PhD-FSTM-2023-060
The Faculty of Sciences, Technology and Medecine

Dissertation

Defence held on the 21st of June 2023 in Luxembourg

to obtain the degree of

Docteur de l'Université du Luxembourg en Informatique

by

Martin Gubri
Born on the 21st of October 1991 in Bordeaux (France)

What Matters in Model Training to Transfer Adversarial Examples

Dissertation defence committee

Dr. Yves Le Traon, Dissertation Supervisor
Professor, University of Luxembourg

Dr. Michail Papadakis, Chairman
Associate professor, University of Luxembourg

Dr. Maxime Cordy, Vice-Chairman
Research scientist, University of Luxembourg

Dr. Seong Joon Oh
Professor, University of Tübingen

Dr. Florian Tramèr
Assistant professor, ETH Zürich

Abstract

Despite state-of-the-art performance on natural data, Deep Neural Networks (DNNs) are highly vulnerable to adversarial examples, i.e., imperceptible, carefully crafted perturbations of inputs applied at test time. Adversarial examples can transfer: an adversarial example against one model is likely to be adversarial against another independently trained model. This dissertation investigates the characteristics of the surrogate weight space that lead to the transferability of adversarial examples. Our research covers three complementary aspects of the weight space exploration: the multimodal exploration to obtain multiple models from different vicinities, the local exploration to obtain multiple models in the same vicinity, and the point selection to obtain a single transferable representation.

First, from a probabilistic perspective, we argue that transferability is fundamentally related to uncertainty. The unknown weights of the target DNN can be treated as random variables. Under a specified threat model, deep ensemble can produce a surrogate by sampling from the distribution of the target model. Unfortunately, deep ensembles are computationally expensive. We propose an efficient alternative by approximately sampling surrogate models from the posterior distribution using cSGLD, a state-of-the-art Bayesian deep learning technique. Our extensive experiments show that our approach improves and complements four attacks, three transferability techniques, and five more training methods significantly on ImageNet, CIFAR-10, and MNIST (up to 83.2 percentage points), while reducing training computations from 11.6 to 2.4 exaflops compared to deep ensemble on ImageNet.

Second, we propose transferability from Large Geometric Vicinity (LGV), a new technique based on the local exploration of the weight space. LGV starts from a pretrained model and collects multiple weights in a few additional training epochs with a constant and high learning rate. LGV exploits two geometric properties that we relate to transferability. First, we show that LGV explores a flatter region of the weight space and generates flatter adversarial examples in the input space. We present the surrogate-target misalignment hypothesis to explain why flatness could increase transferability. Second, we show that the LGV weights span a dense weight subspace whose geometry is intrinsically connected to transferability. Through

extensive experiments, we show that LGV alone outperforms all (combinations of) four established transferability techniques by 1.8 to 59.9 percentage points.

Third, we investigate how to train a transferable representation, that is, a single model for transferability. First, we refute a common hypothesis from previous research to explain why early stopping improves transferability. We then establish links between transferability and the exploration dynamics of the weight space, in which early stopping has an inherent effect. More precisely, we observe that transferability peaks when the learning rate decays, which is also the time at which the sharpness of the loss significantly drops. This leads us to propose RFN, a new approach to transferability that minimises the sharpness of the loss during training. We show that by searching for large flat neighbourhoods, RFN always improves over early stopping (by up to 47 points of success rate) and is competitive to (if not better than) strong state-of-the-art baselines.

Overall, our three complementary techniques provide an extensive and practical method to obtain highly transferable adversarial examples from the multimodal and local exploration of flatter vicinities in the weight space. Our probabilistic and geometric approaches demonstrate that the way to train the surrogate model has been overlooked, although both the training noise and the flatness of the loss landscape are important elements of transfer-based attacks.

Acknowledgments

I am immensely grateful to my supervisor, Prof. Dr. Yves Le Traon, for trusting me to pursue the adventure of the PhD degree, and for his guidance and persistent support. His insights and words of encouragement have often inspired me and set the foundation for this thesis.

I am equally indebted to my co-supervisors, Dr. Maxime Cordy and Prof. Dr. Mike Papadakis, for their advice and constructive feedback. Their expertise and dedication have shaped my research path, and for that, I am truly thankful.

I also would like to express my gratitude to Prof. Dr. Seong Joon Oh and Prof. Dr. Florian Tramèr, the external jury members, whose time, effort, and meticulous review of my work, despite their demanding schedules, have been of immense value. I am thankful for their commitment and engagement in the advancement of my research.

I am thankful to the National Research Fund (FNR) for its financial support. Its funding has been instrumental in facilitating this research.

On a personal note, I extend my greatest thanks to my partner, Rocío Consales, for her infinite patience, understanding and love, especially during the long evenings I spent at the office. Her unwavering faith in my capabilities has been my source of strength and motivation.

I also want to thank my family for their unending love and support throughout my entire educational journey. Their belief in my potential and their constant encouragement have been the pillars of my perseverance. Above all, I want to acknowledge my mother, whose support and encouragement have been continuous and strong. She has been there for me in every way possible, ensuring that I had everything I needed to navigate this educational and academic journey successfully. For her love, her support, and her belief in me, I am forever grateful.

The deepest thanks goes out to my colleagues at the SerVal laboratory of the University of Luxembourg. Their camaraderie and intellectual exchange have created an environment conducive to high-quality research and have made my time here both rewarding and enjoyable. In particular, my sincerest gratitude goes out to Dr. Salah Ghamizi for our enlightening conversations about adversarial machine learning, and to Dr. Renaud Rwemalika for our conversations about research in

general.

I wish to acknowledge Prof. Dr. Christine Thomas-Agnan, Prof. Dr. Anne Ruiz-Gazen, and Dr. Xavier Gendre for their captivating classes, for their support to pursue a PhD degree, and for instilling in me the passion for statistics. Their trust in me and their ability to share their enthusiasm have had a profound influence on my academic path.

Finally, I owe my initial taste for research to Bernard Privat, Martine Loustau, and Prof. Dr. Éric Sopena. The "Math en Jeans" workgroup during high school ignited my interest in research and paved the way for my academic career.

Contents

1

Introduction

This chapter presents the context and challenges of this dissertation. First, we introduce the field of adversarial machine learning and the topic of the transferability of adversarial examples. Second, we present the challenges and open directions arising from the state of knowledge on the transferability of adversarial examples. Finally, we outline the contributions of this dissertation.

Contents

1.1 Context

This dissertation focuses on the transferability of adversarial examples. Adversarial examples, also known as evasion attacks, are the subfield of adversarial machine learning dedicated to the worst-case distributional shift. This security threat aims to find small perturbations in inputs that lead to the largest changes in the output of a machine learning model, at inference time.

1.1.1 Deep Learning: Achievements and Critical Failures

Deep learning has made tremendous progress on natural data over the last decade. However, the state-of-the-art Deep Neural Network (DNN)s fail critically against white-box adversarial examples. Simple projected gradient ascent in the input space is generally enough to decrease performance below random baseline accuracy.

The prolific field of deep learning. Deep learning is a fast pace research field which made tremendous progress in a variety of tasks from numerous fields, including computer vision, natural language processing, bioinformatics, recommender systems, software engineering, among others. The availability of large-scale computing and data had allowed to train larger and more powerful DNNs. Researchers have made large progress on benchmark datasets over the last decade at a fast rate. In particular, DNN is the default solution for many complex computer vision tasks [DKA+19], since explicitly programming rule-based systems would be too cumbersome. Research on deep learning has a vast impact, as numerous services rely on DNNs to analyse and edit images, detect spam and frauds, implement surveillance systems, and recommand content. The recent popularity among the public of large language models is the most striking example of how broad the impacted public is.

Trustworthiness issues. At the same time, DNN can output biased or unintended predictions. Performing a complete audit of trained DNNs is impossible, as interpreting weight values is meaningless to humans. Manual testing and auditing is tedious at best, and often impossible. Often, these blind spots cause trustworthiness issues that damage people or institutions. The prevalence of these unexpected behaviours may be reinforced by the shortness of development cycles. Service providers might rush to deploy models without enough time to properly audit and test them, due to fierce competition between service providers.

Adversarial examples. Despite their success on natural data, state-of-the-art models critically fail on worst-case distributional shift. A common pitfall of these models is that they are highly vulnerable to adversarial examples, i.e., misclassified examples that result from adding tiny and imperceptible carefully crafted perturbations to a well-classified example at test time [BCM+13; SZS+13].

The previously stated success of DNN clashes with the ease of finding adversarial examples. Under a white-box threat model where the model is fully known, an attack generally consists of a simple gradient ascent in the input space projected in a p-norm ball [KGB17; MMS+18]. The small radius of these balls ensure the imperceptibility of the perturbations. Despite the imperceptibility of perturbations added to test examples and the simplicity of the attack algorithm, the accuracy of state-of-the-art models often drops to zero percent [BCM+13; SZS+13; KGB17]. Here lies a key difference with classical distribution shift, such as random noise. Under classical distribution shift, the model accuracy generally remains above the random baseline. However, DNNs critically fails under worst-case distributional shift.

The existence of adversarial examples can be interpreted positively or negatively, depending on the application context. Adversarial examples may constitute a critical security threat if malicious actors exploit this property to enforce some desired outcome. Individuals or organised groups could use these vulnerabilities to bypass undesirable content filters, control the behaviours of autonomous cars, or artificially spread information with recommender systems. Nevertheless, adversarial examples are also a practical opportunity to defend against excessive, illegitimate, or non-consensual uses of deep learning. For example, these vulnerabilities might be a practical way to bypass massive surveillance systems that put civil and political rights at risk.

1.1.2 The Transferability of Adversarial Examples

An adversarial example against one model is likely to be adversarial against another one. This phenomenon is called *transferability* and has two influential outcomes. First, it provides insights about the representations learnt by DNNs, since these errors are not random learning artefacts. Second, it allows transfer-based black-box attacks to fool unknown models without querying them.

Adversarial examples can generalize. Adversarial examples can transfer: an example adversarial against one model is likely to also be adversarial against another independently trained model (of the same or different class of hypothesis). This phenomenon has some implications about the nature and the causes of adversarial examples. Concomitantly with the discovery of adversarial examples, Szegedy et al. [SZS+13] observe their transferability. In particular, a perturbation computed against one DNN can fool another DNN trained on a disjoint subset of training examples. Then, *adversarial examples are not random artefacts of learning*, since the same perturbation can fool two independently trained models. Transferability was also observed across neural network architectures. Therefore, adversarial examples are not specific to one family of architectures. Likewise, cross-technique transferability was observed between classical machine learning models.

For example, despite very different hypotheses about data distribution, adversarial examples against logistic regression transfer well to a decision tree [PMG16]. Therefore, adversarial examples can generalize at multiple levels of the machine learning pipeline: across independent training, across datasets, across neural network architectures, and across modelling assumptions. Studying transferability in deep learning helps us better understand how DNNs learn their representation.

From white-box to black-box attacks. The second influential implication of the transferability of adversarial examples is the possibility of attacking unknown models without any queries. Adversarial attacks have been designed primarily in white-box settings, where the attacker is assumed to have complete knowledge of the target DNN (including its weights). While studying such worst-case scenarios is essential for proper security assessment, in practice the attacker should have limited knowledge of the target model. In such a case, the adversarial attack is applied to a *surrogate model*, with the hope that the crafted adversarial examples transfer to (i.e., are also misclassified by) the target DNN. Transferability demonstrates that security by obscurity is not a robust defence. Indeed, keeping information secret and severely restricting the number of queries cannot prevent an attacker from training his or her own model to be used as a substitute. Transfer-based black-box attacks have some benefits and drawbacks. Transfer-based attacks are applicable where query-based attacks are not. For some applications, getting feedback from a model is costly. API accesses might be heavily restricted. And if the target model is not accessible through a software interface, one may have to go by a physical support. Physical attacks require printing the current adversarial example, manual on-site setup, and digitalisation, to get a single feedback. This process is too tedious and costly for query-based attacks that generally need several thousands of feedbacks, or numerous hundreds at best. Therefore, transferability permits attacks that would not be possible otherwise. However, transfer-based attacks need to train at least one surrogate model, which requires both computation capabilities and training data. Given the scale of modern DNNs, the computation cost of training a surrogate model exceeds by several orders of magnitude the computation cost of applying the attack once this model is trained.

A popular topic. The transferability of adversarial examples is a popular research topic. Figure 1.1 shows the number of newly published articles on transferability listed in the Scopus database and the number of new articles submitted on the arXiv platform, on a given year. The first papers containing the now established lexicon appeared in 2016 on arXiv and in 2017 on Scopus. The number of published articles grew rapidly with an annual growth rate of 33.85%. The year 2022 was particularly prolific: 184 articles were published in total and 8.33 preprints were submitted to arXiv per month on average. The topic of transferable adversarial examples is still a young field, since the published articles are only 1.95 years old

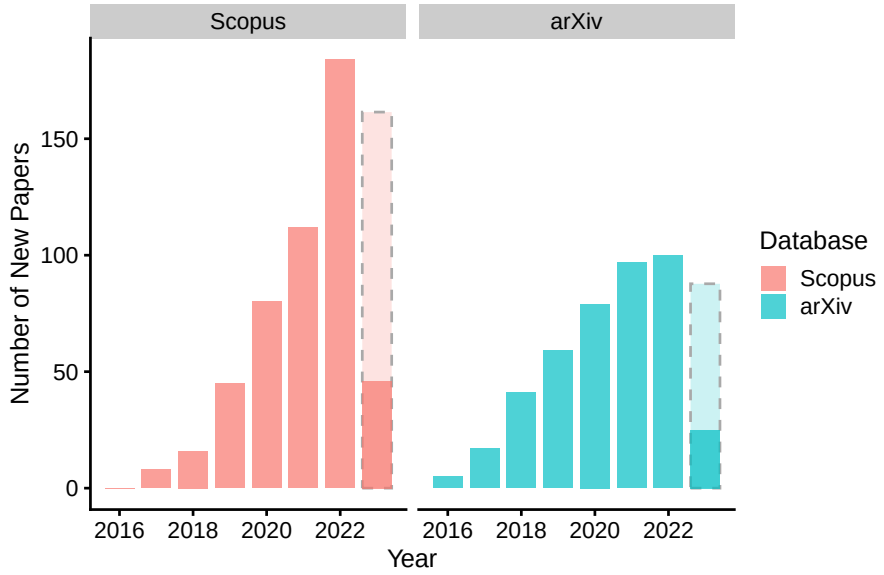


Figure 1.1: Number of published (Scopus) and submitted (arXiv) articles on the transferability of adversarial examples per year. Each bar represents the number of *new* papers on a given year, i.e., this figure is not a cumulative histogram. Papers containing the word "transferability" or "transferable" and either "adversarial example" or "evasion attack" in the title, abstract or keywords, until the 15$^{\text{th}}$ of April 2023. The dashed bars are the linear projection for the year 2023.

on average (on the 15$^{\text{th}}$ of April 2023). Nevertheless, numerous contributions exist. The Scopus database draws up a list of 491 papers on the topic, written by 1,141 authors. Therefore, the subject of the transferability of adversarial examples has gained a lot of attention in the field, and is now a young established topic.

Overall, the transferability of adversarial examples is of importance for both understanding better DNNs and leveraging black-box attacks. The community recognized this role, as demonstrated by the large amount of published papers on the topic over the last half decade.

1.2 Challenges of Adversarial Examples Transferability

Despite, numerous contributions about the transferability of adversarial examples, several challenges and blind-spots persist in this specialized literature. We identify five main challenges to address:

C1. The *knowledge gap* in the scientific literature about training the surrogate

DNNs of transfer-based black-box attacks.

C2. The *low success rate* of existing techniques with small perturbations.

C3. The *high training computational cost* of transfer-based attacks and of the few existing training techniques for transferability.

C4. The *lack of insights* about the transferability of adversarial examples.

C5. The *excessive specificity* of some transferability techniques.

1.2.1 C1. Knowledge Gap on Surrogate Model Training

Despite numerous publications about transferability, only a handful of articles studies how to train DNNs to improve the transferability of their adversarial examples. Most related work boosts the transferability of gradient-based attacks with data augmentation [WZT+18; XZZ+19; DPS+19; LSH+20; WH21; WHW+21], model augmentation [LBZ+18; WWX+20; WWX+20; GLC20], improved attack loss [ZHC+18; HKG+19; WRL+21], or improved attack optimizer [DLP+18; LSH+20; WH21; QFL+22]. To the best of our knowledge, a single article before our first work tackles how to improve transferability during the training phase: Liu et al. [LCL+17] show that an ensemble of architectures is a better surrogate model than a single architecture. They retrieve one standardly trained model for several architectures. We believe that there were *no prior contributions about how to train each architecture* for transferability. Prior work simply uses standard training methods to obtain a surrogate model. The exploration of the weight space of one architecture was in the state of the simplest baseline. We discuss, in Chapter 3, some parallel and posterior work of ours that studies the relation between training and transferability [SMK21; BZK21; YZJ+21; LGZ+23]. Overall, despite numerous publications, very little (to none) has tackled how to train a surrogate model for transferability, and study how the training dynamics, characteristics, and objective of the surrogate model impact transferability.

1.2.2 C2. Low Success Rates of Small Perturbations

Numerous papers about transferability use large perturbations that may be visible. Kurakin et al. [KGB17], Dong et al. [DLP+18], Zhou et al. [ZHC+18], Huang et al. [HKG+19], Dong et al. [DPS+19], Lin et al. [LSH+20], Wu et al. [WWX+20], Guo et al. [GLC20], Wang and He [WH21], Wang et al. [WRL+21], Naseer et al. [NKH+21], and Qin et al. [QFL+22] set the maximum L_∞ norm of the adversarial perturbations to 16/255, and Wu et al. [WZT+18] and Xie et al. [XZZ+19] to 15/255, on ImageNet. These sizes of perturbation are large for the adversarial machine learning literature. Zhao et al. [ZZL+22] find that using a L_∞ norm value of 16/255 on ImageNet sacrifices imperceptibility. Figure 1.2 shows an

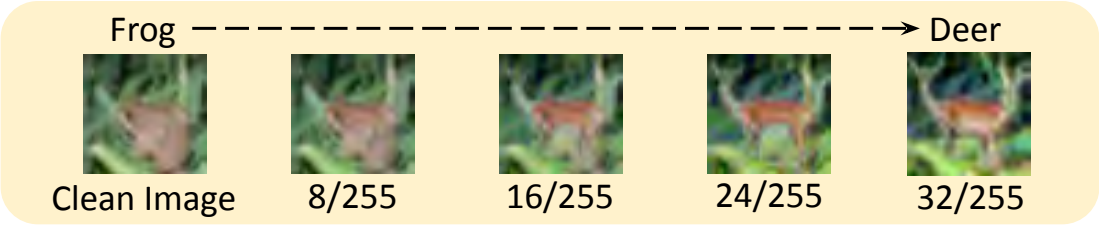


Figure 1.2: Large perturbations may be visible and can even change the label. Illustration from Addepalli et al. [AJS+22] of an original image of a frog altered to the deer class with increasing ε, the L_∞ norm of the perturbation. The frog looks partially like a deer at $\varepsilon = 16/255$, which is the norm widely used to evaluate transferability.

example of an adversarial perturbation on CIFAR-10 with a L_∞ norm of 16/255 that is not only visible to the human eye, but that partially changes the correct class of the image[1]. Reducing the perturbation norm would not only make the perturbation imperceptible, but would also reduce the success rate to a large extent. The success rate of an attack is an increasing function of the maximum size of the perturbations [CAP+19]. In our preliminary experiments, we found that existing transferability techniques achieve a low success rate with perturbations smaller than 16/255. Therefore, imperceptible black-box attacks are challenging.

1.2.3 C3. High Training Cost of the Surrogate

A major challenge of transfer-based black-box attacks is the cost of training the surrogate model. Query-based black-box attacks generally require little computation (at the expense of heavy access to the target model). Transfer-based black-box attacks need first to train a surrogate model and then attack it. To be competitive in cases where both types of attacks can be applied, transfer-based black-box attacks have to be efficient. We argue that transfer-based attack is expensive due to its first requirement, i.e., training one or several models. For example, a standard model, trained on ImageNet for 130 epochs, would need 166,551,710 full forward and backward passes. Whereas, applying a typical gradient-based attack on one image needs only between 5 and 50 full forward and backward passes, which would total to 250,000–2,500,000 passes to attack the entire test set. *Training the surrogate model is at least two orders of magnitude more costly than attacking it.* Despite this highly unbalanced cost, no previous work tackles how to train efficiently the surrogate model. Then, cheap transfer-based black-box attacks are an open challenge.

[1]The values of maximum L_p norm of the adversarial perturbations are not directly comparable between datasets. Figure 1.2 serves here as an illustration of large perturbations.

1.2.4 C4. Lack of Insights About Transferability

Numerous work in the field simply propose a new technique that improves the attack success rate, without providing more in-depth knowledge about adversarial examples or their transferability. The impact of this type of contribution is therefore significantly limited, since the reader is left with speculations, or hypotheses at best, about why the proposed method can fool a diverse set of architectures or DNNs. The lack of evidence-based insights tends to reduce the field to a *collection of scattered techniques, with limited scientific value*. We think that there is a substantial discrepancy between the potential insights that could be gained from the study of transferability and the current state of the field. This represents a missed opportunity for researchers to better understand the representations learnt by DNNs, and in particular why adversarial examples are not simple isolated mistakes of specific representations.

1.2.5 C5. Over-Specialized Transferability Techniques

Numerous techniques designed to increase transferability are specific to particular technical settings, and cannot be applied generally. Many model augmentation techniques are based on a specific component of DNN architectures. In particular, techniques that rely on skip connections [LBZ+18; WWX+20; GLC20] cannot be applied on non-ResNet Convolutional Neural Network (CNN)s or on Vision Transformer (ViT) architectures. The same point holds for techniques designed for ViT [NRK+22]. The techniques based on data augmentation are limited to image datasets. For example, Input Diversity (DI) [XZZ+19] performs random resize and padding. Similarly, Zhou et al. [ZHC+18], Dong et al. [DPS+19], Lin et al. [LSH+20], Wang et al. [WHW+21], and Wang and He [WH21] propose transformations designed for images. New distinct techniques would need to be defined to improve the transferability in, for example, natural language processing tasks. In general, the current literature tends to be overspecific and lacks a way to exploit generic principles in deep learning that can be applied to diverse DNN architectures and datasets.

1.3 Overview of the Contributions

This section presents the contributions of this dissertation to address the aforementioned challenges on the transferability of adversarial examples. Figure 1.3 provides an overall view of these contributions. It outlines and relates all together the training techniques, the concepts and the characteristics of the weight space used in this dissertation. The rest of this section provides a step-by-step description of the elements of Figure 1.3.

1.3.1 Scope and Goal

To tackle the five objectives described in the previous section, this dissertation focuses on **how to train effective surrogate models** for black-box evasion attacks against DNNs, and investigates extensively the **characteristics of the weight space** that lead to the transferability of adversarial examples. The weight space is formally defined as the Euclidean space, where each dimension corresponds to one weight of a specified architecture. As deep neural networks have numerous weights, the weight space of any modern architecture is high-dimensional. The core of this dissertation relies on obtaining transferability from the weight space of the surrogate DNNs (thick arrow, large red and golden boxes in Figure 1.3).

1.3.2 General Outline: What Matters When Exploring the Weight Space

This dissertation provides strong evidence that training techniques are key to obtaining highly transferable adversarial examples. Our three main complementary techniques provide an extensive and practical method to obtain **highly transferable adversarial examples from the multimodal and local exploration of flatter vicinities in the weight space**. Our probabilistic and geometric analyses demonstrate that the way to train the surrogate model has been overlooked, although both the *training noise* and the *flatness of the loss landscape* are important elements of transfer-based attacks.

Using the view of the loss landscape developed by Choromanska et al. [CHM+14] and Goodfellow and Vinyals [GV14], the weight space admits numerous regions of low loss. We call a vicinity each of these regions. The training of a DNN is said to stay in the same vicinity if the loss during the training process stays relatively low. The probability distribution of obtaining a specific weight from a stochastic training process, such as Stochastic Gradient Descent (SGD), is highly multimodal. Similarly, the Bayesian posterior over weights is also highly multimodal. In the entire dissertation, we will use interchangeably the mode of the probability distribution of obtaining a weight, and the vicinity in the weight space, i.e., a region of the weight space with low loss.

Weight space exploration. To achieve the goal of improving the transferability of adversarial examples from the weight space of the surrogate models, this dissertation extensively covers the ways of exploring the weight space (red boxes in Figure 1.3):

- The *multimodal exploration*, where the surrogate is an ensemble of multiple models from different vicinities of the weight space, corresponding to diverse representations of the data. We would like to emphasize that the multimodal exploration refers here to the exploration of multiple vicinities, or equivalently multiple modes of a probability distribution, and not to models that process

multiple modalities, such as images and texts, for example.

- The *local exploration*, where the surrogate is an ensemble of multiple models from the same vicinity of the weight space, corresponding to more similar variations of the representation.
- The *point selection*, where the surrogate is a single model, i.e., one point of the weight space, that corresponds to the search for a transferable representation.

These three types of exploration are *complementary*. Several DNNs can be obtained from different vicinities by independently training models from different random initializations (deep ensemble). Exploring different vicinities is important, since each local minima of the loss typically characterises a meaningfully different representation of the data [GIP+18; FHL19; ZLZ+20]. Each independent training can use point selection techniques to collect improved models from different vicinities. Local exploration techniques can be applied multiple times in parallel, starting from each of these models, to explore each vicinity. In the end, these three types of exploration permit to obtain a surrogate composed of better individual representations from different vicinities which are well characterised locally.

Insights from the training noise and the loss flatness. The primary goal of this dissertation is to provide novel insights, which represents a significant portion of our research endeavour. This dissertation extensively analyses the proposed technique to provide in-depth knowledge of why these techniques are able to train better surrogate models. Through probabilistic and geometrical analysis, we show the key role of training noise and loss flatness to obtain transferability from the weight space (green boxes in Figure 1.3). The probabilistic approach of Chapter 4 considers the unknown weights of the target model as random variables. Under a specific threat model, the training noise of the target model defines a probability distribution of this model (arrow from "Training Noise" to "Distribution of Target Model" in Figure 1.3). One can sample from the distribution of the target model to obtain a surrogate. Sampling may be local, that is, in the same mode, or in different modes (arrows from "Distribution of Target Model" to red boxes in Figure 1.3).

Additionally, the training of one DNN happens in a small subspace of the weight space [GRD18]. Thus, the training noise of the surrogate model defines a weight subspace (arrows from "Training Noise" to "Subspace Geometry", and from "Subspace Geometry" to "Weight Space" in Figure 1.3). Our geometric analysis empirically shows that the geometry of this weight subspace is specific, i.e., unlike a random subspace, and that our local exploration of the weight space improves transferability thanks to the specificity of this geometry (arrow from "Subspace Geometry" to "Local Exploration" in Figure 1.3).

In addition, the local exploration and point selection techniques that improve transferability flatten the loss landscape (arrows to "Loss Flatness" in Figure 1.3). The natural loss changes slowly along one direction of the weight space, when

moving away from one of these good surrogates. To explain the relation between the flatness of the surrogate loss and transferability, we propose *the surrogate-target misalignment hypothesis*: if the surrogate loss is shifted with respect to the target loss in the input space, wide adversarial examples are desirable to keep the difference between the surrogate and the target losses small (arrow from "Loss Flatness" to "Weight Space" in Figure 1.3).

Overall, our probabilistic and geometrical insights show the key role of both training noise and loss flatness to explain why our surrogate models generate highly transferable adversarial examples.

1.3.3 Detailed Contributions per Chapter

This dissertation extensively studies how to leverage training techniques to obtain effective surrogate models for black-box attacks. We propose and analyse in-depth new techniques to better understand the relationship between the surrogate weight space and the transferability of the crafted adversarial examples. First, we identify five challenges in the scientific literature about transferability, including the blind spot on how to train the surrogate model. To address these challenges, we explore three complementary ways to explore the surrogate weight space: the multimodal exploration to obtain multiple models from different vicinities, the local exploration to obtain multiple models in the same vicinity, and the point selection to obtain a single transferable representation. Our combined techniques show how to obtain a surrogate composed of better individual representations from different vicinities, which are well characterised locally.

Transferability from deep ensemble and Bayesian neural networks (Chapter 4). Chapter 4 relates uncertainty to transferability. We develop a *probabilistic* analysis of transferability that shows that what matters is to *sample from the distribution of the unknown target model arising from the training noise*. As the weights of the target DNN are unknown, they can be treated as random variables. Under a specified threat model, the randomness of the target DNN comes from the training noise of SGD. Deep ensemble, i.e., the ensemble of independently trained DNNs, samples from weights from the distribution of the target model (arrow from "Deep Ensemble" to "Distribution of Target Model" in Figure 1.3). We experimentally and extensively show that deep ensemble generates effective surrogate models. Unfortunately, deep ensemble is computationally expensive since an additional sample requires a full independent training. Based on recent work showing that deep ensemble approximates well the Bayesian posterior [MVS+20], we propose an efficient alternative by sampling the surrogate weights from the posterior distribution using Cyclical Stochastic Gradient Langevin Dynamics (cSGLD), a state-of-the-art Bayesian deep learning technique (arrow from "cSGLD" to "Distribution of Target Model" in Figure 1.3). Our extensive experiments on ImageNet,

CIFAR-10 and MNIST show that our approach improves the success rates of four state-of-the-art attacks significantly (up to 83.2 percentage points), in both intra-architecture and inter-architecture transferability. On ImageNet, our approach can reach 94% of success rate while reducing training computations from 11.6 to 2.4 exaflops, compared to deep ensemble. Our vanilla surrogate achieves 87.5% of the time higher transferability than three transferability techniques designed for this purpose. We evaluate seven training methods in total to train a surrogate model. This contribution was the first to investigate how to generate a surrogate from the weight space of a single architecture. Finally, we notice a shortfall of both cSGLD and deep ensemble in sampling inside the same mode (dotted arrow from "Distribution of Target Model" to "Local Exploration"): Section 4.4.5 reveals that cSGLD poorly characterise the local variations in the same mode of the weight space, and deep ensemble obtains a single sample from the mode sampled in each independent training. Section 4.4.6 identifies a promising direction to address this shortfall using a training technique that builds an ensemble from fine-tuning a trained DNN. Our next contribution follows this direction.

Transferability from large geometric vicinity (Chapter 5). Chapter 5 develops a *geometric* analysis to study the transferability from the local exploration of the vicinity around a trained surrogate DNN. First, Chapter 5 establishes the relevance of the local exploration of the weight space for transferability by showing that random directions in the weight space improve transferability, whereas random directions in the input space do not. Both noises differ in the structure of the covariance matrix of the induced Gaussian distribution of input gradients. Second, to improve over the random directions in the weight space, we propose transferability from Large Geometric Vicinity (LGV) to augment a trained surrogate DNN with training noise. LGV starts from a pretrained model and collects multiple weights in a few additional training epochs with a constant and high learning rate. Through extensive experiments, we show that LGV alone outperforms all (combinations of) four established transferability techniques by 1.8 to 59.9 percentage points. We carefully study the hyperparameters of LGV to describe the appropriate type of high learning rate and propose easy solutions to improve the memory and computational efficiencies if needed. We show that the local weight space exploration of LGV complements nicely the multimodal exploration of deep ensemble proposed in Chapter 4. We develop a geometric analysis to explain why LGV increases transferability. We establish that what matters to LGV is the *flatness of the loss* and the *weight subspace obtained from the training noise* (both arrows from "LGV"). First, we show that LGV explores a flatter region of the weight space using four complementary experiments. As a result, we observe that LGV produces flatter adversarial examples in the input space. We present the surrogate-target misalignment hypothesis to explain why flatness

could increase transferability: if the surrogate loss is shifted with respect to the target loss in the input space, wide adversarial examples are desirable to keep the difference between the surrogate and the target losses small. Then, we observe that the LGV weights do not succeed on their own, indicating a missing piece to understand LGV. The second key element is the weight subspace spanned by LGV weights. Our experiments on this topic have three folds. First, we establish that this weight subspace is specific to transferability, i.e., it produces better surrogates than random subspaces, and that it is densely composed of good surrogates, i.e., random sampling inside this subspace produces good surrogates. Second, we show that this subspace is composed of directions whose relative importance depends on the functional similarity between surrogate and target, i.e., the geometry of the subspace is relevant to transferability. Third, we show that this subspace captures generic geometrical properties, in the sense that the subspace shifted to solution from other vicinities still produces good surrogates. While LGV addresses how to augment a base model, our base model is regularly trained. And, our best single flat model "LGV-SWA" is not as competitive as LGV (dotted arrow from "LGV-SWA" to "Point Selection"). Therefore, training a single base model for transferability, i.e, a transferable representation, is an open challenge that requires further investigation.

Transferability from representation in flat neighbourhood (Chapter 6). Chapter 6 develops a *geometric* analysis of the weight space, and shows that the *flatness of the loss landscape* matters to obtain a single surrogate model, i.e., a transferable representation. First, we refute a common hypothesis from previous research to explain why early stopping improves transferability. Previous work proposes the hypothesis that DNN first learns robust features and then non-robust features, explaining why early stopped models are better surrogates, since non-robust features are brittle. However, we provide evidence that tends to refute this hypothesis, showing that early stopping improves transferability from and to non-robust features. We hypothesize that robust and non-robust features are learnt conjointly during training, since the transferability of adversarial examples can provide insights on the closeness of two learnt representations. We propose an alternative explanation to the success of early stopping by establishing links between transferability and the exploration dynamics of the weight space, in which early stopping has an inherent effect. More precisely, we observe that transferability peaks when the learning rate decays, which is also the time at which the sharpness of the loss significantly drops (arrow from "Early Stopping" to "Loss Flatness" in Figure 1.3). This leads us to propose Representation from Flat Neighbourhood (RFN), a new transferability approach that minimises the sharpness of the loss during training (arrow from "RFN" to "Loss Flatness" in Figure 1.3). We show that, by searching for large flat neighbourhoods, RFN always improves over early stopping

(by up to 47 points of success rate) and is competitive with (if not better than) strong state-of-the-art baselines. RFN also complements nicely complementary transferability techniques of other categories, established by our taxonomy of transferability techniques.

Overall, our three complementary techniques provide an extensive and practical method to avoid overly specific adversarial examples from the multimodal and local exploration of flatter vicinities in the weight space. Our probabilistic and geometric approaches demonstrate that the way to train the surrogate model has been overlooked, although both the training noise and the flatness of the loss landscape are important elements of transfer-based attacks.

1.3.4 Addressing the Challenges

This section develops how our contributions address the challenges detailed in Section 1.2.

C1. To address the knowledge gap in the scientific literature on the training of the surrogate DNNs of transfer-based black-box attacks, we empirically evaluate the transferability of numerous training techniques. Chapters 4 to 6 provide a comprehensive overview on how to explore the weight space to obtain good surrogates: we cover the three ways to explore the weight space, either by selecting a single model, multiple models from the same vicinity, or multiple models from different vicinities.

C2. To address the low success rate of small perturbations, we show in Chapters 4 to 6 that our new training-based techniques have higher success rates with smaller L_p norms. In particular, all the Chapters 4 to 6 include the evaluation of the L_∞ norm sets to 4/255, instead of the larger and more popular 16/255. This dissertation contributes to less perceptible transferable adversarial examples.

C3. To reduce the high computational cost of transfer-based attacks, we propose lighter alternatives to existing training techniques. Chapter 4 shows that cSGLD is a significantly cheaper alternative to deep ensemble, thanks to the super convergence. On ImageNet, cSGLD can reach 94% of success rate while reducing the training computations from 11.6 to 2.4 exaflops, compared to deep ensemble. Chapter 5 reveals that LGV beats all combinations of four transferability techniques in only ten additional epochs. The extra computations can be reduced to only five epochs without significantly impacting the success rate. Chapter 6 proposes RFN, a cheaper way to train a surrogate model than slight adversarial training, a competitive technique. Since the transferability of the surrogate trained with RFN does not decrease along epochs, RFN can be seen as a technique that does not waste computing resources compared to standard training.

C4. To tackle the lack of insight on the transferability of adversarial examples, the majority of the experiments in this dissertation are dedicated to the analysis of transferability. We hope that our efforts to develop insight-full contributions will yield scientific value, and that our contributions do not reduce to adding other transferability techniques to a collection of scattered techniques. The insights of this dissertation are summarised in Section 1.3.2. In short, our probabilistic and geometrical analyses exhibit the key role of the training noise and the flatness of the loss to explain why our techniques train better surrogate models.

C5. To face the excessive specificity of some transferability techniques, we propose techniques based on generic training methods for DNN. The transferability techniques proposed and evaluated in Chapters 4 to 6 can be applied on top of any DNN architecture, on other tasks than image classification, and on other applications than computer vision. Their main requirement is the possibility to apply stochastic gradient descent on a loss function, which is how almost all DNNs are trained. Therefore, our generic contributions may impact various fields.

1.4 Organization of the Dissertation

The organization of the dissertation is presented in details in Figure 1.4. In the remaining of this dissertation, Chapter 2 presents the technical background of DNNs and adversarial machine learning, used in this dissertation. Chapter 3 presents the previous work related to the contributions of this dissertation. Chapter 4 presents a probabilistic perspective on transferability and an empirical study that evaluates the improvements of transferability from Bayesian and ensemble training techniques. Chapter 5 presents LGV, an efficient training-based model augmentation transferability technique, and analyses it in depth using a geometrical perspective. Chapter 6 presents an empirical analysis of the relation between transferability and the training dynamics of the surrogate model, and proposes RFN, a new approach to transferability that searches for large flat neighbourhoods. Finally, Chapter 7 concludes this dissertation and presents the limitations and future work.

1.5 Quick Read Guide: Highlighted Sections for Busy Readers

We provide below a list of recommended sections to navigate the dissertation effectively. By highlighting its most important parts, this guide aims to provide a concise roadmap for understanding the outline of this dissertation without the need to read the entire thesis.

- The entire Chapter 1.
- Among Chapter 2:

  - Section 2.4.2,
  - Sections 2.4.4 and 2.4.5.
- Among Chapter 3:
  - Section 3.1.4,
  - Section 3.2.2,
  - Section 3.2.4.
- Among Chapter 4:
  - Section 4.1,
  - Section 4.2,
  - Sections 4.4.5 and 4.4.6,
  - Section 4.6.
- Among Chapter 5:
  - Section 5.1,
  - Section 5.3,
  - Sections 5.4.1, 5.4.2 and 5.4.4,
  - Section 5.5,
  - Section 5.6,
  - Section 5.7.
- Among Chapter 6:
  - Section 6.1,
  - Section 6.3,
  - Section 6.4,
  - Section 6.5,
  - Section 6.7.
- The entire Chapter 7.

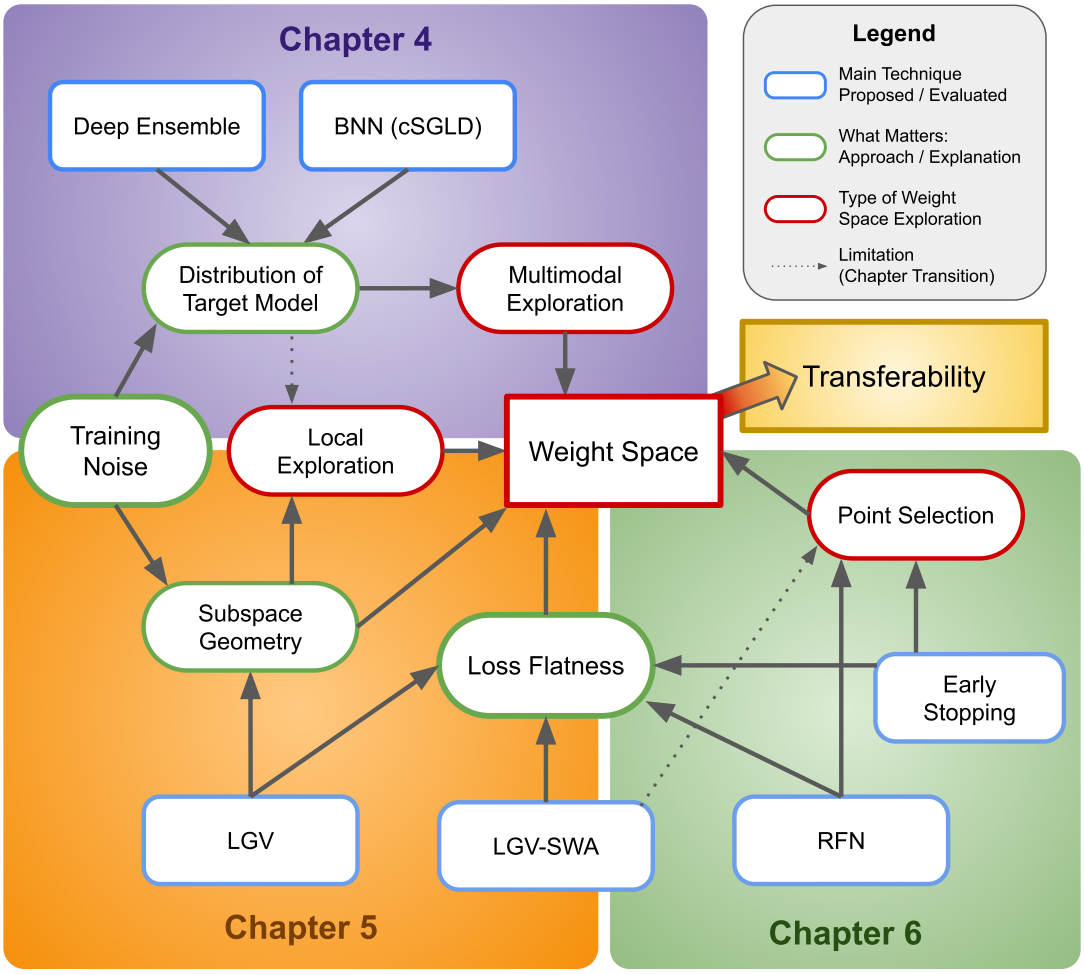


Figure 1.3: Diagram of the outlines of this dissertation. See Section 1.3 for a step-by-step detailed description.

