DenseNet121 [27], ResNet50 [28], GoogLeNet [29], VGG16 with batch normalization [30] and ViT [31], which adapts a transformer architecture instead of a convolutional neural network architecture.

The proposed attacks were compared with two widespread universal attacks, vanilla UAP and Feature UAP. Similar to the proposed attacks, vanilla UAP can take a parameter where the training will be stopped if the fooling rate reaches the parameter value; however, this parameter does not directly impact the actual optimization. Note that the proposed attacks fix the fooling rate and try to minimize the L_p norm; however, the other universal attacks fix the L_p norm and maximize the fooling rate. This makes it rather difficult to compare the performance of the UAPs. To make a controlled experiment, first, the UAPs obtained from the proposed attacks are obtained with fixed fooling rates and varying L_p norms; then, the compared UAPs were trained by fixing the L_p norms as obtained from the corresponding proposed UAP.

In the filtered training phase, each type of filter was used with three different parameters. The radius parameter D of the circular filter, the variance (σ) of the Gaussian filter, and D_0 of the Butterworth filter were set to be 100, 112, and 124. As the Butterworth filter also has the parameter n, it was strictly set to 30, based on several empirical studies for simplicity. The rectangular filter was used with parameters n_x and n_y, which take values as either 10, 20, or 30. The filtering operation was not applied to the vanilla UAP and F-UAP attacks, as it requires changing these attacks substantially.

The code for this thesis was written in Python programming language, which has become the norm for deep learning practices in recent years, thanks to its higher level syntax and interpretation, but most importantly, thanks to the supported GPU acceleration libraries. PyTorch, being one of the most popular GPU accelerated deep learning frameworks, was used in this thesis. Furthermore, the experiments were done in a workstation containing two NVIDIA RTX 3080 GPUs.

4.2 L_2 Attack Results

The results of the L_2 attacks are presented in Table 2; the first three rows show the results of the proposed attacks, whereas the last two rows give the results of the compared attacks whose L_2 norm constraints were set to be equal to the value obtained from P-AL as it was crafted to be the most optimal algorithm out of the three proposed attacks. Note that all the proposed attacks were set to converge to a 95% fooling rate during the optimization.

In this context, B-AL has failed to achieve the target fooling rate against each network while consistently yielding the smallest L_2 norms compared to the EB-AL and P-AL (except against VGG16, where the attack is successful) by respectively 49.60% and 21.24%. However, since the proposed attacks' primary objective is to stabilize around the desired fooling rate, these results cannot be evaluated as successful, despite the marginal fooling rate deviation and small norm values.

On the other hand, EB-AL yields result in the opposite way, where in each case, the attack surpasses the desired fooling rate while yielding considerably higher L_2 norms than the other attacks' norms. Although achieving higher fooling rates is considered to be better in the standard case, for the problem defined in this work, getting farther from the desired fooling rate cannot be evaluated to be better.

The results of L_2 P-AL attacks resemble a combination of the results of the other two attacks; in each case, the attack reaches and converges around the desired 95% fooling rate while yielding norm values that are consistently less than EB-AL but slightly higher than B-AL attacks. Therefore, this attack can be considered successful against any of the networks for the given problem. Also, both the vanilla UAP and the F-UAP attacks fall short in terms of the fooling rate under the same L_2 norm constraints compared to P-AL. Surprisingly, despite still yielding high fooling rates against DenseNet121, ResNet50, and GoogLeNet, F-UAP achieves 70% and 50% fooling rates against VGG16 and ViT. Figure 6 shows visual examples of the output of this attack.

Table 2: The results of the proposed and compared L_2 attacks. The results of the proposed attacks that did not at least reach the target fooling rate were italicized, and the attack result with the least norm value was given as bold.

Method	DenseNet121		ResNet50		GoogLeNet		VGG16		ViT	
	L_2	FR	L_2	FR	L_2	FR	L_2	FR	L_2	FR
B-AL	*9.14*	*0.93*	*9.08*	*0.93*	*9.56*	*0.91*	7.10	0.95	*10.44*	*0.91*
EB-AL	14.11	0.98	14.36	0.98	15.73	0.98	7.39	0.95	17.63	0.98
P-AL	**11.16**	**0.95**	**11.66**	**0.95**	**13.01**	**0.95**	**5.52**	**0.95**	**11.84**	**0.95**
UAP	11.16	0.33	11.66	0.34	13.01	0.43	5.52	0.30	11.84	0.50
F-UAP	11.16	0.90	11.66	0.93	13.01	0.91	5.52	0.70	11.84	0.50

4.3 L_∞ Attack Results

Table 5 shows the results of the L_∞ attacks. In this case, B-AL and EB-AL attacks yield similar fooling rate and norm results, in the that against each network, the fooling rates diverge to 100% which is an undesirable situation for this particular problem; besides, the L_∞ norms tend to get considerably large.

The results of P-AL attacks are better in terms of the deviation from the target fooling rate, which is at most 1 percentage point against the CNNs, however, it also fails against ViT; although it can be seen that the L_∞ norm is 42.11% and 47.39% less than those of B-AL and EB-AL respectively. Several examples outputs from different networks are given in Figure 7.

4.4 Perturbation Accumulation Results

As mentioned earlier, UAPs tend to contain image-like features, unlike image-dependent adversarial perturbations, which are mostly perceived as noise by themselves. This

Table 3: The results of the proposed and compared L_∞ attacks. The results of the proposed attacks that did not at least reach the target fooling rate were italicized, and the attack result with the least norm value was given as bold.

Method	DenseNet121		ResNet50		GoogLeNet		VGG16		ViT	
	L_∞	FR	L_∞	FR	L_∞	FR	L_∞	FR	L_∞	FR
B-AL	0.17	1.00	0.17	1.00	0.20	1.00	0.16	1.00	0.27	1.00
EB-AL	0.16	1.00	0.18	1.00	0.22	1.00	0.17	1.00	0.28	1.00
P-AL	**0.11**	**0.96**	**0.13**	**0.96**	**0.16**	**0.96**	**0.11**	**0.95**	**0.19**	**1.00**
UAP	0.11	0.52	0.13	0.60	0.16	0.79	0.11	0.75	0.19	1.00
F-UAP	0.11	0.99	0.13	0.99	0.16	0.99	0.11	0.99	0.19	1.00

property of UAPs creates an illusion over the network such that it perceives the benign image as noise and the perturbation as the actual image [9]. However, UAPs also contain noisy visible features such as blobs which go against the main feature of the UAPs. It was empirically found that the image-like features tend to accumulate around the edges of the perturbation vector, whereas the noisy features appear around the center; not only are these features perceived easily by the human eye because of their locations, but also they do not contribute to the fooling performance of the UAP. Figure 8 shows the average gradient values of perturbations and their distances to the center of the image. It can be seen that after several UAP optimization iterations, the perturbations which are far from the center tend to generate gradients with higher norms, where the gradients are the partial derivatives of the loss function with respect to each pixel perturbation.

4.5 Filtered L_2 Attack Results

Following the previous findings, the perturbations around the center were masked during the UAP training to reduce the amount of perturbation visually and quantitatively. The masked training results of the L_2 attacks are presented in Table 4. There are four types of filters, and each filter is experimented with three parameter values; for the sake of simplicity, only the best performing(the ones that at least reached the target fooling rate and gave the smallest norm) filter-parameter pairs were given for each attack.