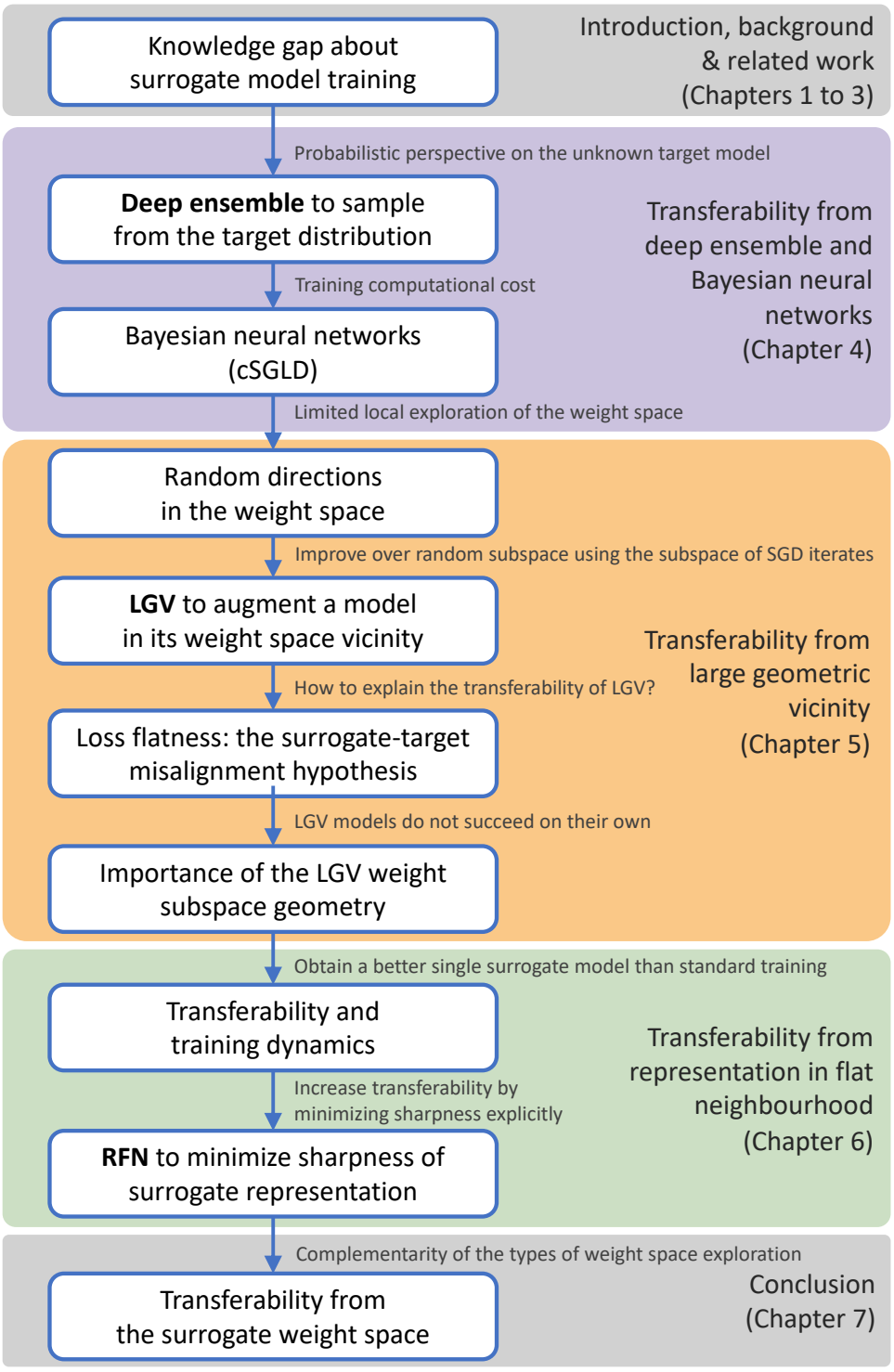


Figure 1.4: Diagram of the storyline and the organization of this dissertation.

2

Background

This chapter discusses the technical background about machine learning, deep neural networks, adversarial machine learning and the transferability of adversarial examples.

Contents

2.1 Machine Learning

This section gives a brief overview of the field of machine learning, including general definitions, applications, the typical project life cycle, and some mathematical definitions.

General definitions. Machine Learning (ML) is a subfield of Artificial Intelligence (AI) that focuses on automating decision-making based on data. ML aims at performing a specific task without being explicitly programmed for, unlike traditional software programming, and using data instead. Data is first fed to a model to train it, i.e, iteratively optimizing the internal parameters of the model to decrease the number of errors. Then, the trained model is used on new unseen data to make predictions. The field has evolved rapidly over the last few decades, resulting in a wide range of applications in diverse domains, including natural language processing, computer vision, robotics, and bioinformatics, among others.

Applications. Numerous real-world tasks have been solved thanks to machine learning. One of these applications is image and speech recognition, where machine learning models can accurately classify or detect objects or spoken words. Another use for ML models is natural language processing, which enables conversational AI, sentiment analysis, and machine translation by analysing or producing text. Recommender systems use ML to predict user interests and suggest content, such as news articles, movies, music, or products. Fraud detection, intrusion detection, and fault identification are examples of anomaly detection, another type of application of ML. Finally, machine learning has been used in bioinformatics to improve protein structure prediction, drug discovery and gene expression analysis.

Task types. Broadly, machine learning can be supervised or unsupervised. In supervised learning, the model is trained on labelled data. The output of the task we aim to solve is known for all the training data. The objective is to learn an input-to-output mapping to be applied on new unlabelled data to predict its unknown output. A classification problem is when the output is a discrete label, and a regression problem is when the output is a continuous variable. Unsupervised learning involves learning from unlabelled data when no output labels are given. The objective is to find hidden patterns, structures, or correlations in the data. The most typical types of tasks are clustering to find similar groups of data, and dimensionality reduction to find a lower-dimensional representation of data that minimizes the loss of information. The rest of this dissertation is dedicated to supervised learning, the most prominent application of machine learning.

Project life cycle. A typical machine learning project consists of several steps, organized in cycle. Data collection is the first step, during which unprocessed data is obtained from a variety of sources, including databases, sensors, scraping, surveys,

human data labelling. The next step is data preparation, which entails cleaning, i.e, removing missing values, outliers and invalid values, and transforming data into an appropriate format. Then, feature engineering transforms raw data intro tidy data with numerical or categorial features, that are directly usable by a classical machine learning model. Model building follows, tuning the parameters of the model using a training dataset to reduce the discrepancy between the expected and predicted outputs. The last step to build a first model is to evaluate it, assessing its performances on unseen data to measure its capacity to generalize. Finally, in the model deployment process, the trained model is integrated into a bigger systems or applications to predict the outputs of new fresh data. The development does not stop after the end of this cycle, since the model needs to be monitored once deployed, and feedback loops are there to improve the previous steps if the performances are inadequate.

Categorisation. ML is divided into two subfields, classical machine learning and representation learning. Classical machine learning uses manually created features as inputs for a variety of learning methods, including logistic regression, support vector machines, and decision trees. Typically, machine learning engineers and domain exports jointly define transformations of raw data that they think would be useful to solve the tasks at stake. This approach is well known and has been demonstrated to be efficient in numerous applications. However, feature engineering can be tedious for complex applications, for example in computer vision. On the other hand, representation learning, automates the feature engineering step, previously described. The model training step jointly learns both the features and the mapping from inputs to outputs, from raw data. This subfield of machine learning is particularly appropriate to complex data, where the structures and patterns are hard to define for a human. The learned features are called the representation, since they aim at giving a high-level abstraction of the data. Deep Learning (DL) is the main branch of representation learning. DL uses artificial neural network with many layers to model the data. DNN are hierarchical, since lower-level layers are generally combined to form higher-level representations of the data.

Mathematical definitions and notations. In its most general mathematical sense, a machine learning model is a function f that maps an input vector $x \in \mathcal{X}$ to a predicted output $y \in \mathcal{Y}$, using a set of learnt parameters $\theta \in \Theta$. The input space $\mathcal{X}$ is generally a subset of $\mathbb{R}^d$, i.e., the input is a vector of d dimensions. In classification, the output space $\mathcal{Y}$ is a set of unordered and mutually exclusive labels $\{1, 2, ..., C\}$, where C is the number of classes. The parameter space Θ depends on the underlying model of the prediction function f. The goal of learning is to find a set of parameters θ that minimizes the average classification error on unseen data, $\theta^* = \arg\min_\Theta \mathbb{E}_{x,y\sim p(x,y)}\, \mathbb{1}(f(x;\theta) \neq y)$, where $p(x, y)$ is the

probability distribution of the data generating process. In practice, this distribution is unknown, and cannot be used explicitly to train the model, i.e., to find θ. But, it can be sampled to form a training dataset $\mathcal{D} = \{(x_i, y_i) \sim p(x, y)\}_{i=1}^{N}$. This dataset $\mathcal{D}$ is used to find the set of parameters $\hat{\theta}$ that minimizes the average error on the training set, $\hat{\theta} = \arg\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(f(x_i; \theta) \neq y_i)$. This process is called Empirical Risk Minimization (ERM). This optimization problem is generally solved with gradient descent. However, we need to replace the indicator function with a smooth approximation. Indeed, the indicator function is step-wise function, so the gradients of the quantity to minimize would be a vector of zeros. This smooth approximation is called the loss function $\mathcal{L}$[1], and takes in inputs the predicted probabilities $\hat{y}_i = f(x_i, \theta)$ and the true label y_i: $\mathcal{L}(\hat{y}_i, y_i)$. A typical loss function in classification is the cross-entropy loss, defined as $\mathcal{L}_{\text{ce}}(\hat{y}_i, y_i) = -\sum_{c=1}^{C} \mathbb{1}(y_i = c) \log \hat{y}_{i,c}$.

2.2 Deep Learning

This section provides a brief overview of the field of deep learning, including the presentations of deterministic and Bayesian neural networks, the architectures of neural networks, stochastic gradient descent, and some benchmark datasets used in this dissertation.

2.2.1 Deterministic Neural Networks

Neural networks are a type of machine learning model loosely inspired by the structure and functions of the human brain. In practice, these models are compositions of deterministic functions. Once trained, a neural network always gives the same output with given input and a set of parameters. Artificial neurons are the first building blocks of neural networks. A neuron is a computational unit that first performs a weighted sum of its inputs from other neurons, then outputs the results of the sum passed through a nonlinear activation function. The activation function is a nonlinear transformation. It introduces nonlinearity into the neural network to learn complex patterns from the data. The two most common activation functions are sigmoid, and rectified linear unit (ReLU). Feed-forward neural networks are composed of multiple layers organized sequentially. Every layer contains several neurons. In its basic form, a neural network is a composition of functions, where the input of one layer is the output of the previous layer.

The first layer is called the input layer because it receives the raw input data. The final layer, named the output layer, produces the predicted output. In between, intermediate layers, or hidden layers, embed increasing complexity representations of the data, from low-level to high-level abstractions. In a fully connected neural network, or multilayer perceptrons, every neuron in a layer is connected to all the

[1]We voluntarily skip the definition of the loss function as used in statistical learning theory for clarity and simplicity to have consistent notations in the remaining of the dissertation.

neurons in the previous layer. Mathematically, a fully connected neural network is composed of numerous matrix multiplications and activation functions. Given an input vector $x \in \mathcal{X}$, the output of the first hidden layer is:

$$h^{(1)} = g^{(1)}(W^{(1)\top}x + b^{(1)}) \ ,$$

where $W^{(1)}$ is the weight matrix connecting the input layer to the first hidden layer, b_1 is the bias vector, and $g^{(1)}$ is the activation function of the first hidden layer. The output of following hidden layers are defined similarly:

$$h^{(l)} = g^{(l)}(W^{(l)\top}h^{(l-1)} + b^{(l)}) \ ,$$

for $l \in \{2, 3, ..., L\}$, where L is the number of hidden layers. Lastly, the output of the neural network is:

$$\hat{y} = g^{(L+1)}(W^{(L+1)\top}h^{(L)} + b^{(L)}) \ ,$$

where $g^{(L+1)}$ is the activation function of the output layer, generally the softmax function for classification. Numerous variants of fully connected neural networks exist, creating many architectures of DNNs. Section 2.2.3 presents some families of DNN architectures.

2.2.2 Bayesian Neural Networks

Contrary to deterministic neural networks, Bayesian Neural Network (BNN)s do not consider the weights as fixed values. BNN is the combination of neural networks with the principles of Bayesian inference. The probabilistic framework quantifies the uncertainty related to the weights, and subsequently the uncertainty of the predictions. BNNs have a long history, since the influential book of Neal [Nea96] dates back to 1996. The use of Bayesian methods addresses the overfitting issue, among others.

The weights of BNNs are treated as random variables with prior distributions. Instead of learning point estimates for the weights, as deterministic neural networks, BNNs aim to learn the posterior distribution over the weights given the observed data. We note $\theta \in \Theta$ the vector of all the weights of the neural network concatenated. The prior distribution over the weights is denoted by $p(\theta)$, and the likelihood of observing the training data given the weights is denoted by $p(\mathcal{D}|\theta)$. According to Bayes' theorem, the posterior distribution $p(\theta|\mathcal{D})$ over the weights can be computed as follows:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} \ ,$$

where $p(\mathcal{D})$ is the marginal likelihood, also called evidence. The objective of training is to maximize the posterior with respect to the weights, i.e., find $\theta^* \in$

$\arg\max_{\theta\in\Theta} p(\theta|\mathcal{D})$. Since the marginal likelihood does not depend on w in our optimization problem, it is considered as a normalization constant, and training a BNN consists of solving the following optimization problem:

$$\max_{\theta\in\Theta} p(\mathcal{D}|\theta)p(\theta) \ ,$$

Generally, deterministic neural networks are trained to maximize the likelihood $p(\mathcal{D}|\theta)$. It becomes apparent that the source of difference between training BNNs and deterministic neural networks is the introduction of the prior over the weights.

In practice, the exact computation of the posterior distribution is often intractable, necessitating the use of Bayesian deep learning methods to approximate the posterior. There are two families of approximate inference techniques to do so:

1. *Simulation-based inference*, that generates a random sample from the posterior distribution, and uses the subsequent empirical distribution as the posterior approximation. Chapter 4 relies heavily on methods based on Markov Chain Monte Carlo (MCMC). Hamiltonian Monte Carlo (HMC) is considered a golden standard, but is also too expensive to scale to bigger datasets than small datasets, like MNIST. A more modern family of approximate Bayesian inference techniques is Stochastic Gradient-Markov Chain Monte Carlo (SG-MCMC), or Stochastic Gradient Langevin Dynamics (SGLD), which was inaugurated by SGLD [WT11]. SG-MCMC combine SGD with MCMC to sample from the posterior during the training with SGD. The original technique [WT11] adds a specific noise to the weight at each SGD iteration. Therefore, the computational overhead of these methods is minimal, and SG-MCMC scales well to large datasets like ImageNet.

2. *Distributional approximation*, that estimates an explicit distribution to approximate the posterior. The approximation is generally a simpler parametric distribution. The most popular methods are Variational Inference (VI). Typically, the goal is to minimize the KL-divergence between the approximate posterior and the true posterior using the evidence lower bound (ELBO) optimization. VI can scale to large datasets, such as ImageNet, using simplistic distribution, like the popular mean-field approximation that is composed of independent Gaussian variables.

Once the approximate posterior distribution over the weights is estimated, the output of a new input x_0 can be predicted by the posterior predictive distribution:

$$p(y_0|x_0, \mathcal{D}) = \int_\Theta p(y_0|x_0, \theta)p(\theta|\mathcal{D})d\theta \ .$$

Approximation techniques, such as MCMC, estimate this predictive distribution, since this integral is generally intractable.

In summary, BNNs offer a principled approach to reasoning about uncertainty by considering the weights as random variables.

2.2.3 Architectures of Neural Networks

Researchers have proposed numerous variants of fully connected DNN, presented in Section 2.2.1. This section gives a brief overview of the most common ones, which we use in this dissertation both as the surrogate and target models.

By utilizing the spatial structure of the input, CNNs are specifically created to process grid-like data, such as images. CNNs are composed of both convolutional and pooling layers. Convolutional layers, which apply filters to specific areas of the input to find patterns, are followed by pooling layers, which shrink the feature maps' spatial dimensions. Convolutional layers scale better than fully connected ones because the weight matrices are much sparser thanks to parameter sharing. They also have the benefit of being (almost) translation invariant, which is considered desirable for computer vision.

Residual neural network (ResNet) is a popular family of architectures. ResNet adds skip connections, or shortcuts, that add the output of one layer to the output of a later layer. ResNets are often CNNs. Skip connections allowed to train deeper DNN which were previously impossible to train due to vanishing gradients. They allow the flow of gradient passing by DNN during backpropagation. ResNets are composed of residual blocks, which consist of a series of convolutional layers followed by the addition of the input to the block, forming a skip connection. Skip connections allowed to train deeper DNNs, which significantly improved various computer vision tasks.

Transformers are another type of architecture that gains a lot of traction lately to modelling long dependances in sequential data, like text. Transformers use self-attention to weigh the importance of different input elements based on the context. ViT is the adaptation of transformers to computer vision data. Unlike CNNs, ViTs divide an image into fixed-size, non-overlapping patches and linearly embed them as sequences. Then, the transformer layers process these sequences to capture both local and global patterns.

2.2.4 Training Deep Neural Networks

In order to train DNNs, gradient-based optimizers minimize the loss function with respect to the model's parameters. Due to the size of modern datasets and the computational limitations, full gradient descent, which computes the gradient using the entire dataset, is computationally impossible for deep neural networks. To solve this issue, SGD uses a mini-batch, i.e., a small randomly selected subset of the data, to approximate the true gradient. This type of optimizer reduces the training computational cost, allowing to train DNNs with many layers on large datasets, by adding *training noise*. Every update of the parameters is performed using a noisy

gradient, where the uncertainty comes from sampling the mini-batches.

Gradient-based optimization for neural networks is built on the backpropagation algorithm. By using the chain rule of differentiation and propagating the gradients backward from the output layer to the input layer, it calculates the gradient of the loss function with respect to each weight. The model's parameters are then updated using the estimated gradients during each SGD iteration, eventually lowering the loss function.

Regularization methods are essential for avoiding overfitting, which happens when a model is excessively complex and performs well on training data but badly on unseen data. Dropout, weight decay, and early stopping are three common regularization techniques. Dropout is a training method that forces a layer to rely on multiple information-flow paths by randomly setting a portion of its neurons to zero. Weight decay, also known as L2 regularization, discourages the model from learning large weights that could result in overfitting, by adding a penalty term to the loss function proportional to the sum of squared weights. Early stopping is a technique that stops training as soon as the model's performance on a validation set begins to deteriorate, preventing the model from fitting the noise in the training data.

Numerous methods, such as learning rate schedules, batch normalization, and variants of SGD, have been developed to enhance convergence and training stability. Learning rate schedules modify the learning rate throughout training, frequently beginning with a higher value to promote exploration and lowering it over time to permit fine-tuning. Using a technique called batch normalization, each layer's activations are normalized to have a mean value of zero and a variance of one. Initially, batch normalization was proposed to lessen the internal covariate shift and encourage quicker convergence. For a more reliable and effective training, sophisticated optimizers or SGD variants, such as momentum, Adam, RMSProp, and AdaGrad, adapt the gradient-based updates.

To sum up, there are several important building blocks to train DNNs. SGD is an efficient optimizer that adds noise to the gradients. Regularization techniques, batch normalization, learning rate schedules and variants of SGD help to train DNNs. The foundation of contemporary deep learning optimization is made up of these techniques.

2.2.5 Classical Datasets

This section describes three computer vision datasets used in this dissertation, and more generally, used classically in deep learning.

MNIST. The MNIST dataset is a classical benchmark dataset for handwritten digit recognition that consists of 60,000 training images and 10,000 test images. Representing a digit, from 0 to 9, each image is a 28x28 greyscale picture of a single

handwritten digit. MNIST has been a well-liked choice for researchers to test and develop various machine learning algorithms because of its small size and simplicity, especially as a first step in investigating deep learning techniques. Nowadays, MNIST is considered too simple and skewed for a benchmark dataset. In this dissertation, we used it only in Chapter 4 to evaluate expensive BNN techniques that cannot scale to larger datasets and DNN. No experiment is restricted to MNIST, since we additionally perform the same evaluations on the two following datasets.

CIFAR-10. Another standard dataset is the CIFAR-10 dataset, which has 60,000 32x32 colour images representing ten different object classes. Each class has 6,000 images. 50,000 images are used for training, and 10,000 for the test dataset. The ten categories are the following: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Compared to MNIST, CIFAR-10 is a more complex task due to its larger images, the presence of colours, and the wider range of objects.

ImageNet. ImageNet is a large-scale dataset with over 14 million high-resolution images encompassing more than 20,000 object categories. The dataset is arranged in accordance with the WordNet hierarchy, with each synset (a collection of related words or phrases) representing a certain category of objects. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC-12), a popular subset of ImageNet, contains over 1.2 million training images, 50,000 validation images, and 100,000 test images, spanning 1,000 object categories. We use this popular dataset in all the chapters of this dissertation. The number and diversity of classes, the amount of training examples, and the input size of images, make ImageNet less of a toy dataset, but greatly increase the computational complexity to train even a single DNN.

In this section, we described briefly key elements in deep learning to understand the rest of this dissertation. Despite, the important success of the techniques described above on natural data, these same techniques can fail critically when facing an adversary. the field of adversarial machine learning studies these cases.

2.3 Adversarial Machine Learning

This section gives an overview of adversarial machine learning, i.e., the study of machine learning models facing an adversary. The field of adversarial machine learning is the application of computer security to machine learning. It should *not* be confused with Generative Adversarial Networks (GAN), which are a completely separate research topic.

2.3.1 Threat Model

This section defines the notion of threat model, from computer security, when applied to machine learning. Threat modelling aims at clearly defining an attack that a system may face, using a security perspective. It should not be confused with a machine learning model that models some data. We present here the definition of a threat model developed by Biggio et al. [BCM+13] and Biggio and Roli [BR18], where an adversary is defined by his/her goal, knowledge, and capability. In the remaining of this dissertation, the machine learning under attack is called the *target model*.

Adversary's goal The first element to define an attack against a machine learning model is the adversary's goal, which defines what would constitute a successful attack. For example, an attacker may aim at misclassification or retrieval of data. This objective is generally solved as an optimization problem expressed in terms of a loss function to minimize or maximize. The goal may impose some constraints on this optimization problem. For instance, when searching for adversarial examples, the adversarial perturbation should be imperceptible. The standard problem formulations are defined in the next section, for each types of attacks.

Adversary's knowledge An attack against a machine learning model is also defined by the attacker's knowledge about the model and the system at stake. Biggio et al. [BCM+13] state that the adversary's knowledge includes the following elements:

- the trained model, which is composed of the functional form of the model, i.e., the type of model and its architecture for DNNs, and all the values of its parameters (the weights for DNNs),
- the training dataset, or a subset of it,
- the data preprocessing, i.e., the transformations of raw data before feeding it into the classifier,
- the learning algorithm used, including the type of optimizer and its hyperparameters,
- the feedbacks from the model, e.g., the prediction of arbitrary inputs chosen by the adversary.

If the adversary has complete knowledge of the target model, the attack is white-box. The robustness evaluation of new defences should be performed with white-box attacks because a truly robust defence against the most capable adversary would also be robust against less capable ones. Black-box attacks correspond to threat models where the adversary has no knowledge of the target models. Empirically, it is a good practice to also use black-box attacks when evaluating new defences to make sure that new defences not only make current white-box attacks fail [CAP+19], as shown by the discovery of defences relying on obfuscated gradients [ACW18].

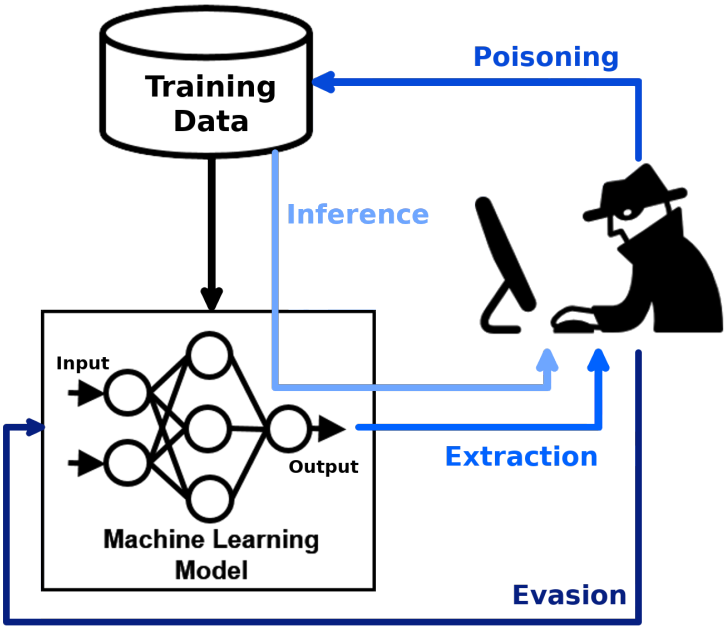


Figure 2.1: Illustration of the typology of attacks developed in adversarial machine learning. Figure from [NST+18].

Grey-box attacks are intermediate cases, where the adversary has partial knowledge of the target model. For example, the architecture and the training data may be known, but not its weights.

Adversary's capability Finally, the adversary comes with a capability. The capability describes which elements of the machine learning pipeline that the adversary can alter. The attacker may be able to modify the test inputs, keeping the target model unchanged. Or the adversary may alter a fraction of training data to achieve his/her goal.

Overall, a threat model defines an attack against a machine learning model, where an adversary aims at achieving a goal using his/her capability based on specific knowledge of the model under attack. The research on adversarial machine learning recognized several standard threat models, leading to a typology of attacks.

2.3.2 Typology of Attacks

In this section, we present a commonly accepted typology of attacks against machine learning models [BR18; NST+18]. The research on adversarial machine learning developed an extensive number of attacks to tackle distinct objectives in a variety of scenarios. These attacks can be grouped into four categories, each defined by a specific threat model. Figure 2.1 illustrates their differences. The

following list summarizes the typology.

- *Evasion attacks.* Evasion attacks produce adversarial examples, i.e., carefully crafted perturbations added to test data designed to fool the model at inference time. This dissertation studies this type of attack.
- *Poisoning attacks.* Poisoning attacks produce poisonous data that are added to the training set at training time.
- *Extraction attacks.* Extraction attacks produce sequences of queries designed to infer information about the model at inference time.
- *Inference attacks.* Inference attacks produce sequences of queries designed to infer information about the training data at inference time.

Evasion attacks (adversarial examples). Adversarial examples are **carefully crafted, imperceptible perturbations applied to the inputs** of a machine learning model to create large changes in output. Figure 2.2 illustrates this definition: starting from a natural and correctly classified test image x (left), one can add a tiny perturbations (middle, here multiplied by several orders of magnitude to visualize colours instead of a plain grey image), to obtain a visually similar image that is classified incorrectly with very high confidence (right). In this dissertation, we use the terms evasion attack and adversarial example interchangeably. Evasion attack is the most studied type of attack against machine learning models.

These perturbations can be seen as a worst-case distributional shift, which causes the model to output incorrect predictions despite the seemingly negligible changes in the input. Distributional shifts are changes in the data distribution between the training and testing phases. While standard distributional shift can happen naturally due to variations in the real-world changes or in the data collection, worst-case distributional shift is intentionally introduced by an adversary to fool the model. This difference is key, since evasion attacks do not introduce random noise, but a carefully crafted perturbation. Despite the random-looking pattern of the perturbations showed in Figure 2.2, the adversarial perturbation is not random noise, but a much more effective perturbations that can easily drop the accuracy of a state-of-the-art model to zero percent. Another key aspect of evasion attacks is that the model under attack remains unaltered, since the capability of the adversary is limited to modifying input at test time.

Adversarial examples can be either targeted or non-targeted. In a targeted adversarial attack, the attacker's objective is to change the input data in such a way that the model outputs the attacker-specified, incorrect output label. This attack corresponds to cases where an adversary attempts to enforce a particular, desired outcome. Non-targeted adversarial attacks, on the other hand, don't specify a specific target label; instead, they try to make the model produce any incorrect prediction. The attacker's primary goal in non-targeted attacks is to cause the model to fail.

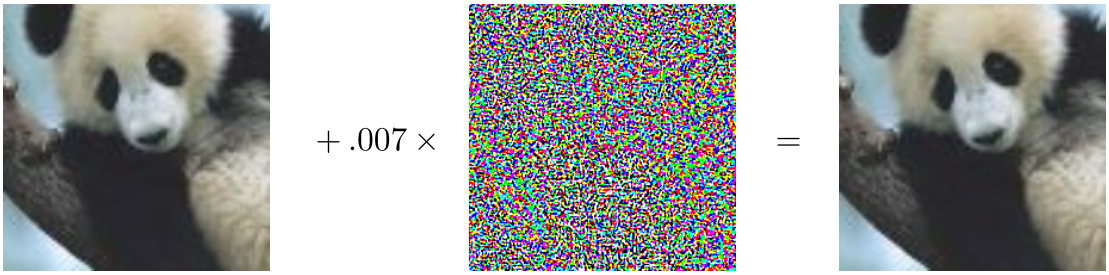


Figure 2.2: Illustration of a correctly classified test input (left), an imperceptible adversarial perturbation crafted with the FGSM attack, here multiplied for visualization (middle), that end produce an adversarial example which is visually similar to the original image, but incorrectly classified with a very high confidence (right). Figure from [GSS14].

Poisoning attacks. In addition to evasion attacks, poisoning attacks represent another threat to machine learning models. In a poisoning attack, an adversary deliberately alters a subset of the training data by introducing precisely constructed, malicious samples, with the aim of harming the model's performance in the learning process. Attacks that poison data, the basic core of machine learning models, can be particularly harmful because they are challenging to recognize and fight against. Poisoning attacks leverage the foundation of machine learning, as shown by the seminal contributions of Barreno et al. [BNS+06; BNJ+10] that dates back as early as 2006 and 2010. The corrupted training data may introduce poor generalization [BNL12; XBB+15; MZ15], skewed decision boundaries [MBD+17], or backdoors [GDG17; CLL+17].

Extraction attacks. Extraction attacks, which focus on the unauthorized acquisition of model information, are another category of threats against machine learning models [BCN+14; FJR15; TZJ+16]. An extraction attack entails the adversary querying the target model with carefully selected inputs and then observing the corresponding outputs in order to copy or approximate the target model. The attacker then trains a surrogate model to try to replicate the behaviour of the target model using the input-output data they've gathered. Extraction attacks provide a serious risk because they can result in the theft of intellectual property, the exposure of private data hidden inside the model, or even the development of surrogate models that enable additional adversarial attacks like evasion or poisoning.

Inference attacks. Inference attacks are the last category of threats against machine learning models [RBH+09; BCN+14; SSS+16]. They aim at inferring information about training data. By observing the model's outputs on a carefully selected set of inputs, the adversary seeks to infer private information about the training data, such as the presence of particular samples or the values of particular attributes. These attacks make use of the unintentional information leakage caused by learned model parameters, which unintentionally encode patterns or features unique to training data. Even though the model itself isn't made to reveal such details, inference attacks might result in privacy violations, since they can divulge private or sensitive information about the people or entities represented in the training data.

All four types of attacks – evasion, poisoning, extraction, and inference – may be turned into physical attacks, posing a potential risk to the systems deployed without exposed API. In the context of a physical attack, an adversary manipulates the physical environment or input devices to produce adversarial manipulations that trick the model into producing inaccurate predictions or reveal sensitive data. For example, a poisoning attack could include tampering with sensor data needed to train a predictive maintenance model for industrial applications, while an evasion attack can take the form of adversarial perturbations on a traffic sign to trick an autonomous vehicle's vision system.

The following of this dissertation is dedicated to evasion attacks. We will consider a threat model where the adversary aims at *misclassification* (untargeted attacks), under *limited knowledge* (gray-box attack), and can only modify inputs at test time in an imperceptible way. We use both terms evasion attacks and adversarial examples interchangeably.

2.4 Transferability of Adversarial Examples

This section provides a technical background about the transferability of adversarial examples. First, we present the seminal papers on the topic, followed by a formal definition of the adversarial examples and of their transferability. Then we present the main metric used in this dissertation to measure transferability. We expose the current hypotheses for the existence of transferability, and finally the standard gradient-based attacks.

2.4.1 Seminal Papers

The transferability of adversarial examples was observed early on, in the field of adversarial machine learning. Szegedy et al. [SZS+13], one of the two main seminar papers of adversarial machine learning with Biggio et al. [BCM+13], concurrently finds adversarial examples and observes their transferability. Szegedy et al. [SZS+13]

is the first to use the expression of "adversarial examples" to designate test examples that are altered imperceptibly to fool a model. They show that adversarial examples against one DNN can also fool other DNNs of different architectures or trained on disjoint subsets of data. This article paved the way for understanding the transferability of adversarial cases and activated more research on the subject.

Later, Goodfellow et al. [GSS14] further analyse the transferability of adversarial examples. They show that a simple linear model can have adversarial examples if its input has sufficient dimensionality. Then, they propose the linear hypothesis to explain why adversarial examples transfer: transferability across architectures and training sets would come from the local linearity of DNNs. Goodfellow et al. [GSS14] provide evidence in favour of this hypothesis by proposing the Fast Gradient Sign Method (FGSM) attack, that computes a single gradient, and showing its transferability. This groundwork motivated other competitive or complementary hypotheses to explain the source of transferability and techniques to improve transferability (Section 2.4.5).

Finally, Papernot et al. [PMG16] provide in-depth analysis of the transferability phenomenon. They propose attacks against DNNs, logistic regression, support vector machines, decision trees, nearest neighbours, and ensembles. Then, Papernot et al. [PMG16] show that adversarial examples transfer between models of the same type (intra-technique transferability) and between models of different types (cross-technique transferability). Last, but not least, Papernot et al. [PMG16] are the first to show the feasibility of transfer-based black-box attacks against commercial APIs. They train a *surrogate model* to apply white-box attacks against it, and finally feed these adversarial examples to an unknown *target model*.

2.4.2 Formal Definitions

First, this section gives a formal definition of an adversarial example, and then a formal definition of a transferable adversarial example.

Adversarial examples. Given an input $x \in \mathcal{X}$ associated with the label $y \in \mathcal{Y}$ and a classification model f parametrized by $\theta \in \Theta$, an adversarial example $x_{\text{adv}} \in \mathcal{X}$ can be defined as follows:

$$f(x_{\text{adv}}; \theta) \neq y \ ,$$

The adversarial example is defined from its corresponding adversarial perturbation $\delta \in \Delta$ that is constrained to be in the feasible set Δ of imperceptible perturbations:

$$x_{\text{adv}} = x + \delta \ .$$

This feasible set Δ would ideally include any input that a human would associate with the same true label. These inputs would be produced by all label-preserving

data transformations such as rotation, scaling, background change, change in the pose of the underlying 3D object considered. This set is hard to define mathematically. So, researchers generally consider the subset of perturbations that are imperceptible by bounding them with a small p-norm ε. Using this L_p norm definition, the feasible set of adversarial perturbations Δ, is:

$$\Delta = \{\delta \in \mathbb{R}^d \mid \|\delta\|_p \leq \varepsilon,\, x + \delta \in \mathcal{X}\}\,.$$

In practice, finding an adversarial example is transformed into an optimization problem. Instead of minimizing the loss with respect to the model's parameters, as done during training, an adversarial example aims at maximizing the loss $\mathcal{L}$ with respect to the input:

$$\delta \in \arg\max_{\delta \in \Delta} \mathcal{L}(f(x+\delta;\theta), y).$$

Transferability. To define a transferable adversarial example, we consider two classifiers: the target model f_t parametrized by $\theta_t \in \Theta_t$ which is unknown to the adversary, and the surrogate model f_s parametrized by $\theta_s \in \Theta_s$ which is known to the adversary and used as a replacement to the target model. An adversarial example $x_{\text{adv}} = x + \delta \in \mathcal{X}$ is said to be transferable if it is crafted against the surrogate model only, and if it is misclassified by the unseen target model:

$$\begin{aligned} \delta \in \arg\max_{\delta \in \Delta} \mathcal{L}(f_s(x+\delta;\theta_s), y)\ , \\ f_t(x+\delta;\theta_t) \neq y\ . \end{aligned}$$

The remaining of this dissertation focuses on transfer-based attacks, specifically in the context of the "NoBox" threat model. Under the "NoBox" threat model, the adversary has no feedback from the target model. Therefore, the optimization problem to find the perturbation δ cannot be solved using information on the target model, nor the outputs of this target model corresponding to chosen inputs.

2.4.3 Metric

We measure the transferability of adversarial examples using the success rate, defined as the misclassification rate by the target model, of adversarial examples crafted on the distinct surrogate model. All adversarial examples originate from correctly classified examples by all target models under study.

Given a set of N' adversarial examples and their corresponding original true label $\{(x'_{\text{adv},i}, y'_i)\}_{i=1}^{N'}$, the top-1 success rate is defined as:

$$\frac{1}{N'} \sum_{i=1}^{N'} \mathbb{1}\left(f_t(x'_{\text{adv},i};\theta_t) \neq y'_i\right)$$

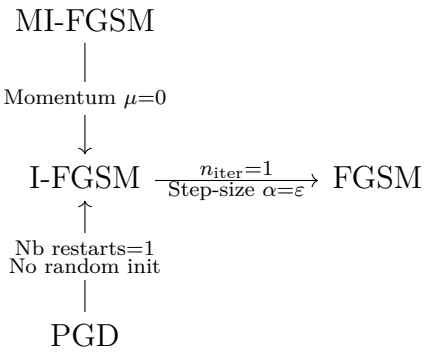


Figure 2.3: Relationships between gradient-based attacks.

There are other definitions and metrics for measuring the transferability of adversarial examples. However, in this study, we aim to compare diverse surrogate models, each consisting of a different number of models with varying accuracies. To ensure fairness in our comparison, we use a common set of original examples that are correctly classified by all target models.

2.4.4 Standard Gradient-based Attacks

This section presents the standard gradient-based attacks used to generate transferable adversarial examples in the adversarial machine learning literature in general and in this dissertation in particular. We present here the attacks in their general form for any L_p norm. The attack has to depend exclusively on the surrogate model, leaving the target model unseen. As a reminder, the optimization objective here is the following:

$$\delta \in \arg\max_{\delta \in \Delta} \mathcal{L}(f_s(x+\delta;\theta_s), y) \ .$$

We present here the four most popular gradient-based attacks. Figure 2.3 illustrates the relationships between them.

First, FGSM was proposed early on by Goodfellow et al. [GSS14]. FGSM is a single-step, white-box attack that computes the gradient of the loss with respect to the input using standard backpropagation, and uses the normalized gradient to create adversarial perturbations. Algorithm 1 contains the pseudocode of the attack. We want to emphasize that FGSM should not be used to evaluate the robustness of a defence, since it is a fragile attack [CAP+19]. This attack was indeed created to explore the linearity hypothesis regarding why adversarial examples exist and transfer, as presented in Section 2.4.1.

Algorithm 1 FGSM Attack.

Input: (x, y) natural example and its corresponding label, θ_s weights of the surrogate DNN f_s, ε p-norm perturbation, $\mathcal{L}$ loss function

Output: x_{adv} adversarial example

1: $g \leftarrow \nabla_x \mathcal{L}(f_s(x; \theta_s), y)$ ▷ Compute the input gradient of the surrogate loss

2: $x_{\text{adv}} \leftarrow x + \text{project}(g, S_\varepsilon[\mathbf{0}])$ ▷ Add the normalized gradient, projected in the p-norm sphere of ε radius

3: $x_{\text{adv}} \leftarrow \text{clip}(x_{\text{adv}}, 0, 1)$ ▷ Clip to pixel range values

Second, the most popular attack to produce transferable adversarial examples is Iterative Fast Gradient Sign Method (I-FGSM), also called Basic Iterative Method (BIM). This attack was proposed by Kurakin et al. [KGB17] and is now the workhorse of transferability. I-FGSM is the iterative algorithm based on FGSM. Algorithm 2 presents the generic pseudocode of I-FGSM for any L_p norm. In essence, I-FGSM is *normalized projected gradient ascent* in the input space. This attack is the backbone of this dissertation. Our preliminary experiments show that I-FGSM has better success rate than FGSM.

Third, Projected Gradient Descent (PGD) [MMS+18], a variant of I-FGSM, is the most popular gradient-based attack to evaluate the robustness of DNNs. Contrary to I-FGSM, PGD starts from a random point in the L_p ball (line 1 of Algorithm 2), and performs several of those random restarts until to find a successful adversarial example against the surrogate model. In Chapter 4, we found that PGD has lower transferability than I-FGSM.

Lastly, I-FGSM has become the standard gradient-based attack to generate transferable adversarial examples in the scientific literature, and numerous I-FGSM variants have been proposed to enhance transferability. The most popular is Momentum Iterative attack (MI), or MI-FGSM, that adds momentum to the gradient g in Algorithm 2 (line 3): $g_i = \mu\, g_{i-1} + \frac{\nabla_x \mathcal{L}(f_s(x;\theta_s),y)}{\|\nabla_x \mathcal{L}(f_s(x;\theta_s),y)\|_1}$ with μ the decay factor. We present numerous other techniques for transferability in the next section.

2.4.5 Typology of Transferability Techniques

This section proposes a typology of existing techniques designed to boost the transferability of adversarial examples. In the remaining of this dissertation, we call them *transferability techniques*. A large portion are variants of the I-FGSM attack. We propose to classify them in seven categories, presented below. The typology developed here is partially based on the work of Zhao et al. [ZZL+22].

Input augmentation for transferability. The first type of transferability technique relies on augmenting inputs during the attack. Typically, at each I-FGSM iteration, a data transformation function is applied to the current adversarial

Algorithm 2 I-FGSM Attack.

Input: (x, y) natural example and its corresponding label, θ_s weights of the surrogate DNN f_s, n_{iter} number of iterations, ε p-norm perturbation, α step-size, $\mathcal{L}$ loss function

Output: x_{adv} adversarial example

$x_{\text{adv}} \leftarrow x$
for $i \leftarrow 1$ **to** n_{iter} **do**
    $g \leftarrow \nabla_x \mathcal{L}(f_s(x_{\text{adv}}; \theta_s), y)$ ▷ Compute the input gradient of the surrogate loss
    $x_{\text{adv}} \leftarrow x_{\text{adv}} + \text{project}(g, S_\alpha[\mathbf{0}])$ ▷ Add the normalized gradient, projected in the p-norm sphere of α radius
    $x_{\text{adv}} \leftarrow \text{project}(x_{\text{adv}}, B_\varepsilon[x])$ ▷ Project in the p-norm ball centred on x of ε radius
    $x_{\text{adv}} \leftarrow \text{clip}(x_{\text{adv}}, 0, 1)$ ▷ Clip to pixel range values
end for

perturbation before computing the gradient. The objective is to obtain gradients that are less specific to the surrogate model, and therefore generalize better to another model. The most popular of such technique is *Input Diversity* (DI) [XZZ+19], which applies random transformations (random resize followed by random padding) to the input images at each attack iteration. Zhou et al. [ZHC+18] add Gaussian noise to ensemble gradients at each iteration. To target defended models, Dong et al. [DPS+19] propose Translation Invariance (TI) that convolves the gradient with a pre-defined kernel to approximate the gradient of the ensemble of translated images. Lin et al. [LSH+20] develop scale invariance that computes several gradients per iteration over pixels values scaled with a factor of $\frac{1}{2^i}$ where $i \in \mathbb{N}$. Wang et al. [WHW+21] improve transferability with Admix by computing the gradient at the input image composed with a small portion of a random image of another label. VT (Variance Tuning) [WH21] uses samples from the uniform distribution over a small neighbourhood around the adversarial example at the current iteration. These techniques can also be applied several times per iteration to compute an average gradient that leads to more transferable adversarial examples [ZZL+22]. Averaging gradients over random noise in the input space has a smoothing effect on the loss function [ZHC+18] (see Section 3.2.3 for a more detailed discussion). Techniques should be compared with the same number of gradient per iteration for fairness [ZZL+22]. The input augmentation transferability techniques are specific to the type of data considered (images, here) and to the domain of application. New separated techniques need to be defined to improve the transferability in, for example, natural language processing tasks.

Model augmentation for transferability. Another popular type of transferability technique augments the surrogate model during the attack, either by applying transformations to the weights or to the architecture. Model augmentation techniques transform the model at *test time*, i.e., after training the base model, when performing the attack, and do not modify the training of the base surrogate model, contrary to training techniques presented below. Ghost Networks (GN) [LBZ+18] use dropout or skip connection erosion to generate on-the-fly diverse sets of surrogate models from one or more base models. Skip Gradient Method (SGM) [WWX+20] favours the gradients from skip connections rather than residual modules through a decay factor applied to the latter during the backward pass. LinBP [GLC20] also modifies the backward pass to increase linearity, but differently by skipping some non-linear activation functions while rescaling the activations. Naseer et al. [NRK+22] develop a model augmentation transferability technique designed specifically for vision transformers. Zhang et al. [ZCB+21] find a slight increase in transferability by filtering the surrogate weight with the smallest L_1 norm. These techniques are cheap to apply, since they generally do not require additional computations. However, they are most of the time specific to a family of architecture: GN either applies skip connection erosion or dropout depending on the architecture, and SGM can only be applied on architectures of the ResNet family. Furthermore, the reasons why some techniques improve transferability are not well understood. It remains unclear why SGM and LinBP apply their transformations in the backward pass only, keeping the forward pass unchanged. As pointed by Zhang et al. [ZBC+21], backpropagating smoothly may be more important than backpropagating linearly.

Training technique for transferability. Another direction to improve the transferability of adversarial example is to improve the training of the surrogate model. To the best of our knowledge, a single article before our first work proposed such a technique: Liu et al. [LCL+17] show that an ensemble of architectures is a better surrogate model than a single architecture. Later, Springer et al. [SMK21] train the surrogate model with adversarial training on adversarial examples of small norm bound ε, called *slight adversarial training*. They generally train the surrogate using another L_p norm than the one used for the attack. Benz et al. [BZK21] and Zhang et al. [ZCB+21] show the benefit of early stopping the surrogate model (see Chapter 3 for more details). Benz et al. [BZK21] also show that removing batch-normalization layers improves transferability. Yuan et al. [YZJ+21] propose a way to exploit a large number of pretrained models as the surrogate when such a model zoo is available. Such techniques are easily applied in complement to other types of transferability techniques: for example, one can apply model augmentation on a better base model, or on every model of multiple architectures. This category of techniques is broad, since they can rely on all the training elements of a DNN:

the variants of SGD, the training hyperparameters, the choice of architectures or pre-trained models, the regularization scheme, etc. This dissertation focuses on this type of technique.

Optimization for transferability. Some works improve upon the optimization of I-FGSM to perform gradient ascent in the input space without finding adversarial examples overly specific to the surrogate model. Dong et al. [DLP+18] propose MI, also called MI-FGSM, that adds momentum to the attack gradients to stabilize them and escape from local maxima with poor transferability (see Section 2.4.4 for more details). Lin et al. [LSH+20] ameliorate upon MI with Nesterov Iterative attack (NI). VT [WH21] smooths the update using the average gradients over the data augmentation described above. Zhao et al. [ZLL] show that a significant increase in success rate can be obtained from the unmodified I-FGSM simply by performing more iterations, especially for targeted attacks. Qin et al. [QFL+22] propose Reverse Adversarial Perturbation (RAP) to find flat adversarial examples by solving a min-max bi-level optimization problem, similar to Sharpness-aware Minimizer (SAM) but in the input space (see Chapter 3 for more details on SAM). Wang et al. [WWY] develop a variant of MI by adding momentum over the spatial component of the image to stabilize further the updates. Wang et al. [WLH+21] offer another variant of MI. These optimizers prove successful on their own or in combinations of data or model augmentation to smooth the updates during the attack.

Loss function for transferability. Another type of transferability technique focuses on the loss function used to perform gradient ascent with I-FGSM. Zhou et al. [ZHC+18] develop a new loss called TAP that includes an additional regularization component to penalize high-frequency perturbations. Wang et al. [WRL+21] improve transferability with the addition of a penalization to the interactions between pixels in the adversarial perturbation.

Intermediate representation for transferability. Another line of research to improve the transferability of adversarial example was to target the intermediate features of DNNs, i.e., computing the gradients from an intermediate layer instead of the last one. This type of attack may require to adapt the attack loss, similarly to the previous category, but here the objective is to attack directly the intermediate layers. The intermediate representations are known to be more generic. TAP [ZHC+18] also adds a component to the loss to maximize the distance of intermediate feature maps. Inkawhich et al. [IWL+19] compute the adversarial perturbation by matching the internal representation of the perturbed image with the internal representation of another image of the targeted class. Huang et al. [HKG+19] fine-tune an initial perturbation computed on the last layer, using an intermediate layer. Wang et al. [WGZ+21] and Zhang et al. [ZWH+22] perturb only important features that are

selected based on, respectively, gradients information, and a neuron attribution method.

Generative modelling for transferability. Finally, a more distinct line of research on transferability is to use generative modelling to produce transferable adversarial examples. Poursaeed et al. [PKG+17] propose to use generative adversarial network (GAN) to output image-agnostic and image-dependent perturbations for targeted and non-targeted attacks. In particular, a surrogate model is used as a discriminator and a generator is trained using the cross-entropy loss. Naseer et al. [NKK+19] improve the training of the generator with another loss function. Naseer et al. [NKH+21] propose another generative approach that improves over the first two generative techniques presented here, for targeted adversarial examples. Zhang et al. [ZLC+22] and Fang et al. [FLL+22] propose other GAN variants to output more transferable adversarial examples. Generative modelling techniques are generally considered separately to the other categories, since they are not variants of I-FGSM, and therefore are hard to combine with techniques of other categories.

Overall, we provide a comprehensive typology of transferability techniques. These categories are of particular importance to design suitable benchmarks of transferability techniques. A good practice pointed by Zhao et al. [ZZL+22] is to only compare competitively transferability techniques of the same category, since they aim at the same objective. Two transferability techniques belonging to different categories should be evaluated complementarily.

2.5 Summary

This chapter presented the technical background that serves as a foundation for the contributions of this dissertation. Overall, this chapter briefly reminds fundamental concepts of machine learning, neural networks, and adversarial machine learning. Finally, we introduce the building blocks of this dissertation regarding the transferability of adversarial examples: the seminal papers, a definition, the metrics, the current hypothesis about the causes of transferability, standard attacks, and a typology of transferability techniques.

3

Related Work

This chapter discusses the existing work related to the contributions of the dissertation.

Contents

This chapter presents the work related to this dissertation. In particular, we give an overview of the existing transferability techniques, and their relation to ours. We summarize the previous geometrical approaches of transferable adversarial examples, and of DNNs in general. In particular, we describe research about the flatness of the natural loss, which receives recently an important attention by the scientific community. Next, we present Bayesian and ensemble approaches to adversarial examples and to natural accuracy. Finally, we present the studies of SGD with constant learning rate.

3.1 Transferability Techniques (Chapters 4 to 6)

We discuss the relation of this dissertation to the transferability techniques based on the typology developed in Section 2.4.5: our work is either complementary or in competition with other transferability techniques. Chapters 4 and 6 propose new techniques that all belong to the surrogate training category. Chapter 5 proposes a model augmentation technique based on training.

3.1.1 Complementary Techniques to Training (Chapters 4 to 6).

Among the categories presented in Section 2.4.5, input augmentation, optimization, loss function, intermediate representation, tackle different objectives to ours. They transform the model or the input at test time, i.e., after training, when performing the attack. These variants of I-FGSM alter distinct elements of the attack than ours, and are therefore straightforward to implement concurrently [ZZL+22]. Input augmentation transforms the inputs to find invariant adversarial perturbations, or to smooth the loss landscape. As such, input augmentation techniques can be directly applied on an improved single surrogate model, as in Chapter 6, or on multiple models, such as the ones outputted by our techniques in Chapters 4 and 5. Both approaches are complementary. Similarly, transferability techniques based on optimization can be directly applied on ours. These attack optimizers prove successful to smooth the updates during the attack on their own, in combinations of data augmentation, or when applied on multiple surrogate DNNs [XZZ+19; LBZ+18]. Loss functions for transferability add some regularization terms to the loss to find more generic adversarial perturbations. Replacing the loss functions in I-FGSM is complementary to our approaches, which do not alter the standard cross-entropy loss. Attacking the intermediate representations of the transferable representations evaluated in Chapter 6 is a promising extension to our work. Chapter 6 propose ways to train more transferable representations. Therefore, attacking the intermediate layers of these better surrogate representations should further improve our line of research, since intermediate representations are known to be more generic and less specialized in a task. Attacking the intermediate layers of

the multiple DNNs outputted by our techniques in Chapters 4 and 5 complements them.

If it is straightforward to complement our work with the variants of I-FGSM presented above, the relation of this dissertation to generative modelling is less direct. Since this category does not rely on I-FGSM, such techniques have a more separated lineage in the scientific literature compared to the other categories. To the best of our knowledge, the complementarity of generative modelling with the other categories of transferability techniques has not been studied previously. However, numerous generative approaches of transferability use a surrogate model as the discriminator in the GAN framework (cf. Section 2.4.5). We think that our techniques to train better surrogate models might be used for the discriminator, and in turn train a better generator model. We left for future research the comprehensive evaluation of the complementarity of our work with generative approaches to transferability.

Despite the complementarity of our work with the previously listed categories of transferability techniques, comparing them competitively indicates which objective gives the highest boost of transferability. As such, our Chapters 4 and 5 find a more effective way to do so, than other categories of transferability techniques.

3.1.2 Model Augmentation Techniques (Chapters 4 to 6)

Transferability techniques based on model augmentation are complementary to our work in Chapters 4 and 6, and competitive to our LGV technique proposed in Chapter 5.

Chapters 4 and 6 propose training techniques to train respectively several and one surrogate models. As such, transferability techniques based on model augmentation are complementary to ours, since they can be directly applied on top of each of our surrogate models. These techniques transform the model at test time, i.e., after training, when performing the attack, using cheap transformation of the weights or of the architecture. Following the approach of Fort et al. [FHL19], described below in Section 3.2.2 and summarized in Figure 3.1, the training techniques evaluated in Chapter 4 samples different modes of the loss landscape. Whereas, model augmentation techniques explore locally each mode of the loss landscape. For example, Fort et al. [FHL19] show that dropout at test time, such as used by Li et al. [LBZ+18] for transferability, captures only the local uncertainty of a mode. As illustrated by Figure 3.1, the techniques that train models from different modes are complementary to the techniques that sample locally in the weight space. Moreover, these techniques can naturally be combined to ours in Chapters 4 and 6: (i) cSGLD can provide at a low computation cost a diverse set of base models to build GN [LBZ+18]; SGM modifies backward passes during the attack, independently of the training method. As our evaluation in Chapters 4 and 6 will reveal, our train-time methods further improve the transferability of the

above techniques.

Our LGV technique proposed in Chapter 5 is of the model augmentation category, and shows the key role of the local exploration of the weight space. LGV augments the base surrogate model by fine-tuning with a high learning rate. As revealed by Section 5.4, our LGV approach alone consistently beats SGM [WWX+20] and GN [LBZ+18] by a large margin. SGM favours the gradients from skip connections rather than residual modules, and Wu et al. [WWX+20] claims that the formers are of first importance to generate highly transferable adversarial examples. We find the local loss geometry to have such relevance. In line with our results in Chapter 5, residual connections flatten the natural loss [YGK+19] and increase transferability. GN [LBZ+18] uses dropout or skip connection erosion to augment the base model, and identifies the diversity of surrogate models as key. Our results strongly suggest that exploring locally the loss landscape in the weight space is at most importance to avoid adversarial examples overfitting to their surrogate model. Neither SGM nor GN improve LGV when combined, suggesting that they may be poor local loss geometry proxies. This is also suggested by Fort et al. [FHL19], as explained in the previous paragraph.

3.1.3 Training Surrogate Models (Chapters 4 to 6)

Despite the important amount of work on transferability as established in Chapter 1, the way to train an effective single surrogate base model has received little attention in the literature [ZZL+22]. To the best of our knowledge, a single article before our first work tackles how to improve transferability during the training phase: Liu et al. [LCL+17] show that an ensemble of architectures is a better surrogate model than a single architecture. They retrieve one standardly trained model for several architectures. Our work leans on theirs and complements it by demonstrating that exploring the weight space of one architecture improves transferability. The techniques evaluated in Chapters 4 to 6 operate on a single weight space (one architecture), so one can apply them to collect models of multiple architectures as Liu et al. [LCL+17]. For example, Section 4.4.3 shows that our weight space exploration approaches complement nicely the ensembling of several architectures. The work before our first contribution simply uses standard training methods to obtain a surrogate model. We believe that there were *no prior contributions about how to train each architecture* for transferability.

Benz et al. [BZK21], Nitin [Nit21], and Zhang et al. [ZCB+21] point that early stopping SGD improves transferability. Springer et al. [SMK21] propose SAT, slight adversarial training that uses tiny perturbations to filter out some non-robust features. Our work in Chapter 4 that analyse transferability with a probabilistic perspective of the weight space, happens before or in parallel to the above-mentioned papers [SMK21; BZK21; Nit21; ZCB+21], since we published the first version of Chapter 4 on the arXiv platform on November 2020. The exploration of the weight

space of one architecture for transferability was at the state of the simplest baseline before our first contribution. Chapter 5 proposes a model augmentation technique that is complementary to these training techniques. Chapter 6 evaluates in-depth early stopping for transferability, by studying extensively the dynamics of training, and the hypothesis developed by Benz et al. [BZK21], Nitin [Nit21], and Zhang et al. [ZCB+21] to explain why early stopping improves transferability. and SAT (see the next section for a more detailed discussion). Section 6.5 evaluate competitively early stopping, SAT, and LGV-SWA (Chapter 5) the weight average (Stochastic Weight Averaging (SWA)) of the models collected by LGV, that all produce a single model, i.e., a transferable representation.

Our Chapters 5 and 6 shed new light on the relation between flatness and transferability. Springer et al. [SMK21] implicitly flatten the surrogate model, since adversarial trained models are flatter than their naturally trained counterparts Stutz et al. [SHS21]. We observe a similar implicit link with early stopping in Section 6.4. Our Chapter 5 proposes the surrogate-target misalignment hypothesis to explain why flat minima in the parameter space are better surrogate models. In Chapter 6, we improve on the single-model baseline (LGV-SWA) of Chapter 5 by explicitly minimizing sharpness with RFN, and we show that LGV, the full model augmentation technique of Chapter 5, is complementary to RFN.

Recently, posterior to our contributions in Chapters 4 and 5, Li et al. [LGZ+23] developed another Bayesian formulation of transferability, and proposed to use Stochastic Weight Averaging-Gaussian (SWAG) to fine-tune a regularly trained model. Their Bayesian approach is different but overlaps ours, developed in Chapter 4. Their proposed approach boils down to LGV with SWAG applied on top. We already evaluated the use of SWAG for transferability in Chapter 4. They report improved success rates compared to LGV. However, their comparison appears to be unfair, since it seems that their technique relies on a much higher number of gradients per attack iteration. Zhao et al. [ZZL+22] show that a fair comparison must compare transferability techniques with the same number of gradients per iteration. In our preliminary experiments, we reported some significant improvements when averaging gradients over an increasing number of LGV models. Furthermore, we are sceptical that their technique improved upon LGV because we actually discovered LGV in our preliminary experiments by observing that SWAG slightly decades the transferability compared to its base models. We found that the transferability of the base models collected by SWAG, i.e. LGV with a standard learning rate, was higher than the transferability of the samples from the SWAG posterior built on these same models. We made the same observation for Subspace Inference (SI).

3.1.4 Early Stopping for Transferability (Chapter 6).

Several works [BZK21; ZCB+21; Nit21] point out that fully trained surrogate models are not optimal for transferability. To explain this observation, they propose a hypothesis based on *the perspective of robust and non-robust features (RFs/NRFs)* from Ilyas et al. [IST+19]. Ilyas et al. [IST+19] disentangle features that are highly predictive and robust to adversarial perturbations (RFs), and features that are also highly predictive but non-robust to adversarial perturbations (NRFs). According to Benz et al. [BZK21] and Nitin [Nit21], the training of DNNs mainly learns RFs first and then learns NRFs. NRFs are transferable [IST+19], but also brittle. RFs in a tiny input neighbourhood, called *slightly RFs*, improve the transferability of larger adversarial examples [ZCB+21; SMK21]: the input neighbourhood is sufficiently small for an attack to find adversarial examples in a larger radius, and slightly RFs are less brittle than NRFs. Models at earlier epochs would be composed of more slightly RFs, thus being better surrogate models. Section 6.3 provides some observations that tend to refute this hypothesis. Instead, Sections 6.4 and 6.5 suggest that the success of early stopping is correlated with the dynamics of the training and the flatness of the surrogate.

3.2 Geometrical Analysis of Adversarial Examples and DNNs (Chapters 5 and 6)

3.2.1 Geometry of Transferable Adversarial Examples

Previous studies [TPG+17; CRP20] analyse the geometry of transferable adversarial examples to gain insights on the input space. Whereas, our Chapters 5 and 6 study them from the perspective of the weight space and provide both insights to improve surrogates and actionable methods. On MNIST, [TPG+17] shows that among the 44 dimensions adversarial input space, a dense 25 dimensions subspace is shared between models, thus enabling transferability. [CRP20] proves with a geometric perspective that transferable adversarial directions exist with high probability for linear classifiers trained on independent sets drawn from the same distribution.

3.2.2 Geometry of the Weight Space of DNNs

Numerous work study the generalization of DNNs and SGD with a geometric perspective. [LFL+18] establishes that the intrinsic dimension of the objective landscapes is smaller than expected by applying SGD in a randomly oriented parameter subspace. Despite the high dimensionality of the weight space, [GRD18] observes that SGD happens in a tiny parameter subspace, which is mostly preserved during training. Our analysis in Section 5.6 relates to the work of Gur-Ari et al. [GRD18] but ours is focus on transferability not natural generalization. We analyse

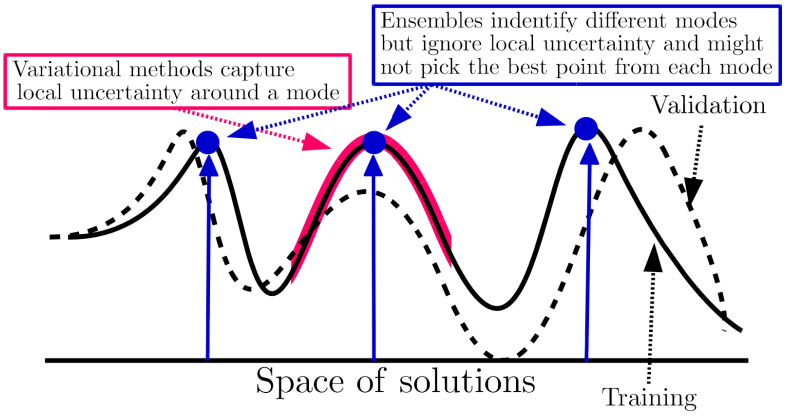


Figure 3.1: Illustration of the hypothesis of Fort et al. [FHL19]. Deep ensemble produces a set of diverse representations from different modes. Some training techniques, such as VI, capture the local uncertainty inside each mode. This hypothesis supports our explanation in Chapter 1 regarding the three complementary ways to explore the weight space proposed in Chapters 4 to 6. The x-axis represents the weight space and the y-axis plots the loss.

in-depth the transferability from the subspace spanned by SGD with high learning rate.

Fort et al. [FHL19] provide insight on deep ensemble from the perspective of the weight space. Figure 3.1 illustrates their hypothesis. Deep ensemble contains independently trained DNNs that land in different modes. The model from one mode disagrees with the model from other mode, showing that deep ensemble average a diverse set of predictions. Despite the roughly similar individual accuracies of each DNN, the authors exhibit an important variety in the learnt representations between modes of the loss landscape. This view supports our explanation regarding the complementary ways to explore the weight space, described in Chapter 1. Chapter 4 evaluates deep ensemble to obtain several base surrogate models, each from a different mode, containing a variety of representations. Chapter 5 proposes LGV to locally explore the mode of a base surrogate model, capturing the uncertainty around a mode, as illustrated by Figure 3.1. Finally, Chapter 6 evaluate which mode to choose, showing that some modes are more desirable than other, to obtain a transferable representation.

3.2.3 Flatness and Transferable Adversarial Examples

A few papers relate flatness and smoothness to the transferability of adversarial examples. As concomitant work to ours, Qin et al. [QFL+22] propose Reverse Adversarial Perturbation (RAP). They formulate a min-max bi-level optimization problem, similar to SAM but in the input space and to craft transferable adversarial

examples. This variant of I-FGSM adds an extra gradient at each attack iteration to compute the worst-case perturbation, i.e., the reverse adversarial perturbation. Qin et al. [QFL+22] propose an analysis similar to our surrogate-target misalignment hypothesis (Chapter 5). These ideas emerged in parallel, as both works were under review during the same period (NeurIPS 2022 for [QFL+22], ECCV 2022 for our work in Chapter 5). This independent development of related hypotheses highlights the significance and relevance of the research topic. Nevertheless, both work differ in the methods to leverage flatness. Qin et al. [QFL+22] propose a new optimizer in the input space, a variant of I-FGSM for transferability. Our Chapter 5 proposes LGV, a model augmentation technique that collects models by exploring the weight space. We develop the surrogate-target misalignment hypothesis to explain LGV, but LGV does not explicitly minimize sharpness. Our Chapter 6 analyses the dynamics of training and proposes RFN, based on SAM which predates RAP [QFL+22], to explicitly minimize sharpness in the weight space during training. RAP [QFL+22] is to be applied on a regularly trained surrogate model. RFN and RAP are therefore complementary techniques: the former is a surrogate training technique for the weight space, and the latter is an attack optimizer for the input space.

Prior to our work, Wu et al. [WZT+18] propose the transferability technique called Variance Reduced (VR) that smooths the loss of the surrogate model by averaging gradients under Gaussian noise in the input space, at each attack iteration. They analyse the success of VR by showing that VR smooths the surrogate loss in the input space, which reduces the effect of shattered gradients illustrated in Figure 3.2. Gradients from the surrogate model are noisy, and therefore not well aligned with the gradients of the target model. Their VR method is complementary to ours, since averaging gradients of LGV weights also improves transferability. VR is an input augmentation technique, complementary to LGV, our model augmentation, and RFN, our surrogate training technique. In Section 5.3, we show that Gaussian noise in the input space does not improve transferability on its own, i.e., without averaging them, contrary to Gaussian noise, LGV and RFN in the weight space. Their analysis of smoothness complements ours of sharpness (Chapters 5 and 6), and the intersection of both might lead to new direction for future work.

3.2.4 Flatness and Natural Generalization

The relationship between natural generalization and the flatness of the solution in the weigh space of DNNs has been studied extensively. Nevertheless, this topic is currently subject to a *scientific controversy*. Some papers show that flatter solutions in the weight space have lower natural generalization gap. A sharp minimum (the opposite of a flat minimum) is one where the variations of the objective function in a neighbourhood are important, whereas a flat minimum shows low variations [HS97]. Sharpness, the opposite of flatness, can be defined as the highest increase

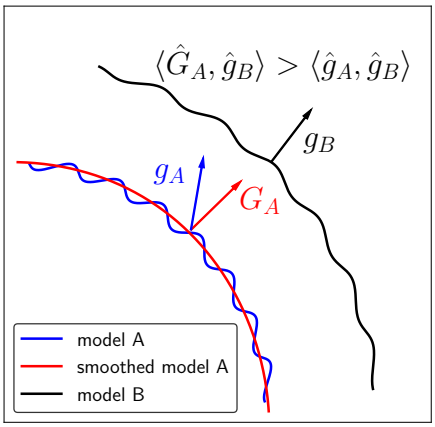


Figure 3.2: Illustration of the concept of shattered gradients from Wu et al. [WZT+18]. The gradient g_A from the surrogate model A is noisy and hurts transferability to the target model B, since their gradients are not well aligned. Wu et al. [WZT+18] smooth the loss of the surrogate model A by averaging gradients under Gaussian noise in the input space, at each attack iteration. After smoothing, the gradients of the surrogate model have a higher cosine similarity with the gradients from the target model. Their approach is complementary to ours, since averaging gradients of LGV weights also improves transferability.

in the training loss $\mathcal{L}_\mathcal{D}$ in a small neighbourhood in the weight space [FKM+20]:

$$\max_{\|\gamma\|_2 \leq \rho} \mathcal{L}_\mathcal{D}(w + \gamma) - \mathcal{L}_\mathcal{D}(w) \ .$$

Keskar et al. [KNT+16] correlate large-batch SGD to both sharp solutions and a generalization gap compared to small-batch SGD. Flat solutions would be desirable because under a shift between training and test losses, the generalization gap would remain small for flat minima, whereas it would be high for sharp ones. Keskar et al. [KNT+16] give another interpretations of why flatness would be desirable: under an information theory perspective and using the minimum description length theory, flat minima can be described with fewer bits of information, i.e., are of lower complexity [Ris83]. Less complex statistical models would generalize better. Chaudhari et al. [CCS+16] propose Entropy-SGD to find wide valleys and develop another explanation through the lens of free Gibbs energy. Izmailov et al. [IPG+18] show that averaging weights along the trajectory of SGD iterates leads to wider optima and better natural generalization than SGD. Xing et al. [XAT+18] show that high learning rates and small batch sizes drive SGD towards flatter minima with better generalization. Some techniques explicitly minimize the sharpness of the loss for natural [FKM+20; KKP+21; ZGY+22] or robust generalization [WXW20]. Kaddour et al. [KLS+22] benchmark such techniques in a variety of application

domains and derive insights from the loss landscape. Recently, Möllenhoff and Khan [MK22] develop a Bayesian interpretation of SAM proposed by Foret et al. [FKM+20].

Our work in Chapters 5 and 6 differs in that we study flatness under the scope of transferability. We adapt the explanation of Keskar et al. [KNT+16] about the shift between training and test losses for natural accuracy, to transferability by proposing our surrogate-target misalignment hypothesis in Section 5.5. Figure 5.10 illustrates this adaptation. We use the Hessian-based sharpness metrics developed by Yao et al. [YGK+19] in Chapter 5. We iterate on some work listed above by evaluating SWA [IPG+18], SAM [FKM+20] and two SAM variants [KKP+21; ZGY+22], in the context of transferability respectively in Chapter 5 and Chapter 6.

On the other side of the controversy, some works point to the limitations of sharpness either as a metric or as a cause for the reduced generalization gap. Dinh et al. [DPB+17] show that some scaling-based reparametrizations can make any minimum arbitrary sharp, while representing the same function. Therefore, there are some cases where sharpness is a meaningless metric. Zhang et al. [ZRP+21] show that the correlation between flatness and generalization might be spurious, due to the confounding factor of the volume of a function consistent with the training data. This volume can be defined as a Bayesian prior, which is inversely correlated to the complexity of the function [PLC18; MSV+19]. Andriushchenko and Flammarion [AF22] analyse the implicit bias of SAM. Recently, Andriushchenko et al. [ACM+23] propose a new reparametrization-invariant measure of sharpness, and show that the correlation between flatness and generalization does not hold in modern experimental settings. Similarly, Kaur et al. [KCL22] exhibit experimentally cases where a smaller sharpness increases the generalization gap. Andriushchenko et al. [AVP+22] study SGD with a high learning (such as the one used by LGV developed in Chapter 5) and show an implicit regularization effect that leads to sparse features. Li et al. [LWM19] also show a regularization effect of large learning rates. Overall, an increasing number of work show correlate flatness with regularization.

During our extensive preliminary work of Chapter 6, after our work of Chapter 5, we did our best to analyse transferability from the other perspective of the controversy. We extensively experimented with various types of regularization to balance the explanation of LGV in terms of flatness developed in Section 5.5. But we found no evidence of a particular link between regularization and transferability. In relation to Andriushchenko et al. [AVP+22] who show that large learning rates have an implicit sparse regularization effect (instead of the flattening effect shown by Xing et al. [XAT+18]), we trained several surrogate models with L_1 regularization, which favours sparse weights [GBC16]. Despite our best attempt at tuning the hyperparameter controlling the strength of the L_1 regularization, we found

systematically a significative lower transferability from these surrogates compared to a standard surrogate trained with L_2 regularization. Moreover, we explored in-depth if the strength of the L_2 regularization (i.e. weight decay) has an effect on transferability. This is particularly interesting, since the weight decay is related to the covariance matrix of the Bayesian prior (see Chapter 4). If the weight decay optimal for transferability is smaller than the optimal weight decay for natural generalization, then a broader prior is beneficial to the surrogate. However, we found that the best weight decay for the transferability of the surrogate is the same as the best one for natural accuracy on CIFAR-10, even when ensembled. Then we rejected our preliminary hypothesis that an ensemble of simpler representations is a better surrogate, which was suggested by the transferability gained from dropout [LBZ+18], from the linearity in the backward pass [WWX+20; GLC20], from the large learning rate used by LGV (Chapter 5, [LWM19; AVP+22]). Additionally, and contrary to Wu et al. [WZT+18], we found no evidence that smaller architectures, i.e., less complex functions, are better surrogates. We studied the transferability from and to different sizes of architecture (for five families of architectures). We found that the closer the size of the surrogate to the size of the target, the higher transferability is, i.e., smaller architectures are not better surrogates. We also compared the transferability of each pair of sizes. If simpler functions transfer better, then the transferability from small to big architectures should be higher than the transferability from big to small architectures. We found evidence that contradicts this hypothesis most of the time. Instead, during our preliminary experiments for Chapter 6, we quickly found strong and reliable transferability improvements from SAM that explicitly minimize sharpness. This finding echoes the transferability from slight adversarial training [SMK21], which has an implicit flattening effect [MFU+19]. Nevertheless, we think that many directions remain to be explored, and that the community studying transferability should follow the evolution of this controversy.

3.3 Bayesian and Ensemble Approaches (Chapter 4)

3.3.1 Ensemble and Transferable Adversarial Examples

As stated previously, Liu et al. [LCL+17] show the benefit of ensembling architectures for inter-architecture transferability. Our Chapter 4 complements their by exploring how to build an ensemble of a single surrogate architecture. We show in Section 4.4.3 that transferability is higher by applying our ensembling and Bayesian techniques on several surrogate architectures. Therefore, both techniques complement each other nicely.

3.3.2 Bayesian Neural Network and Adversarial Examples

Though not our goal, past research aimed at generating adversarial examples for BNNs (we rather use Bayesian Deep Learning as a way to attack deterministic DNNs). Grosse et al. [GPS+18] show that BNN uncertainty measures are vulnerable to high-confidence-low-uncertainty adversarial examples crafted on Gaussian Processes. Palacci and Hess [PH18] show that several SG-MCMC sampling schemes are not secure against white-box attacks. Wang et al. [WVL+18] use SGLD and Generative Adversarial Network to detect adversarial examples instead of crafting them.

Carbone et al. [CWL+20] claim that BNNs are robust against gradient-based attacks because gradients vanish in expectation under the true posterior distribution. Their conclusions hold theoretically under the restrictive assumption of the large-data overparametrized limit, and experimentally for HMC and VI on MNIST and Fashion MNIST. In Section 4.4.2, our experiments reveal opposite conclusions about cSGLD: our surrogates DNNs suffer more often from vanished gradients than our cSGLD surrogates. On MNIST, we observe that 60.6-86.6% of individual gradients of HMC or VI vanish before averaging them. Therefore, the theoretical development of Carbone et al. [CWL+20] does not seem to explain most gradient vanishing. Furthermore, VI on larger datasets (ImageNet and CIFAR-10) do not suffer from vanishing gradients.

3.3.3 Bayesian and Ensemble Training Techniques

Following the work of Ashukha et al. [ALM+20], we consider the following training techniques in Chapter 4: Deep Ensemble [LPB16], cSGLD [ZLZ+20], SWAG [MGI+19a], VI, Snapshot Ensembles (SSE) [HLP+17], and Fast Geometric Ensembling (FGE) [GIP+18]. These training techniques were developed for natural accuracy. Chapter 4 in general, and Section 4.4.6 in particular, evaluate the transferability and the computational cost of these training techniques. These training techniques are the building blocks that we apply to a new problem they were not designed to solve initially.

Deep Ensemble. Deep ensemble [LPB16] simply trains several DNNs independently with random initialization and random subsampling (mini-batch on shuffled data in practice). All DNNs have the same standard hyperparameters for training. For classification, the predictions of individual DNNs are averaged. In Chapter 4, we train 15 PreResNet110, 4 PreResNet164, 4 VGG16bn, 4 VGG19bn, and 4 WideResNet28x10 DNNs on CIFAR-10. We retrieve 15 ResNet50 DNNs trained by Ashukha et al. [ALM+20] on ImageNet, and trained on our own 1 DNN for each of the remaining studied architectures (ResNeXt50 32x4d, DenseNet121, MNASNet 1.0, and EfficientNet-B0).

cSGLD. We refer the reader to Section 4.2 for a detailed description of cyclical Stochastic Gradient Langevin Dynamics (cSGLD). Figure 4.2 illustrates both the cyclical cosine annealing learning rate schedule and the separation of each cycle into an exploration phase (called the burn-in period of MCMC algorithm) and a sampling phase.

SWAG. Stochastic Weight Averaging-Gaussian (SWAG) [MGI+19a] is a Bayesian deep learning method that fits a Gaussian onto SGD iterates to approximate the posterior distribution over weights. Its first moment is the SWA solution, and its second moment a diagonal plus low-rank covariance matrix. Both are estimated from SGD iterates with constant learning rate (0.001 on ImageNet and 0.01 on CIFAR-10). On ImageNet, SWAG performs 10 additional epochs to collect SGD iterates from one of the Deep Ensemble DNNs. On CIFAR-10, a regular pre-training phase of 160 epochs precedes 140 epochs to collect checkpoints. Once fitted, models are sampled from the Gaussian distribution. For every sample, batch normalization statistics are updated in a forward pass over the entire CIFAR-10 train set and over a random subset of 10% on ImageNet. Apart from the fixed initial cost, the marginal computational cost to obtain a sample is very low. In Chapter 4, we sample a maximum of 50 models because iterative attacks perform 50 iterations of one model per iteration, and further samples would be discarded. Thus, the lines corresponding to SWAG in Figures 4.8 and 4.9 are shorter than the ones of other methods. The rank of the estimated covariance matrix is 20. Batch-size is 128 on CIFAR-10, and 256 on ImageNet.

VI. Variational Inference (VI) approximates the true posterior distribution with a variational approximation, here a fully-factorized Gaussian distribution, and maximizes a corresponding lower bound. A Gaussian prior is chosen. Once trained, the variational approximation is used as the posterior. There is no additional sampling phase to perform Bayesian model averaging. Therefore, we cannot tune the number of samples and a single VI point is plotted in Figures 4.8 and 4.9 (Chapter 4). We follow the solutions of Ashukha et al. [ALM+20] to avoid underfitting: pre-training and annealing of β. The first moment of the Gaussian variational approximation is initially set to a DNN pre-trained similarly to Deep Ensemble (300 epochs on CIFAR-10 with initial learning rate of 10^{-4}, and 130 epochs on ImageNet starting at 10^{-3}). The log of its second moment is initially set to -5 on CIFAR-10 and -6 on ImageNet, and further optimized for 100 epochs (45 on ImageNet) with Adam and a learning rate of 10^{-4}. β is set to 10^{-5} on CIFAR-10 and 10^{-4} on ImageNet. Batch-size is 128 on CIFAR-10, and 256 on ImageNet. On MNIST, we train VI using the code and the hyperparameters of Carbone et al. [CWL+20].

SSE. Snapshot ensembles technique (SSE) [HLP+17] is the foundation of cSGLD. The learning rate is cyclical with a cosine annealing schedule. Contrary to cSGLD, SSE saves a single snapshot per cycle and does not add gradient noise. In Chapter 4, the cycles are 40 epochs long on CIFAR-10, 45 on ImageNet. The maximum learning rate is 0.2, batch size is 64 on CIFAR-10, respectively 0.1 and 256 on ImageNet.

FGE. Fast Geometric Ensembling (FGE) [GIP+18] is a method developed after the empirical observation of Mode Connectivity on CIFAR-10 and CIFAR-100: it's possible to find a path in the parameters space that connects two independently trained DNNs such that the models along the path have low loss and high test accuracy. In practice, it uses a cyclical triangular learning rate and collects one model during each cycle. It is quite similar to SSE, except for the learning rate schedule, the much shorter cycles (4 epochs on CIFAR-10, 2 epochs on ImageNet), and a pre-training phase. Pre-training lasts for 160 epochs on CIFAR-10. On ImageNet, FGE is initialized from one Deep Ensemble checkpoint. The learning rate varies between 5×10^{-5} and 5×10^{-3} on CIFAR-10 and 10^{-6} and 10^{-4} on ImageNet. Batch-size is 128 on CIFAR-10, and 256 on ImageNet.

HMC. Hamiltonian Monte Carlo (HMC) is considered a golden standard to train BNN. In Chapter 4, we trained the small Fully Connected (FC) architecture on MNIST, using the code and the hyperparameters of Carbone et al. [CWL+20]. Unfortunately, HMC does not scale to larger DNNs, even on MNIST.

3.4 SGD With Constant Learning Rate (Chapter 5)

Chapter 5 proposes LGV which rests upon sampling weights along the trajectory of SGD with constant learning rate. This idea has been explored to improve natural accuracy or calibration in deep learning [MHB17; IPG+18; MGI+19a]. [MHB17] proves that under some assumptions, SGD with constant learning rate simulates a Markov chain with a stationary distribution, which can be tuned to approximate the Bayesian posterior. Our results in Chapter 5 corroborate the relationship between the posterior predictive distribution and transferability established in Chapter 4, with a new plug-in technique. LGV is inspired by the SGD trajectories used in SWA [IPG+18], SWAG [MGI+19a], and SI [IMK+19]. A key difference is that LGV uses a higher learning rate to improve attack transferability that degrades the natural accuracy of the surrogate ensemble (Figure 5.5). We analyse extensively SWA on top of our LGV surrogate in Chapter 5. We observed during our preliminary experiments that applying SWAG or SI on top of our LGV surrogate degrades transferability.