Like the unfiltered results, Filtered B-AL (FB-AL) struggles to reach the target fooling rate regardless of the filter type. However, unlike circular and rectangular filters, Gaussian and Butterworth filters considerably reduce the fooling rates. On the other hand, a rectangular filter consistently reduces the L_2 norms while maintaining the fooling rate performance from the original attack.

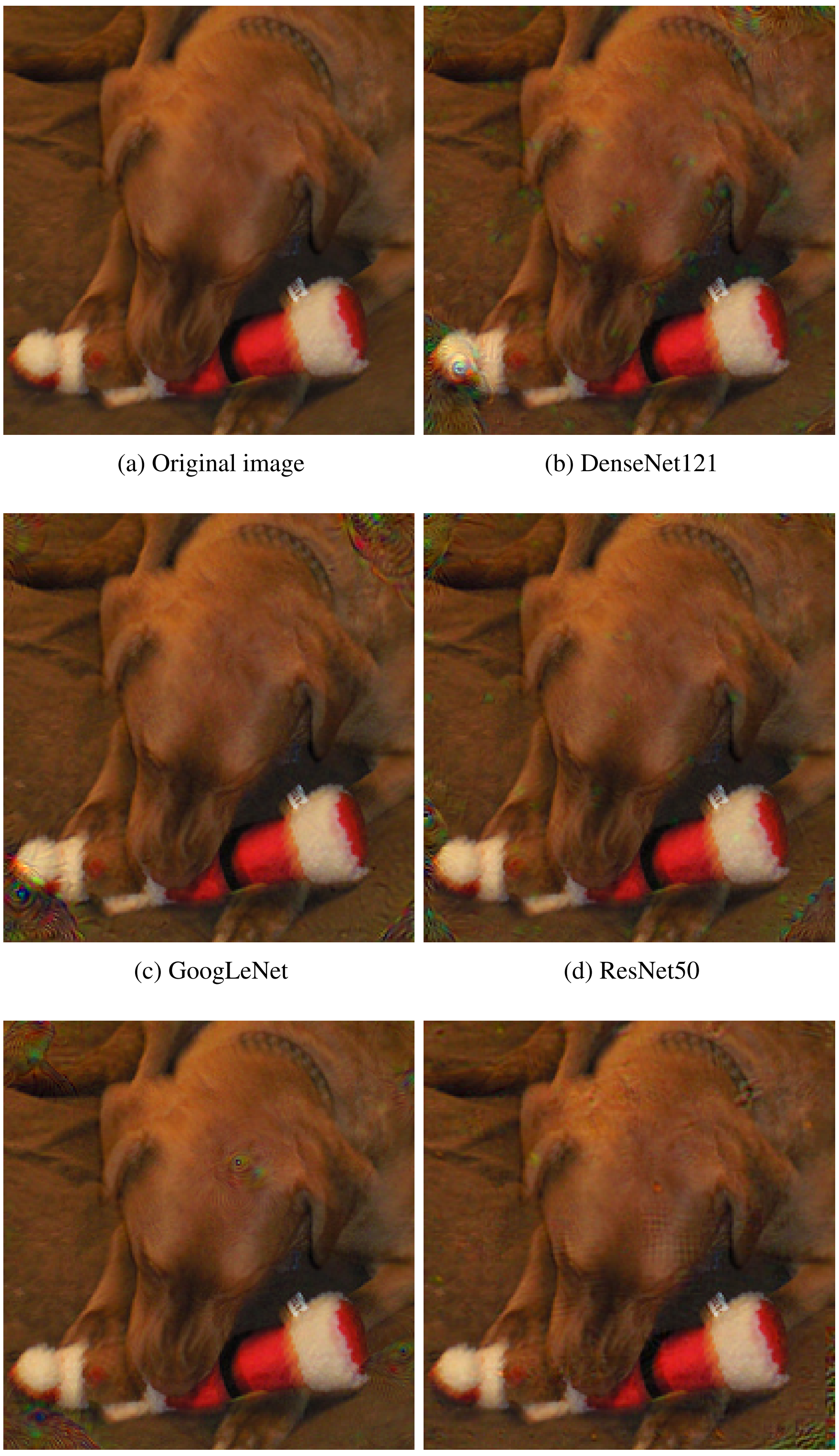
In the Filtered EB-AL (FEB-AL) results, each filtered attack except the Gaussian at least stays over the target fooling rate while consistently reducing the L_2 norms. It is also possible to see small reductions in the fooling rates, which in this case is beneficial; the original EB-AL attack consistently yields fooling rates higher than the desired level; thanks to the filters, the difference between the achieved and desired fooling rates get smaller. Example outputs of the attack are given in Figure 9

Filtered P-AL (FP-AL) attack not only consistently (except for the Gaussian filter) holds the already successful fooling rates but also considerably reduces the L_2 norms against almost any target network. Therefore the filtering operation gives even more optimal results than the unfiltered P-AL attack. Outputs of this attack can be found in Figure 10.

4.6 Filtered L_∞ Attack Results

The stabilization problem of the L_∞ attack persists in the filtered training paradigm as well, in that the fooling rates tend to converge to 100%. Unlike the success of the filtering operation applied to the L_2 attacks, the filtering operations increase the norm values while not significantly affecting the fooling rate. As an exception, the Butterworth filter reduced the average fooling rate to 95.2%, which is only marginally different from the target fooling rate; however, the average L_∞ norm has increased by 58.7%, which is an unacceptably high deviation from the unfiltered variant of the attack. On the other hand, the results obtained from FEB-AL(whose example outputs can be found in Figure 12) are very close to FB-AL; in fact, most of the norm and fooling rate values only fractionally differ from their counterparts. Note that the unfiltered L_∞ attacks also yielded similar values.