3.5 Summary

The work in the literature that is related to the contributions of this dissertation was presented in this chapter. Overall, this chapter shows that the work regarding how to train a surrogate model is scarce. In particular, prior to our first contribution in Chapter 4, no previous research studied how to explore a surrogate weight space to improve the transferability of adversarial examples.

4

Transferability From Deep Ensemble and Bayesian Neural Networks, a Probabilistic Perspective

This chapter develops a probabilistic perspective on the transferability of adversarial examples. Since the target is unknown, we can treat its weights as random variable. Under a specified threat model, deep ensemble can obtain a surrogate with samples from the distribution of the target model that arises from the training noise. Since deep ensemble is costly, we propose an efficient approximation using cSGLD, a state-of-the-art Bayesian deep learning technique that samples from the posterior distribution. We evaluate the transferability of seven training techniques in total. Our extensive experimental evaluation shows the key role of training the surrogate model in transfer-based black-box attacks.

Contents

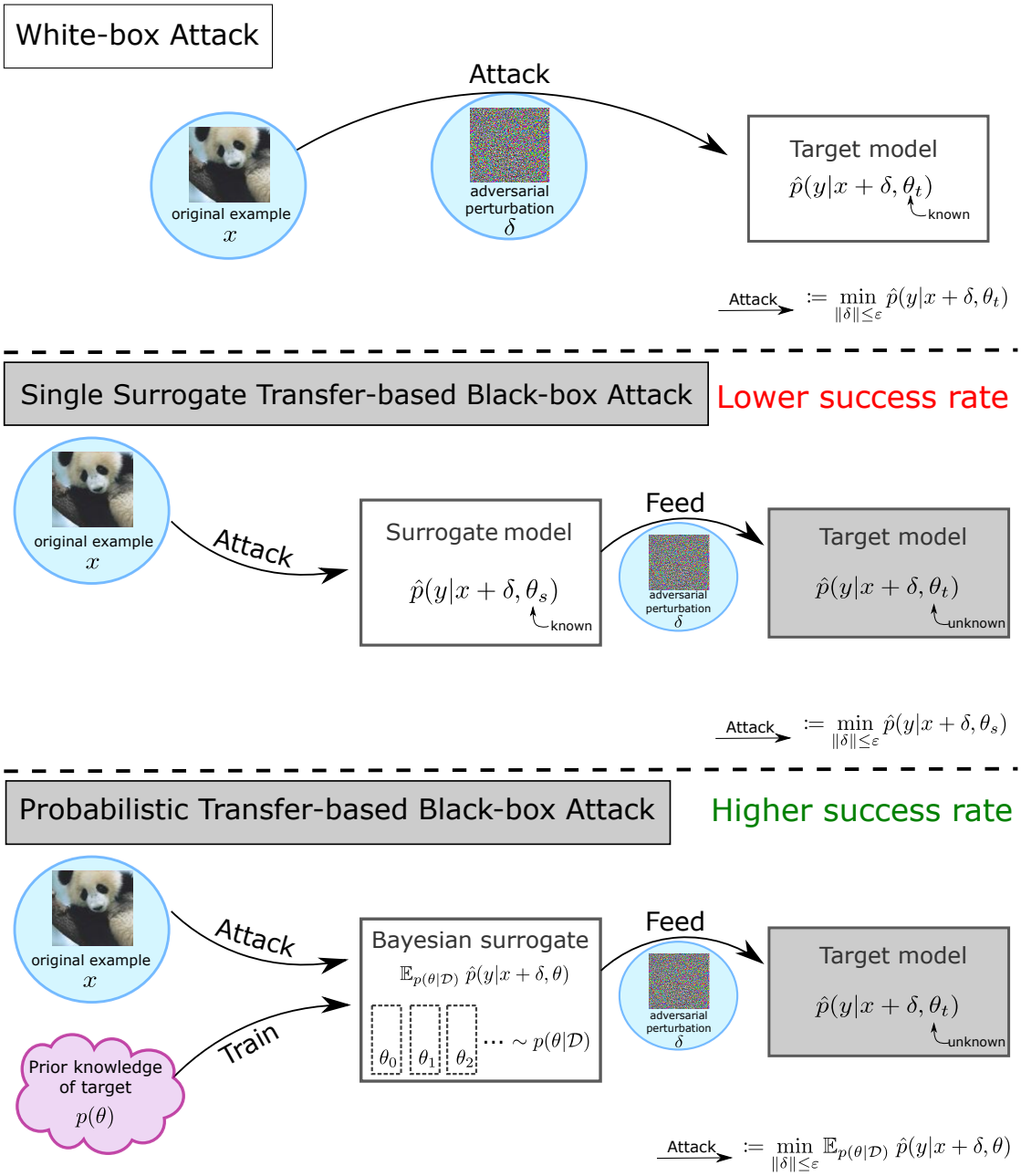


Figure 4.1: Illustration of the proposed approach.

This chapter is based on the following paper:

- Martin Gubri et al. Efficient and Transferable Adversarial Examples from Bayesian Neural Networks. In *UAI 2022*, 2022. URL: `https://gubri.eu/publication/transferable_adv_ex_from_bnn/`

4.1 Introduction

Adversarial attacks have been first designed in white-box settings, where the attacker is assumed to have complete knowledge of the target DNN (including its weights). While studying such worst-case scenarios is essential for proper security assessment, in practice the attacker should have limited knowledge of the target model. In such a case, the adversarial attack is applied to a surrogate model, with the hope that the crafted adversarial examples transfer to (i.e., are also misclassified by) the target DNN.

Achieving transferability remains challenging, though. This is because adversarial attacks were designed to optimize the loss function of a specific model [GSS14; KGB17], different from that of the target model. As a result, Liu et al. [LCL+17] improved transferability by attacking an *ensemble of architectures*, composed of one standardly trained model per architecture. The key intuition is that adversarial examples that fool a diverse set of models are more likely to generalize. However, the way to train each surrogate architecture to obtain a diverse set of surrogate representations is left unexplored.

In this chapter, we analyse the unknown target model with a probabilistic eye, and relate transfer-based attacks to uncertainty. We propose a new method to improve the transferability of adversarial examples using deep ensemble to train each surrogate architecture, complementarily to Liu et al. [LCL+17]. Then, we evaluate an approximate Bayesian inference to build a surrogate – and do so with less computation overhead compared to deep ensemble. Our approach, shown in Figure 4.1, leans upon recent results in Bayesian Deep Learning. More precisely, we train our surrogate with a cyclical variant of Stochastic Gradient Markov Chain Monte Carlo (i.e., *cSGLD* [ZLZ+20]) to sample from the posterior distribution of neural network weights. We then perform efficient approximate Bayesian model averaging during the attack with minimal modifications of the attack algorithms.

We evaluate our approach on the ImageNet, the CIFAR-10 and the MNIST datasets with a variety of DNN architectures, four adversarial attacks, and three test-time transformations. Overall, our results indicate that applying cSGLD significantly improves the success rate compared to training single DNNs and outperforms classical ensemble-based attacks in terms of computation cost. Deep Ensemble requires at least 2.51 times more flops to achieve the same success rates as cSGLD when the targeted architecture is known. This can represent, on ImageNet, a saving of 3.56 exaflops (2.36 vs 5.92). At constant computation costs, our method increases the intra-architecture transfer success rates between 1.6 and 82.0 percentage points and the inter-architecture transfer success rates between -2.3 and 83.2. cSGLD always raises the effectiveness of test-time techniques designed for transferability between 3.8 and 56.2 percentage points. Applied alone, it is more effective than these techniques applied to a single DNN in 105/120 cases.

To summarize, our contributions are:

- We relate uncertainty and transferability of adversarial examples with a Bayesian perspective. The posterior distribution represents a belief about the unknown target model.
- We are the first to propose and evaluate ways to explore a single surrogate weight space, i.e, to evaluate training methods to be applied on a surrogate architecture. We propose the first method based on a Bayesian deep learning technique to generate transferable adversarial examples. Existing iterative attacks can be easily modified to perform approximate Bayesian model averaging at no additional computational cost.
- We pave the way for improving surrogates at train-time by evaluating six Bayesian and ensemble techniques. cSGLD is a strong competitor, though other techniques open promising avenues.
- We advocate the use of a new metric, T-DEE, to compare the effectiveness of transferability techniques with the strong baseline of deep ensemble.
- Our evaluation on ImageNet, CIFAR-10 and MNIST reveals significant improvements over the single-DNN per architecture baseline in diverse experimental settings. Our train-time method improves existing test-time techniques, and is better in most cases on a competitive basis. We open new ways to understand transferability from the surrogate weight space.

4.2 Approach

A probabilistic perspective on transferability. Under a specified threat model, we relate uncertainty and posterior predictive distributions to transferability. We consider a classification problem with a training dataset $\mathcal{D} = \{(x_i, y_i) \sim p(x, y)\}_{i=1}^{N}$ and C class labels. A probabilistic classifier parametrized by θ maps x_i into a predictive distribution $\hat{p}(y|x_i, \theta)$. A white-box adversarial perturbation of a test example $(x, y) \sim p(x, y)$ against such classifier is defined as:

$$\delta_\theta \in \underset{\|\delta\|_p \leq \varepsilon}{\arg\min}\, \hat{p}(y|x + \delta, \theta).$$

In practice, this optimization problem is solved by replacing the predictive distribution with a loss function (see Chapter 2). The *transferability* phenomenon is the empirical observation that an adversarial example for one model is likely to be adversarial for another one [GSS14]. Black-box attacks can leverage this property by crafting adversarial examples using white-box attacks against a surrogate model to target an unseen model [PMG16].

Assumption 1 (Threat model)**.** We define our threat model with the following assumptions on the targeted classifier:

1. Its architecture is known and so is its prediction function $\hat{p}(y|x, \bullet)$ [1].

2. Its training set $\mathcal{D}$ is known.

3. Its parameters θ_t, estimated by maximum likelihood, are unknown.

4. A reasonable prior on its parameters $p(\theta_t)$ is known[2].

5. No oracle access (test-time feedback) is possible.

Assuming the threat model described in Assumption 1, *uncertainty on target parameters arises from the stochastic nature of training*, and more specifically from two sources of randomness: (i) every SGD update depends on a random batch of training examples[3], (ii) weights are randomly initialized at the beginning of training[4]. From the attacker subjective view, the target parameters obtained at the end of training are random variables.

Under this threat model, deep ensemble [LPB16] samples from the distribution of the parameters of the target model. Deep ensemble builds an ensemble from independently trained DNNs, where each model starts with an independent random initialization and is trained with independent training noise (random batch). Since the target architecture is known and the target weights θ_t were obtained from training on the same dataset with independent random initialization and random batch, deep ensemble samples from the distribution of the target parameters θ_t. Therefore, *deep ensemble is a natural way to build a surrogate* ensemble of a single DNN architecture.

We argue that θ_t is also approximately distributed according to the posterior distribution $p(\theta|\mathcal{D})$. Mingard et al. [MVS+20] observe a strong correlation between the probability to obtain with SGD or its variants a function consistent with a training set and the Bayesian posterior probability of this function. Mandt et al. [MHB17] show that SGD with constant learning rate has a stationary distribution centred on an optimum, which approximates a posterior. Marginalizing over local optima, we obtain a posterior that is the distribution of SGD endpoints with a step decay learning rate schedule (as widely used).

Then, *the best transferable adversarial example approximately minimizes the Bayesian posterior predictive distribution* $p(y|x, \mathcal{D}) = \mathbb{E}_{p(\theta|\mathcal{D})}\, \hat{p}(y|x, \theta)$ and our black-box attack objective is:

[1] We discuss the unknown architecture case further on.

[2] In practice, it corresponds to knowing the weight decay hyperparameter, see discussion below.

[3] The same argument holds for SGD variants.

[4] Despite being independent and identically distributed random variables, weights initialization values play an important role in guiding the SGD trajectory [FC19].

$$\delta^* \in \arg\min_{\|\delta\|_p \leq \varepsilon} \mathop{\mathbb{E}}_{\theta_t \sim p(\theta|\mathcal{D})} \hat{p}(y|x+\delta, \theta_t). \tag{4.1}$$

Usually in adversarial machine learning, transferable adversarial examples are optimized against one surrogate model. This is similar to solving problem (4.1) deterministically by approximating the expectation of the posterior predictive with a "plug-in" estimation of the parameters, $\hat{\theta}_{\text{MAP}}$ the maximum a posteriori probability (MAP) estimate: $\delta^* \approx \delta_{\hat{\theta}_{\text{MAP}}}$. To avoid overfitting to the surrogate model, random transformations of inputs or prediction functions were developed in the literature (see Chapter 3).

A fundamental issue is that the closed form of the posterior predictive distribution is intractable for DNNs. Our contribution lies in *sampling from the posterior distribution to build a surrogate in black-box adversarial attacks*. We replace the crude MAP approximation of the posterior predictive distribution with a more accurate one to generate transferable adversarial examples. Therefore, we focus on the training phase by considering the methods and the computational costs of obtaining the surrogate model, whereas most previous work searches to optimize adversarial examples crafting at the time of the attack ("test-time").

SG-MCMC & cSGLD. In practice, we perform Bayesian model averaging using samples obtained from Stochastic Gradient-Markov Chain Monte Carlo (SG-MCMC). SG-MCMC is a family of approximate Bayesian inference techniques, inaugurated by SGLD [WT11], that combines SGD with MCMC. Adding noise during training allows to sample from the posterior distribution of parameters. The empirical distribution of the samples approximates the posterior. Then, our method aims to solve the following optimization problem:

$$\delta_{\{\theta_s\}} \in \arg\min_{\|\delta\|_p \leq \varepsilon} \frac{1}{S} \sum_{s=1}^{S} \hat{p}(y|x+\delta, \theta_s) \ , \tag{4.2}$$

where $\{\theta_s \sim p(\theta|\mathcal{D})\}_{s=1}^{S}$ are samples of the posterior.

We choose to apply the recently proposed *cyclical Stochastic Gradient Langevin Dynamics* (cSGLD) [ZLZ+20], a state-of-the-art SG-MCMC technique. cSGLD performs warm restarts by dividing the training into cycles that all start from the initial learning rate value (cf. Figure 4.2). Each cycle consists of (1) an exploration stage with larger learning rates which corresponds to the burn-in period of MCMC algorithms; (2) a sampling stage that samples parameters at regular intervals and operates with smaller learning rates and added noise. Starting a new cycle with a large learning rate allows the exploration of another vicinity of the loss landscape. Contrary to most SG-MCMC methods, cSGLD has the compelling benefit of sampling from both several modes of the posterior distributions and locally inside each mode, avoiding mode collapse. Another major advantage of cSGLD is that its

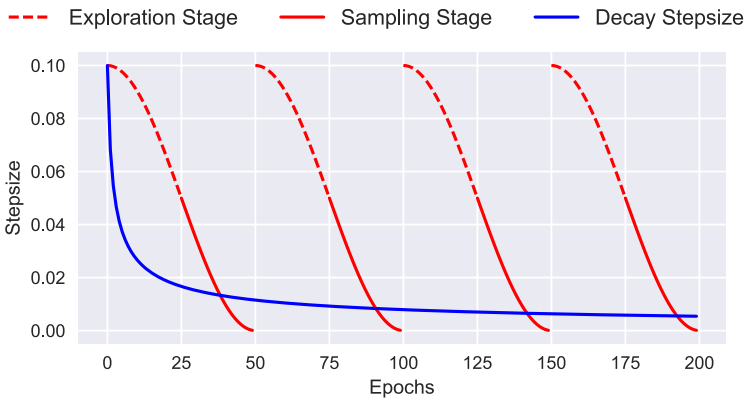


Figure 4.2: Illustration of the cSGLD cyclical learning rate schedule (red) and the traditional decreasing learning rate schedule (blue). Each cSGLD cycle is composed of an exploration phase (burn-in period of MCMC algorithms — red doted) and of a sampling phase (red plain). Figure taken from [ZLZ+20].

computation overhead compared to SGD/Adam is negligible (0.019% flops for one epoch on PreResNet110 on CIFAR-10 and 0.015% for ResNet50 on ImageNet).

Difference with ensembling. Our work differs from previous research [LCL+17; LBZ+18; XZZ+19] that relates diversity with transferability in the same way that Ensembling and Bayesian Model Averaging do [Min02]. The latter "assumes that the true model lies within the hypothesis class of the prior, and performs soft model selection [...]. In contrast, ensembles [...] combine the models to obtain a more powerful model; ensembles can be expected to be better when the true model does not lie within the hypothesis class" [LPB16]. Under Assumptions 1, the unknown target model, our true model here, lies within the hypothesis class of its prior by definition. Therefore, we argue that under these conditions, a Bayesian approach is a more natural way to select a surrogate model.

Target prior. We express the prior of a standard target DNN. Deterministic DNNs are classically trained using the cross-entropy loss regularized by weight decay:

$$\min_{\theta_t} -\frac{1}{N}\sum_{i=1}^{N} \log \hat{p}(y_i|x_i,\theta_t) + \frac{\lambda}{2}\|\theta_t\|^2,$$

with λ its weight decay hyperparameter. This maximum likelihood estimation (MLE) procedure corresponds to the maximum a posteriori inference (MAP) of this implied probabilistic model:

$$p(y,\theta_t|x) = p(y|x,\theta_t)p(\theta_t),$$

where $p(y|x, \theta_t)$ is the likelihood function and $p(\theta_t) = \mathcal{N}(\theta_t|0, \frac{1}{N\lambda}I)$ a Gaussian prior. Therefore, in this standard setting, the hypothesis 4 reduces to knowing the weight decay hyperparameter λ.

Extension to unknown architecture. Let $\mathcal{A} = \{a_i\}_i$ be a countable set of candidate architectures, $p(a)$ a prior on $\mathcal{A}$, θ^a the parameters of the architecture a and $\hat{p}^a(y|x, \theta^a)$ its predictive distribution. Discarding hypothesis 1 of Assumption 1 on the knowledge of the architecture, the architecture of the target a becomes a random variable. We perform *Bayesian Model Comparison* to compute the posterior over models:

$$p(a|\mathcal{D}) \propto p(\mathcal{D}|a)p(a). \tag{4.3}$$

We marginalize over architectures to express the complete posterior predictive distribution as the average across architectures weighted by their posterior probabilities:

$$\begin{aligned} p(y|x, \mathcal{D}) &= \sum_{a\in\mathcal{A}} p(a|\mathcal{D})p(y|x, \mathcal{D}, a) \\ &\propto \mathop{\mathbb{E}}_{p(a)} p(\mathcal{D}|a) \mathop{\mathbb{E}}_{p(\theta^a|\mathcal{D})} \hat{p}^a(y|x, \theta^a) \end{aligned} \tag{4.4}$$

If $\mathcal{A}$ is finite and small, we can approximate this quantity with a weighted average of one cSGLD empirical posterior predictive distribution per architecture. Otherwise, we estimate it with MCMC by sampling according to $p(a)$ a finite subset $A = \{a_i \sim p(a)\}_{i=1}^{S_A} \subset \mathcal{A}$ of architectures, where the number of architectures S_A is fixed by the computational budget. We sample S parameters $\{\theta_s^a\}_{s=1}^S$ for all $a \in A$. Then, our inter-architecture attack that minimizes our approximation of $p(y|x, \mathcal{D})$ becomes:

$$\delta_A \in \mathop{\arg\min}_{\|\delta\|_p\leq\varepsilon} \frac{1}{S_A S} \sum_{a\in A} p(\mathcal{D}|a) \sum_{s=1}^{S} \hat{p}^a(y|x + \delta, \theta_s^a) \tag{4.5}$$

Various methods exist to approximate model evidence [FW12]. To simplify empirical conclusions, we assume that all architectures in $\mathcal{A}$ have approximately equal evidence. This strong assumption is reasonable here, since we select widely used architectures which are well-specified on the standard benchmark datasets evaluated. For fairness to ensemble baselines, our experiments on unknown architectures do not include the target architecture in the set $\mathcal{A}$.

Attack algorithm. One can approximate the solution of Equations 4.2 and 4.5 with minor modifications of existing adversarial attack algorithms, i.e. simply cycling surrogate models throughout iterations. To efficiently approximate Bayesian model averaging during iterative attacks, we compute the gradient of every iteration

Algorithm 3 Variant of I-FGSM attack to perform approximate Bayesian Model Averaging efficiently on numerous models from several architectures.

Input: (x, y) natural example and its corresponding label, S_A ordered sets of model parameters $(\theta_s^1)_{s=1}^S, \ldots, (\theta_s^{S_A})_{s=1}^S$ each sampled from the corresponding posterior distribution $\theta_s^i \sim p(\theta_s|\mathcal{D})$ of the surrogate DNN f_s, n_{iter} number of iterations, ε p-norm perturbation, α step-size, $\mathcal{L}$ loss function

Output: x_{adv} adversarial example

Shuffle each ordered set of model samples $(\theta_s^1)_{s=1}^S, \ldots, (\theta_s^{S_A})_{s=1}^S$
$x_{\text{adv}} \leftarrow x$
for $i = 1$ to n_{iter} **do**
    $g \leftarrow \frac{\alpha}{S_A} \sum_{a=1}^{S_A} \nabla_x \mathcal{L}(f_a(x_{\text{adv}}; \theta^a_{i \bmod S}), y)$ ▷ Compute the average input gradient of the loss of a posterior sample for each surrogate architecture
    $x_{\text{adv}} \leftarrow x_{\text{adv}} + \text{project}(g, S_\alpha[\mathbf{0}])$ ▷ Add the normalized gradient, projected in the p-norm sphere of α radius
    $x_{\text{adv}} \leftarrow \text{project}(x_{\text{adv}}, B_\varepsilon[x])$ ▷ Project in the p-norm ball centred on x of ε radius
    $x_{\text{adv}} \leftarrow \text{clip}(x_{\text{adv}}, 0, 1)$ ▷ Clip to pixel range values
end for

on a single model sample per architecture. If multiple architectures are attacked, we average their gradients (see Algorithm 3). The cost of iterative attacks, measured as the number of backward passes, does not increase with the number of samples S.

Clarifications. In the following, the intra-architecture transferability represents the case of known target architecture. The mass of the prior concentrates on a single architecture, thus the posterior too. Respectively, the inter-architecture transferability corresponds to an unknown target architecture not sampled in the surrogate set. The prior of the target architecture may not be zero, given the extension to unknown architecture described above. But we hold-out this architecture from the surrogate set during empirical evaluation for fairness to baseline and to simplify result interpretations.

4.3 Experimental Settings

Setup summary. The target models are deterministic DNNs and are never used as a surrogate. For a fair comparison between DNNs and cSGLD, we train the surrogate DNNs on CIFAR-10 and MNIST using the same process as the target models. ImageNet targets are third-party pretrained models. Each cSGLD cycle lasts 50 epochs and samples 5 models on CIFAR-10, 10 epochs/4 models on MNIST, 45 epochs/3 models on ImageNet. We report the success rate (misclassification

rate of untargeted adversarial examples) averaged over three attack runs. We craft adversarial examples from correctly predicted test examples (all examples for CIFAR-10 and MNIST, and a random subset of 5000 examples for ImageNet). The iterative attacks (I-FGSM, MI-FGSM, and PGD) perform 50 iterations such that the transferability rates plateaus (see Figures 4.3 and 4.4). Each attack computes the gradient of one model per architecture. Therefore, their computation cost and volatile memory are not multiplied by the size of the surrogate, except for FGSM which computes its unique gradient against all available models. The source code is publicly available[5]. The remaining of this section presents the experimental setup in details.

Datasets. We consider ImageNet (ILSVRC2012; [RDS+15]), CIFAR-10 [Kri09] and MNIST. In all cases, we train the surrogate and target models on the entire training set. For each CIFAR-10 and MNIST target model, we select all the examples from the test set that are correctly predicted by it. In the case of ImageNet, we use a random subset of 5000 correctly predicted test images.

Architectures. We cover a diverse set of architectures in terms of heterogeneity (similar and different families of architecture), computation cost, and release date. For ImageNet, we select five architectures with $3 \times 224 \times 244$ input size. Three classical architectures: ResNet-50 [HZR+16a][6], ResNeXt-50 32x4d [XGD+17] and Densenet-121 [XGD+17]; and two mobile architectures: MNASNet 1.0 [TCP+18] and EfficientNet-B0 [TL19]. Following the work of Ashukha et al. [ALM+20], we consider the following five architectures for CIFAR-10: PreResNet110, PreResNet164 [HZR+16b], VGG16BN, VGG19BN [SZ15], and WideResNet28x10 [ZK16]. We study three architectures on MNIST: "FC" a fully connected neural network with two hidden layers 1200-1200, "Small FC" with a single fully connected hidden layer of size 512, and "CNN" a convolutional neural network composed of two convolutional layers with 32 filters each followed by two fully connected hidden layers 200-200.

Target models. The target models are deterministic DNNs. For ImageNet, we use the pre-trained models provided by PyTorch [PGM+19] and the pre-trained EfficientNet-B0 provided by PyTorch Image Models (*timm*). In the case of CIFAR-10, they are trained using Adam optimizer for 300 epochs with step-wise learning rate decay that divides it by 10 every 75 epochs (MNIST: 50 epochs in total, learning rate divided by 10 every 20 epochs). The benign accuracy of all target models exceeds 73% (ImageNet), 83% (CIFAR-10) and 98% (MNIST); see Table 4.1 for exact values.

[5] `https://github.com/Framartin/transferable-bnn-adv-ex`

[6] Ashukha et al. [ALM+20] study ResNet-50 only on ImageNet. We used their shared trained models as surrogate DNNs.

Table 4.1: Top-1 natural test accuracy of target DNNs.

Dataset	Target DNN	Accuracy
CIFAR-10	PreResNet110	93.26 %
	PreResNet164	93.03 %
	VGG16bn	83.68 %
	VGG19bn	83.62 %
	WideResNet28x10	92.13 %
ImageNet	ResNet50	76.15 %
	ResNeXt50 32x4d	77.62 %
	Densenet121	74.65 %
	MNASNet 1.0	73.51 %
	EfficientNet-B0	77.70 %
MNIST	CNN	99.33 %
	FC	98.65 %
	Small FC	98.41 %

Surrogate models (deep ensemble). For CIFAR-10 and MNIST, the DNNs used to form surrogate ensembles are trained using the same process as the target models. Therefore, the comparison between deterministic DNNs and cSGLD is fair, since one can expect the deterministic DNNs surrogate to be "close" to the target. As for ImageNet, we retrieve an ensemble of 15 ResNet-50 models trained independently by Ashukha et al. [ALM+20] using SGD with momentum during 130 epochs. For the experiments in Section 4.4.3, we train similarly one model for every 4 other ImageNet architectures.

Surrogate models (cSGLD). Following the work of Ashukha et al. [ALM+20] and Zhang et al. [ZLZ+20], we train models with cSGLD on CIFAR-10 for 6 learning rate cycles (which, as our RQ4 experiments reveal, is where the transfer rate starts plateauing). cSGLD performs 5 cycles on ImageNet, and 10 on MNIST. The learning rate is set with cosine annealing schedule for fast convergence. Each cycle lasts 45 on ImageNet, 50 epochs on CIFAR-10 and 10 on MNIST. The last epochs of every cycle form the sampling phase: noise is added and one sample is drawn at the end of each epoch. On CIFAR-10, we obtain 5 samples per cycle (resp. 3 on ImageNet and 4 MNIST), so 30 samples in total (resp. 15 and 20). Figure 5.3 is an illustration of a cSGLD cyclical learning rate schedule. To train ResNet-50 models on ImageNet, we re-use the original cSGLD hyperparameters.

Surrogate models (other training methods). Additionally, to Deep Ensemble cSGLD and following Ashukha et al. [ALM+20], we consider 2 Bayesian Deep Learning techniques (SWAG and VI) and 2 Ensemble ones (SSE and FGE). We

train every technique on CIFAR-10 and cSGLD and SWAG on ImageNet. We retrieve trained Deep Ensemble, SSE, FGE and VIImageNet models from Ashukha et al. [ALM+20]. Technique descriptions and experimental setup of surrogates trained with SWAG, VI, FGE, or SSE are detailed below in the Bayesian and Ensemble Training Techniques section.

Adversarial attacks. We applied our variant of 4 gradient-based attacks as described in the approach section. The attacker's goal is misclassification (untargeted adversarial examples). We perform both 2-norm and ∞-norm bounded adversarial attacks, and report means and standard deviations computed on 3 random seeds. In accordance to values commonly used in the literature [CH20], the maximum perturbation norm ε is set respectively to 0.5 and $\frac{4}{255}$ on CIFAR-10, and respectively to 3 and $\frac{4}{255}$ on ImageNet. MNIST ones are respectively 3 and 0.1. The step-size α is set to $\frac{\varepsilon}{10}$. We choose to perform 50 iterations such that the transferability rates plateaus for all iterative attacks (I-FGSM, MI-FGSM and PGD) on both norms, both datasets, for both several DNNs and cSGLD (see Figures 4.3 and 4.4). PGD runs with 5 random restarts. FGSM aside, *every iteration computes the gradient of 1 model per architecture*. Therefore, the attack computation cost and volatile memory are not multiplied by the size of the surrogate, except for FGSM which computes its unique gradient against all available models. cSGLD samples are attacked in random order. The MI-FGSM decay factor is set to 0.9.

Test-time transformations. In the dedicated section, we consider three test-time transformations applied during attack designed for transferability (see related work section): Ghost Networks (GN) [LBZ+18], Input Diversity (DI) [XZZ+19] and Skip Gradient Method (SGM) [WWX+20]. We implemented the first two in PyTorch with their original hyperparameters. To extend Input Diversity to the smaller input sizes of CIFAR-10, we keep the same maximum resize ratio of 0.9. We reuse the original implementation of the third one on ResNet50, and extend it to PreResNet110 (we set its hyperparameter to 0.7 via grid-search).

Implementation. The source code of the experiments are publicly available on GitHub[7]. Our attack is built on top of the Python ART library [NST+18]. cSGLD, VI, SSE, and FGE models were trained thanks to the implementation of Ashukha et al. [ALM+20] available on GitHub[8]. All models were trained with PyTorch [PGM+19]. We use EfficientNet-B0 from timm[9]. We train SWAG on ImageNet with the original implementation [MGI+19a]. We use the following software versions: Python 3.8.8, Pytorch 1.7.1 (1.9.0 for Flops measurement), torchvision 0.8.2, Adversarial Robustness Toolbox 1.6.0, and timm 0.3.2.

[7] `https://github.com/Framartin/transferable-bnn-adv-ex`
[8] `https://github.com/bayesgroup/pytorch-ensembles`
[9] `https://github.com/rwightman/pytorch-image-models`

Table 4.2: Hyperparameters used to train cSGLD or Deep Ensemble. The $\star$ symbols refer to the inter-architecture and test-time techniques sections, and $\star\star$ to the Bayesian and Ensemble training methods section. We do not include target DNNs on ImageNet, since they are pretrained models from PyTorch and timm.

		CIFAR-10		**ImageNet**	
Method	**Hyperparameter**	**cSGLD**	**DNN Surrogate & Target**	**cSGLD**	**DNN Surrogate**
All	Number epochs	50 per cycle	300	45 per cycle	130 (135 for $\star$)
	Initial learning rate	0.5	0.01	0.1	0.1
	Learning rate schedule	Cosine Annealing	Step size decay ($\times$0.1 each 75 epochs)	Cosine Annealing	Step size decay ($\times$0.1 each 30 epochs)
	Optimizer	cSGLD	Adam	cSGLD	SGD
	Momentum	0	0.9	0.9	0.9
	Weight decay	5e-4 (3e-4 for PreResNet)	1e-4	1e-4	1e-4
	Batch-size	64	128	256 for ResNet50, 64 for others	256 for ResNet50, 64 for others
cSGLD	Sampling interval	1 sample per epoch	-	1 sample per epoch	-
	Nb cycles	6 (18 for $\star\star$)	-	5 (3 for $\star$, 6 for $\star\star$)	-
	Nb samples per cycle	5	-	3	-
	Nb epochs with noise	5	-	3	-

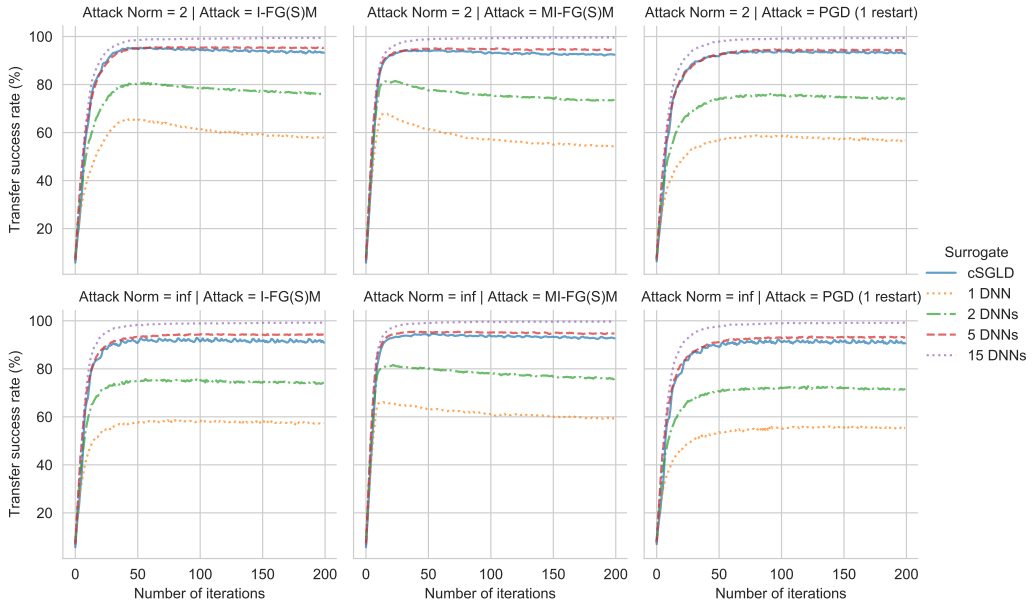


Figure 4.3: Transfer success rates on ImageNet of three iterative gradient-based attacks on the same architecture (ResNet-50) with respect to the number of iterations.

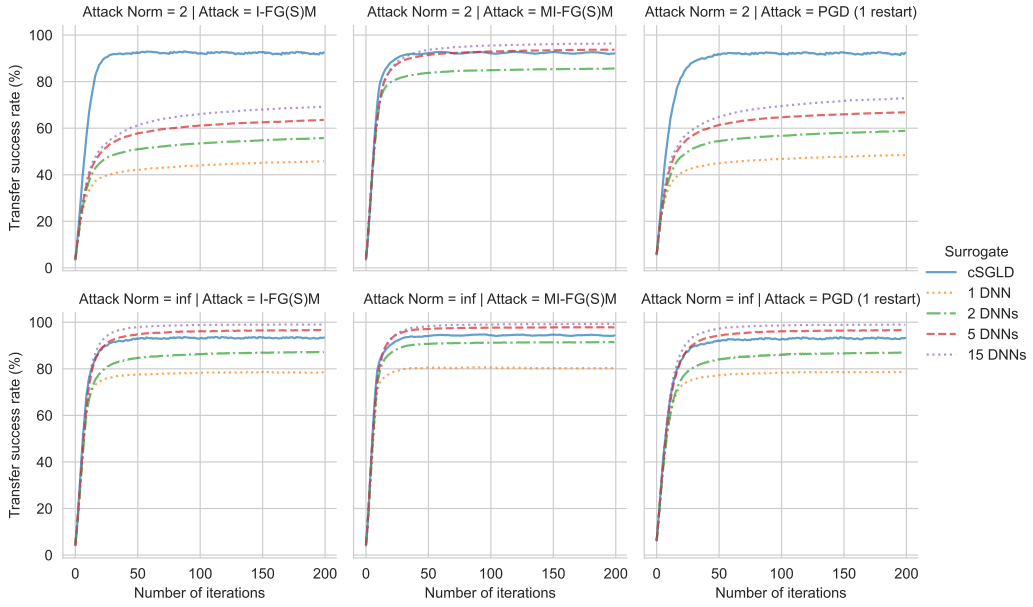


Figure 4.4: Transfer success rates on CIFAR-10 of three iterative gradient-based attacks on the same architecture (PreResNet110) with respect to the number of iterations.

Table 4.3: Hyperparameters of attacks and test-time transferability techniques.

Attack / Technique	**Hyperparameter**	**ImageNet**	**CIFAR-10**	**MNIST**
All attacks	Perturbation 2-norm ε	3	0.5	3
	Perturbation ∞-norm ε	$\frac{4}{255}$	$\frac{4}{255}$	0.1
Iterative Attacks	Step-size α	$\frac{\varepsilon}{10}$	$\frac{\varepsilon}{10}$	$\frac{\varepsilon}{10}$
	Number iterations	50	50	50
MI-FGSM	Momentum term	0.9	0.9	0.9
PGD	Number random restarts	5	5	5
GN	Skip connection erosion random range	[1-0.22, 1+0.22]	[1-0.22, 1+0.22]	-
DI	Minimum resize ratio	90%	90%	-
	Probability transformation	50%	50%	-
SGM	Residual Gradient Decay γ	0.2 (ResNet50)	0.7 (PreResNet110)	-

Flops. We measure the training computational complexity in Flops using the PyTorch profiler. The computation overhead of one epoch with cSGLD compared to one with SGD/Adam is negligible. The main difference is the addition of noise to the weights during the sampling phase. On CIFAR-10, the overhead of 1 cSGLD epoch of PreResNet110 with added noise compared to one of a DNN trained with Adam (SGD) is 0.0187% Flops (respectively 0.0146% for ResNet50 on ImageNet).

Infrastructure. Experiments were run on Tesla V100-DGXS-32GB GPUs. The server has the following specifications: 256GB RDIMM DDR4, CUDA version 10.1, Linux (Ubuntu) operating system.

4.4 Experimental Results

The goal of our approach is to increase the transferability of adversarial examples by using a surrogate sampled from the distribution of the target model or from the posterior distribution to attack a deterministic DNN.

4.4.1 Transferability From Deep Ensemble

First, we show that deep ensemble can generate highly transferable adversarial examples. That is to say, we assess the possibility of independently training several surrogate DNNs *of the same architecture* from independent random initializations

and independent batch sampling during training. Deep ensemble samples from the distribution of the target model under the threat model presented in Assumption 1.

At each attack iteration, we sample one independently trained model to compute its input gradient. Therefore, the cost of the attack is kept constant. This section exceptionally borrows the experimental settings of Chapter 5 on ImageNet, detailed in Section 5.2.

The upper left sub-figure in Figure 4.5 shows the intra-architecture transferability, i.e., the surrogate and target models have the same architecture. We observe that transferability increases with the number of independently trained DNNs. Therefore, sampling from the distribution of the unknown target does indeed increase transferability. This is the most favourable setting for the threat model in Assumption 1.

Next, we investigate the inter-architecture case, where the surrogate and target models have distinct architectures. This case is not favourable to deep ensemble since we know that the surrogate is not sampled from the target distribution, contrary to the intra-architecture case. As observed in Figure 4.5, and despite this unfavourable setting, deep ensemble is effective in increasing transferability by a large margin. As we train more surrogate models of the same architecture, the transferability to other architectures increases. This observation stays true even for the transferability between families of architecture, as shown in Figure 4.5 for the success rate from a ResNet-50 surrogate to DenseNet, VGG and Inception targets.

Therefore, deep ensemble is an effective surrogate for black-box attacks based on transferability. However, since it requires training numerous DNNs from scratch, deep ensemble surrogates are costly. This cost is an issue, as training the surrogate model is the main computational cost of transfer-based attacks, as established in Chapter 1. Deep ensemble is effective, but not efficient. In the following, we explore ways to address this shortfall.

4.4.2 Intra-architecture Transferability

Since SG-MCMC methods sample the weights of a given architecture, we expect our approach to work particularly well in settings where the architecture of the target model is known, but not its weights. To demonstrate this, we compare the intra-architecture transfer success rates of cSGLD with the ones of deep ensemble surrogates (using 1 up to 15 independently trained DNNs). Architectures are ResNet-50 (ImageNet), PreResNet110 (CIFAR-10) and fully connected 1200-1200 (MNIST).

Tables 4.4 and 4.5 provide the detailed results for four classical gradient-based attacks on the three datasets. In summary, for a similar computation cost on ImageNet and CIFAR-10, cSGLD systematically increases the success rate of iterative attacks by 13.8 (ImageNet, MI-FGSM, L∞) to 49.2 (CIFAR-10, I-FGSM, L2) percentage points, and of FGSM by 12.18 to 22.2. On MNIST, it ranges from

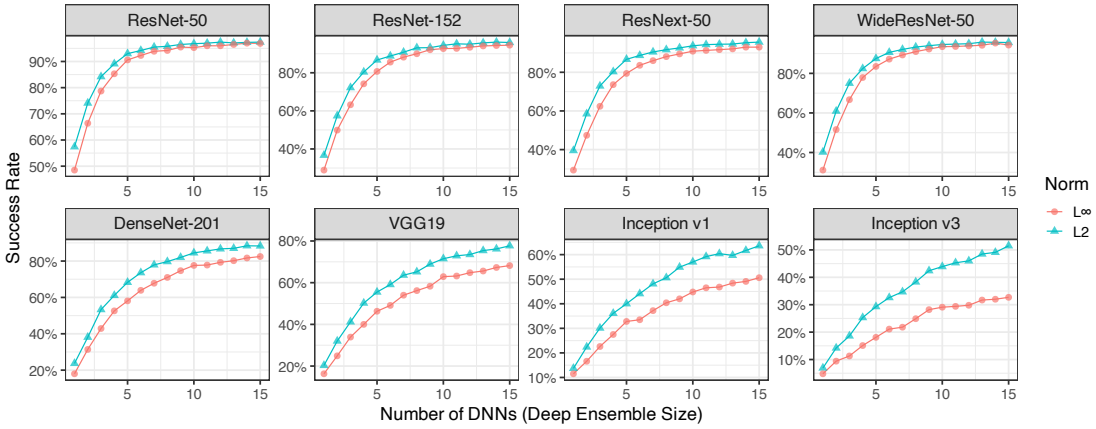


Figure 4.5: Transfer success rate with respect to the size of the deep ensemble surrogate (number of independently trained models) on ImageNet. The sub-figure title is the target architecture. The deep ensemble surrogate has a ResNet-50 architecture. The upper-left sub-figure is intra-architecture transferability, the other subplots are inter-architectures.

6.8 to 80.5. One explanation for the highest improvements is that DNN-based L2 norm attacks suffer from vanishing gradients on CIFAR-10 and MNIST, whereas cSGLD avoids it thanks to fast convergence and warm restarts (Table 4.6 for proportions of vanished gradients).

Inspired by DEE [ALM+20], we propose the **Transferability-Deep Ensemble Equivalent (T-DEE)** metric as the number of independently trained DNNs needed to achieve the same success rate as the technique considered (computed with linear interpolation). Under some assumptions[10], deep ensemble samples exactly from the distribution of target parameters, and is thus optimal for intra-architecture transferability with infinite computing power.

Table 4.7 reports the T-DEE and the *computing ratio*, i.e., the total number of flops to train such DNNs ensemble divided by the number of flops used to trained cSGLD. This ratio represents the computational gain factor achieved by our approach[11]. In the worst case across the three datasets, an ensemble of 3 surrogate DNNs is required to beat the cSGLD surrogate, while requiring at least 2.51 times more flops during the training phase. On CIFAR-10 and MNIST and considering L2 attack specifically (MI-FGSM CIFAR-10 aside), it even outperforms

[10]Besides Assumptions 1, we suppose that deep ensemble uses the target optimizer, and that the minimum in Eq. 4.2 is reached, i.e., that the attack doesn't fail due to vanished or obfuscated gradients [ACW18].

[11]The ImageNet computing ratios don't equal to T-DEE since 1 DNN is trained for 130 epochs and cSGLD for 225.

Table 4.4: Intra-architecture transfer success rates of four attacks on PreResNet110 (CIFAR-10) and ResNet50 (ImageNet), in %. Bold is best. Higher is better. The last two columns are respectively the number of epochs and attack backward passes.

Dataset	Attack	Surrogate	L2 Norm	L∞ Norm	Epochs	Backwards
ImageNet	I-FGSM	cSGLD	94.41 ±0.46	90.77 ±0.09	225	50
		1 DNN	64.95 ±0.54	57.79 ±0.17	130	50
		2 DNNs	80.39 ±0.83	74.25 ±0.71	260	50
		5 DNNs	94.53 ±0.43	92.81 ±0.45	650	50
		15 DNNs	**98.51 ±0.11**	**98.28 ±0.16**	1950	50
	MI-FGSM	cSGLD	93.42 ±0.73	93.61 ±0.41	225	50
		1 DNN	61.11 ±0.35	63.70 ±0.21	130	50
		2 DNNs	77.93 ±0.44	79.27 ±0.76	260	50
		5 DNNs	94.41 ±0.47	95.32 ±0.25	650	50
		15 DNNs	**98.89 ±0.13**	**99.19 ±0.13**	1950	50
	PGD	cSGLD	91.81 ±0.38	88.76 ±0.24	225	250
		1 DNN	57.47 ±0.52	53.79 ±0.45	130	250
		2 DNNs	74.04 ±0.47	70.90 ±0.41	260	250
		5 DNNs	91.99 ±0.41	91.27 ±0.59	650	250
		15 DNNs	**97.83 ±0.20**	**97.65 ±0.21**	1950	250
	FGSM	cSGLD	58.91 ±0.11	67.17 ±0.26	225	15
		1 DNN	37.37 ±0.19	44.55 ±0.72	130	1
		2 DNNs	46.73 ±0.34	53.91 ±0.60	260	2
		5 DNNs	58.17 ±0.18	65.53 ±0.10	650	5
		15 DNNs	**68.48 ±0.52**	**76.57 ±0.62**	1950	15
CIFAR-10	I-FGSM	cSGLD	**92.38 ±0.23**	92.74 ±0.33	300	50
		1 DNN	43.17 ±0.97	77.59 ±0.01	300	50
		2 DNNs	52.08 ±1.03	84.75 ±0.20	600	50
		5 DNNs	58.74 ±0.98	94.81 ±0.17	1500	50
		15 DNNs	62.08 ±0.92	**97.83 ±0.03**	4500	50
	MI-FGSM	cSGLD	92.29 ±0.25	94.20 ±0.14	300	50
		1 DNN	72.34 ±0.23	80.43 ±0.04	300	50
		2 DNNs	84.10 ±0.33	90.70 ±0.07	600	50
		5 DNNs	91.66 ±0.26	97.04 ±0.07	1500	50
		15 DNNs	**93.87 ±0.30**	**98.30 ±0.11**	4500	50
	PGD	cSGLD	**91.65 ±0.33**	92.10 ±0.25	300	250
		1 DNN	51.08 ±0.10	77.58 ±0.38	300	250
		2 DNNs	60.60 ±0.06	83.67 ±0.27	600	250
		5 DNNs	67.55 ±0.21	94.19 ±0.07	1500	250
		15 DNNs	70.42 ±0.23	**97.37 ±0.06**	4500	250
	FGSM	cSGLD	**43.13 ±0.00**	58.85 ±0.01	300	30
		1 DNN	20.92 ±0.00	38.89 ±0.01	300	1
		2 DNNs	23.75 ±0.00	45.83 ±0.01	600	2
		5 DNNs	25.60 ±0.00	54.62 ±0.01	1500	5
		15 DNNs	26.71 ±0.00	**61.81 ±0.00**	4500	15

Table 4.5: Intra-architecture transfer success rates of four attacks on the FC architecture (MNIST), in %. Bold is best. Higher is better. The last two columns are respectively the number of epochs and attack backward passes.

Dataset	Attack	Surrogate	L2 Norm	L∞ Norm	Epochs	Backwards
MNIST	I-FGSM	cSGLD	**97.65%** ±0.02	**41.49%** ±0.02	50	50
		1 DNN	17.17% ±0.00	34.53% ±0.00	50	50
		2 DNNs	18.52% ±0.01	36.44% ±0.01	100	50
		5 DNNs	26.21% ±0.10	43.12% ±0.16	250	50
		15 DNNs	26.46% ±0.19	45.22% ±0.27	750	50
	MI-FGSM	cSGLD	**97.62%** ±0.05	**42.07%** ±0.09	50	50
		1 DNN	80.72% ±0.00	34.52% ±0.00	50	50
		2 DNNs	82.63% ±0.05	39.83% ±0.06	100	50
		5 DNNs	91.83% ±0.12	44.74% ±0.23	250	50
		15 DNNs	92.08% ±0.09	46.99% ±0.37	750	50
	PGD	cSGLD	**97.78%** ±0.04	**41.64%** ±0.18	50	250
		1 DNN	31.99% ±0.08	34.80% ±0.07	50	250
		2 DNNs	33.61% ±0.07	37.26% ±0.17	100	250
		5 DNNs	43.27% ±0.37	43.61% ±0.29	250	250
		15 DNNs	44.56% ±0.29	45.50% ±0.29	750	250
	FGSM	cSGLD	**75.09%** ±0.00	**34.90%** ±0.00	50	20
		1 DNN	8.62% ±0.00	22.52% ±0.00	50	1
		2 DNNs	7.42% ±0.00	25.76% ±0.00	100	2
		5 DNNs	7.95% ±0.00	29.52% ±0.00	250	5
		15 DNNs	7.52% ±0.00	31.08% ±0.00	750	15

the ensemble of 15 DNNs by a significant factor (up to 71.2 percentage points). On ImageNet, cSGLD achieves the same success rate as 4.38–5.98 DNNs, which corresponds to dividing the number of flops by 2.51–3.46.

Then, the uncertainty on parameter estimation captured by cSGLD is useful to discover generic adversarial directions.

4.4.3 Inter-architecture Transferability

We now focus on black-box settings where the architecture of the target model is unknown (and not used to build the surrogate model). We consider ten architectures (five for both ImageNet and CIFAR-10). Following Liu et al. [LCL+17], Xie et al. [XZZ+19], Li et al. [LBZ+18], and Dong et al. [DLP+18], we hold-out one architecture to act as the target model and use the four remaining ones as surrogates. We apply I-FGSM with 1 model per surrogate architecture per iteration to keep attack cost constant. Due to computational limitations, we limit the training to 135 epochs on ImageNet (3 cycles of 45 epochs for cSGLD). For every architecture, cSGLD and 1 DNN are trained for the same number of epochs.

Table 4.6: Proportion of vanished gradients of each 15 individual models and of the ensemble of 15 models (in %). Gradients disappear before and after averaging in similar proportion (except in one case for VI where there is more gradient vanishing after averaging). A gradient vanishes if its L2 norm is lower than 10^{-8}, the numerical tolerance of the Adversarial Robustness Toolbox library. Gradients are on 10 000 original test examples. Means and standard deviations of 15 models are reported when not ensembled.

Dataset	**Arch**	**Surrogate**	**Vanished individual model gradients**	**Vanished ensemble gradients (averaging)**
ImageNet	RN50	cSGLD (ours)	0.06 ± 0.06	0.00
		VI	0.15 ± 0.02	0.05
		DNN	0.11 ± 0.03	0.00
CIFAR-10	PRN110	cSGLD (ours)	3.04 ± 0.72	2.52
		VI	2.79 ± 0.11	2.08
		DNN	59.15 ± 0.73	63.96
MNIST	CNN	cSGLD (ours)	30.94 ± 2.00	31.67
		DNN	91.53 ± 2.36	94.20
	FC	cSGLD (ours)	11.16 ± 0.51	11.31
		VI	84.73 ± 1.95	91.72
		DNN	90.60 ± 1.71	92.14
	Small FC	cSGLD (ours)	4.63 ± 0.48	4.71
		VI	60.61 ± 4.61	82.00
		HMC	85.61 ± 0.02	85.62
		DNN	77.56 ± 2.84	79.88

As shown in Tables 4.8 (ImageNet) and 4.9 (CIFAR-10), our method significantly improved transferability on all five hold-out architectures for both datasets, except for the L∞ VGG19 target (with a difference of 0.4 percentage point). On CIFAR-10, the differences range from 15.0 to 35.2 percentage points (2-norm), and from -0.4 to 9.9 (∞-norm). Our method outperforms 4 DNNs per architecture on the L2 attack, despite been trained for 4 times fewer epochs. On ImageNet, cSGLD improves over the one DNN counterpart by 11.8 and 29.9 percentage points of success rate at constant computational train and attack budget.

Table 4.7: Number of DNNs (T-DEE) and training computation budget (in flops) to achieve the intra-architecture transferability of cSGLD with deep ensemble. Higher is better. ">15" means that 15 DNNs always transfer less than cSGLD.

Dataset	**Attack**	**Norm**	**T-DEE**	**Flops Ratio**
ImageNet	I-FGSM	L2	4.91 ±0.11	2.84 ±0.06
		L∞	4.34 ±0.13	2.51 ±0.08
	MI-FGSM	L2	4.69 ±0.18	2.71 ±0.10
		L∞	4.38 ±0.03	2.53 ±0.02
	PGD	L2	5.00 ±0.11	2.89 ±0.06
		L∞	4.42 ±0.16	2.56 ±0.09
	FGSM	L2	5.81 ±0.34	3.35 ±0.19
		L∞	5.98 ±0.03	3.46 ±0.02
CIFAR10	I-FGSM	L2	>15 ±nan	>15 ±nan
		L∞	3.76 ±0.08	3.76 ±0.08
	MI-FGSM	L2	5.56 ±0.80	5.56 ±0.80
		L∞	2.88 ±0.03	2.87 ±0.03
	PGD	L2	>15 ±nan	>15 ±nan
		L∞	3.74 ±0.12	3.74 ±0.12
	FGSM	L2	>15 ±nan	>15 ±nan
		L∞	8.72 ±0.01	8.72 ±0.01
MNIST	I-FGSM	L2	>15 ±nan	>15 ±nan
		L∞	3.42 ±0.17	3.42 ±0.17
	MI-FGSM	L2	>15 ±nan	>15 ±nan
		L∞	2.79 ±0.07	2.79 ±0.07
	PGD	L2	>15 ±nan	>15 ±nan
		L∞	3.26 ±0.28	3.26 ±0.28
	FGSM	L2	>15 ±nan	>15 ±nan
		L∞	>15 ±nan	>15 ±nan

Table 4.8: Transfer success rates of I-FGSM attack on ImageNet hold-out architectures. Higher is better.

		Target Architecture					
Norm	**Surrogate**	−RN50	−RNX50	−DN121	−MNASN	−EffNetB0	**Epochs**
L2	1 cSGLD per arch.	**93.28** ±0.12	**90.61** ±0.24	**92.25** ±0.26	**95.98** ±0.19	**81.88** ±0.38	4 × 135
	1 DNN per arch.	72.99 ±0.52	72.31 ±0.44	64.72 ±0.59	84.21 ±0.18	53.99 ±0.76	4 × 135
L∞	1 cSGLD per arch.	**92.21** ±0.23	**89.83** ±0.22	**90.86** ±0.19	**95.85** ±0.46	**79.40** ±0.42	4 × 135
	1 DNN per arch.	69.65 ±0.47	69.01 ±0.70	61.00 ±0.66	82.25 ±0.03	49.71 ±1.37	4 × 135

Table 4.9: Transfer success rates of I-FGSM attack on CIFAR-10 hold-out architectures. The ⋆ symbol indicates that 1 DNN per architecture is better than 1 cSGLD per architecture. Higher is better.

		Target Architecture					
Norm	**Surrogate**	−PRN110	−PRN164	−VGG16	−VGG19	−WRN28	**Epochs**
L2	1 cSGLD per arch.	**95.56** ±0.04	**95.72** ±0.06	**45.96** ±0.07	**42.60** ±0.08	**84.04** ±0.05	4 × 300
	1 DNN per arch.	60.38 ±1.09	60.93 ±1.06	29.97 ±0.48	27.57 ±0.66	57.86 ±0.74	4 × 300
	4 DNNs per arch.	77.12 ±1.32	77.21 ±1.14	40.89 ±0.63	40.18 ±0.76	77.54 ±0.93	4 × 1200
L∞	1 cSGLD per arch.	96.38 ±0.06	96.51 ±0.08	49.19 ±0.06	45.17 ±0.03	84.75 ±0.01	4 × 300
	1 DNN per arch.	87.02 ±0.04	88.86 ±0.04	44.99 ±0.10	⋆45.55 ±0.02	74.84 ±0.03	4 × 300
	4 DNNs per arch.	**96.50** ±0.01	**97.01** ±0.02	**59.80** ±0.01	**59.08** ±0.01	**89.23** ±0.04	4 × 1200

Table 4.10: Inter-architecture transfer success rates of I-FGSM of single architecture surrogate on ImageNet (in %). All combinations of surrogate and targeted architectures are evaluated. Diagonals are intra-architecture. 1 DNN and cSGLD have similar computation budget (135 epochs). Bold is best. Higher is better.

	Surrogate		**Target Architecture**				
Norm	Arch	Training	RN50	RNX50	DN121	MNASN	EffNetB0
L2	RN50	cSGLD	**84.93** ±0.59	**74.70** ±0.91	**71.32** ±0.63	**60.09** ±0.60	**39.70** ±0.29
		1 DNN	56.98 ±0.62	41.13 ±0.97	29.81 ±0.33	27.90 ±0.43	16.39 ±0.46
	RNX50	cSGLD	**79.25** ±0.24	**77.34** ±0.39	**68.53** ±0.19	**62.16** ±0.19	**43.51** ±0.62
		1 DNN	37.48 ±0.52	36.35 ±0.22	23.77 ±0.41	23.69 ±0.21	14.32 ±0.24
	DN121	cSGLD	**63.23** ±1.16	**59.89** ±1.12	**73.28** ±0.45	**60.84** ±0.33	**40.27** ±0.44
		1 DNN	32.61 ±0.29	32.06 ±0.61	39.18 ±0.47	32.01 ±0.44	17.72 ±0.49
	MNASN	cSGLD	**7.81** ±0.19	**5.97** ±0.37	**9.81** ±0.31	30.41 ±1.45	**15.46** ±0.44
		1 DNN	7.04 ±0.51	5.29 ±0.36	8.41 ±0.20	**32.65** ±0.22	13.13 ±0.06
	EffNetB0	cSGLD	**18.93** ±2.17	**14.16** ±1.69	**19.89** ±1.21	**65.97** ±3.60	**49.41** ±3.64
		1 DNN	15.15 ±0.30	13.33 ±0.33	16.12 ±0.71	58.73 ±0.25	48.85 ±0.56
L∞	RN50	cSGLD	**78.67** ±1.19	**65.21** ±1.48	**61.54** ±0.83	**51.75** ±1.39	**31.11** ±1.13
		1 DNN	48.03 ±0.94	32.17 ±0.43	23.37 ±0.34	22.60 ±0.40	12.59 ±0.21
	RNX50	cSGLD	**71.67** ±1.00	**69.33** ±0.85	**59.18** ±1.14	**54.75** ±1.33	**35.13** ±0.71
		1 DNN	31.19 ±0.42	28.68 ±0.76	19.12 ±0.07	19.53 ±0.51	11.20 ±0.33
	DN121	cSGLD	**54.13** ±1.70	**50.66** ±1.62	**65.80** ±0.66	**53.43** ±1.30	**32.49** ±0.36
		1 DNN	25.49 ±0.81	23.73 ±0.59	30.78 ±0.21	26.05 ±0.66	13.41 ±0.20
	MNASN	cSGLD	**6.77** ±0.29	4.72 ±0.27	**8.26** ±0.36	25.27 ±1.83	**12.21** ±0.84
		1 DNN	6.52 ±0.23	**5.06** ±0.12	7.83 ±0.13	**29.19** ±0.05	11.13 ±0.16
	EffNetB0	cSGLD	**17.81** ±1.58	**13.91** ±1.45	**19.71** ±1.29	**63.67** ±3.16	46.91 ±3.44
		1 DNN	15.83 ±0.32	13.51 ±0.52	16.78 ±0.38	60.14 ±0.37	**50.16** ±0.64

We also present the results for an alternative protocol where we use a single architecture as surrogate (Tables 4.10, 4.11 and 4.12). In summary, in this setup cSGLD achieves a higher inter-architecture success rate in 39/40 cases on ImageNet (Table 4.10), 38/40 cases on CIFAR-10 (Table 4.11), and in 18/18 cases on MNIST (Table 4.12), compared to a single DNN trained for the same number of epochs. Differences range between -0.3 and 44.8 percentage points on ImageNet, -2.3 and 62.1 on CIFAR-10 and 0.2 and 83.2 on MNIST.

We conclude that our method improves transferability even when the target architecture is unknown. This tends to indicate that the adversarial directions against posterior predictive distribution are partially aligned across different architectures. In other words, given a common classification task, the variability of an architecture parameters might be informative of the variability of another architecture parameters.

4.4.4 Test-time Transferability Techniques

Given that our approach works at train time, we evaluate its combination with test-time techniques. We apply three test-time transformations to cSGLD samples and one DNN obtained with the same number of epochs (300 for CIFAR-10, 135 for ImageNet). The ImageNet surrogates are ResNet50 (respect. PreResNet110 on CIFAR-10). The targets are the same as in Section 4.4.3. Following Li et al. [LBZ+18], Xie et al. [XZZ+19], and Wu et al. [WWX+20], we also combine every test-time technique with momentum[12].

Table 4.13 shows the results on ImageNet (Table 4.14 for CIFAR-10). We observe that our approach and the test-time techniques complement well to each other. Indeed, the best success rates are always achieved by a technique applied on cSGLD (in bold). All three techniques combined with momentum applied on cSGLD achieve a systematically higher success rate than the same technique applied on 1 DNN, with differences ranging from 10.7 to 41.7 percentages points on ImageNet and from 3.8 to 56.2 on CIFAR-10. Overall, the addition of a technique (excluding momentum alone) to our vanilla cSGLD surrogate never decrease the success rate on CIFAR-10 and only in 10% of the averaged cases considered on ImageNet, as indicated by the † symbols.

Besides, our vanilla cSGLD surrogate achieves better transferability than any of the test-time techniques applied to 1 DNN in 90% of the cases on CIFAR-10 and 93.3% on ImageNet, using the I-FGSM attack. Similarly, for MI-FGSM, we observe 76.7% for the former and 90% for the latter. This demonstrates that despite previous efforts in providing effective test-time techniques for transferability (see Chapter 3), *improving the training of the surrogate – in our case, through efficient sampling*

[12]All rows with "MI" correspond to MI-FGSM, an attack variant designed to improve transferability using momentum [DLP+18].

Table 4.11: Inter-architecture transfer success rates of I-FGSM of single architecture surrogate on CIFAR-10 (in %). All combinations of surrogate and targeted architectures are evaluated. Diagonals are intra-architecture. Symbols ⋆ indicate 1 DNN having higher transferability than cSGLD. 1 DNN and cSGLD have similar computation budget (300 epochs). Bold is best. Higher is better.

	Surrogate		**Target Architecture**				
Norm	Arch	Training	PRN110	PRN164	VGG16	VGG19	WRN28
L2	PRN110	cSGLD	**88.96** ±0.02	**88.57** ±0.00	26.18 ±0.02	24.38 ±0.00	**63.35** ±0.01
		1 DNN	34.42 ±0.00	34.39 ±0.01	12.66 ±0.01	12.54 ±0.00	26.29 ±0.00
		4 DNNs	50.50 ±0.00	50.49 ±0.00	**27.45** ±0.01	**27.30** ±0.00	46.10 ±0.00
	PRN164	cSGLD	**88.28** ±0.01	**87.52** ±0.01	25.83 ±0.01	23.64 ±0.01	**62.79** ±0.01
		1 DNN	33.89 ±0.00	34.36 ±0.01	11.93 ±0.00	12.07 ±0.01	25.95 ±0.01
		4 DNNs	50.36 ±0.01	50.45 ±0.00	**26.79** ±0.01	**27.13** ±0.00	45.94 ±0.00
	VGG16	cSGLD	**69.22** ±0.06	**69.03** ±0.03	43.70 ±0.04	38.54 ±0.02	**55.62** ±0.07
		1 DNN	27.22 ±0.04	27.23 ±0.05	29.28 ±0.08	28.73 ±0.02	22.22 ±0.00
		4 DNNs	55.14 ±0.06	54.96 ±0.04	**73.65** ±0.00	**71.24** ±0.04	44.89 ±0.09
	VGG19	cSGLD	**69.82** ±0.05	**68.27** ±0.07	44.59 ±0.10	39.76 ±0.13	**54.40** ±0.08
		1 DNN	18.09 ±0.10	18.09 ±0.06	⋆44.63 ±0.03	⋆46.76 ±0.03	14.38 ±0.03
		4 DNNs	34.30 ±0.06	33.77 ±0.01	**66.20** ±0.03	**68.87** ±0.05	27.44 ±0.02
	WRN28	cSGLD	**82.25** ±0.03	**85.06** ±0.02	**26.34** ±0.08	**23.81** ±0.03	**69.31** ±0.07
		1 DNN	22.14 ±0.01	23.00 ±0.00	9.43 ±0.00	9.54 ±0.00	26.85 ±0.00
		4 DNNs	41.07 ±0.00	41.75 ±0.04	22.91 ±0.04	22.65 ±0.03	43.00 ±0.01
L∞	PRN110	cSGLD	88.70 ±0.00	88.48 ±0.01	26.32 ±0.00	24.27 ±0.01	62.95 ±0.01
		1 DNN	72.73 ±0.00	74.57 ±0.00	22.26 ±0.00	20.98 ±0.00	47.59 ±0.01
		4 DNNs	**91.98** ±0.00	**92.25** ±0.00	**38.24** ±0.00	**35.56** ±0.00	**72.64** ±0.01
	PRN164	cSGLD	87.99 ±0.01	87.74 ±0.00	26.33 ±0.00	23.67 ±0.01	61.83 ±0.02
		1 DNN	68.97 ±0.01	71.76 ±0.00	20.29 ±0.00	18.86 ±0.00	45.07 ±0.00
		4 DNNs	**90.67** ±0.00	**92.22** ±0.00	**37.62** ±0.00	**35.23** ±0.00	**73.18** ±0.00
	VGG16	cSGLD	**66.97** ±0.13	**67.48** ±0.11	42.91 ±0.05	37.91 ±0.02	**50.52** ±0.01
		1 DNN	35.57 ±0.02	35.89 ±0.03	38.35 ±0.00	35.82 ±0.00	26.77 ±0.02
		4 DNNs	52.59 ±0.00	53.12 ±0.00	**70.89** ±0.00	**68.53** ±0.00	41.34 ±0.00
	VGG19	cSGLD	**67.11** ±0.00	**66.55** ±0.02	43.50 ±0.01	38.72 ±0.02	**49.69** ±0.02
		1 DNN	20.50 ±0.02	20.97 ±0.00	⋆45.90 ±0.02	⋆48.60 ±0.02	16.37 ±0.01
		4 DNNs	32.43 ±0.06	32.25 ±0.04	**63.11** ±0.07	**65.64** ±0.06	25.34 ±0.02
	WRN28	cSGLD	**81.99** ±0.01	**85.63** ±0.01	27.04 ±0.02	23.46 ±0.01	68.43 ±0.01
		1 DNN	49.24 ±0.16	52.84 ±0.03	20.23 ±0.04	18.53 ±0.02	60.84 ±0.09
		4 DNNs	77.45 ±0.01	79.55 ±0.13	**36.33** ±0.13	**33.60** ±0.22	**83.24** ±0.00

Table 4.12: Inter-architecture transfer success rates of I-FGSM of single architecture surrogate on MNIST (in %). All combinations of surrogate and targeted architectures are evaluated. Diagonals are intra-architecture. cSGLD has always higher transferability than 1 DNN. Symbols ⋆ indicate Bayesian methods (SVI or HMC) having lower transferability than 1 DNN. 1 DNN and cSGLD have similar computation budget (50 epochs). Bold is best. Higher is better.

	Surrogate		**Target Architecture**		
Norm	Arch	Training	Small FC	FC	CNN
L2	Small FC	cSGLD	**99.17** ±0.01	**97.15** ±0.05	**46.04** ±0.15
		HMC	⋆2.66 ±0.01	⋆2.04 ±0.01	⋆0.37 ±0.01
		SVI	⋆5.67 ±0.09	⋆4.04 ±0.09	⋆0.62 ±0.02
		1 DNN	44.19 ±0.00	43.98 ±0.00	19.35 ±0.00
		5 DNNs	48.01 ±0.01	47.78 ±0.04	24.76 ±0.02
		10 DNNs	52.36 ±0.09	51.97 ±0.11	26.52 ±0.05
		15 DNNs	53.13 ±0.09	52.84 ±0.08	27.05 ±0.12
	FC	cSGLD	**98.61** ±0.00	**97.36** ±0.03	**49.27** ±0.17
		SVI	17.16 ±0.17	15.47 ±0.17	⋆4.85 ±0.06
		1 DNN	15.37 ±0.00	15.32 ±0.00	10.40 ±0.00
		5 DNNs	23.13 ±0.06	23.07 ±0.08	16.03 ±0.06
		10 DNNs	24.55 ±0.14	24.46 ±0.13	16.96 ±0.21
		15 DNNs	23.46 ±0.13	23.44 ±0.12	16.44 ±0.21
	CNN	cSGLD	**46.86** ±0.27	**47.06** ±0.32	**92.57** ±0.14
		1 DNN	10.73 ±0.00	10.43 ±0.00	14.80 ±0.00
		5 DNNs	22.20 ±0.09	22.22 ±0.05	28.69 ±0.03
		10 DNNs	19.18 ±0.23	19.27 ±0.34	23.84 ±0.40
		15 DNNs	19.71 ±0.22	19.83 ±0.22	24.33 ±0.26
L∞	Small FC	cSGLD	61.75 ±0.25	37.66 ±0.25	**1.25** ±0.01
		HMC	⋆1.24 ±0.01	⋆0.91 ±0.03	⋆0.10 ±0.01
		SVI	⋆1.76 ±0.02	⋆1.25 ±0.01	⋆0.16 ±0.03
		1 DNN	58.77 ±0.00	32.15 ±0.00	0.95 ±0.00
		5 DNNs	66.81 ±0.02	37.40 ±0.04	1.04 ±0.01
		10 DNNs	67.88 ±0.18	38.22 ±0.02	1.02 ±0.02
		15 DNNs	**68.07** ±0.13	**38.35** ±0.08	1.04 ±0.03
	FC	cSGLD	**60.06** ±0.01	41.04 ±0.02	**1.33** ±0.01
		SVI	⋆4.29 ±0.02	⋆3.18 ±0.05	⋆0.30 ±0.01
		1 DNN	40.15 ±0.00	34.01 ±0.00	1.11 ±0.00
		5 DNNs	51.62 ±0.05	42.66 ±0.17	1.25 ±0.02
		10 DNNs	54.05 ±0.52	44.44 ±0.15	1.26 ±0.02
		15 DNNs	55.03 ±0.45	**44.78** ±0.27	1.27 ±0.01
	CNN	cSGLD	3.07 ±0.08	2.89 ±0.04	5.42 ±0.03
		1 DNN	2.40 ±0.00	2.30 ±0.00	3.83 ±0.00
		5 DNNs	3.50 ±0.03	3.09 ±0.06	6.05 ±0.04
		10 DNNs	3.79 ±0.04	**3.39** ±0.01	6.37 ±0.03
		15 DNNs	**3.81** ±0.09	3.37 ±0.04	**6.55** ±0.05

Table 4.13: cSGLD improves all techniques compared to 1 DNN. Transfer success rates of (M)I-FGSM improved by our approach combined with test-time techniques on ImageNet (in %). Target in column. "RN50" column is intra-architecture transferability, other columns are inter-architecture. Bold is best. Symbols ⋆ are DNN-based techniques better than our vanilla cSGLD surrogate, † are techniques that degrade their vanilla surrogate.

		Target Architecture				
Norm	**Surrogate**	RN50	RNX50	DN121	MNASN	EffNetB0
L2	1 DNN	56.60 ±0.71	41.09 ±0.61	29.73 ±0.30	28.13 ±0.17	16.64 ±0.33
	+ DI	83.15 ±0.30	73.17 ±0.80	61.24 ±0.58	58.16 ±0.36	⋆42.10 ±0.36
	+ SGM	65.64 ±0.88	52.75 ±0.42	38.58 ±0.55	43.40 ±0.61	29.11 ±0.30
	+ GN	78.84 ±0.46	62.46 ±0.38	45.76 ±0.02	41.44 ±0.58	25.77 ±0.11
	+ MI	†52.53 ±0.80	†37.15 ±0.76	†26.33 ±0.48	†25.21 ±0.42	†14.74 ±0.31
	+ DI	80.81 ±0.72	69.55 ±0.83	56.73 ±0.39	54.16 ±0.05	37.07 ±0.03
	+ SGM	65.65 ±0.95	53.25 ±0.18	38.79 ±0.62	44.33 ±0.63	29.45 ±0.28
	+ GN	71.50 ±0.12	53.45 ±0.65	37.39 ±0.47	34.53 ±0.69	20.29 ±0.36
	cSGLD	84.83 ±0.55	74.73 ±0.82	71.45 ±0.56	60.14 ±0.44	39.71 ±0.20
	+ DI	**93.87 ±0.19**	**89.12 ±0.24**	**88.52 ±0.16**	**82.78 ±0.28**	**66.13 ±0.35**
	+ SGM	†83.17 ±0.85	†72.79 ±1.06	†66.19 ±0.89	71.71 ±0.41	52.66 ±0.31
	+ GN	92.99 ±0.13	85.69 ±0.24	82.81 ±0.42	72.88 ±0.30	50.30 ±0.29
	+ MI	†82.44 ±0.19	†70.93 ±1.04	†66.19 ±0.56	†55.51 ±0.59	†34.49 ±0.59
	+ DI	93.48 ±0.23	87.87 ±0.15	86.81 ±0.33	80.37 ±0.20	60.26 ±0.02
	+ SGM	†82.35 ±0.10	†71.54 ±0.58	†64.50 ±0.18	70.47 ±0.22	50.80 ±0.23
	+ GN	90.11 ±0.18	80.35 ±0.61	75.10 ±0.67	64.08 ±0.12	39.85 ±0.52
L∞	1 DNN	47.81 ±1.09	32.29 ±0.64	23.43 ±0.32	22.52 ±0.45	12.77 ±0.32
	+ DI	76.55 ±1.01	62.57 ±0.56	50.17 ±0.33	49.31 ±0.18	⋆32.64 ±0.09
	+ SGM	66.36 ±0.50	51.60 ±0.36	39.05 ±0.24	45.60 ±0.72	30.69 ±0.03
	+ GN	67.02 ±0.17	46.74 ±0.63	32.57 ±0.17	31.12 ±0.77	17.68 ±0.05
	+ MI	55.12 ±0.82	38.47 ±0.82	28.19 ±0.14	27.55 ±0.67	16.34 ±0.37
	+ DI	⋆82.47 ±0.41	⋆69.69 ±0.81	57.79 ±0.57	⋆55.99 ±0.37	⋆38.63 ±0.29
	+ SGM	68.39 ±0.53	54.57 ±0.60	41.48 ±0.37	47.97 ±0.41	⋆33.16 ±0.37
	+ GN	71.27 ±0.54	51.46 ±0.84	36.91 ±0.48	34.54 ±0.32	20.51 ±0.30
	cSGLD	78.71 ±1.19	65.11 ±1.45	61.49 ±0.59	51.81 ±1.45	31.11 ±0.99
	+ DI	90.03 ±0.10	82.13 ±0.45	81.19 ±0.34	74.48 ±0.39	53.51 ±0.39
	+ SGM	81.37 ±0.72	69.88 ±1.31	65.20 ±0.75	71.68 ±0.53	52.15 ±0.32
	+ GN	87.33 ±0.73	76.00 ±1.33	71.67 ±0.97	61.45 ±0.25	37.19 ±0.68
	+ MI	82.89 ±0.70	70.42 ±1.26	66.39 ±0.74	56.68 ±0.97	36.00 ±1.15
	+ DI	**93.97 ±0.26**	**87.69 ±0.44**	**86.78 ±0.16**	**81.08 ±0.14**	**60.87 ±0.48**
	+ SGM	84.19 ±0.21	73.14 ±0.99	67.35 ±0.26	74.36 ±0.47	55.30 ±0.16
	+ GN	89.53 ±0.05	78.69 ±0.19	73.33 ±0.58	63.56 ±0.35	39.79 ±0.52

Table 4.14: Transfer success rates of (M)I-FGSM attack improved by our approach combined with test-time transformations on CIFAR-10 (in %). Columns are targets. PreResNet110 columns are intra-architecture transferability, others are inter-architecture. Bold is best. Symbols ⋆ are DNN-based techniques better than our vanilla cSGLD surrogate, and † are techniques that do not improve the corresponding vanilla surrogate. The success rate of every cSGLD-based technique is better than its 1 DNN-based counterpart.

		Target Architecture				
Norm	**Surrogate**	PRN110	PRN164	VGG16	VGG19	WRN28
L2	1 DNN	34.42 ±0.00	34.39 ±0.01	12.67 ±0.00	12.54 ±0.00	26.29 ±0.01
	+ DI	59.63 ±0.80	59.79 ±0.75	24.37 ±0.16	23.25 ±0.12	46.09 ±0.47
	+ SGM	57.00 ±0.00	57.66 ±0.04	20.87 ±0.03	20.10 ±0.09	41.80 ±0.04
	+ GN	79.22 ±0.30	80.38 ±0.16	⋆32.03 ±0.25	⋆28.63 ±0.17	56.65 ±0.24
	+ MI	67.12 ±0.07	67.80 ±0.00	20.49 ±0.02	19.15 ±0.01	44.11 ±0.04
	+ DI	81.44 ±0.32	82.69 ±0.29	27.64 ±0.03	25.82 ±0.42	57.29 ±0.12
	+ SGM	73.52 ±0.00	75.23 ±0.01	24.52 ±0.00	22.76 ±0.00	49.73 ±0.00
	+ GN	77.44 ±0.28	79.13 ±0.12	⋆28.98 ±0.57	25.74 ±0.18	54.06 ±0.04
	cSGLD	90.67 ±0.39	89.74 ±0.31	28.05 ±0.33	26.12 ±0.14	67.27 ±0.89
	+ DI	92.45 ±0.14	91.80 ±0.14	33.69 ±0.28	31.35 ±0.28	72.41 ±0.76
	+ SGM	92.46 ±0.17	92.10 ±0.28	31.96 ±0.53	29.84 ±0.34	71.04 ±1.23
	+ GN	**92.73 ±0.21**	**92.20 ±0.07**	**36.17 ±0.39**	**33.08 ±0.32**	**74.77 ±0.10**
	+ MI	†90.35 ±0.37	89.77 ±0.28	†26.89 ±0.37	†25.02 ±0.29	†65.98 ±0.52
	+ DI	92.31 ±0.33	91.58 ±0.23	31.92 ±0.49	29.72 ±0.46	70.94 ±0.31
	+ SGM	92.33 ±0.34	91.94 ±0.41	31.95 ±0.29	29.85 ±0.28	70.96 ±0.65
	+ GN	92.42 ±0.16	91.93 ±0.25	33.02 ±0.60	29.77 ±0.14	72.28 ±0.53
L∞	1 DNN	72.73 ±0.00	74.58 ±0.01	22.26 ±0.00	20.98 ±0.00	47.59 ±0.01
	+ DI	81.29 ±0.18	82.77 ±0.12	28.10 ±0.22	26.17 ±0.25	57.04 ±0.10
	+ SGM	77.92 ±0.00	79.50 ±0.01	27.43 ±0.00	25.31 ±0.01	53.39 ±0.00
	+ GN	74.92 ±0.08	77.23 ±0.26	⋆29.61 ±0.19	26.31 ±0.30	52.93 ±0.05
	+ MI	76.12 ±0.01	78.05 ±0.00	23.77 ±0.02	22.33 ±0.01	50.49 ±0.01
	+ DI	84.66 ±0.19	86.38 ±0.12	⋆31.47 ±0.05	⋆28.89 ±0.31	61.60 ±0.16
	+ SGM	79.72 ±0.02	80.80 ±0.02	28.75 ±0.01	26.12 ±0.00	55.74 ±0.00
	+ GN	80.34 ±0.34	82.59 ±0.42	⋆34.17 ±0.48	⋆29.37 ±0.18	60.62 ±0.40
	cSGLD	90.98 ±0.40	90.26 ±0.35	29.26 ±0.53	26.97 ±0.43	67.18 ±1.03
	+ DI	92.46 ±0.14	91.62 ±0.16	33.81 ±0.25	30.84 ±0.34	71.15 ±0.92
	+ SGM	93.38 ±0.50	92.84 ±0.25	35.68 ±0.61	32.43 ±0.52	73.55 ±1.08
	+ GN	91.66 ±0.40	91.32 ±0.19	34.77 ±0.09	31.01 ±0.27	71.60 ±0.40
	+ MI	92.84 ±0.18	92.18 ±0.28	32.03 ±0.49	28.53 ±0.38	71.56 ±0.25
	+ DI	94.05 ±0.31	93.53 ±0.21	37.31 ±0.38	33.23 ±0.23	75.40 ±0.25
	+ SGM	**94.64 ±0.26**	**94.29 ±0.31**	**38.08 ±0.27**	**34.28 ±0.17**	**76.62 ±0.50**
	+ GN	93.76 ±0.14	93.75 ±0.13	38.01 ±0.44	33.15 ±0.36	76.23 ±0.29

from the posterior distribution – yields significantly higher improvements. Hence, while training approaches have been overlooked, canonical elements that have been related to transferability, i.e., skip connections [WWX+20], input [XZZ+19] and model diversity [LBZ+18], should be put into perspective compared to the importance that the posterior distribution appears to have.

4.4.5 Multimodal vs. Local Exploration

While studying the effect of the hyperparameters of cSGLD, we show that the most important cSGLD capability is its multimodal exploration. The local mode exploration of cSGLD has limited ability to increase transferability.

First, we evaluate the importance of the number of cSGLD cycles for transferability. The number of cycles is a strong proxy for the number of explored modes. cSGLD is the only SG-MCMC method we know that avoids mode collapse and, therefore, can effectively explore complex multimodal distributions [ZLZ+20]. We train cSGLD on CIFAR-10 for 18 cycles. Figure 4.6 shows that transferability of both L2 and L∞ attacks increases significantly for the first new five modes explored. After eight cycles, the transferability plateaus. Therefore, the multimodal exploration of cSGLD is key to its success.