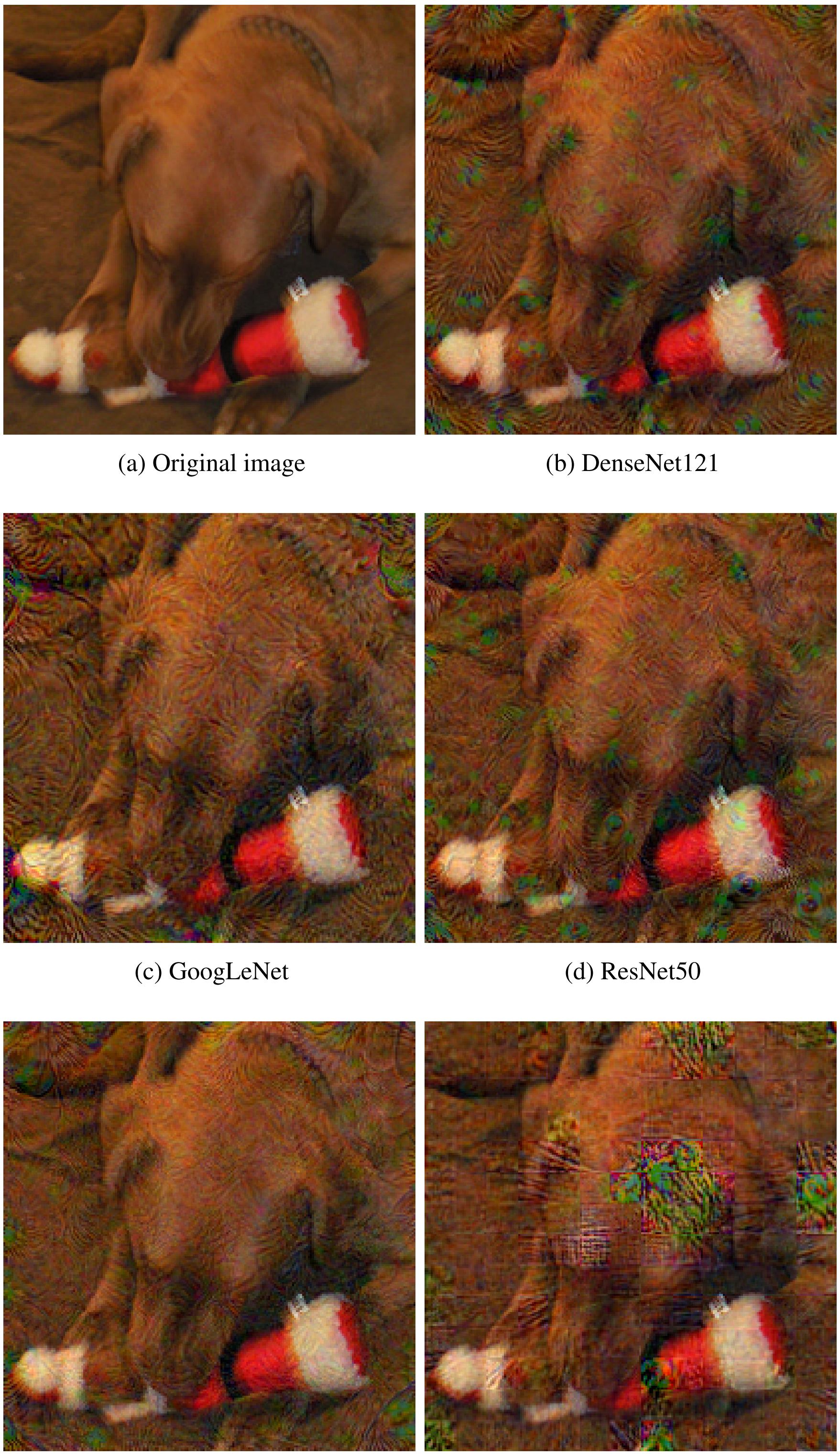
Filtering the L_∞ P-AL attack increases the norm values drastically in most cases while having a minimal impact on the fooling rate. Even though, based on the previous results, this situation can be expected, it still makes the attack inferior to the unfiltered one. Visual outputs of this attack are given in Figure 11.


(a) Original image (b) DenseNet121

(c) GoogLeNet (d) ResNet50

(e) VGG16 BN (f) ViT

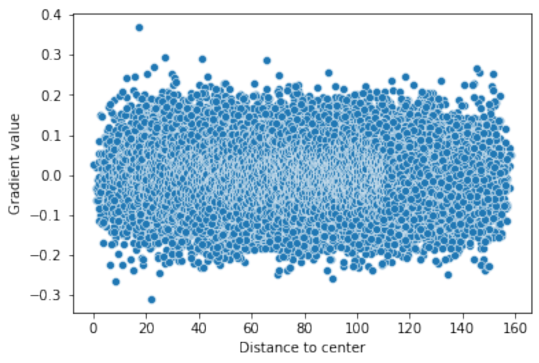
Figure 6: Attack results obtained by attacking the indicated network using L_2 P-AL. The original class of the image is "Chesapeake Bay retriever".


(a) Original image (b) DenseNet121

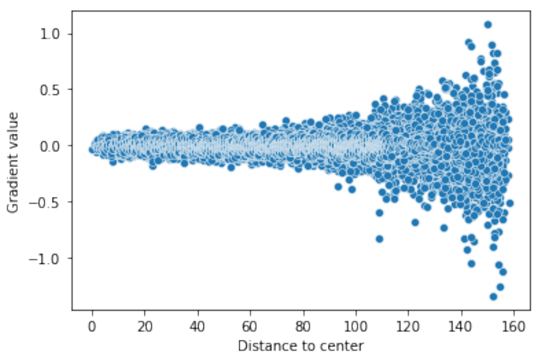
(c) GoogLeNet (d) ResNet50

(e) VGG16 BN (f) ViT

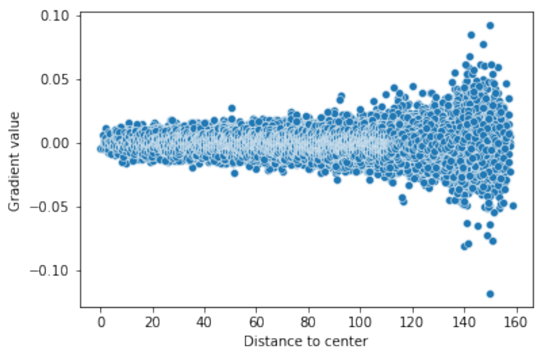
Figure 7: Attack results obtained by attacking the indicated network using L_∞ P-AL. The original class of the image is "Chesapeake Bay retriever".



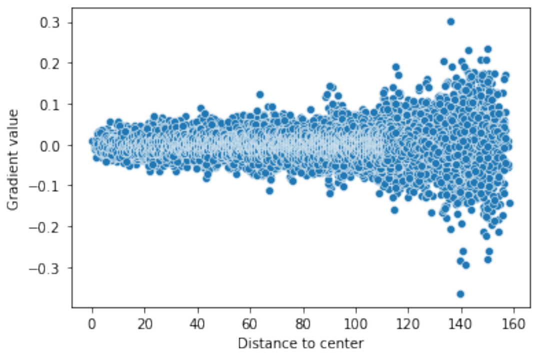
(a) Iteration 1



(b) Iteration 30



(c) Iteration 150



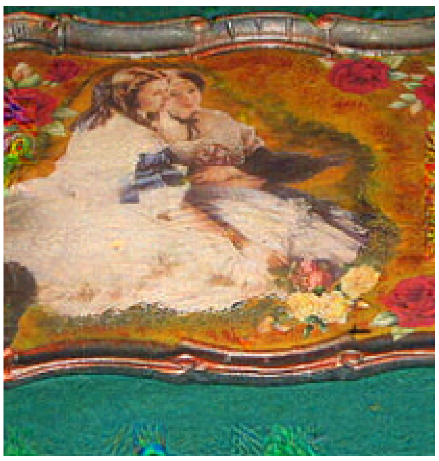
(d) Iteration 300

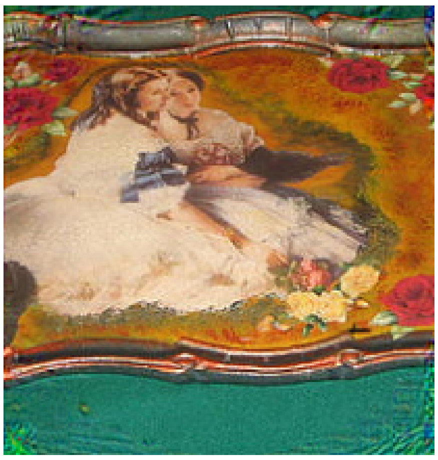
Figure 8: Vertical axes show the mean gradient value, horizontal axes show the distance between the pixel containing the corresponding mean gradient, and the center of the image. The scatter plots are extracted from UAP states after iteration number 1, 30, 150 and 300.

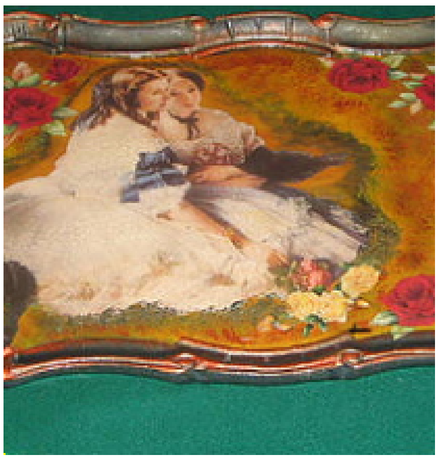
Table 4: The results of the proposed and compared filtered L_2 attacks. The results of the proposed attacks that did not at least reach the target fooling rate were italicized, and the attack result with the least norm value was given as bold. The filter types are identity (Iden.), circular (Circle), Gaussian (Gaus.), Buttherworth (Bwth.) and rectangular (Rect.).

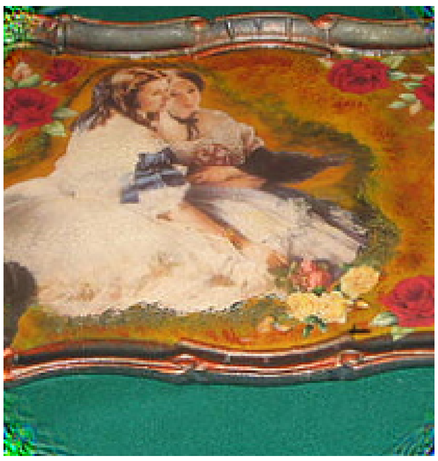
Method	Filter	Param.	DenseNet121		ResNet50		GoogLeNet		VGG16		ViT	
			L2	FR	L2	FR	L2	FR	L2	FR	L2	FR
FB-AL	Iden.	-	*9.08*	*0.93*	*8.76*	*0.93*	*9.75*	*0.91*	*7.26*	*0.94*	*10.44*	*0.91*
	Circle	$r = 100$	*9.24*	*0.92*	*8.48*	*0.93*	*9.17*	*0.91*	*6.97*	*0.94*	*10.72*	*0.91*
	Gaus.	$\sigma = 100$	*3.44*	*0.24*	*3.07*	*0.22*	*2.87*	*0.30*	*3.32*	*0.27*	*2.92*	*0.24*
	Bwth.	$D_0 = 100$	*9.07*	*0.90*	*8.22*	*0.90*	*10.59*	*0.88*	*7.11*	*0.92*	*11.60*	*0.87*
	Rect.	$n = 30$	*8.80*	*0.93*	*8.39*	*0.93*	*9.08*	*0.91*	*7.22*	*0.94*	*10.31*	*0.91*
FEB-AL	Iden.	-	14.68	0.98	14.47	0.98	15.93	0.98	7.18	0.95	17.63	0.98
	Circle	$r = 112$	12.98	0.97	13.31	0.97	14.24	0.98	11.06	0.99	16.77	0.97
	Gaus.	$\sigma = 112$	*3.34*	*0.23*	*3.16*	*0.22*	*3.06*	*0.30*	*3.40*	*0.27*	*3.32*	*0.24*
	Bwth.	$D_0 = 100$	13.81	0.96	12.07	0.96	14.49	0.95	10.00	0.96	17.39	0.95
	Rect.	$n = 30$	13.63	0.98	13.62	0.97	15.15	0.98	11.46	0.99	16.77	0.97
FP-AL	Iden.	-	9.93	0.95	10.30	0.95	**10.11**	**0.95**	7.19	0.95	11.84	0.95
	Circle	$r = 112$	**7.73**	**0.95**	**7.76**	**0.95**	12.53	0.95	**6.81**	**0.95**	**10.63**	**0.95**
	Gaus.	$\sigma = 124$	*3.43*	*0.23*	*3.10*	*0.22*	*3.09*	*0.30*	*3.37*	*0.27*	*3.21*	*0.23*
	Bwth.	$D_0 = 100$	12.42	0.95	12.99	0.95	12.79	0.94	8.40	0.95	*16.92*	*0.94*
	Rect.	$n = 30$	8.61	0.95	8.44	0.95	8.10	0.95	7.07	0.95	12.68	0.95

(a) Original image


(b) Identity


(c) Circular


(d) Gaussian


(e) Butterworth

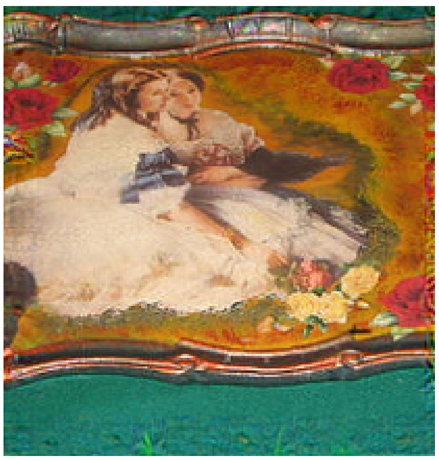
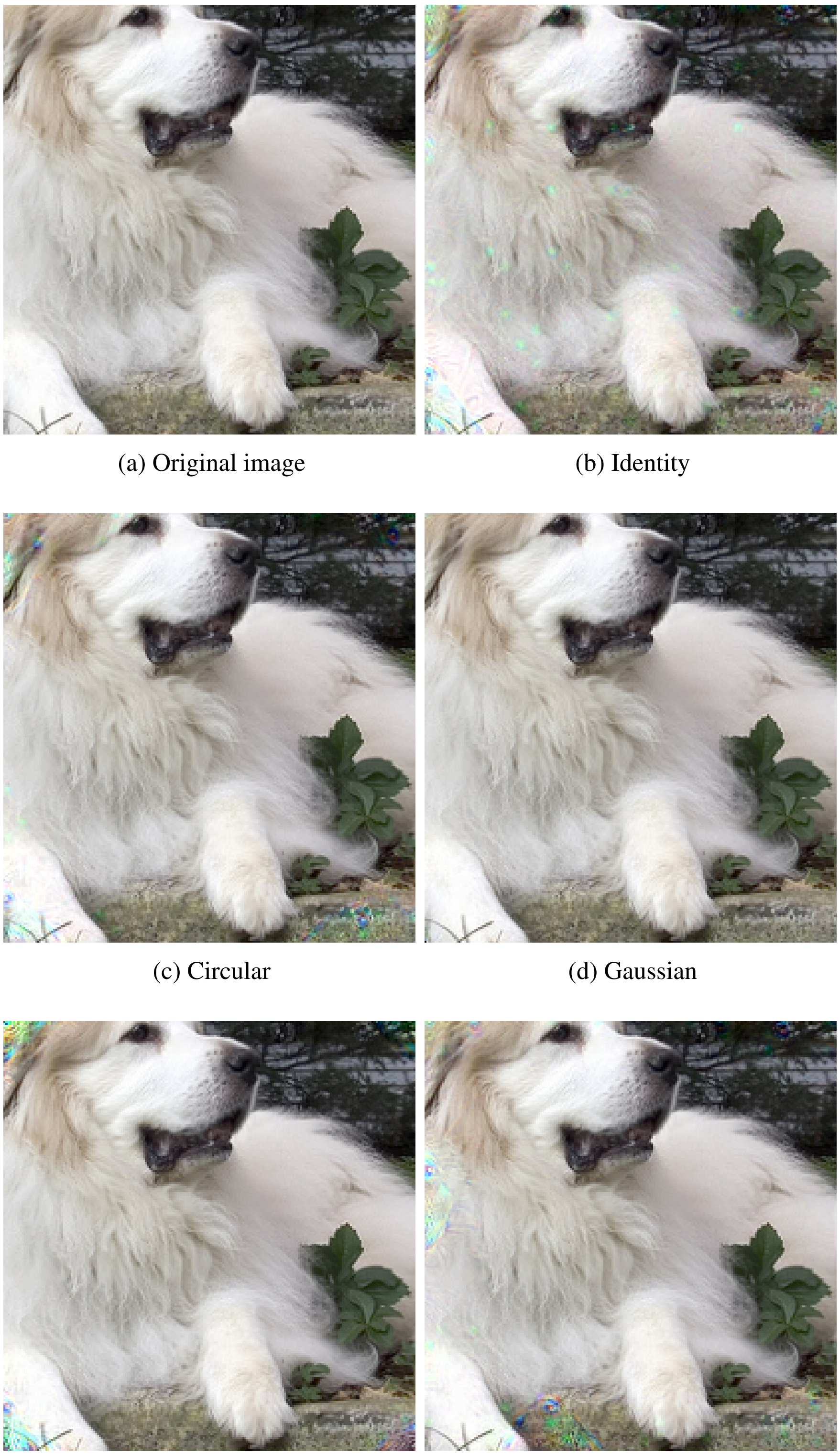