Secondly, we access the limited impact of the number of samples per cycle on transferability. We train cSGLD on CIFAR-10 ten times, from one to ten samples per cycle, each using 5 cycles. Each additional sample per cycle increases the training cost by one epoch per cycle (starting at 48 epochs per cycle). Figure 4.7 shows that the number of cycles per epoch has a relatively low impact on transferability. The variations are contained within 91.0% and 93.0% for the L2 attack, and within 91.75% and 93.5% for the L∞ attack. We conclude that a more fine-grained exploration of a single mode by cSGLD only marginally increases transferability.

Overall, cSGLD increases transferability mainly thanks to warm restarts and only marginally thanks to local mode exploration. Improving upon the local mode exploration is a promising research direction.

4.4.6 Bayesian and Ensemble Techniques

In addition to cSGLD and deep ensemble, we explore the use of other training techniques to improve transferability: two other Bayesian techniques – VI and SWAG – and two other ensembling techniques – SSE and FGE. We train each for an equivalent computational cost of 3 DNNs on CIFAR-10 and 2 DNNs on ImageNet (except for VI and SWAG, see discussion in Section 3.3.3). Figure 4.8 presents the success rate of L∞ I-FS(S)M attack with the corresponding training computational cost (in flops), as we increase the number of models in each surrogate. Section 3.3.3 contains details on the methods and the results for L2 I-FS(S)M.

On CIFAR-10, the success rate of the first 4 cycles of cSGLD increases substantially from one cycle to the next (from 76.58% to 81.56% for the first to the second

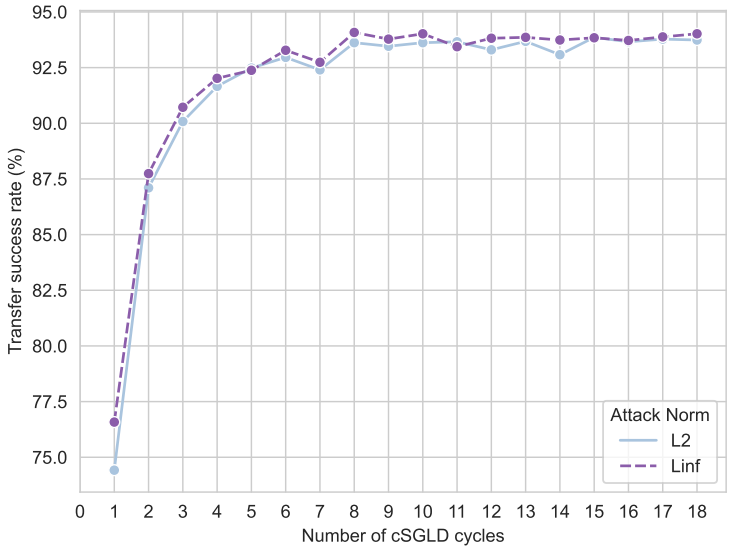


Figure 4.6: Intra-architecture transfer success rate of I-FGSM with respect to the number of cSGLD cycles on CIFAR-10 (PreResNet110).

cycle) and within a single cycle (from 81.56% to 87.20% between the start and the end of the second cycle). This reveals that exploring modes of the posterior plays an important role to generate transferable adversarial examples, and that there is some local geometric discrepancy of the loss landscape among local maxima. On ImageNet, transferability improves mainly by sampling from several local optima.

Interestingly, even though FGE and SWAG build an ensemble around a single local optimum, their flexibility allows capturing general adversarial directions. The FGE surrogates trained for more than 0.30 petaflops have systematically higher success rates than cSGLD and SSE on CIFAR-10. However, the opposite is observed on ImageNet: FGE is not competitive with methods exploring several local optima (cSGLD, SSE, and deep ensemble). We hypothesize that modes are not as well-connected on larger datasets.

The efficiency of SWAG on both datasets opens new directions to create hybrid attacks based on a few additional iterations over the training set. SWAG approximates the posterior with a Gaussian fitted on some additional SGD epochs from a pretrained DNN. It captures well the shape of the true posterior [MGI+19a], reinforcing our views on the strong relationship between the posterior and transferability. The success rate gap between cSGLD/SSE and SWAG on ImageNet suggests higher geometrical discrepancies between local loss maxima on larger datasets.

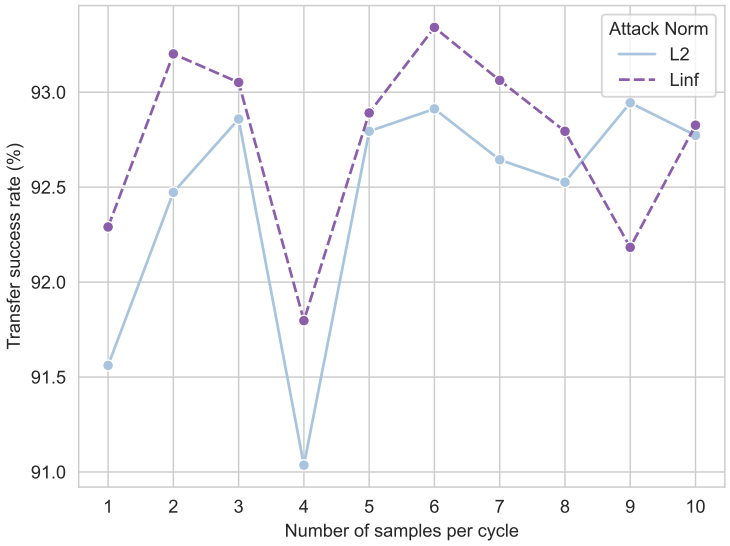


Figure 4.7: Intra-architecture transfer success rate of I-FGSM with respect to the number of cSGLD samples per cycle. We train one PreResNet110 cSGLD on CIFAR-10 for every number of cycles, from 1 to 10 samples per cycle. Each additional sample per cycle increases the training cost by one epoch per cycle (starting at 48 epochs per cycle). A fixed number of 5 cSGLD cycles is used.

VI fails to compete with deep ensemble on both success rate and computational efficiency for the L∞ attack on CIFAR-10, but beats it on L2 bound and on ImageNet.

On CIFAR-10, the marginal impact beyond 6 cSGLD cycles, 17 SWAG samples, 7 SSE models, and 35 FGE models becomes noisy. We hypothesize that correlated samples produce these limitations. Hence, the use of multiple runs is a promising direction for greater transferability.

Finally, we train one surrogate model with SWA [IPG+18] on CIFAR-10. SWA keeps a running average of the weights being trained with SGD. We reuse the original hyperparameter of SWA to train one PreResNet110 model on CIFAR-10, extended to 300 epochs for a fair comparison. The final SWA model is the average of weights obtained at the end of every epoch from epoch 161 to epoch 300. This surrogate model obtains a success rate of 70.52% for the L∞ attack, and of 67.42% for the L2 attack. SWA improves over a regularly trained DNN, for the L2 attack (67.42% vs. 46.20%), but not for the L∞ attack (70.52% vs. 78.14%). More importantly, SWAG improves consistently over SWA, with a respective success rate of 92.38% and 70.52% for the L∞ attack, and of 92.28% and 67.42% for the

L2 attack. Estimating the covariance matrix of the Gaussian distribution used by SWAG to sample deviations from its SWA expected value, improves transferability. This observation confirms our observation about SWAG that the local diversity of models is a promising direction to improve diversity at low cost.

4.5 Threats to Validity

External validity threats arise from the generalization outside the context of the study. First, our results may not generalize to non p-norm constrained adversarial examples. However, this way of ensuring imperceptibility is common to all the related work we know on transferability. We also systematically evaluate L2 and L∞ attacks, while most previous studies do not. Second, similar to all competitive approaches, we consider benchmark datasets of classification in computer vision. The generalization of our conclusions to other domains and tasks could require a dedicated study. Finally, we fed adversarial examples directly to the target model. Evaluating adversarial examples through the physical domain may degrade success rates significantly.

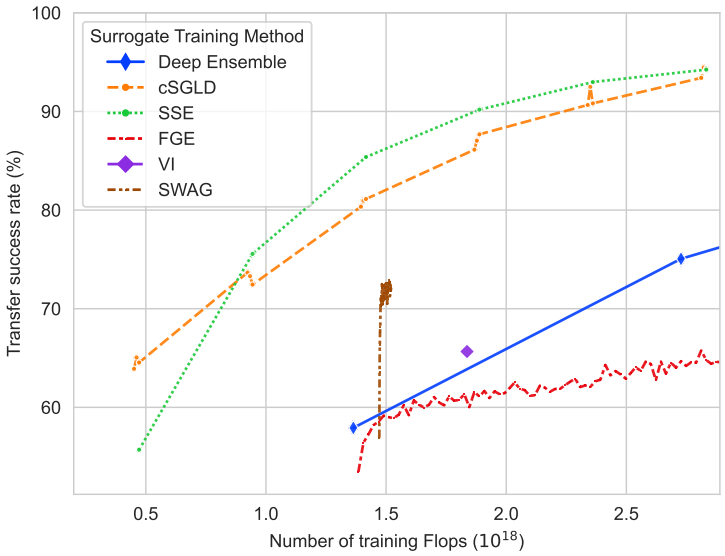
Internal validity threats come from the design of the study. Our approach relies on the empirical fact that SGD is approximately a Bayesian sampler [MVS+20]. A definitive proof would strengthen the premise of our contributions. Moreover, despite our best effort to control confounding factors, some may exist, such as training hyperparameters.

Threat to construct validity is a consequence of metrics not suitable for evaluation. Our T-DEE metric might not be reliable when the success rate is not increasing with the number of independently trained DNNs. None of our experiences exhibit this (except L2 FSGM on MNIST).

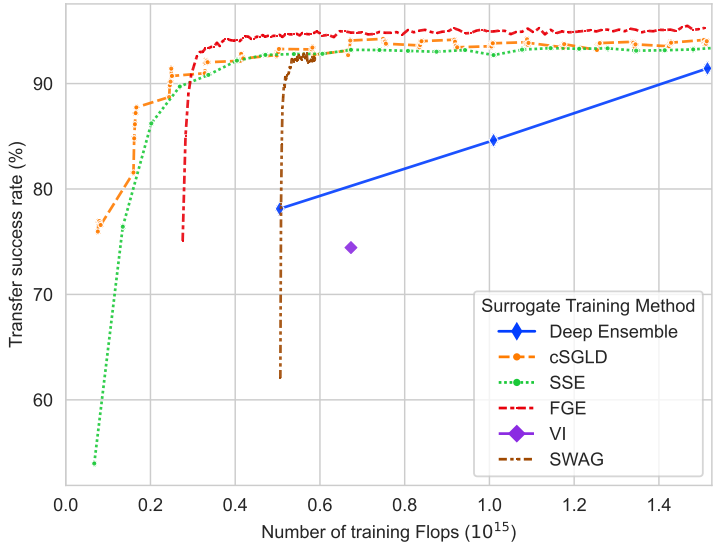
4.6 Conclusion

The contribution presented in this chapter was the first to investigate how to explore a surrogate weight space to enhance transfer-based black-box attacks. Through extensive empirical experiments, we show the key role of the training of the surrogate model in boosting the transferability of adversarial examples.

We connect uncertainty and transferability by considering a threat model, where we can characterize the probabilistic distribution of the unknown weight of the target model. Deep ensemble can sample a surrogate from this distribution. We exhibit experimentally the effectiveness of the deep ensemble surrogate. Unfortunately, deep ensemble requires training numerous DNNs from scratch, and is therefore extensive. We use a cheaper approximation based on cSGLD, a state-of-the-art Bayesian training techniques. This training technique samples from the Bayesian posterior over weights, which approximates the distribution of the target model. We establish empirically that our Bayesian surrogate is efficient and effective to

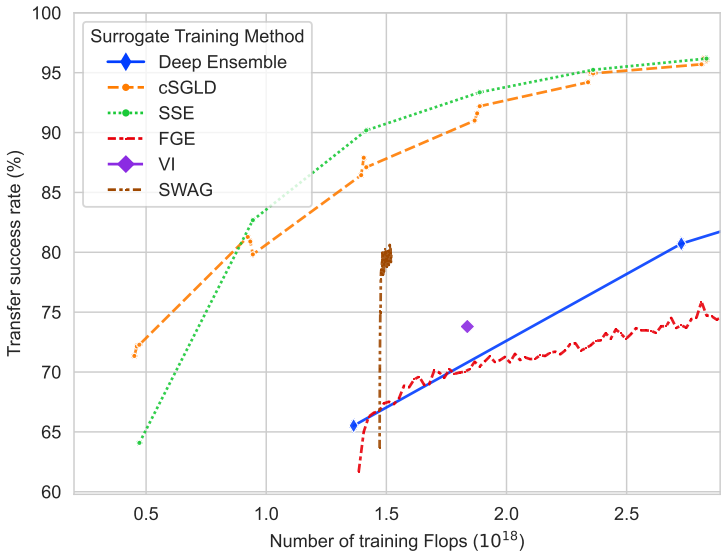


(a) ImageNet

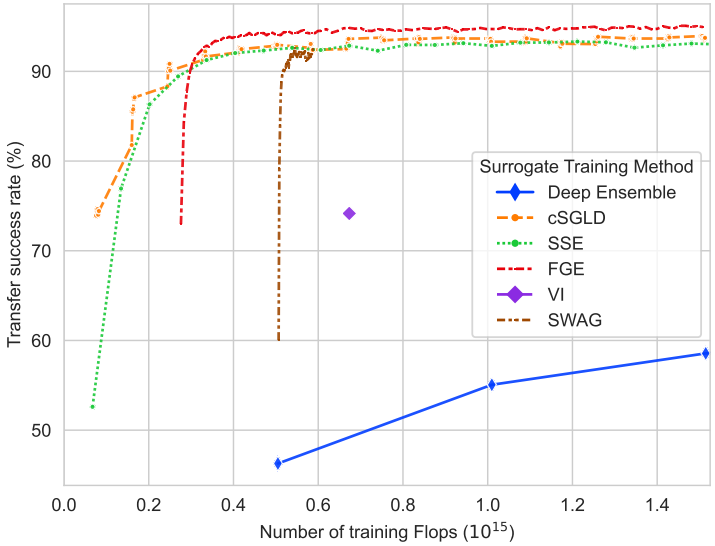


(b) CIFAR-10

Figure 4.8: Intra-architecture L∞ I-FGSM success rate with respect to the training computational complexity, measured by the number of Flops, of an increasing number of samples from six training techniques. Every curve starts with one model, and each successive point is obtained by forming an ensemble with one more model.



(a) ImageNet



(b) CIFAR-10

Figure 4.9: Intra-architecture L2 I-FGSM success rate with respect to the training computational complexity, measured by the number of Flops, of six Bayesian and Ensemble methods. Every curve starts with one model, and each successive point is obtained by forming an ensemble with one more model.

craft adversarial examples transferable to deterministic DNNs.

Our approach further improves existing adversarial attacks and transferability techniques, as one can use it on top of them to perform approximate Bayesian model averaging efficiently and with minimal modifications. We show that our simple training-time approach improves transferability more than previous test-time techniques. Overall, we provide new evidence that the training of the surrogate model is an overlooked direction for research on transferability.

The evaluated approaches increase transferability from the multimodal exploration of the weight space, i.e., the exploration of several separated vicinities. Deep ensemble independently trains a set of models, where each model belongs to a separate vicinity [FHL19]. In Section 4.4.5, we establish that cSGLD boosts transferability mainly due to its multimodal exploration of the loss landscape, and that its local exploration of each vicinity is poor for transferability. Therefore, an effective and efficient local exploration of the loss landscape is an open challenge. The following chapter tackles this research direction.

We also pave the way for new hybrid transferability techniques in-between training time and testing time, i.e., at the moment of the attack. In particular, Section 4.4.6 identifies that SWAG can increase transferability in just a few additional epochs from a fully trained surrogate DNN. SWAG reaches a similar success rate than cSGLD on CIFAR-10 for the same computational cost. However, SWAG is not competitive on ImageNet. The next chapter addresses this limitation by proposing a technique, inspired by SWAG, that collects models in a few additional epochs.

5

Transferability From Large Geometric Vicinity

The local exploration of the weight space is key to improve a surrogate, since random directions in the weight space improve transferability, whereas random directions in the input space do not. This chapter proposes LGV (Large Geometric Vicinity), a new technique to increase transferability from a regularly trained DNN. We analyse extensively the success of LGV through the lens of the geometry of the weight space. We point two key elements: the flatness of the loss landscape that we relate to transferability with the surrogate-target misalignment hypothesis, and the geometry of the subspace spanned by LGV weights.

Contents

This chapter is based on the following paper:

- Martin Gubri et al. LGV: Boosting Adversarial Example Transferability from Large Geometric Vicinity. In *ECCV 2022*, 2022. URL: `https://gubri.eu/publication/lgv_eccv22/`

5.1 Introduction

Chapter 4 delved into the underexplored topic of training the surrogate model of a transfer-based black-box attacks. While our findings highlight the critical role surrogate model training plays in boosting the transferability of adversarial examples, the journey is far from over. One key revelation from our analysis was the need for a more effective and efficient strategy for local exploration of the loss landscape, i.e, collecting surrogate models from the same mode. This chapter is dedicated to addressing this open challenge, building on the groundwork laid in the preceding chapter.

In Chapter 4, we demonstrated the benefits of a multimodal exploration of the weight space, using deep ensemble or cSGLD, which involves sampling models from several separate vicinities. However, the local exploration of each vicinity via cSGLD was less fruitful for transferability, and deep ensemble outputs a single model per vicinity (see Figure 3.1 for an illustration of the phenomenon). To overcome this, this chapter will introduce and evaluate new techniques to explore efficiently a vicinity of the loss landscape to build strong surrogate models.

Moreover, Chapter 4 revealed an interesting potential for hybrid transferability techniques, that apply training techniques at the time of the attack. Specifically, SWAG emerged as a promising candidate, capable of enhancing transferability with only a few additional epochs of training from a fully trained surrogate DNN. Inspired by SWAG, we will propose the LGV technique that collects models in a few additional epochs. Our preliminary experiments reveal that SWAG applied on top of our LGV technique degrades its success rate.

Transferability is challenging to achieve consistently, and the factors behind transferability (or lack thereof) remain an active field of study (see Section 1.2). A type of transferability technique relies on augmenting the surrogate model to build diversity in the surrogate representation [LBZ+18; WWX+20; GLC20] (see Section 2.4.5 for a detailed discussion). While these approaches typically report significantly higher success rates than a classical surrogate, the relationships between the properties of the surrogate and transferability remain obscure. Understanding these relationships would enable the efficient construction of attacks (which would directly target the properties of interest) that effectively improve transferability. By focusing on the local exploration of the loss landscape, this chapter attempts to advance our understanding of surrogate model training, surrogate model augmentation and their impacts on the transferability of adversarial examples. The majority

of this chapter is dedicated to building strong empirical evidence to explain why our proposed technique LGV increases transferability.

As a first step to motivate our approach, we exhibit the importance of exploring locally the geometry of the surrogate loss in improving transferability. We show that random directions in the weight space from a regularly trained DNN increases its transferability, whereas random directions in the input space applied on gradients do not.

Then, we propose Transferability from Geometric Vicinity (LGV), an efficient technique to increase the transferability of black-box adversarial attacks. LGV starts from a pretrained surrogate model and collects multiple weight samples from a few additional training epochs with a constant and high learning rate. Through extensive experiments, we show that LGV outperforms competing techniques by 3.1 to 59.9 percentage points of transfer rate. LGV also complements nicely the deep ensemble surrogate proposed in Chapter 4, combining local and multimodal exploration of the weight space.

We relate this improved transferability to two properties of the weights that LGV samples. First, LGV samples weights on a flatter surface of the loss landscape in the weight space, leading to flatter adversarial examples in the feature space. Our observations support our hypothesis that misalignment between surrogate and target alters transferability, which LGV avoids by sampling from flatter optima. Second, the span of LGV weights forms a dense subspace whose geometry is intrinsically connected to transferability, even when the subspace is shifted to other local optima.

DNN geometry has been intensively studied from the lens of natural generalization [LFL+18; GRD18; KNT+16; IPG+18; FKM+20; WXW20]. However, the literature on the importance of geometry to improve transferability is scarcer [TPG+17; CRP20] and has not yielded actionable insights that can drive the design of new transferability methods (more in Chapter 3).

Our main contribution is, therefore, to shed new light on the importance of the surrogate loss geometry to explain the transferability of adversarial examples, and the development of the LGV method that improves over state-of-the-art transferability techniques.

5.2 Experimental Settings

Setup summary. Our study uses standard experimental settings to evaluate transfer-based black-box attacks. The surrogates are trained ResNet-50 models from [ALM+20]. The targets are eight trained models from PyTorch [PGM+19] with a variety of architectures – including ResNet-50. Therefore, we cover both the intra-architecture and inter-architecture cases. We craft adversarial examples from a random subset of 2000 ImageNet test images that all eight targets classify correctly.

Table 5.1: Natural accuracy and loss of target models computed on the test set.

Name	Architecture	Test Accuracy	Loss (NLL)
RN50	ResNet-50	76.01%	0.963
RN152	ResNet-152	78.25%	0.876
RNX50	ResNext-50	77.63%	0.941
WRN50	WideResNet-50	78.46%	0.883
DN201	DenseNet-201	76.93%	0.926
VGG19	VGG19	72.36%	1.115
IncV1	Inception v1 (GoogLeNet)	69.74%	1.283
IncV3	Inception v3	76.25%	1.041

We compare LGV with four test-time transformations and their combinations, all applied on top of I-FGSM. We do not consider query-based black-box attacks because the threat model of transfer attacks does not grant oracle access to the target. To select the hyperparameters of the attacks, we do cross-validation on an independent subset of well-classified training examples. We provide results for L_∞ and L_2 norm bounded perturbations. We report the average and standard deviation of the attack success rate, i.e., the misclassification rate of untargeted adversarial examples, over 3 independent runs. Each run involves independent sets of examples, different surrogate models, and different random seeds. All code and models are available on GitHub[1].

Notation. In the following, we denote (x, y) an example in $\mathcal{X} \times \mathcal{Y}$ with $\mathcal{X} \subset \mathbb{R}^d$, w a vector of p DNN weights in $\mathbb{R}^p$, and $\mathcal{L}(x; y, w)$ the loss function at input x of a DNN parametrised by w. The weights of a regularly trained DNN are noted w_0. Our LGV approach samples K weights $w_1, \cdots, w_K$. We name **LGV-SWA** the model parametrised by the empirical average of weights collected by LGV, i.e. $w_{\text{SWA}} = \frac{1}{K} \sum_{k=1}^{K} w_k$. The dot product between two vectors u, v is noted $\langle u, v \rangle$.

Target models. We select 8 pretrained models distributed by the *torchvision* library [PGM+19]. The architectures are diverse and belong to different families. We choose ResNet-50 to study the intra-architecture transferability, ResNet-152 for the effect of increasing the number of layers, ResNeXt-50 32Œ4d and WideResNet-50-2 for other variants in the ResNet family, DenseNet-201, VGG19, Inception v1 (GoogLeNet) and Inception v3 to represent other families. Table 5.1 contains their natural accuracy and negative log likelihood (NLL).

Surrogate models. We retrieve the independently trained ResNet-50 DNNs from [ALM+20]. All DNNs share the same hyperparameters, and have different

[1] `https://github.com/Framartin/lgv-geometric-transferability`

Table 5.2: Natural accuracy and loss of surrogate models computed on the test set.

Method	Test Accuracy	Loss (NLL)	Number of models
1 DNN	76.14% ±0.14	0.945 ±0.003	1
1 DNN + RD	76.17% ±0.10	0.948 ±0.003	50
LGV-SWA	72.17% ±0.10	1.128 ±0.002	1
LGV (ours)	70.83% ±0.10	1.310 ±0.011	40

random initializations. For each experiment run, we select without replacement a random DNN, and call it interchangeably "1 DNN", the initial DNN, or the DNN with weights w_0. LGV starts from the weights w_0 of this DNN. . "1 DNN + RD" is defined in Section 5.3 and "LGV-SWA" in Section 5.2. Table 5.2 contains their natural accuracy and negative log likelihood.

Threat model. We study untargeted adversarial examples, the attack objective is misclassification. We consider the less restrictive threat model for transfer-based attack, where no query access to the target model is granted. Therefore, we do not compare with query-based black-box attacks. This experimental setup is standard for transfer-based attack evaluation.

Attack. The I-FGSM attack performs 50 iterations with a step size equal to one tenth of the maximum perturbation norm ε. For the L_∞ attack ε equals $\frac{4}{255}$, and for the L_2 one it equals 3. The number of iterations is selected on a validation set for both the initial DNN surrogate and its resulting LGV surrogate (Figure 5.9). Each iteration compute a single gradient on a randomly selected model if several are available for the method considered. If the number of iterations is higher than the number of models, we cycle on models in the same order. Therefore, the attack cost, measured as the number of backward passes, is kept constant regardless of the number of the size of the surrogate. We do not consider the ensemble of models for fairness with the single model surrogate baselines.

Batch normalisation. Each time our experiments change weights (e.g. when we apply random directions), we perform an additional forward pass over 10% of the training data to update the batch-normalization statistics, following previous studies [IMK+19; MGI+19a]. Translations in the weight space cause internal covariate shift, which may cause our surrogate to fail, regardless of the quality of the corresponding point in the weight space. We control this undesired experimental artefact by updating batch-normalization statistics. LGV does not need such extra computational cost, since regular training updates batch-normalization statistics on the fly.

Table 5.3: Hyperparameters used to train LGV and the initial DNN.

Hyperparameter	1 DNN	LGV
Learning rate schedule	Step size decay $\times 0.1$ each 30 epochs	Constant
Initial learning rate	0.1	0.05
Number of epochs	130	10
Optimizer	SGD	SGD
Momentum	0.9	0.9
Batch-size	256	256
Weight decay	1×10^{-4}	1×10^{-4}

Figures. Figures containing multiple subplots report the success rate on the target indicated in subplot title of adversarial examples crafted against the surrogate indicated in legend or in caption. In all figures containing lines surrounded with a lighter coloured area, the lines are smoothed means[2] over 3 independent runs, and the coloured areas correspond to one standard deviation around the mean.

Implementation. All experiments source code and models are available on GitHub[3]. We adapt the I-FGSM attack from the Python ART library to support the four state-of-the-art transferability techniques. The training of LGV and some experiments are adapted from the code of [IMK+19] on PyTorch. We use the following software versions: Python 3.8.8, PyTorch 1.7.1, Torchvision 0.8.2, Adversarial Robustness Toolbox 1.6.0, and Scikit-learn 0.23.2.

Infrastructure. The GPU used for the experiments is Tesla V100-DGXS-32GB, on a server with the following specifications: 256GB RDIMM DDR4, CUDA version 10.1, Linux (Ubuntu) operating system.

Hyperparameters We use the hyperparameters for training indicated in Table 5.3, and for the attack in Table 5.4.

5.3 Preliminaries: Transferability From the Weight Space

We aim to show the importance of the geometry of the surrogate loss in improving transferability. As a first step to motivate our approach, we experimentally demonstrate that adding random directions in the weight space to a regularly trained DNN increases its transferability, whereas random directions in the input

[2]The smooth means are local polynomial regressions computed by the "loess()" function of the R stats package.

[3]`https://github.com/Framartin/lgv-geometric-transferability`

Table 5.4: Hyperparameters of the I-FGSM attack and transferability techniques.

Attack	Hyperparameter	Values
I-FGSM	Perturbation norm ε	3 for L_2, $\frac{4}{255}$ for L_∞
I-FGSM	Step-size α	$\frac{\varepsilon}{10}$
I-FGSM	Number iterations	50
Momentum (MI)	Momentum term	0.9
Ghost Network (GN)	Skip connection erosion random range	$[1-0.22, 1+0.22]$
Input Diversity (DI)	Minimum resize ratio	90%
Input Diversity (DI)	Probability transformation	0.5
Skip Gradient Method (SGM)	Residual gradient decay γ	0.2

Table 5.5: Success rates of random directions (RD) in the weight and input spaces under the L∞ attack. In %.

	Target							
Surrogate	RN50	RN152	RNX50	WRN50	DN201	VGG19	IncV1	IncV3
1 DNN (baseline)	45.3$_{\pm 2.4}$	29.6$_{\pm 0.9}$	28.8$_{\pm 0.2}$	31.5$_{\pm 1.6}$	17.5$_{\pm 0.6}$	16.6$_{\pm 0.9}$	10.4$_{\pm 0.5}$	5.3$_{\pm 1.0}$
+ **RD Weights**	**60.6**$_{\pm 1.5}$	**40.5**$_{\pm 3.0}$	**39.9**$_{\pm 0.2}$	**44.4**$_{\pm 3.2}$	**22.9**$_{\pm 0.8}$	**22.7**$_{\pm 0.5}$	**13.9**$_{\pm 0.2}$	**6.6**$_{\pm 0.7}$
+ RD Inputs	46.4$_{\pm 1.8}$	29.0$_{\pm 2.2}$	28.7$_{\pm 1.2}$	32.7$_{\pm 1.5}$	17.5$_{\pm 0.6}$	17.5$_{\pm 0.6}$	10.3$_{\pm 0.7}$	5.6$_{\pm 0.7}$

space applied on gradients do not. Then, we develop a theoretical connection between both noises. We show that the difference boils down to structuring the covariance matrix of the Gaussian noise added to input gradients from local variations in the weight space (at the first order approximation).

5.3.1 Gaussian Noise on Inputs

We first establish that random directions in input space do not increase transferability (when not ensembled). [TKP+18] observe that adding a random step to the single-step FGSM attack hinders transferability. We extend this conclusion to the I-FGSM attack. We add Gaussian white noise to the input gradients of the loss function during the attack, $\nabla_x \mathcal{L}(x'_k; y, w_0) + e'_k$ with $e'_k \sim \mathcal{N}(\mathbf{0}, \sigma'^2 I_d)$ where x'_k is the adversarial example at the kth attack iteration. Gradient noise does not improve the success rate (over all considered architectures), regardless of the standard deviation value σ' used (from 1×10^{-7} to 1×10^{-2} ; Figure 5.1). Table 5.5 contains the final success rate of random directions in the input space, with the standard deviation selected by cross-validation.

Our results may appear contradictory to the previous findings of [WZT+18]. However, technical distinctions and the objective of the analysis account for the differences. Wu et al. [WZT+18] improves transferability by averaging several gradients computed from the current inputs perturbed by Gaussian noise in the input space. We do not average over input noise to study the fundamental relations of noises independently of the ensemble effect. Without averaging noises, we observe the pure effect of exploring the space considered, controlling the gradient smoothness from averaging.

5.3.2 Gaussian Noise on Weights

Next, we demonstrate that sampling random directions in the surrogate weight space increases transferability. We build a surrogate called **RD** (see Table 5.6) by adding Gaussian white noise to a DNN with weight w_0:

$$\left\{w_0 + e_k \mid e_k \sim \mathcal{N}(\mathbf{0},\, \sigma I_p),\, k \in [\![1, K]\!]\right\}. \tag{5.1}$$

At each I-FGSM iteration k, we add Gaussian white noise to every weight w_0 to compute the input gradient, $\nabla_x \mathcal{L}(x'_k;\, y, w_0 + e_k)$ with $e_k \sim \mathcal{N}(\mathbf{0},\, \sigma^2 I_p)$. As weights belong to high dimensions[4], the resulting surrogate is approximately uniformly distributed on the sphere centred on w_0 with radius $\sigma\sqrt{p}$.

The noise standard deviation is selected by cross-validation on a validation set (Figure 5.2). Due to computational limitations to update the batch normalisation statistics, we sample only 10 random directions for cross-validation and cycle between samples during the 50 attack iterations.

We found a consistent and significant improvement of transferability – from 1.1 to 20.8 percentage points of success rate – for all eight target architectures and both L_2 and L_∞ attacks, compared to the initial weights w_0. Table 5.5 reports the success rates of both random directions in the input and weight spaces under the L_∞ attack. We call the attack *RD* for random directions in Table 5.6 (L_∞) and Table 5.7 (L_2). RD is reported with standard deviation σ equal to 5×10^{-3} and one random direction per attack iteration.

As a side note and in line with our hypothesis about flatness and transferability developed in Section 5.5, we observe in Section 5.5.2 that "RD" produces flatter adversarial examples than its vanilla DNN counterpart.

Therefore, we showed experimentally that the addition of random directions in the weight space increases transferability, whereas random directions in the input space do not. In the next section, we will study how both induced distributions of gradients differ.

[4]The number of ResNet-50 weights p is 25 557 032.

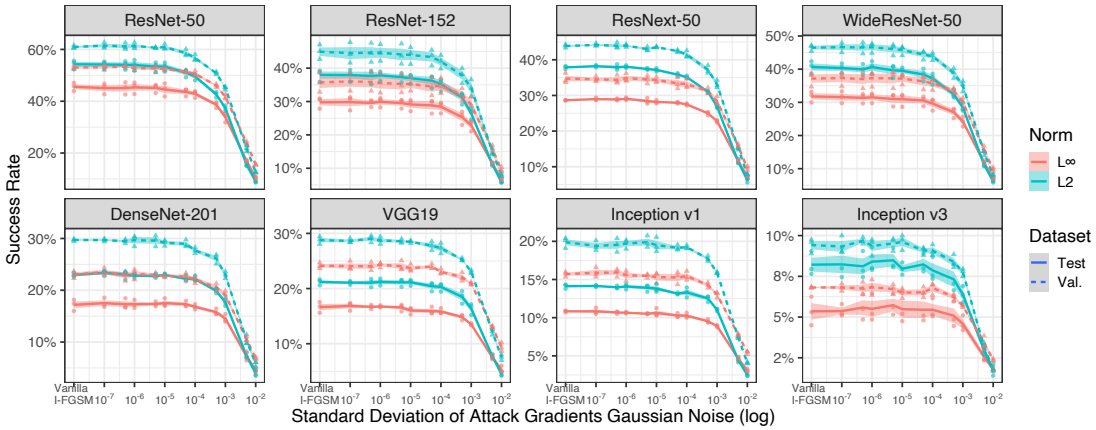


Figure 5.1: Transfer success rate of I-FGSM with respect to the standard deviation of the Gaussian white noise added to the inputs gradients (pseudo-log scale). The null standard deviation is vanilla I-FGSM. The subplot title is the target architecture. The first subplot is intra-architecture transferability.

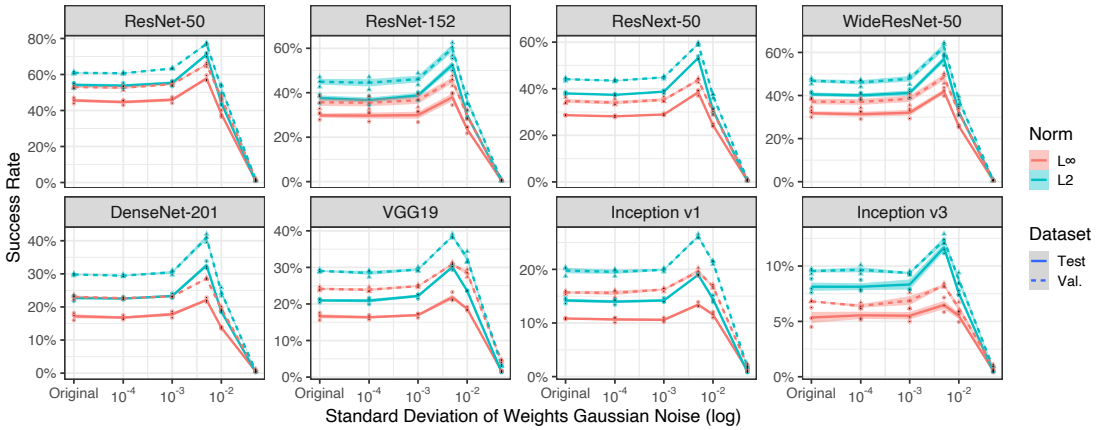


Figure 5.2: Transfer success rate of I-FGSM with respect to the standard deviation of the Gaussian white noise added to the weight of the initial DNN. Ten random directions are sampled in weight space. The subplot title is the target architecture. The first subplot is intra-architecture transferability.

5.3.3 Theoretical Connection Between Both Noises

We develop the theoretical relation between the addition of Gaussian white noise to the gradients in input space, and the addition of Gaussian white noise in the weight space. We show that the distinction boils down to structuring the covariance matrix of the Gaussian noise added to input gradients from local variations in the weight space (at the first order approximation).

According to the Section 5.3.1, after the addition of Gaussian noise to the input gradients, the attack gradients are distributed according to this distribution:

$$\mathcal{N}\left(\nabla_x \mathcal{L}(x'_k;\, y, w_0),\ \sigma'^2\, I_d\right) \tag{5.2}$$

We suppose that the loss function $\mathcal{L}(x;\, y, w)$ is twice continuously differentiable both with respect to x in the L_p ball $B_\varepsilon[x]$, and to w at w_0. To understand the failure of noise in input space and the success of noise in the weight space, we consider the linear approximation of the input loss gradient function $\nabla_x \mathcal{L}(x'_k;\, y, \cdot) : \mathbb{R}^p \longrightarrow \mathcal{X}$, around w_0,

$$\nabla_x \mathcal{L}(x'_k;\, y, w_0 + e_k) = \nabla_x \mathcal{L}(x'_k;\, y, w_0) + \mathbf{J}_{\nabla_x \mathcal{L}(x'_k;\, y, \cdot)}(w_0)\, e_k + o(\|e_k\|), \tag{5.3}$$

with $\mathbf{J}_{\nabla_x \mathcal{L}(x'_k;\, y, \cdot)}(w)$ the Jacobian matrix of the input loss gradient function at w, x'_k the adversarial example at iteration k, and $e_k \sim \mathcal{N}(\mathbf{0},\, \sigma^2 I_p)$. Empirically σ is set to 5×10^{-3}, justifying the local approximation. So, at the first order approximation, the attack gradient is approximately sampled from:

$$\mathcal{N}\left(\nabla_x \mathcal{L}(x'_k;\, y, w_0),\ \sigma^2\, \mathbf{J}_{\nabla_x \mathcal{L}(x'_k;\, y, \cdot)}(w_0)\, \mathbf{J}_{\nabla_x \mathcal{L}(x'_k;\, y, \cdot)}(w_0)^T\right) \tag{5.4}$$

Comparing the distributions of gradients in Equation (5.2) and Equation (5.4), only the noise covariance matrix changes. Therefore, the *structured input noise* induced by local variations of input gradients in the weight space improves transferability.

Overall, we show that sampling random directions in the weight space increases transferability due to the *structured input noise* induced by the surrogate architecture. We develop the connection between input space noise and weight space noise, and show that the latter boils down to adding a structured Gaussian noise in input space with a covariance matrix based on local variations in weight space (to the first order approximation).

These findings reveal that exploiting local variations in the weight space is a promising avenue to increase transferability. However, this success is sensitive to

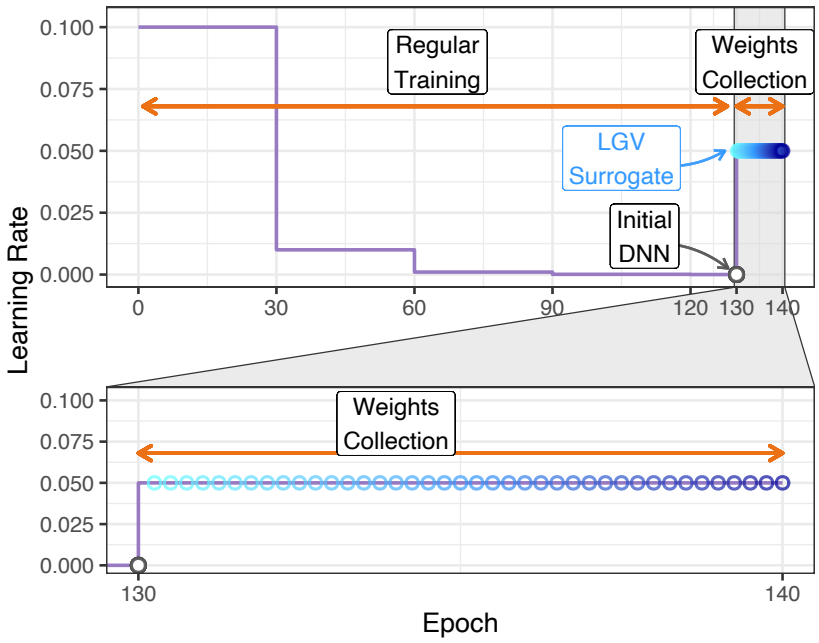


Figure 5.3: Representation of the proposed approach.

the length of the applied random vectors, and only a narrow range of σ values increase the success rate.

5.4 LGV: Transferability From Large Geometric Vicinity

Based on the insights of Section 5.3, we develop LGV (Transferability from Geometric Vicinity), our approach to efficiently build a surrogate from the vicinity of a regularly trained DNN. Despite its simplicity, it beats the combinations of four state of the art competitive techniques. The effectiveness of LGV confirms that the weight space of the surrogate is of first importance to increase transferability.

5.4.1 Algorithm

Our LGV approach performs in two steps: weight collection (Algorithm 4) and iterative attack (Algorithm 5).

First, LGV performs a few additional training epochs from a regularly trained model with weights w_0. LGV collects weights in a single run along the SGD trajectory at regular interval (4 per epoch). The *high constant learning rate* is key for LGV to sample in a sufficiently large vicinity. On the ResNet-50 surrogate we use in our experiments, we run SGD with half the learning rate at the start of the regular training (Figure 5.3). It allows SGD to escape the basin of attraction of

Algorithm 4 LGV Weights Collection.