(f) Rectangular

Figure 9: Attack results obtained by L_2 FEB-AL and indicated filter, against ResNet50. The original class of the image is "Great Pyrenees".


(a) Original image (b) Identity

(c) Circular (d) Gaussian

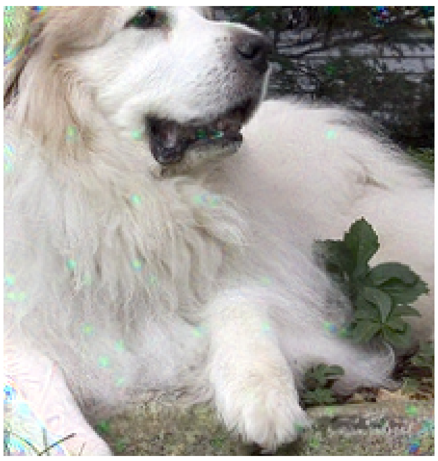
(e) Butterworth (f) Rectangular

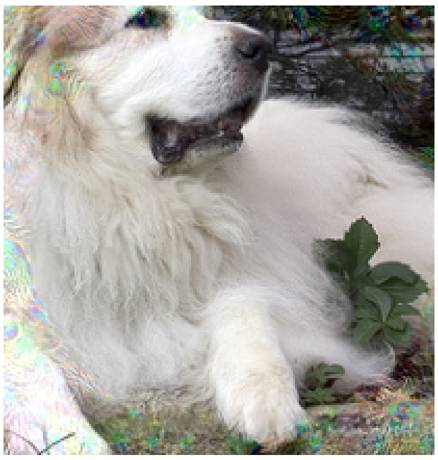
Figure 10: Attack results obtained by L_2 FP-AL and indicated filter, against ResNet50. The original class of the image is "Great Pyrenees".

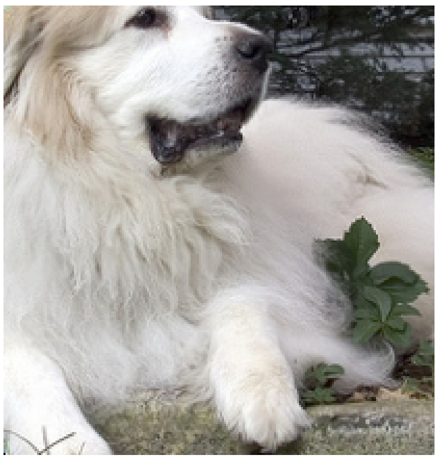
Table 5: The results of the proposed and compared filtered L_∞ attacks. The results of the proposed attacks that did not at least reach the target fooling rate were italicized, and the attack result with the least norm value was given as bold. The filter types are identity (Iden.), circular (Circle), Gaussian (Gaus.), Buttherworth (Bwth.) and rectangular (Rect.).

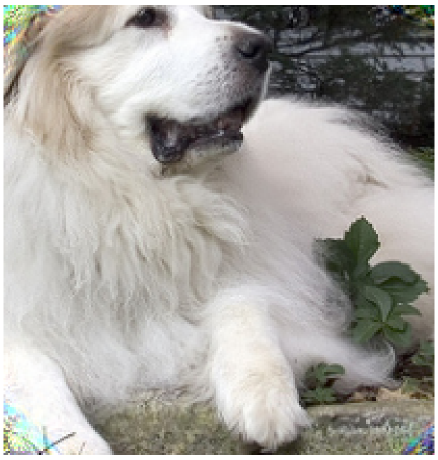
Method	Filter	Param.	DenseNet121		ResNet50		GoogLeNet		VGG16		ViT	
			L_∞	FR	L_∞	FR	L_∞	FR	L_∞	FR	L_∞	FR
FB-AL	Iden.	-	0.17	1.00	0.18	1.00	0.20	1.00	0.15	1.00	0.27	1.00
	Circle	$r = 100$	0.21	1.00	0.22	1.00	0.24	1.00	0.18	1.00	0.31	1.00
	Gaus.	$\sigma = 124$	*1.00*	*0.24*	*1.00*	*0.22*	*1.00*	*0.30*	*1.00*	*0.27*	*1.00*	*0.24*
	Bwth.	$D_0 = 100$	0.29	0.96	0.29	0.95	0.33	0.95	0.26	0.96	0.37	0.94
	Rect.	$n = 30$	0.20	1.00	0.22	1.00	0.23	1.00	0.17	1.00	0.29	1.00
FEB-AL	Iden.	-	0.17	1.00	0.18	1.00	0.22	1.00	0.16	1.00	0.28	1.00
	Circle	$r = 100$	0.21	1.00	0.22	1.00	0.23	1.00	0.19	1.00	0.32	1.00
	Gaus.	$\sigma = 112$	*1.00*	*0.23*	*1.00*	*0.22*	*1.00*	*0.30*	*1.00*	*0.27*	*1.00*	*0.23*
	Bwth.	$D_0 = 100$	0.29	0.96	0.30	0.95	0.38	0.95	0.24	0.97	0.37	0.95
	Rect.	$n = 30$	0.20	1.00	0.21	1.00	0.23	1.00	0.19	1.00	0.32	1.00
FP-AL	Iden.	-	**0.12**	**0.96**	**0.13**	**0.96**	**0.14**	**0.96**	**0.12**	**0.97**	**0.19**	**0.96**
	Circle	$r = 100$	0.14	0.96	0.14	0.96	0.16	0.96	0.13	0.97	0.23	0.96
	Gaus.	$\sigma = 112$	*1.00*	*0.23*	*1.00*	*0.22*	*1.00*	*0.30*	*1.00*	*0.27*	*1.00*	*0.24*
	Bwth.	$D_0 = 100$	0.26	0.95	0.32	0.95	0.34	0.95	0.23	0.95	0.51	0.95
	Rect.	$n = 20$	0.15	0.96	0.15	0.95	0.17	0.96	**0.12**	**0.96**	0.23	0.96