Input: n_{epochs} number of epochs, K number of weights, η learning rate, γ momentum, w_0 pretrained weights, $\mathcal{D}$ training dataset
Output: $(w_1, \ldots, w_K)$ LGV weights

$w \leftarrow w_0$ ▷ Start from a regularly trained DNN
for $i \leftarrow 1$ **to** K **do**
    $w \leftarrow \text{SGD}(w, \eta, \gamma, \mathcal{D}, \frac{n_{\text{epochs}}}{K})$ ▷ Perform $\frac{n_{\text{epochs}}}{K}$ of an epoch of SGD with η learning rate and γ momentum on $\mathcal{D}$
    $w_i \leftarrow w$
end for

the initial local minimum. Section 5.4.4 includes an in-depth discussion on the type of high learning rates used by LGV. Compared to adding white noise to the weights, running SGD with a high constant learning rate changes the shape of the Gaussian covariance matrix to a non-trivial one [MHB17]. As Table 5.6 shows, LGV improves over random directions (RD).

Second, LGV iteratively attacks the collected models (Algorithm 5). At each iteration k, the attack computes the gradient of one collected model with weights w_k randomly sampled without replacement. If the number of iterations is greater than the number of collected models, we cycle on the models. Because the attack computes the gradient of a single model at each iteration, this step has a negligible computational overhead compared to attacking a single model.

LGV offers multiple benefits. It is efficient (requires 5 to 10 additional training epochs from a pretrained model – see Section 5.4.4), and it requires only minor modifications to training algorithms and adversarial attacks. If memory is limited, we can approximate the collected set of LGV weights by their empirical average (see Section 5.4.5). The most important hyperparameter is the learning rate. In Section 5.4.4, we show that LGV provides reliable transferability improvements for a wide range of learning rate.

5.4.2 Comparison With Other Transferability Techniques

We evaluate the transferability of LGV and compare it with four state-of-the-art techniques.

MI [DLP+18] adds momentum to the attack gradients to stabilize them and escape from local maxima with poor transferability. Ghost Networks (GN) [LBZ+18] use dropout or skip connection erosion to efficiently generate diverse surrogate ensembles. DI [XZZ+19] applies transformations to inputs to increase input diversity at each attack iteration. Skip Gradient Method (SGM) [WWX+20] favors the gradients from skip connections rather than residual modules, and claims that the formers are of first importance to generate highly transferable adversarial examples.

Algorithm 5 I-FGSM Attack on LGV.

Input: (x, y) natural example and its corresponding label, $(w_1, \ldots, w_K)$ LGV weights of the surrogate DNN f_s, n_{iter} number of iterations, ε p-norm perturbation, α step-size, $\mathcal{L}$ loss function

Output: x_{adv} adversarial example

Shuffle $(w_1, \ldots, w_K)$ ▷ Shuffle weights
$x_{\text{adv}} \leftarrow x$
for $i \leftarrow 1$ **to** n_{iter} **do**
    $g \leftarrow \nabla_x \mathcal{L}(f_s(x_{\text{adv}}; w_{i \bmod K}), y)$ ▷ Compute the input gradient of the loss of a randomly picked LGV model
    $x_{\text{adv}} \leftarrow x_{\text{adv}} + \text{project}(g, S_\alpha[\mathbf{0}])$ ▷ Add the normalized gradient, projected in the p-norm sphere of α radius
    $x_{\text{adv}} \leftarrow \text{project}(x_{\text{adv}}, B_\varepsilon[x])$ ▷ Project in the p-norm ball centred on x of ε radius
    $x_{\text{adv}} \leftarrow \text{clip}(x_{\text{adv}}, 0, 1)$ ▷ Clip to pixel range values
end for

We discuss these techniques more deeply in Chapter 3.

Table 5.6 reports the success rates of the ∞-norm attack. We see that LGV alone improves over all (combinations of) other techniques (simple underline). Compared to individual techniques, LGV raises the success rate by 10.1 to 59.9 percentage points, with an average of 35.6. When the techniques are combined, LGV still outperforms them by 1.8 to 55.4 percentage points, and 26.6 on average.

Table 5.7 reports the results of the 2-norm attack. As for the L_∞ attack, LGV alone improves over all (combinations of) other techniques (simple underline). LGV is even more effective on the L_2 attack than the L_∞ one. The vanilla LGV intra-architecture L_2 attack outperforms all techniques applied on LGV, with a margin larger than the sum of the respective standard deviations. In six out of seven inter-architecture targets, input diversity (DI) on top of LGV is best, and in one case, vanilla LGV is.

We also see that combining LGV with test-time techniques does not always improve the results and can even drastically decrease success rate. Still, LGV combined with input diversity (DI) and momentum (MI) generally outperforms LGV alone (by up to 20.5%) and ranks the best or close to the best. Indeed, both techniques tackle properties of transferability not covered by LGV: DI captures some input invariances learned by different architectures, and MI smooths the attack optimization updates in a moving average way.

The incompatibility of GN and SGM with LGV leads us to believe that their input perturbations are *cheap and bad proxies for local weight geometry*. Eroding randomly skip connection, applying dropout on all layers, or backpropagating more

linearly, may (poorly) approximate sampling in the weight space vicinity. LGV does this sampling explicitly.

Overall, our observations lessen both the importance of skip connections to explain transferability claimed by Wu et al. [WWX+20], and what was believed to hurt most transferability, i.e., the optimization algorithm [DLP+18] and lack of input diversity [XZZ+19]. Our results demonstrate that the diversity of surrogate models (one model per iteration) is at most importance to avoid adversarial examples overfitting to their surrogate model. LGV does so more effectively than [LBZ+18].

5.4.3 Comparison With Deep Ensemble and SWAG

This section evaluates the complementarity between LGV and the surrogate based on deep ensemble proposed in Chapter 4, and compares LGV with the SWAG surrogate proposed in Chapter 4.

Complementarity of deep ensemble and LGV. We are interested in investigating the complementarity between the local exploration of the weight space of LGV with the multimodal exploration of deep ensemble. Deep ensemble trains multiple DNNs with independent random initializations and independent SGD training noise. To complement both approaches of Chapter 4 and Chapter 5, we use several independently trained DNNs as the base models for LGV, and collect one set of LGV models per base model. At each iteration of the attack, we average the input gradients across the LGV sets, using one LGV model per set. Since a fair comparison of transferability techniques requires the same number of gradients per iteration [ZZL+22], the deep ensemble baseline also averages the input gradients of all base models.

Figure 5.4 shows the success rates of an increasing number of base models, i.e., the size of deep ensemble, for both norms. We observe that applying LGV on top of deep ensemble is beneficial for all ensembles (from one to five DNNs here) and for both norms. We also note that LGV benefits from deep ensemble: collecting models from different vicinities and ensembling them improves transferability compared to LGV on a single vicinity.

Overall, the local exploration of the weight space by LGV is complementary to the multimodal exploration by deep ensemble. Both are straightforward to complement, since we can simply collect LGV models from several independently trained DNNs.

Comparison of SWAG and LGV. We evaluate competitively LGV with the SWAG surrogate, evaluated in Chapter 4, since the limitations of SWAG was the motivation to propose LGV. SWAG samples weights from a Gaussian distribution with an expected value and covariance matrix estimated from collected models [MGI+19b]. For fairness, we sample 40 models using SWAG and compare their transferability to the original 40 collected models, and the 40 models of

Table 5.6: Success rates of baselines, state-of-the-art and LGV under the L∞ attack. Simple underline is best without LGV combinations, double is best overall. Gray is LGV-based techniques worse than vanilla LGV. "RD" stands for random directions in the weight space. In %.

	Target							
Surrogate	RN50	RN152	RNX50	WRN50	DN201	VGG19	IncV1	IncV3
Baselines (1 DNN)								
1 DNN	45.3±2.4	29.6±0.9	28.8±0.2	31.5±1.6	17.5±0.6	16.6±0.9	10.4±0.5	5.3±1.0
MI	53.0±2.2	36.3±1.5	34.7±0.4	38.1±2.0	22.0±0.1	21.1±0.3	13.9±0.4	7.3±0.8
GN	63.9±2.4	43.8±2.4	43.3±1.3	47.4±0.9	24.8±0.3	24.1±1.0	14.6±0.3	6.8±1.2
GN+MI	68.4±2.3	49.3±2.5	47.9±1.2	52.1±1.7	28.4±0.8	28.0±0.7	17.5±0.5	8.7±0.5
DI	75.0±0.2	56.4±1.9	59.6±1.5	61.6±2.4	41.6±1.1	39.7±0.9	27.7±1.0	15.2±1.0
DI+MI	81.2±0.3	63.8±1.9	67.6±0.9	68.9±1.5	49.3±0.7	46.7±0.4	33.0±1.0	19.4±0.9
SGM	64.4±0.8	49.1±3.1	48.9±0.6	51.7±2.8	30.7±0.9	33.6±1.3	22.5±1.5	10.7±0.9
SGM+MI	66.0±0.6	51.3±3.5	50.9±0.9	54.3±2.3	32.5±1.3	35.8±0.7	24.1±1.0	12.1±1.2
SGM+DI	76.8±0.5	62.3±2.7	63.6±1.7	65.3±1.4	45.5±0.9	49.9±0.8	36.0±0.7	19.2±1.7
SGM+DI+MI	80.9±0.7	66.9±2.5	68.7±1.2	70.0±1.7	50.9±0.6	56.0±1.4	42.1±1.4	23.6±1.6
Our techniques								
RD	60.6±1.5	40.5±3.0	39.9±0.2	44.4±3.2	22.9±0.8	22.7±0.5	13.9±0.2	6.6±0.7
LGV-SWA	84.9±1.2	63.9±3.7	62.1±0.4	61.1±2.9	44.2±0.4	42.4±1.3	31.5±0.8	12.2±0.8
LGV-SWA+RD	90.2±0.5	71.7±3.4	69.9±1.2	69.1±3.3	49.9±1.0	47.4±2.0	34.9±0.3	13.5±0.9
LGV (ours)	95.4±0.1	85.5±2.3	83.7±1.2	82.1±2.4	69.3±1.0	67.8±1.2	58.1±0.8	25.3±1.9
LGV combined with other techniques								
MI	97.1±0.3	88.7±2.3	87.0±1.0	86.6±2.1	73.2±1.4	71.6±1.4	60.7±0.6	27.4±0.8
GN	94.2±0.2	83.0±2.2	80.8±0.7	79.5±2.4	66.9±0.7	66.6±0.7	56.2±0.5	24.4±1.4
GN+MI	96.4±0.1	87.2±2.0	85.3±0.8	84.4±2.3	70.4±1.0	71.2±0.8	59.2±0.5	26.5±0.4
DI	93.8±0.1	84.4±1.6	84.1±0.6	81.8±1.6	74.9±0.2	76.2±0.7	71.5±1.3	38.9±1.1
DI+MI	96.9±0.0	89.6±1.7	89.6±0.4	88.4±1.1	82.3±0.9	82.2±0.9	78.6±0.8	45.4±0.5
SGM	86.9±0.7	74.8±2.6	73.5±1.2	72.8±2.4	60.6±0.9	69.0±1.8	61.5±1.7	31.7±1.8
SGM+MI	89.1±0.5	77.1±2.8	76.7±1.1	75.6±2.1	62.7±1.1	72.3±1.0	64.7±2.2	34.2±1.7
SGM+DI	84.3±0.6	72.5±2.4	72.8±0.7	70.7±1.8	62.1±0.9	71.8±1.4	67.0±1.8	37.7±1.8
SGM+DI+MI	87.7±0.6	76.4±2.5	77.2±0.8	75.6±1.1	66.4±1.0	76.6±0.7	72.1±1.4	42.9±1.7

Table 5.7: Success rates of baselines, state-of-the-art and LGV under the L2 attack. Simple underline is best without LGV combinations, double is best overall. Gray is LGV-based techniques worse than vanilla LGV. "RD" stands for random directions in the weight space. In %.

	Target							
Surrogate	RN50	RN152	RNX50	WRN50	DN201	VGG19	IncV1	IncV3
Baselines (1 DNN)								
1 DNN	53.9±2.0	37.9±2.0	37.9±0.1	40.5±2.0	22.7±0.5	21.1±0.6	13.6±0.2	7.9±0.7
MI	48.7±1.2	33.5±1.3	33.7±0.7	36.5±1.8	19.9±0.3	19.3±0.8	12.0±0.5	6.6±0.5
GN	77.2±0.9	60.1±1.3	59.6±2.0	63.4±2.0	37.3±1.4	33.6±0.5	21.1±0.8	12.1±1.0
GN+MI	68.4±2.0	50.4±1.9	49.8±1.4	53.3±1.2	29.0±1.2	27.3±0.2	16.5±0.5	9.0±1.0
DI	82.2±0.6	68.0±1.6	71.8±0.6	72.5±1.9	53.8±0.4	49.8±1.1	37.5±0.9	25.4±1.3
DI+MI	79.2±0.4	63.7±1.1	66.8±0.6	68.2±1.4	47.3±0.9	45.8±1.0	32.0±0.7	20.6±0.9
SGM	63.3±0.5	49.7±3.1	50.1±0.7	51.6±1.7	30.2±0.7	33.3±1.3	20.6±0.9	11.1±0.7
SGM+MI	63.3±0.4	49.2±3.7	50.1±0.6	51.9±1.5	29.6±0.3	33.9±1.4	21.4±0.5	11.6±0.8
SGM+DI	79.5±0.7	67.5±2.3	69.0±0.9	69.6±1.1	50.1±0.2	54.6±1.3	40.5±0.8	25.8±1.0
SGM+DI+MI	78.5±0.8	66.4±2.4	68.5±1.7	69.1±1.2	49.1±1.4	54.5±0.9	39.7±0.8	25.6±0.3
Our techniques								
RD	74.4±0.6	55.6±3.1	55.9±0.7	59.7±3.3	34.5±0.3	31.5±1.4	19.6±0.8	11.2±1.0
LGV-SWA	85.8±0.7	68.0±3.4	67.0±0.4	65.1±1.8	48.4±0.7	47.0±1.6	34.8±0.5	15.8±1.1
LGV-SWA+RD	92.0±0.5	77.9±3.0	76.2±1.4	75.2±2.8	58.1±0.3	55.6±1.9	42.7±0.6	20.2±0.5
LGV (ours)	96.3±0.2	90.1±0.9	88.7±0.5	87.2±1.8	79.6±1.2	78.0±1.6	71.8±0.5	42.8±0.4
LGV combined with other techniques								
MI	96.0±0.1	88.3±1.7	85.8±0.7	84.3±2.6	72.6±0.8	71.8±1.9	62.7±0.7	31.1±0.3
GN	95.8±0.5	89.3±1.6	87.6±0.6	85.8±1.8	77.7±1.0	77.5±0.6	71.0±0.6	41.5±1.5
GN+MI	95.3±0.4	86.1±2.2	84.1±0.6	82.6±2.4	71.0±1.2	71.1±1.1	62.0±0.8	30.2±0.5
DI	95.3±0.3	89.5±0.9	89.5±0.5	87.3±0.9	83.9±0.9	83.7±0.2	82.2±0.9	59.0±0.8
DI+MI	95.2±0.4	88.6±0.7	88.0±0.7	85.7±1.5	81.2±0.7	81.6±0.7	79.3±1.6	50.8±0.7
SGM	85.8±0.5	74.1±2.5	73.4±0.7	71.6±2.1	59.5±0.8	68.5±1.4	62.2±2.1	34.4±1.9
SGM+MI	85.0±0.7	73.3±2.3	72.5±0.9	70.1±1.9	57.5±0.3	67.6±1.2	60.7±1.9	33.0±1.5
SGM+DI	85.0±1.1	75.2±1.4	75.5±0.7	72.5±1.7	65.2±0.8	74.2±1.6	71.6±1.6	46.0±1.7
SGM+DI+MI	84.4±0.6	74.0±1.4	74.9±0.8	71.7±1.2	63.8±0.7	73.3±1.4	70.3±1.4	44.6±1.4

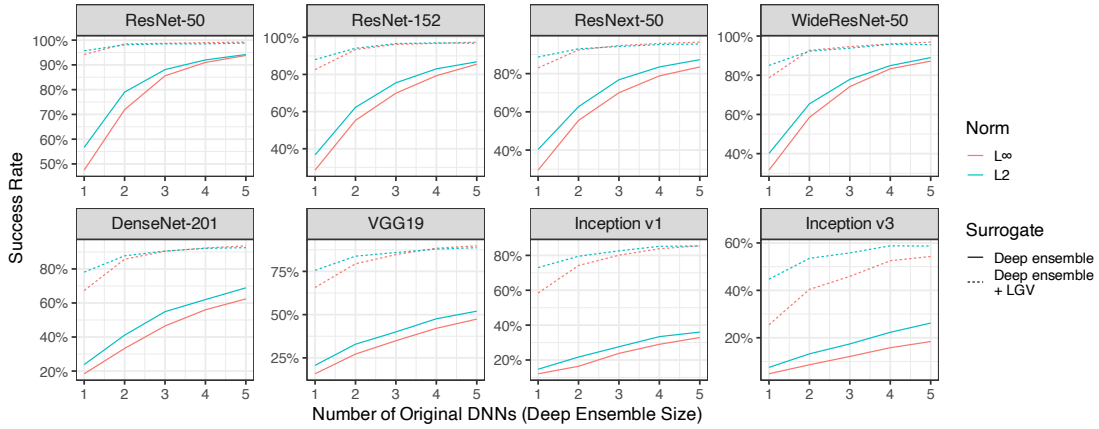


Figure 5.4: Transfer success rate with respect to the number of independently trained base models (deep ensemble size).

Table 5.8: Success rates of SWAG, the collected models used to sample SWAG models, and LGV, on ImageNet. Bold is best. In %.

Surrogate	RN50	RN152	RNX50	WRN50	VGG19	DN201	IncV1	IncV3
L∞ Attack								
Collected models	63.8	41.6	39.9	45.6	21.4	24.4	12.7	6.5
+ SWAG	63.0	42.1	38.1	46.7	20.7	24.8	13.9	6.6
+ high LR (LGV)	**94.7**	**82.5**	**82.5**	**79.5**	**65.0**	**67.6**	**58.3**	**26.2**
L2 Attack								
Collected models	71.8	53.5	51.3	57.8	27.9	35.3	20.1	11.1
+ SWAG	73.0	54.9	53.8	60.0	28.0	36.0	19.3	11.8
+ high LR (LGV)	**95.7**	**88.3**	**88.8**	**85.5**	**75.8**	**77.8**	**72.8**	**44.9**

LGV. Table 5.8 shows that SWAG does not consistently and significantly improve transferability, compared to its collected models. We would like to mention that SWAG may benefit from sampling a higher number of surrogate weights, but a fair comparison of training techniques should compute the same number of gradient per iteration [ZZL+22].

LGV boils down to the models collected for SWAG (without applying SWAG) using a high learning rate. Table 5.8 shows that the high learning rate of LGV is crucial to its success: thanks to its high learning rate, LGV significantly and consistently beats SWAG on both norm and eight targets. Overall, sampling surrogate weights using SWAG on top of collected models with SGD only provides minor improvements in transferability, whereas collecting models with a high learning rate provides important gains in transferability.

5.4.4 Hyperparameters Analysis

This section reports the success rates of the I-FGSM transfer attack for the LGV and I-FGSM attack hyperparameters: the LGV learning rate, the number of LGV epochs, the number of LGV weights collected per epoch, and the number of I-FGSM attack iterations. The most important points are summarised below.

- **Learning rate sensibility.** We show that LGV provides reliable transferability improvements for a wide range of learning rate.
- **Typologies of high learning rates.** We precisely describe the type of high learning rates suitable for LGV.
- **Computational efficiency.** If computational resources are limited, we can decrease the number of additional epochs of LGV to five instead of ten with limited degradation of transferability.
- **Memory efficiency.** If persistent memory is limited, we can decrease the number of epochs as previously, and the number of models collected per epoch to two instead of four. In the most severe case, a single model can be saved, since LGV-SWA improves over the initial DNN baseline.

Hyperparameters selection. We select all hyperparameters by cross-validation. A random subset of 2000 examples from the ImageNet train set is used as validation set to craft adversarial examples. The selected hyperparameter value is unique and does not depend on the target to respect the black-box threat model where the architecture is unknown. Each figure includes the eight studied targets (*subfigure title*), both L_∞ (*red*) and L_2 (*blue*) attacks, and the adversarial examples from the validation set (*dashed*) for hyperparameter selection and from the test set (*plain*) for independent evaluation.

Sensibility to the learning rate. We study the sensitivity of LGV on the constant learning rate value. LGV provides reliable transferability improvements for a wide range of learning rate, between 0.01 and 0.1 (Figure 5.5). The effectiveness of LGV degrades quickly as the learning rate goes larger than 1×10^{-1} or smaller than 5×10^{-3}. We suppose that small learning rates produce surrogates with gradients that overfit the initial model.

We select a learning rate equal to 0.05, half of the learning rate at the beginning of training, based on a validation set, both attack norms, and the eight target models. These observations are valid for the three other target architectures of the ResNet family (Figure 5.6). However, the best learning rate against the Inception v1 and v3 targets is 0.1 for both norms. This tolerance to a higher learning rate is consistent with our observation in Section 5.6 that transferability to these targets is less sensitive to the locally meaningful directions in the subspace spanned by LGV weights.

Typologies of high learning rates. We describe the type of high learning rates suitable for LGV[5]. We can identify several kinds of high learning: (a) the highest possible learning rate that does not make the model leave the current local minimum; (b) *the highest possible learning rate that makes the model jump between different local minima but does not cause deterministic chaos*; (c) the highest possible learning rate that causes deterministic chaos but does not lead to numerical divergence. "High" in our case refers to the definition (b). With a learning rate of 0.05, LGV exits the initial local minimum, as indicated by the spike of the training loss during the first LGV epochs from 0.95 of the initial DNN to 3.1. This creates a drop of 5.31 percentage points in natural test accuracy between the initial DNN and the ensemble of 40 LGV models (Figure 5.5). Our learning rate allows SGD to explore a larger vicinity in the weight space. This leaves (mostly) definition (a) out. Numerical divergence appears for a learning rate of 50, which is three orders of magnitude above the optimal one. Transferability drops suddenly when deterministic chaos appears (from 95% to 9% along with the natural test accuracy from 67% to 33% when changing the learning rate from 0.1 to 1). Deterministic chaos is more dangerous to LGV than exploring without leaving the local minimum. Very low LGV success rates might be an indication of convergence to local a maximum due to an excessively high learning rate. These observations exclude definition (c), leaving definition (b) coherent with our results.

Number of LGV epochs. Figure 5.7 reports the transferability of LGV with respect to the number of additional epochs. Ten additional epochs on the training set is enough to reach convergence. The computational cost of LGV is low, as it represents less than 7.7% of the training of the initial DNN. If the attacker has limited computational capability, five epochs are enough to obtain close results for most targets.

Number of LGV weights per epoch. Figure 5.8 reports the transferability of LGV with respect to the number of weights collected at each epoch. A threat model with intermediary limitation memory-wise could sample two LGV weights per epoch with minor success rate loss. In the case of an even more restricted threat model, the attacker can save a single model, LGV-SWA, to improve transferability over the baseline of the initial DNN (Section 5.4.5). In the rest of the paper, LGV saves four weights per epoch.

Number of attack iterations. To ensure the fairness of our analysis regarding the 1 DNN baseline, we report in Figure 5.9 the transferability with respect to the number of iterations of the I-FGSM attack. The number of I-FGSM iterations is set to 50 based on the validation success rate of both the initial DNN ("1 DNN") and the LGV surrogate. The attack on the initial DNN converges to its maximum

[5]We would like to thank the reviewers for raising this interesting discussion.

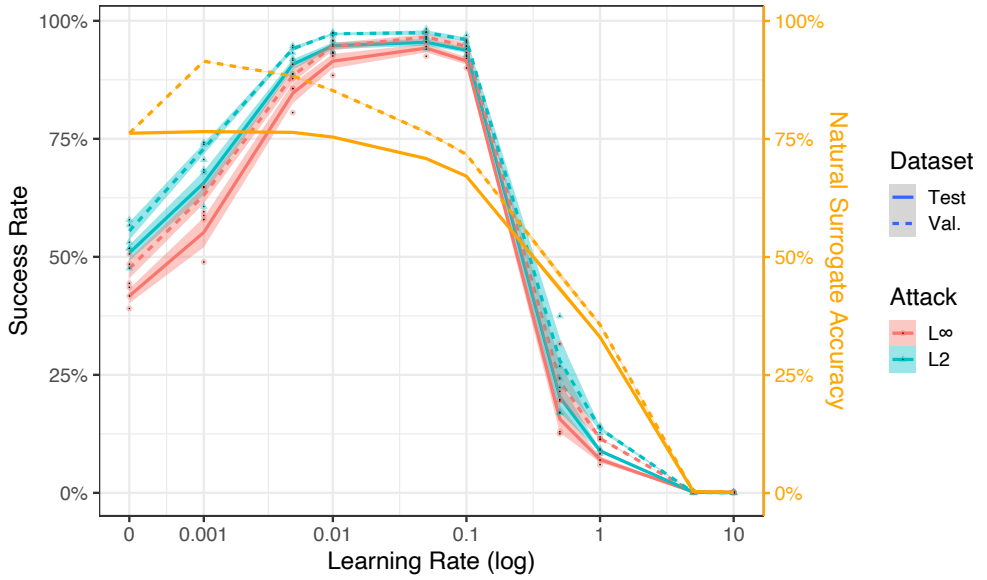


Figure 5.5: Transfer success rate against the ResNet-50 target (*red, blue*) and natural test accuracy (*orange*) of the LGV surrogate trained with a wide range of constant learning rate, in pseudo-log scale. The null learning rate refers to the initial DNN.

around 50 iterations for all targets. The same is true for the LGV surrogate against the ResNet family targets, but not against the Inception v1 and v3 architectures, where the success rate is already decreasing. For fairness, we choose 50 iterations in favour of the 1 DNN surrogate.

5.4.5 Connection Between LGV-SWA and LGV Ensemble

We demonstrate that the gradient of LGV-SWA approximates the gradient of the ensemble of LGV models. We show empirically in Section 5.4 that LGV-SWA is a good single model surrogate. We develop here its relation to the LGV weights. We extend the analysis from the original SWA paper [IPG+18] on the connection between the natural generalization of SWA and the one of local ensemble methods. Here, we suppose the loss function $\mathcal{L}(x;\, y, w)$ to be twice continuously differentiable both with respect to x in the L_p ball $B_\varepsilon[x]$, and to w at every w_k, for k in $[\![1, K]\!]$.

We perform a local analysis, since by construction, the weights collected by LGV w_k are close in the weight space and concentrated around their mean w_{SWA}. We consider the linear approximation of the input loss gradient function $\nabla_x \mathcal{L}(x;\, y, \cdot) : \mathbb{R}^p \longrightarrow \mathcal{X}$ around w_k,

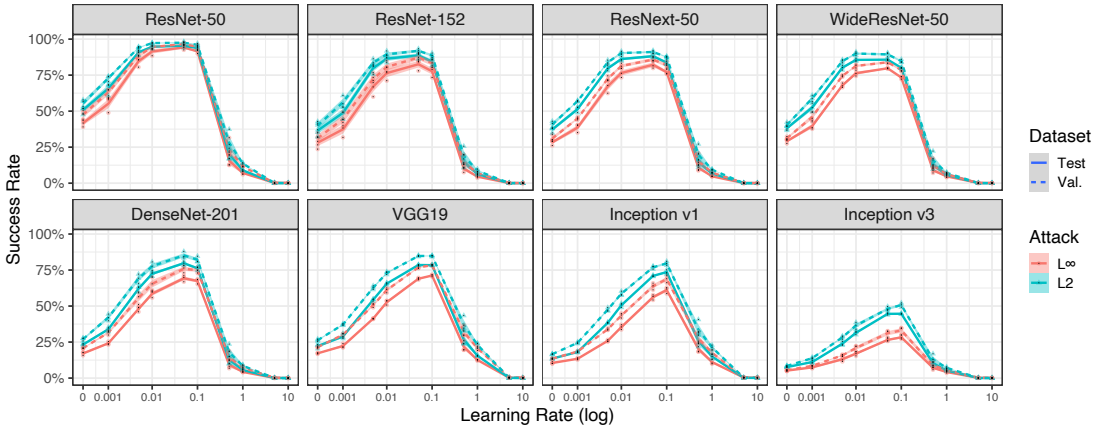


Figure 5.6: Transfer success rate with respect to the LGV learning rate, for the eight targets.

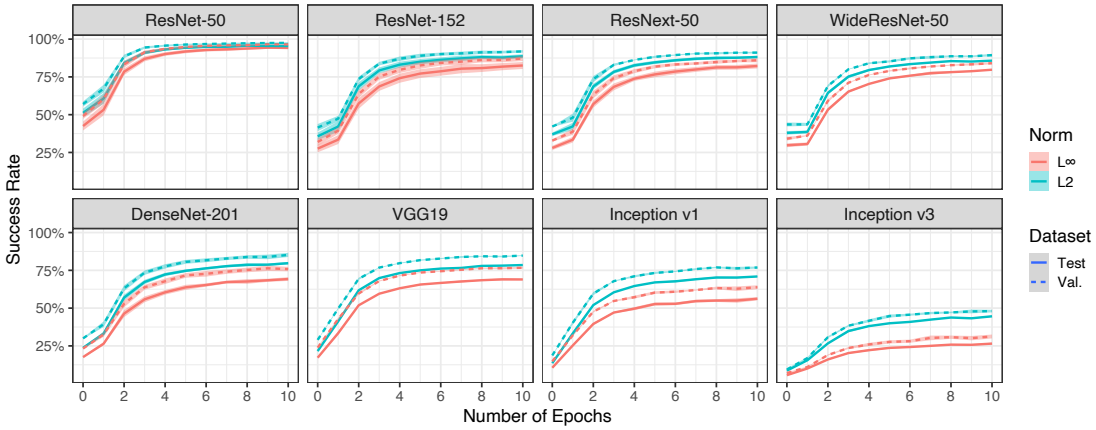


Figure 5.7: Transfer success rate with respect to the number of LGV epochs.

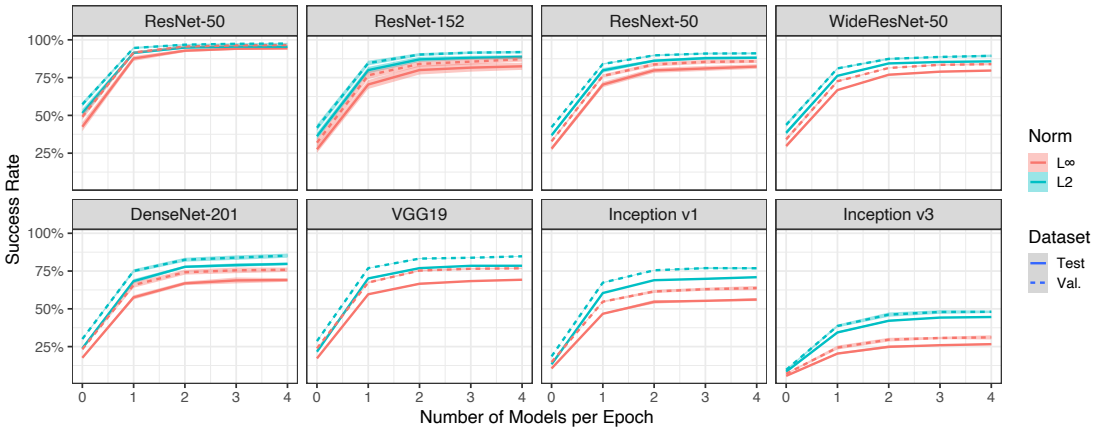


Figure 5.8: Transfer success rate with respect to the number of LGV weights saved per epoch.

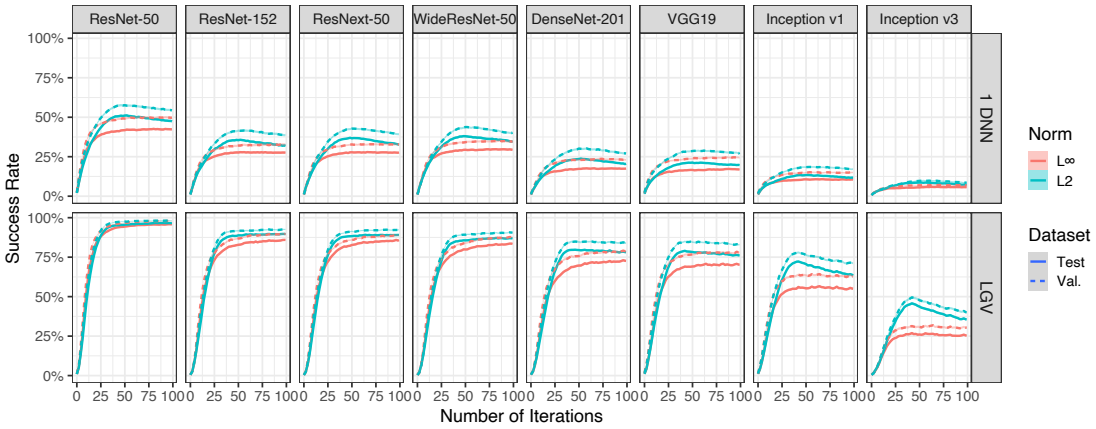


Figure 5.9: Transfer success rate with respect to the number of iterations of the I-FGSM attack.

$$\nabla_x \mathcal{L}(x;\, y, w_k) = \nabla_x \mathcal{L}(x;\, y, w_{\text{SWA}}) + \mathbf{J}_{\nabla_x \mathcal{L}(x;\, y, \cdot)}(w_{\text{SWA}})(w_k - w_{\text{SWA}}) + o(\|w_k - w_{\text{SWA}}\|)\,,$$

with $\mathbf{J}_{\nabla_x \mathcal{L}(x;\, y, \cdot)}(w)$ the Jacobian matrix of $\nabla_x \mathcal{L}(x;\, y, w)$ at w. The gradient of the ensemble of LGV models is the ensemble of individual gradients, $\overline{\nabla}_x \coloneqq \nabla_x \frac{1}{K} \sum_{k=1}^{K} \mathcal{L}(x;\, y, w_k) = \frac{1}{K} \sum_{k=1}^{K} \nabla_x \mathcal{L}(x;\, y, w_k)$. Then, the difference between the average of gradients and the gradient of the weights average is

$$\begin{aligned}
&\overline{\nabla}_x - \nabla_x \mathcal{L}(x;\, y, w_{\text{SWA}}) \\
&= \frac{1}{K} \sum_{k=1}^{K} \left[\mathbf{J}_{\nabla_x \mathcal{L}(x;\, y, \cdot)}(w_{\text{SWA}})(w_k - w_{\text{SWA}}) + o(\|w_k - w_{\text{SWA}}\|) \right] \\
&= \mathbf{J}_{\nabla_x \mathcal{L}(x;\, y, \cdot)}(w_{\text{SWA}}) \left(\frac{1}{K} \sum_{k=1}^{K} w_k - w_{\text{SWA}} \right) + o(\|\Delta_w\|) \\
&= o(\|\Delta_w\|),
\end{aligned}$$

with $\Delta_w = \max_{k=1}^{K}(\|w_k - w_{\text{SWA}}\|)$. It follows that LGV-SWA is a good single-model approximation of the ensemble of LGV models for gradient-based attacks. It captures some diversity of gradients in the vicinity of the weight space.

Overall, we show that LGV consistently increases transfer-based attacks success. However, it is not trivial why sampling surrogate weights in the vicinity of a local minimum helps adversarial examples to be successful against a model from another local minimum. In the following, we analyse the LGV success with a geometrical perspective.

5.5 Loss Flatness: the Surrogate-Target Misalignment Hypothesis

In this section and the following one, we relate the increased transferability of LGV to two geometrical properties of the weight space. First in this section, we show that LGV collects weights on flatter regions of the loss landscape than where it started (the initial, pretrained surrogate). These flatter surrogates produce wider adversarial examples in input space, and improve transferability in case of misalignment between the surrogate loss (optimized function) and the target loss (objective function). Second, we show in Section 5.6 that the span of LGV weights forms a dense subspace whose geometry is intrinsically connected to transferability, even when the subspace is shifted to other independent solutions. The geometry

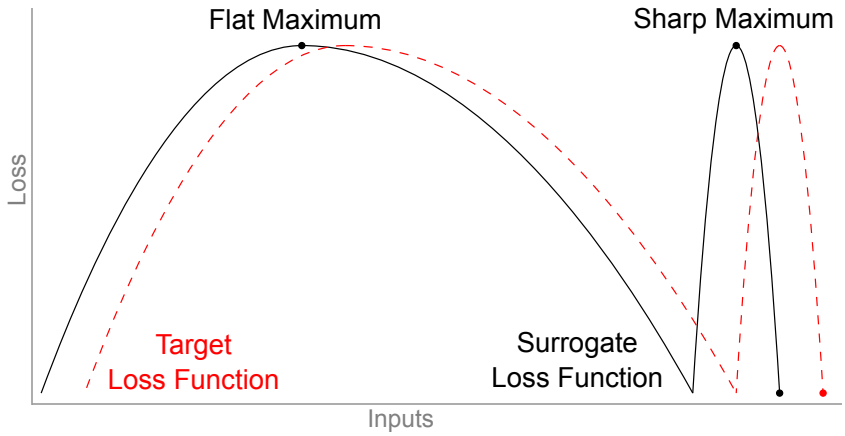


Figure 5.10: Conceptual sketch of flat and sharp adversarial examples. Adapted from [KNT+16].

plays a different role depending on the functional similarity between the target and the surrogate architectures.

We first explain why LGV is a good surrogate through the *surrogate-target misalignment hypothesis*. We show that LGV samples from flatter regions in the weight space and, as a result, produces adversarial examples flatter in the input space. This leads to surrogates that are more robust to misalignment between the surrogate and target prediction functions.

Sharp and flat minima have been discussed extensively in machine learning (see Chapter 3). A sharp minimum is one where the variations of the objective function in a neighbourhood are important, whereas a flat minimum shows low variations [HS97]. Multiple studies [IPG+18; KNT+16] correlate (natural) generalization with the width of the solution in the weight space: if the train loss is shifted w.r.t. the test loss in the weight space, wide optima are desirable to keep the difference between train and test losses small.

We conjecture that a similar misalignment occurs between the surrogate model and the target model *in the input space*. See Figure 5.10 for an illustration of the phenomenon. Under this hypothesis, adversarial examples at wider maxima of the surrogate loss would transfer better than sharp ones. The assumption that surrogate and target models are shifted with respect to each other seems particularly reasonable when both are the same function parametrised differently (intra-architecture transferability), or are functionally similar (same architecture family). We do not expect all types of loss flatness to increase transferability, since entirely vanished gradients would be the flatter loss surface possible and annihilate gradient-based attacks.

We provide two empirical evidences for this hypothesis. First, LGV flattens

Table 5.9: Sharpness metrics in the weight space, i.e., the largest eigenvalue and the rank of the Hessian, computed on three types of surrogate and 10,000 training examples.

	Hessian			
Model	Max EV		Trace	
1 DNN	558	ś57	16258	ś725
LGV indiv.	168	ś127	4295	ś517
LGV-SWA	30	ś1	1837	ś70

weights compared to the initial DNN. Second, LGV similarly flattens adversarial examples in the input space.

5.5.1 Flatness in the Weight Space

We establish that LGV weights and their mean (LGV-SWA) are in a flatter region of the loss than the initial DNN. The reason we consider LGV-SWA is that this model lies at the center of the loss surface explored by LGV and attacking this model yields a good first-order approximation of attacking the ensemble of LGV weights (cf. Section 5.4.5). First, we compute Hessian-based sharpness metrics. Second, we study the variations of the loss in the weight space along random directions from the solutions. Third, we confirm our observations by interpolating LGV-SWA and the initial DNN in the weight space. Finally, we report the same findings regarding the lengths of random and LGV deviations vectors that are optimal for transferability.

Hessian-based metrics. First, Table 5.9 reports two sharpness metrics in the weight space: the largest eigenvalue of the Hessian which estimates the sharpness of the sharpest direction, and the trace of the Hessian which estimates the sharpness of all directions. Both metrics conclude that the initial DNN is significantly sharper than the LGV and LGV-SWA weights.