(a) Original image


(b) Identity


(c) Circular


(d) Gaussian


(e) Butterworth

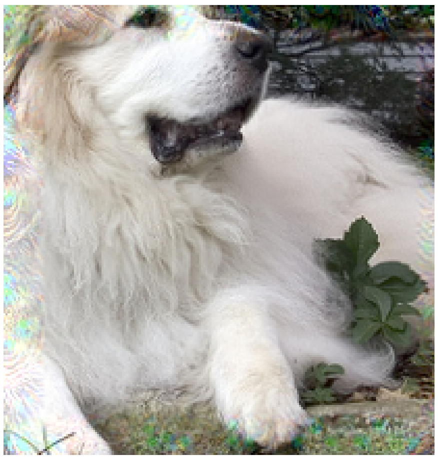
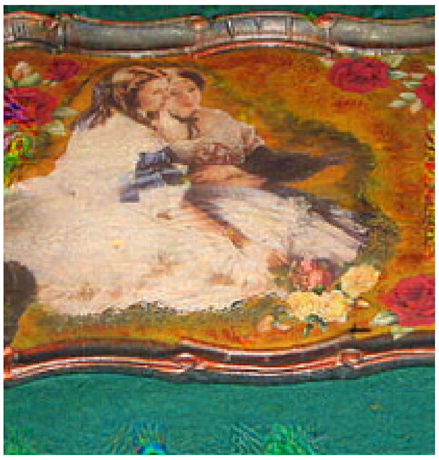
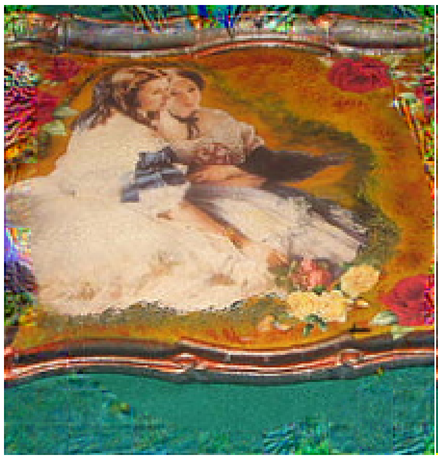
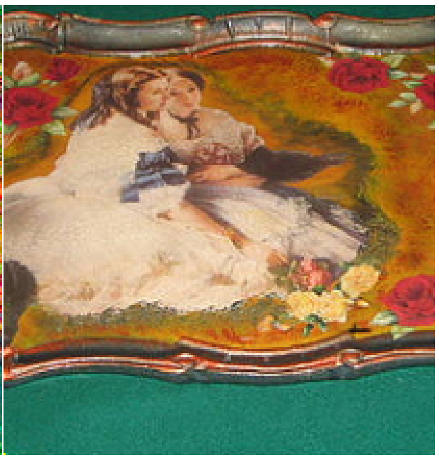

(f) Rectangular

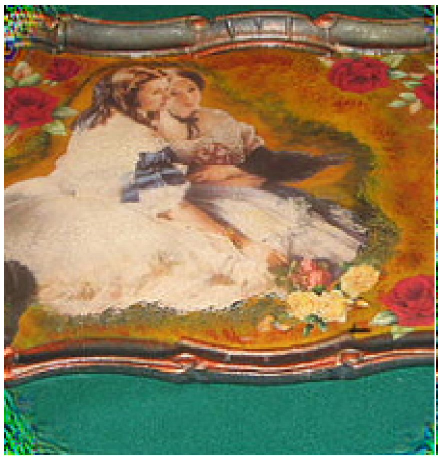
Figure 11: Attack results obtained by L_∞ FP-AL and indicated filter, against ResNet50. The original class of the image is "Great Pyrenees".

(a) Original image


(b) Identity


(c) Circular


(d) Gaussian


(e) Butterworth

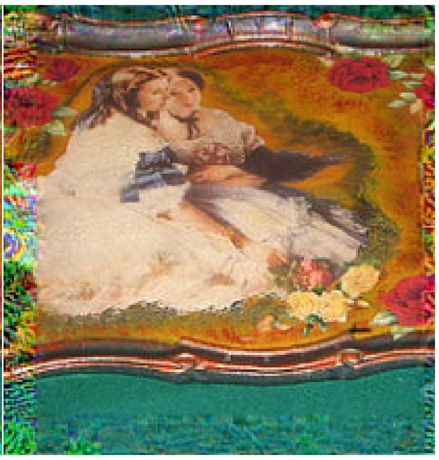

(f) Rectangular

Figure 12: Attack results obtained by L_∞ FEB-AL and indicated filter, against ResNet50. The original class of the image is "Great Pyrenees".

CHAPTER 5

DISCUSSION

The results given in the previous section are discussed in this section. As there are many results from several experiments, it is valuable that the quantitative data is interpreted in multiple ways, in order to extract conclusive information. The section starts by discussing the results of the standard unfiltered attacks, which includes comparisons between the results of both the same attack and other attacks, furthermore, the filtered attacks are discussed in the same manner. Finally, the visual interpretation of different attack results is given, as the visual aspect of the adversarial examples covers a crucial part of their usability.

5.1 Standard Attacks

L_2 P-AL attack shows significantly better results than any other attack under the given constraints. This attack consistently converges to the desired fooling rate, while B-AL never reaches and EB-AL oscillates around that level. A reason for these behaviors is that both B-AL and EB-AL are highly dependent on the batch size hyperparameter since both algorithms assume that a sufficiently large batch size can be a good approximation of the performance of the UAP against the actual population. Although a batch size of 32 was selected in the experimental settings, not all GPUs can handle such computational load, making these algorithms mostly unusable in practice. On the other hand, P-AL is not affected by the batch size, thanks to its altering condition that is only dependent on the cumulative performance against the dataset itself. Thanks to this altering scheme, the optimization becomes much more stable, and the convergence is achieved around the desired fooling rate while also yielding noticeably lower L_2 norms. Also, although the problems differ, P-AL outperforms UAP and F-UAP regarding the fooling rate with the same fixed norm.

It should be noted that the proposed attacks are more suitable for the L_2 norm counterpart, as optimizations done over L_∞ norms tend to be very unstable as the slightest deviation on L_∞ value creates a significant difference in the overall amount of perturbation. For instance, when the L_∞ norm is increased by 0.01, for a perturbation vector with a resolution of $224 \times 224 \times 3$, the total amount of perturbation (L_1 norm) can increase by 1,491.87, which can have considerable effects over the fooling rate. L_∞ P-AL attack is also the most optimal choice among the proposed algorithms since the fooling rates are controlled around the desired level while also yielding considerably smaller L_∞ norms. However, P-AL falls short under standard UAP attack settings against F-UAP, as this attack reaches higher fooling rates with a norm bound equal to that P-AL achieved during its minimization optimization. Overall, P-AL is not only the best choice as an attack whose fooling rate can be predefined but also can be a viable option under standard adversarial settings. Furthermore, EB-AL can be a better option to maximize the fooling rate, as seen in Table 2 and 3, it reaches higher fooling rates while also optimizing the norm. In the case of P-AL, when the target fooling rate is set to 100%, the loss will be adversarial in most iterations, maximizing the norm indefinitely. To be exact, if the attack is unsuccessful against a single image, for the rest of the optimization, the loss will strictly be altered to adversarial since the fooling rate will never reach 100% until the end of the epoch. Therefore, under standard adversarial settings, EB-AL is the more viable option.

5.2 Perturbation Features

Figure 13 shows UAPs trained with the proposed algorithms and the corresponding adversarial examples. It can be seen that more image-like features are accumulated around the edges, while the noisy features are generated around the center. It is hypothesised that the partial derivative of the cross-entropy loss with respect to the edge pixels becomes relatively high compared to the central pixels since the features of the target class are less likely to be found around the center; therefore, the central pixels contribute much less than the edge pixels which do not usually contain features of the original class. The correlation between the magnitude of the gradients and their distance from the center can be seen in Figure 8. This also implies that the image-like features accumulate around the edges because of the attacks being targeted rather than their usages of alternating loss schemes. This also supports the findings given in [9] that the UAPs fool the networks thanks to their image-like features, which are perceived by the networks as real images.

5.3 Filtering Operation

Incorporating different filtering operations into the UAP training was proven to be successful under L_2 attack settings. Furthermore, while P-AL was objectively the best performer amongst the proposed attacks, FP-AL consistently improved the results even further. Although there are several cases where the filtering operation increases the norm or marginally decreases the fooling rate, it can be argued that it is an effective addition to the alternating loss training procedure. Besides, the rectangular and circular filters are the best performing filters in terms of their consistent effects over the norm values. On the other hand, while the Butterworth filter has also been shown to be effective in preserving the fooling rate, it consistently increased the norm values; conversely, the Gaussian filter decreased the L_2 norm substantially while decreasing the fooling rates drastically up to the point that renders the UAP unusable. The commonality between these filters is that their values start from the middle as 0, and *progressively* increase up to 1, which makes the large portion of the filter have fractional values. It was mentioned that UAPs benefit from the image-like features of the perturbations. However, the fractional filter values damage the visual quality of the images as they fade away in those parts. This argument also explains the performance gap between the Butterworth filter and the Gaussian filter, as the Butterwork filter has a sharper transition from values closer to 0 to 1; the Gaussian filter is relatively smoother. Therefore, it can be argued that if the filtering operation is to be applied during AL UAP training, the circular and rectangular filters should be used.

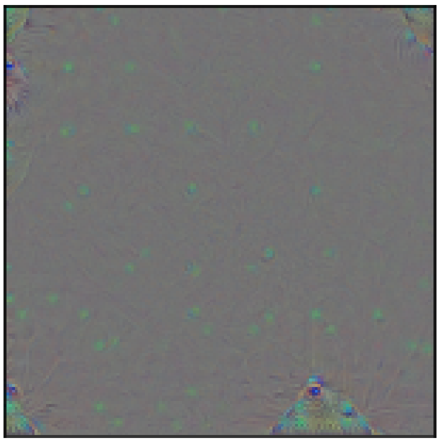
L_∞ attack results are highly different from the L_2 counterpart in that the filtering operation never improves the norm value while not changing the fooling rate performance either, which is expected as most of the attacks already reach 100% in the unfiltered settings. This situation is not surprising, as the filtering operation was originally proposed to eliminate the smaller artifacts around the center of the image; such filtering can decrease L_1 and L_2 norms but hypothetically cannot decrease L_∞ norm substantially, as it is highly probable that most of the high-value perturbations are generated around the edges. On the other hand, limiting the optimization to be done inside a smaller search space is likely to result in the occurence of perturbations that are higher in magnitude; it can also be argued that the filtering operation clears not only the central area but also *mitigates* those perturbations around the edges. Therefore the perturbation with the highest L_∞ norm contains adversarial energy from perturbations from both the center and edges. Hence, it can be concluded that the filtering operation is unsuitable for L_∞ norm AL-based UAP training.

5.4 Visual Interpretation

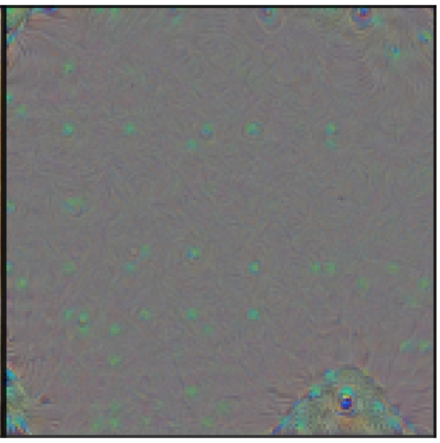
In this section, the visual properties of the generated UAPs will be discussed, as this plays a crucial role in the effectiveness of adversarial attacks in real-world applications. Ideally, an adversarial attack should be as imperceptible to the human eye as possible, which is mostly achieved with image-dependent attacks; however, as UAPs need to carry more visual information, their imperceptibility is a different and possibly harder goal to achieve. In the case of the proposed attacks, imperceptibility is not guaranteed, although mitigating the perturbations to the edges may positively affect their noticeability. When looking at Figure 6, each perturbation contains a similar structure in that the more image-like features are accumulated around the corners. However, all of them also contain small artifacts around the center, where the head of the dog is placed. The L_∞ counterpart of the previous figure is Figure 7, where a similar situation is observed. At this point, it is worth mentioning that the results of ViT are slightly different from the others in that the artifacts tend to form in small rectangular shapes. This is likely to happen because of the network's structure, where the input image is divided into 16-by-16 patches in the first layer, and these patches then go through a series of linear projections and attention mechanisms. The pixels inside a block may receive similar gradient values, which eventually makes the blocks' perturbations very similar. ViT is the only example where apparent image-like features generate around the center of the image.

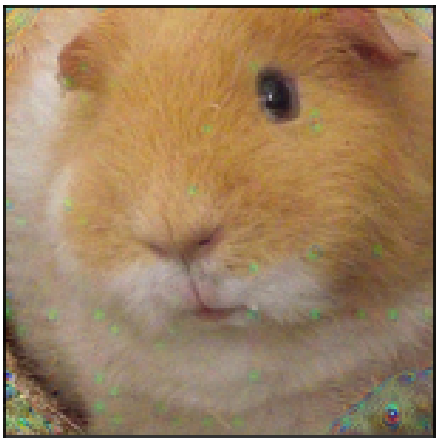
Figure 10 shows the results of the L_2 FP-AL attack, where it can be seen that the density of the perturbations increases as the filters get more restrictive. The difference between the Butterworth filter and the identity filter can be spotted by looking at the top left corners of the images. This phenomenon supports the proposition in the previous section about mitigating perturbations. Regardless of the argument, it can be seen that the green accumulations around the center of the standard P-AL attack are cleared in the filtered attacks, which can arguably decrease the noticeability of the perturbations. Figures 9 and 10 also show that circular and rectangular filters give different outputs in terms of the locations to where the perturbations are mitigated; the dense perturbations are accumulated around the corners when the circular filter is used, on the other hand, the accumulations occur around the edges when the filter of choice is rectangular. Although it is difficult to argue about the superiority of either of the filters, the fact that different filters mitigate the perturbations to different locations can be a factor when a choice needs to be made for the type of filtering operation to be applied on an attack.

(a) B-AL image


(b) B-AL UAP

(c) EB-AL image


(d) EB-AL UAP


(e) P-AL image

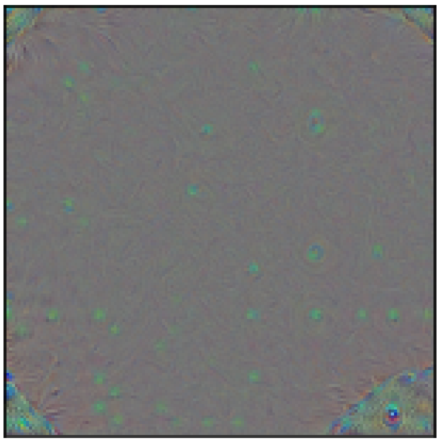

(f) P-AL UAP

Figure 13: UAPs trained with B-AL, EB-AL and P-AL as DenseNet121 being the target network, "peacock" being the target class, and the obtained adversarial examples whose original labels are "hamster".

CHAPTER 6

CONCLUSION

This thesis introduced the problem of parametrizing the fooling rate for UAPs and proposed 3 new UAP training algorithms that take advantage of the alternating loss scheme, where the loss function is changed during each iteration depending on the current state of the UAP. These algorithms were B-AL, where the performance against each batch is assumed to be a reasonable projection of the performance against the dataset; therefore, the norm loss is selected if the target fooling rate is reached; otherwise, the adversarial loss is backpropagated. This attack tends to converge to a fooling rate just below the target; to fix this issue, EB-AL solely selects the adversarial loss until the target fooling rate is achieved at the end of the epoch. Then, the same altering condition as B-AL is applied. Different from B-AL, this attack oscillates around the target fooling rate, with uncontrollably large changes. To overcome these drawbacks, P-AL alters the loss function based on the cumulative fooling rate up to each optimization point. It was also found that image-like features were accumulated around the edges of the UAP; therefore, a filtering operation that eliminates the perturbations around the center was proposed, using four well-known masks. The filtering operation was found to be only effective against L_2 attacks, and the best performing filter type was circular.

As future work, several methods that improve the visual quality of image-dependent adversarial attacks [32, 33] can be adapted to AL UAP training. Also, an effective scaling method can be developed to decrease the scaling problem between the adversarial and norm losses; when training a UAP using any of the L_2 attacks, the adversarial loss, which is cross-entropy, gets close to a very small value such as 0.1 in several epochs. On the other hand, the norm loss stays around 10 (as can be seen in Tables 2 and 4), which is 100 times higher than the adversarial loss. This creates a situation where the gradient descent step taken using the norm loss high enough to prevent the optimization to find a local minimum in the loss curve. Several scaling methods were tried in this thesis but a reliable method could not be found, therefore they were not included in the methodology section. These methods were annealed scale variables for the altered losses and varying learning rates. Furthermore, the filtering operation can also be improved and applied to other UAP training and generation algorithms, whether they are minimization or norm bounded attacks; this may improve the real-life usability of UAPs in AI model validation systems such as CAPTCHAs [12].

REFERENCES

[1] A. Kurakin, I. J. Goodfellow, and S. Bengio, “Adversarial examples in the physical world,” in *Artificial intelligence safety and security*, pp. 99–112, Chapman and Hall/CRC, 2018.

[2] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” *arXiv preprint arXiv:1706.06083*, 2017.

[3] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” *arXiv preprint arXiv:1412.6572*, 2014.

[4] H. Kannan, A. Kurakin, and I. Goodfellow, “Adversarial logit pairing,” *arXiv preprint arXiv:1803.06373*, 2018.

[5] H. Lee, S. Han, and J. Lee, “Generative adversarial trainer: Defense to adversarial perturbations with gan,” *arXiv preprint arXiv:1705.03387*, 2017.

[6] P. Samangouei, M. Kabkab, and R. Chellappa, “Defense-gan: Protecting classifiers against adversarial attacks using generative models,” *arXiv preprint arXiv:1805.06605*, 2018.

[7] A. E. Aydemir, A. Temizel, and T. T. Temizel, “The effects of jpeg and jpeg2000 compression on attacks using adversarial examples,” 2018.

[8] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, “Universal adversarial perturbations,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[9] C. Zhang, P. Benz, T. Imtiaz, and I. S. Kweon, “Understanding adversarial examples from the mutual influence of images and perturbations,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14521–14530, 2020.

[10] D. Sen, B. T. Karli, and A. Temizel, “Training universal adversarial perturbations with alternating loss functions,” in *The AAAI-22 Workshop on Adversarial Machine Learning and Beyond*, 2022.

[11] J. Su, D. V. Vargas, and K. Sakurai, “One pixel attack for fooling deep neural networks,” *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.

[12] C. Shi, X. Xu, S. Ji, K. Bu, J. Chen, R. Beyah, and T. Wang, “Adversarial captchas,” 2019.

[13] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[14] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2574–2582, 2016.

[15] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy)*, pp. 39–57, 2017.

[16] Z. Zhao, Z. Liu, and M. Larson, "Towards large yet imperceptible adversarial image perturbations with perceptual color distance," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[17] C. Xiao, J.-Y. Zhu, B. Li, W. He, M. Liu, and D. Song, "Spatially transformed adversarial examples," in *International Conference on Learning Representations*, 2018.

[18] K. R. Mopuri, U. Ojha, U. Garg, and R. V. Babu, "Nag: Network for adversary generation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 742–751, 2018.

[19] C. Zhang, P. Benz, A. Karjauv, and I. S. Kweon, "Universal adversarial perturbations through the lens of deep steganography: Towards a fourier perspective," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 3296–3304, 2021.

[20] K. R. Mopuri, U. Garg, and R. V. Babu, "Fast feature fool: A data independent approach to universal adversarial perturbations," *arXiv preprint arXiv:1707.05572*, 2017.

[21] P. Benz, C. Zhang, T. Imtiaz, and I. S. Kweon, "Double targeted universal adversarial perturbations," in *Proceedings of the Asian Conference on Computer Vision*, 2020.

[22] G. Sharma, W. Wu, and E. N. Dalal, "The ciede2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations," *Color Research & Application: Endorsed by Inter-Society Color Council*, vol. 30, no. 1, pp. 21–30, 2005.

[23] A. Aydin, D. Sen, B. T. Karli, O. Hanoglu, and A. Temizel, *Imperceptible Adversarial Examples by Spatial Chroma-Shift*, p. 8–14. New York, NY, USA: Association for Computing Machinery, 2021.

[24] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[25] S. Butterworth *et al.*, "On the theory of filter amplifiers," *Wireless Engineer*, vol. 7, no. 6, pp. 536–541, 1930.

[26] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[27] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[28] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[29] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[30] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[31] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv preprint arXiv:2010.11929*, 2020.

[32] B. T. Karli, D. Sen, and A. Temizel, “Improving perceptual quality of adversarial images using perceptual distance minimization and normalized variance weighting,” in *The AAAI-22 Workshop on Adversarial Machine Learning and Beyond*, 2022.

[33] B. Aksoy and A. Temizel, “Attack type agnostic perceptual enhancement of adversarial images,” *International Workshop on Adversarial Machine Learning And Security (AMLAS), IEEE World Congress on Computational Intelligence (IEEE WCCI)*, 2020.