Variations along random directions. Second, like [IPG+18], we study the variations of the loss in the weight space along random directions. We sample a random direction vector d on the unit sphere, $d = \frac{e}{\|e\|_2}$ with $e \sim \mathcal{N}(\mathbf{0}, I_p)$ and we study the following rays,

$$w_0(\alpha, d) = w_0 + \alpha d, \tag{5.5}$$
$$w_k(\alpha, d) = w_k + \alpha d, \tag{5.6}$$
$$w_{\text{SWA}}(\alpha, d) = w_{\text{SWA}} + \alpha d, \tag{5.7}$$

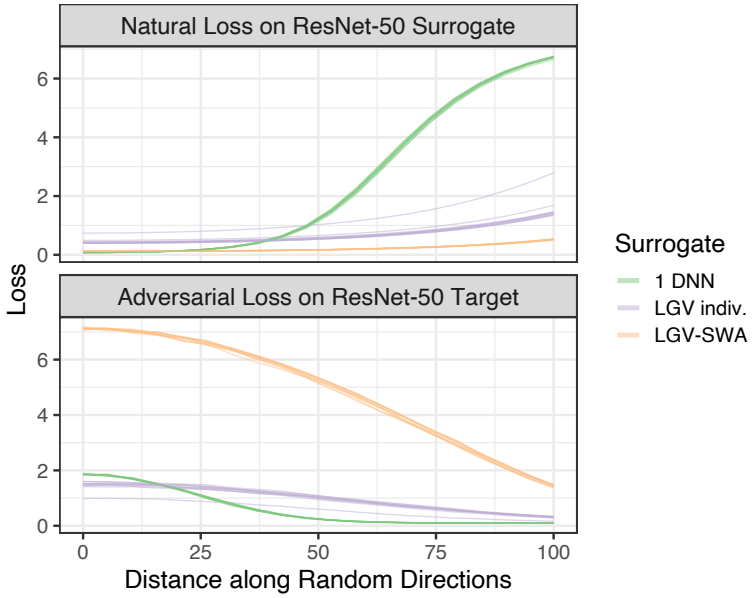


Figure 5.11: L_∞ attack crafted on surrogate with natural loss (*up*), evaluated on target (*down*) with respect to the 2-norm distance along 10 random directions in the weight space from the LGV-SWA solution (*orange*), random LGV weights (*purple*), and the initial DNN (*green*).

with $\alpha \in \mathbb{R}^+$. That is, we follow the same direction d for the three studied solutions. Figure 5.11 reports the intra-architecture results for 10 random directions (see Figure 5.12 for other settings). The natural loss in the weight space is wider at the individual LGV weights and at LGV-SWA than it is at the initial model weights (upper plot). When adding the random vector αd, the natural loss of LGV-SWA barely increases, while that of the initial model w_0 reaches high values: 0.40 vs. 6.67 for $\|\alpha \cdot d\|_2$ from 0 to 100. The individual LGV models are in between, with an 1.12 increase on average. Additionally, Figure 5.12 reports the loss of adversarial examples crafted along 10 random directions in the weight space, for both norms and evaluated on the eight targets. We confirm on the L_2 attack and on the inter-architecture case that the increased flatness of LGV-SWA in the weight space comes with an increased transferability of LGV-SWA adversarial examples, compared to the initial DNN.

Interpolation between LGV-SWA and the initial DNN. Third, we confirm the previous observations in about flatness in weight space along another specific direction. None of the 10 studied random directions increases transferability on

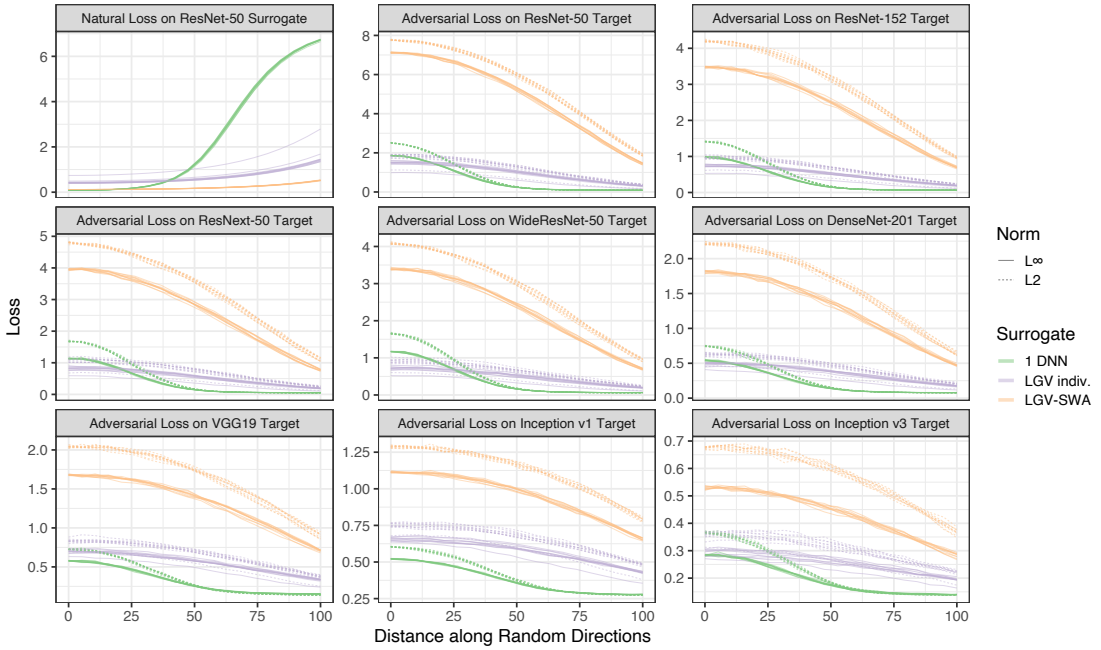


Figure 5.12: Surrogate natural loss (*first subplot*) and adversarial target loss (*other subplots*) with respect to the 2-norm distance along 10 random directions in the weight space from the initial model (*green*), LGV-SWA (*orange*) and randomly drawn individual LGV weights (*purple*). For adversarial target losses, plain lines are L_∞ and dashed ones are L_2. Ordinate scale not shared.

their own[6]. However, we know that at least one direction behaves differently, since transferability increases from the initial DNN to LGV-SWA (Section 5.4). As [IPG+18], we study the path in the weight space connecting both with $\alpha \in \mathbb{R}$:

$$w(\alpha) = \alpha w_0 + (1 - \alpha) w_{\text{SWA}}$$

We observe the same correlation between the flatness of the natural surrogate loss (orange) and the target adversarial loss (blue and red) in Figure 5.13. Around the LGV-SWA solution, the natural loss is flatter than around the initial DNN where it explodes at α close to 1.2. The same conclusions hold for all target architectures. Interestingly, LGV-SWA is not always the best single surrogate. The best surrogate in the studied segment is achieved for values of α between 0.154 and 0.538 for target architectures that belong to the ResNet family. The natural loss looks also flat in this region, so this does not contradict our observation. In

[6]This monotonic decrease in random directions does not contradict our findings in Section 5.3. Here, all the I-FGSM attack iterations are applied on a single surrogate, whereas previously each iteration was performed on a new *iid* sample.

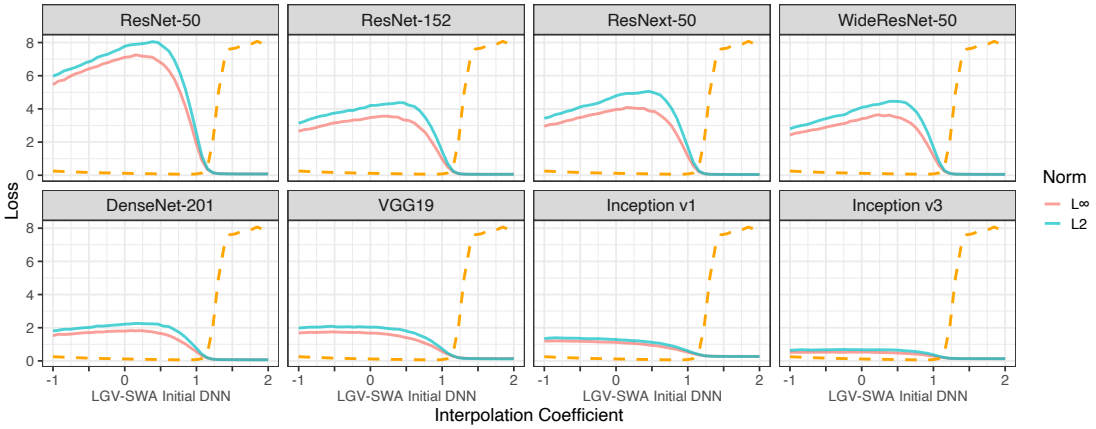


Figure 5.13: Adversarial target loss (*plain*) and surrogate natural loss (*orange dashed*) with respect to the interpolation coefficient α between the LGV-SWA solution and the initial model. The subplot title is the target architecture. The first subplot is intra-architecture transferability.

conclusion, LGV produces weights on a flatter region of the loss landscape than where it starts from.

Optimal lengths of random and LGV deviations vectors. Finally, Section 5.6.3 confirm our observations about flatness in a fourth experiment: the optimal lengths of random and LGV deviations vectors to generate a set of surrogate weights are larger for LGV-SWA than for the initial DNN. The length σ' (1×10^{-2}) to generate the surrogate "LGV-SWA + RD" described in Section 5.6.1 is the double of the length σ to generate the surrogate "RD" from the initial DNN in Section 5.3.2 (5×10^{-3}). Therefore, one can randomly sample further away from LGV-SWA in the weight space than from the initial DNN. The same conclusion holds for the γ scalar of the "1 DNN + γ (LGV' - LGV-SWA')" surrogate used to rescale the LGV deviations to apply them to the initial DNN (Section 5.6.3). This γ factor is set by cross-validation to 0.5. Therefore, one also needs to divide by two the length of the LGV deviations vectors to apply them to the initial model. Overall, LGV-SWA is flatter in the weight space than the initial model regarding the lengths of random and LGV deviation vectors that are optimal for transferability.

Overall, we show empirically that the LGV weights are flatter in the weight space than in the initial DNN using four techniques: the Hessian-based sharpness metrics, and the variations of the loss in random directions and along the path in weight space connecting LGV-SWA and the initial DNN, and the optimal length to generate multiple surrogate weights.

As Figures 5.11 to 5.13 also reveal, the increased flatness of LGV-SWA in the weight space comes with an increased transferability. We investigate this phenomenon deeper in what follows.

5.5.2 Flatness in the Input Space

Knowing that LGV yields loss flatness in the weight space, we now connect this observation to the width of basins of attractions in the input space when we craft adversarial examples. That is, we aim to show that flat surrogates in the weight space produce flatter adversarial examples in the input space.

To study flatness of adversarial examples in the input space, we consider the plane containing 3 points: the original example x, a LGV adversarial example $x^{\text{adv}}_{\text{LGV}}$, and an adversarial example crafted against the initial DNN $x^{\text{adv}}_{\text{DNN}}$. We build an orthonormal basis $(u', v') \coloneqq (\frac{u}{\|u\|}, \frac{v}{\|v\|})$ using the first two steps of the GramSchmidt process,

$$(u, v) = \left(x^{\text{adv}}_{\text{LGV}} - x,\ (x^{\text{adv}}_{\text{DNN}} - x) - \frac{\langle x^{\text{adv}}_{\text{DNN}} - x, u \rangle}{\langle u, u \rangle} u \right). \tag{5.8}$$

We focus our analysis on the 2-norm attack. It constrains adversarial perturbations inside the L_2-ball centered on x of radius ε. This has the convenient property that the intersection of this ball with our previously defined plane (containing x) is a disk of radius ε.

Figure 5.14 shows the loss of the ensemble of LGV weights and the loss of the initial DNN in the (u', v') coordinate system. We report the average losses over 500 disks, each one centered on a randomly picked test example. It appears that LGV has a much smoother loss surface than its initial model. LGV adversarial examples are in a wide region of the LGV ensemble's loss. The maxima of the initial DNN loss is highly sharp and much more attractive for gradient ascent than the ones found by LGV – the reason why adversarial examples crafted from the initial DNN overfit.

Additionally, we report in Figures 5.15 to 5.21 the visualizations of the plane in input space for all eight targets and with several combinations of surrogates.

As shown in Section 5.5.1, the initial DNN is sharper than individual LGV models in the weight space, and LGV-SWA is flatter than both. We show here that the order of flatness is the same in the weight space and in input space. Figure 5.16 shows that LGV-SWA produces flatter adversarial examples. A randomly sampled individual LGV weights surrogate leads to flatter adversarial examples than the initial DNN (Figure 5.18), but sharper than the LGV-SWA (Figure 5.19) and the LGV (Figure 5.20) surrogates.

Section 5.4.5 shows that LGV-SWA is a good approximation to the ensemble of LGV weights. LGV transfers better than LGV-SWA (Tables 5.6 and 5.7). We

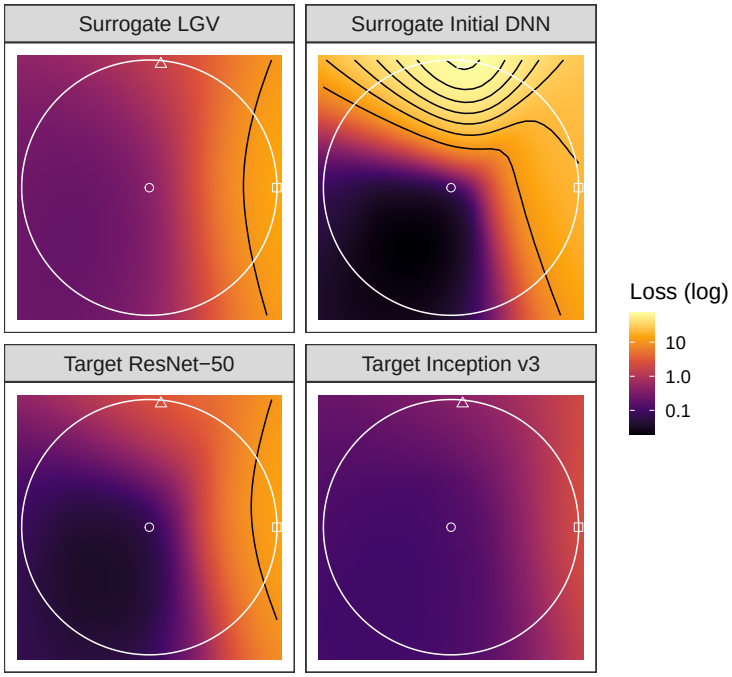


Figure 5.14: Surrogate (*upper*) and target (*bottom*) losses in the plane containing the original example (*circle*), an adversarial example against LGV (*square*) and one against the initial DNN (*triangle*), in the (u', v') coordinate system. Colours are in log-scale, contours in natural scale. The white circle represents the intersection of the 2-norm ball with the plane.

observe in Figure 5.17 that the adversarial examples of the former are flatter on average than the ones of the latter. The optimization of the I-FGSM attack overfits the single set of weights approximation, leading to sharper minima. We also observe that the form of the contours around the LGV-SWA surrogate can be explained by a shift between target and surrogate.

Section 5.3 exhibits that noise applied to the weights of a DNN increases transferability. Figure 5.21 establishes that this noise also slightly flattens the adversarial examples in input space.

5.5.3 Flatness and Transferability

Figure 5.14 also shows the losses of two target models in the (u', v') coordinate system. The LGV loss appears particularly well aligned with the one of the ResNet-50 target (intra-architecture transferability). We observe a *shift between the contour of both models, with the same functional form*. These observations are valid for other targets and on planes defined by adversarial examples of other surrogates (see

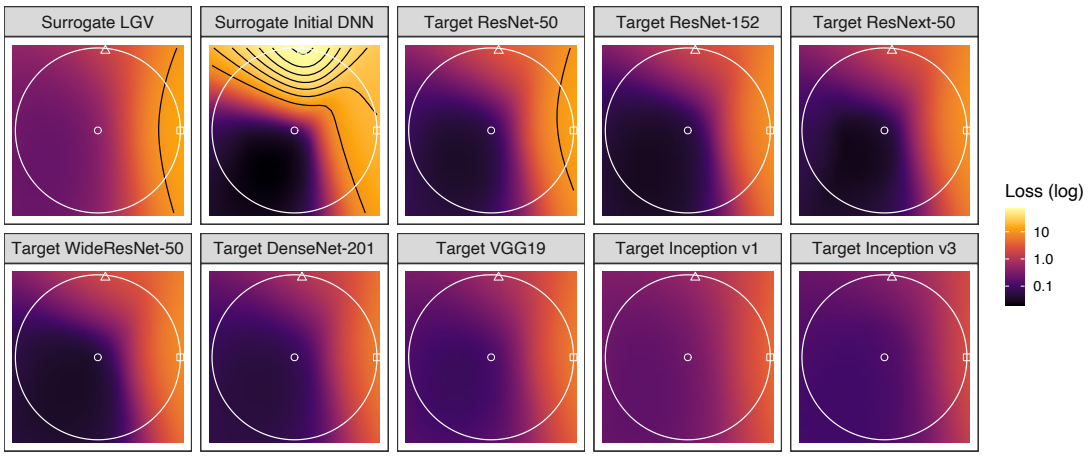


Figure 5.15: **LGV** surrogate (*first up*), the **initial DNN** surrogate (*second up*) and targets (*others*) losses in the plane containing the original example (*circle*), an adversarial example against LGV (*square*) and one against the initial DNN (*triangle*), in the (u', v') coordinate system. Colours are in log-scale, contours in natural scale. The white circle represents the intersection of the 2-norm ball with the plane.

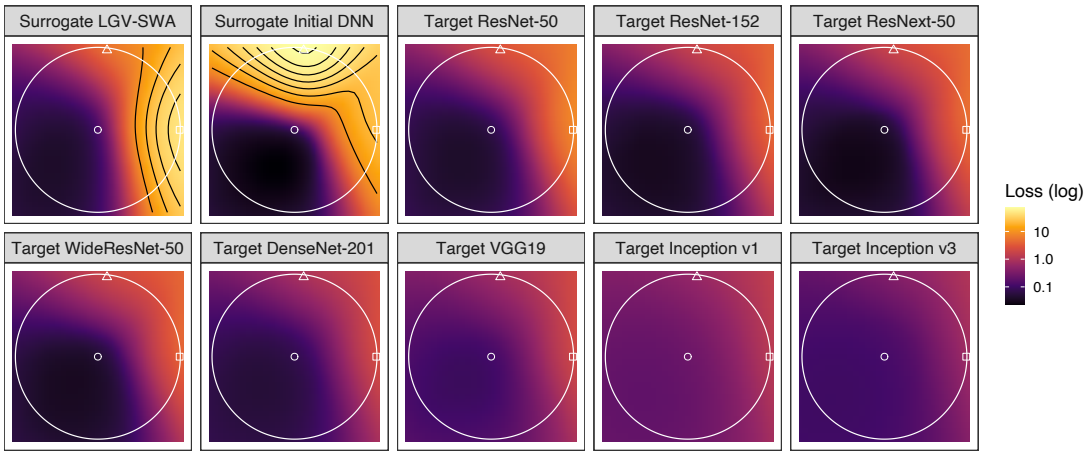


Figure 5.16: **LGV-SWA** surrogate (*first up*), the **initial DNN** surrogate (*second up*) and targets (*others*) losses in the plane containing the original example (*circle*), an adversarial example against LGV-SWA (*square*) and one against the initial DNN (*triangle*), in the (u', v') coordinate system. Colours are in log-scale, contours in natural scale. The white circle represents the intersection of the 2-norm ball with the plane.

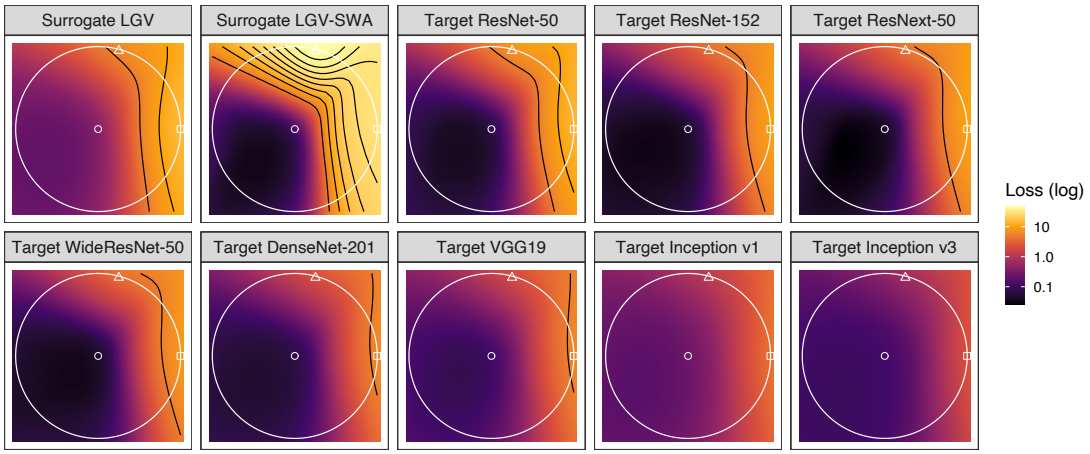


Figure 5.17: **LGV** surrogate (*first up*), the **LGV-SWA** surrogate (*second up*) and targets (*others*) losses in the plane containing the original example (*circle*), an adversarial example against LGV (*square*) and one against LGV-SWA (*triangle*), in the (u', v') coordinate system. Colors are in log-scale, contours in natural scale. The white circle represents the intersection of the 2-norm ball with the plane.

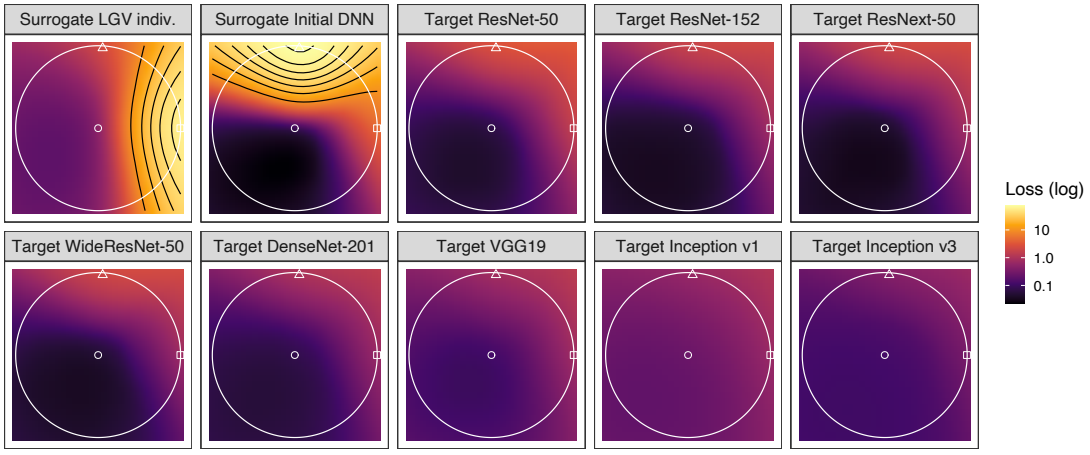


Figure 5.18: A randomly sampled **individual LGV weights** surrogate (*first up*), the **initial DNN** surrogate (*second up*) and targets (*others*) losses in the plane containing the original example (*circle*), an adversarial example against the individual LGV weights (*square*) and one against the initial DNN (*triangle*), in the (u', v') coordinate system. Colors are in log-scale, contours in natural scale. The white circle represents the intersection of the 2-norm ball with the plane.

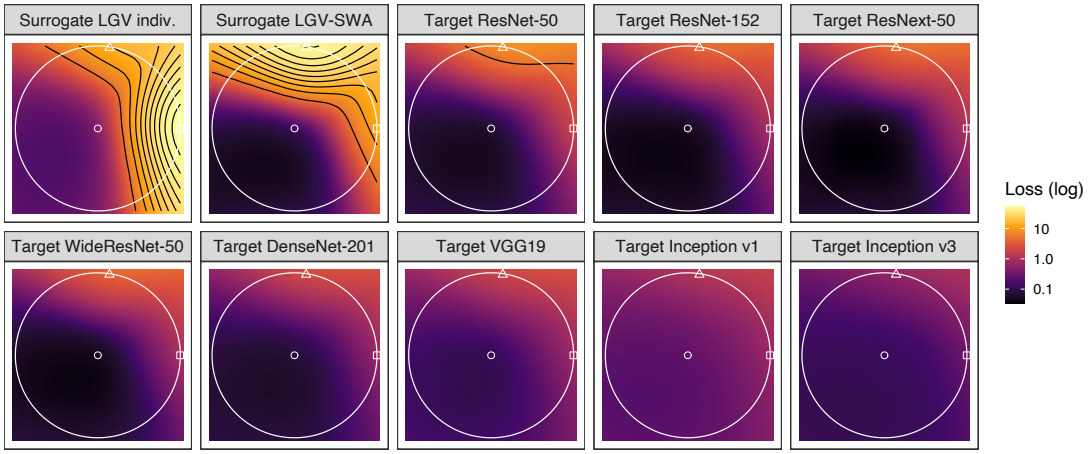


Figure 5.19: A randomly sampled **individual LGV weights** surrogate (*first up*), **LGV-SWA** surrogate (*second up*) and targets (*others*) losses in the plane containing the original example (*circle*), an adversarial example against the individual LGV weights (*square*) and one against LGV-SWA (*triangle*), in the (u', v') coordinate system. Colours are in log-scale, contours in natural scale. The white circle represents the intersection of the 2-norm ball with the plane.

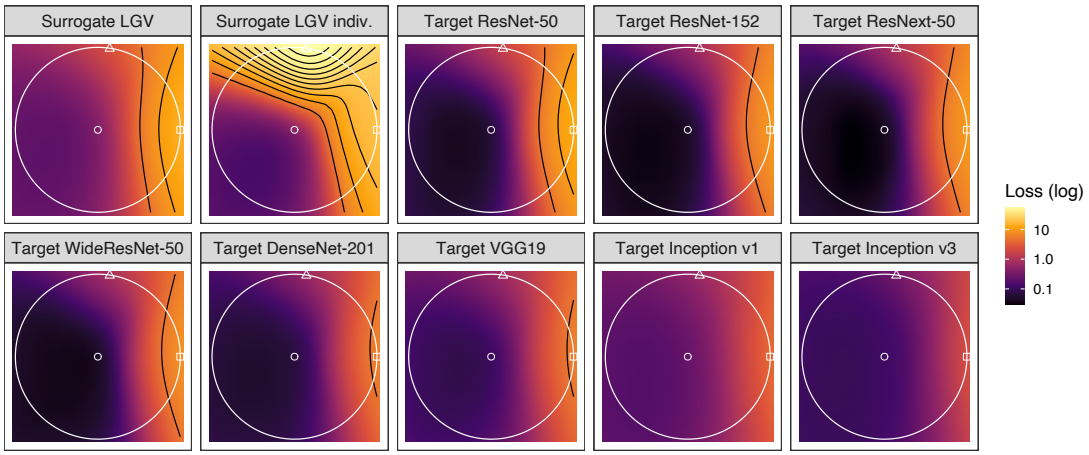


Figure 5.20: **LGV** surrogate (*first up*), a randomly sampled **individual LGV weights** surrogate (*second up*) and targets (*others*) losses in the plane containing the original example (*circle*), an adversarial example against LGV (*square*) and one against the individual LGV weights (*triangle*), in the (u', v') coordinate system. Colours are in log-scale, contours in natural scale. The white circle represents the intersection of the 2-norm ball with the plane.

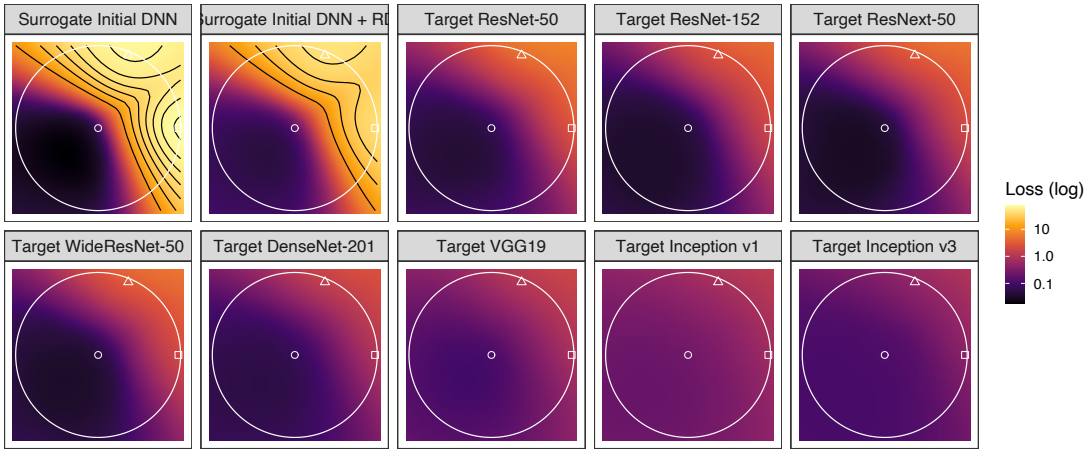


Figure 5.21: The **initial DNN** surrogate (*first up*), the **initial DNN + random directions** surrogate (*second up*) and targets (*others*) losses in the plane containing the original example (*circle*), an adversarial example against the initial DNN (*square*) and one against the initial DNN + random directions (*triangle*), in the (u', v') coordinate system. Colours are in log-scale, contours in natural scale. The white circle represents the intersection of the 2-norm ball with the plane.

Section 5.5.2). All these observations corroborate our surrogate-target misalignment hypothesis.

In Section 5.5.1, the experiment interpolating between LGV-SWA and the initial DNN corroborates our findings. By interpolating the weights between LGV-SWA and the initial model, i.e. moving along a non-random direction, we confirm that (i) the surrogate loss is flatter at LGV-SWA than at the initial model weights, (ii) that the adversarial loss of target models gets higher as we move from the initial model to LGV-SWA.

Section 5.5 – Conclusion. LGV weights lie in flatter regions of the loss landscape than the initial DNN weights. Flatness in the weight space correlates with flatness in the input space: LGV adversarial examples are wider maxima than sharp adversarial examples crafted against the initial DNN. These conclusions support our surrogate-target misalignment hypothesis: if surrogate and target losses are shifted with respect to each other, a wide optimum is more robust to this shift than a sharp optimum.

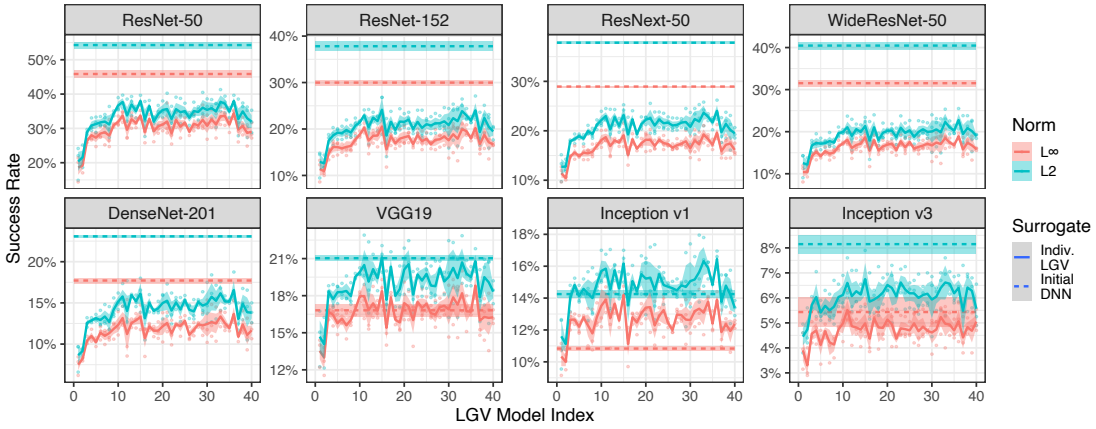


Figure 5.22: Transfer success rate of each individual LGV weights indexed by the sampling order (*plain*) and the initial DNN baseline (*dashed*). The subplot title is the target architecture. The first subplot is intra-architecture transferability. Ordinate scale not shared.

5.6 Importance of the LGV Weight Subspace Geometry

Although we have demonstrated the link between the better transferability that LGV-SWA (and in extenso, the LGV ensemble) achieves and the flatness of this surrogate's loss, additional experiments have revealed that the LGV models – taken individually – achieve lower transferability, although they also have a flatter loss than the initial model. This indicates that other factors are in play to explain LGV transferability.

Individual LGV weights do not succeed on their own. The success of LGV cannot be explained by the intrinsic properties of each of its models taken on their own. Figure 5.22 shows that no single model sampled by LGV improves consistently upon the baseline of the initial model it originates from. On the contrary, the initial DNN is generally a better surrogate. Therefore, flatness alone is not enough to explain the transferability of LGV.

In what follows, we show the importance of the geometry of the subspace formed by LGV models in increasing transferability. More precisely, deviations of LGV weights from their average spans a weight subspace which is (i) densely related to transferability (i.e., *it is useful*), (ii) composed of directions whose relative importance depends on the functional similarity between surrogate and target (i.e., *its geometry is relevant*), (iii) remains useful when shifted to other solutions (i.e., *its*

geometry captures generic properties). Similarly to [IMK+19], the K-dimensional subspace of interest is defined as,

$$\mathcal{S} = \{w \,|\, w = w_{\text{SWA}} + \mathbf{P}z\}\,, \tag{5.9}$$

where w_{SWA} is called the shift vector, $\mathbf{P} = (w_1 - w_{\text{SWA}}, \ldots, w_K - w_{\text{SWA}})^\intercal$ is the projection matrix of LGV weights deviations from their mean, and $z \in \mathbb{R}^K$.

5.6.1 A Subspace Useful for Transferability

In this section, we show that the subspace $\mathcal{S}$ of LGV weights has importance for transferability. First, we show that the subspace $\mathcal{S}$ is *specific* to transferability in the sense that $\mathcal{S}$ improves transferability significantly more than a random subspace. Second, we provide evidence that the subspace $\mathcal{S}$ *densely* relate to transferability, in the sense that randomly sampling $\mathcal{S}$ generates good surrogate models.

Our findings relate to the work of Gur-Ari et al. [GRD18], who analyse the weight subspace spanned by SGD updates during training. They show that despite the high dimensionality of the weight space, SGD updates are concentrated in a tiny subspace. Our hypothesis is that LGV generates a subspace where SGD is likely to explore. With the probabilistic perspective developed in Chapter 4, the probability distribution of the unknown target trained with SGD spreads more mass in this subspace than in a random subspace.

LGV subspace vs. random subspace. This paragraph shows that the subspace $\mathcal{S}$ of LGV weights is significantly better for transferability than a random subspace, i.e., the subspace $\mathcal{S}$ is *specific* to transferability. Similarly to our previous RD surrogate, we build a new surrogate "LGV-SWA + RD" by sampling random directions in the full weight space around LGV-SWA. It is defined as:

$$\left\{w_{\text{SWA}} + e'_k \mid e'_k \sim \mathcal{N}(\mathbf{0},\, \sigma' I_p),\, k \in [\![1, K]\!]\right\}, \tag{5.10}$$

where the standard deviation σ' is selected by cross-validation. Figure 5.23 reports the success rate for various values of σ'. Similarly to "RD" in Section 5.3, we tune this hyperparameter on 10 random directions, and generate the final "LGV-SWA + RD" surrogate on 50 directions.

In accordance with our findings about the respective flatness of the DNNs and LGV-SWA, the optimal standard deviation for "LGV-SWA + RD" (1×10^{-2}) is larger than the one for "RD" (5×10^{-3}). A flatter solution implies that we can sample further along random directions before exiting the vicinity of low loss.

Table 5.6 reports the transferability of this surrogate for the L_∞ attack (see Table 5.7 for L_2). We observe that random deviations drawn in the entire weight space

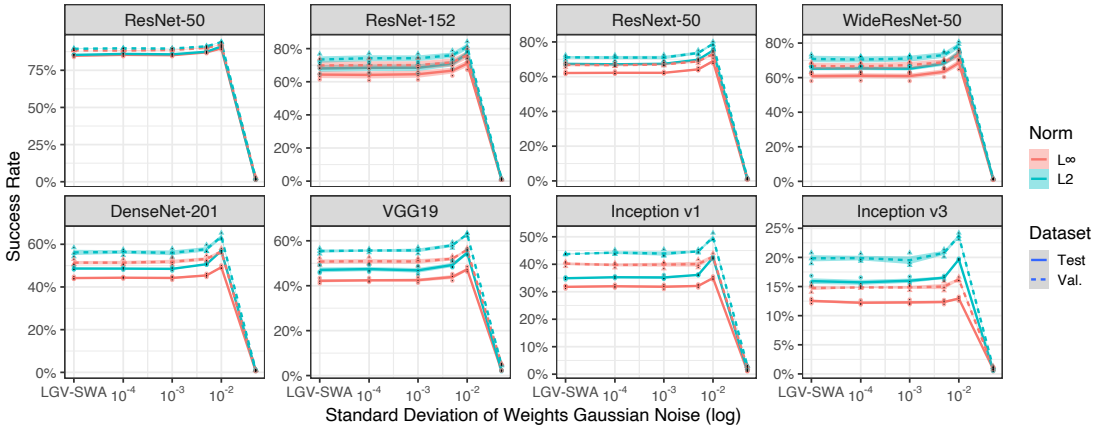


Figure 5.23: Transfer success rate of I-FGSM with respect to the standard deviation σ' of the Gaussian white noise added to the weight of LGV-SWA. Ten random directions are sampled in the weight space. The subplot title is the target architecture. The first subplot is intra-architecture transferability.

do improve the transferability of LGV-SWA (increase of 1.32 to 10.18 percentage points, with an average of 6.90). However, the LGV surrogate systematically outperforms "LGV-SWA + RD". The differences range from 4.33 to 29.15 percentage points, and average to 16.10. Therefore, the subspace $\mathcal{S}$ has specific geometric properties related to transferability that make this ensemble outperforms the ensemble formed by random directions around LGV-SWA.

Random directions in the LGV subspace. We show that the LGV deviation subspace is *densely* related to transferability, in the sense that it is a dense subspace of good surrogates. We form a new surrogate called "LGV-SWA + RD in $\mathcal{S}$" by sampling random directions in the LGV deviations subspace $\mathcal{S}$,

$$\{w_{\text{SWA}} + \mathbf{P}z_k \mid z_k \sim \mathcal{N}(\mathbf{0},\, I_K),\ k \in [\![1, K]\!]\} \subset \mathcal{S}, \tag{5.11}$$

where $\mathbf{P} = (w_1 - w_{\text{SWA}}, \ldots, w_K - w_{\text{SWA}})^\intercal$ is the projection matrix of LGV weights deviations from their mean. Table 5.10 reports the success rates of this surrogate along with other techniques. We observe that the transferability of random directions in the subspace is close to the original LGV surrogate (average difference of 1.45 percentage point, with values between -0.6 and 5.65), especially for ResNet-like targets. The negative difference corresponds to the intra-architecture transferability, where "LGV-SWA + RD in $\mathcal{S}$" outperforms LGV (significantly for L∞ and non-significantly for L2). Sampling random directions in the full weigh space ("LGV-SWA + RD") instead of the subspace hinders transferability of 14.7

Table 5.10: Transfer success rate of random directions sampled in LGV deviations subspace.

		Target							
Norm	Surrogate	RN50	RN152	RNX50	WRN50	DN201	VGG19	IncV1	IncV3
L∞	LGV	95.5±0.1	85.5±2.1	83.6±1.1	82.2±2.4	69.6±1.0	67.8±0.9	58.4±0.6	25.6±1.7
L∞	LGV-SWA + RD in $\mathcal{S}$	96.0±0.2	85.6±2.5	83.6±0.6	82.1±2.8	68.6±1.1	65.7±1.5	54.5±0.9	23.5±0.4
L∞	LGV-SWA + RD	90.4±0.3	71.9±3.4	70.0±1.2	69.2±3.4	50.0±1.0	47.4±1.9	34.9±0.4	13.4±0.7
L2	LGV	96.3±0.1	90.1±1.0	88.8±0.4	87.5±1.6	79.8±1.1	78.1±1.6	71.9±0.6	43.1±0.6
L2	LGV-SWA + RD in $\mathcal{S}$	96.6±0.3	90.1±1.4	88.7±0.5	87.3±2.0	77.6±1.0	75.6±1.5	67.4±1.9	37.4±0.4
L2	LGV-SWA + RD	91.9±0.6	78.2±2.9	76.2±1.3	75.4±2.5	58.1±0.3	55.8±1.6	42.7±0.6	20.0±0.6

percentage points on average (4.7—24.73). Therefore, the subspace $\mathcal{S}$ is densely and intrinsically related to transferability.

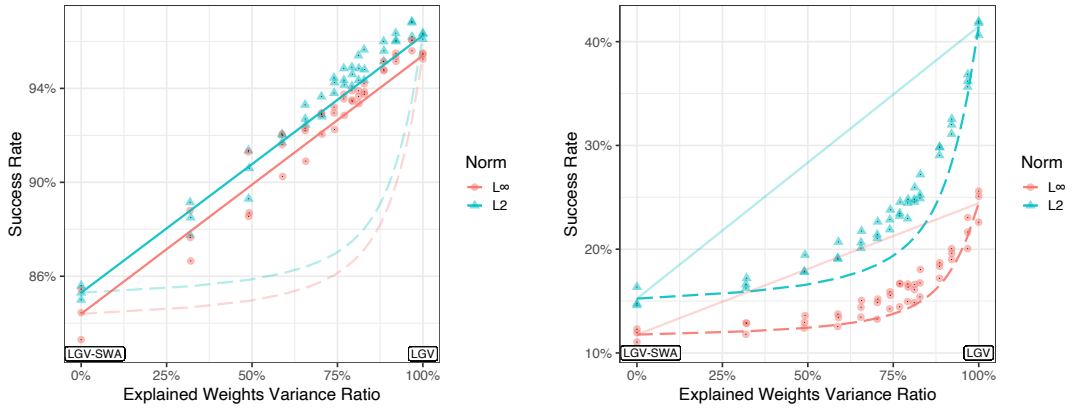
5.6.2 Decomposition of the LGV Projection Matrix

In this section, we analyse the contribution of subspace basis vectors to transferability through a decomposition of their projection matrix. Doing so, we build alternative LGV surrogates with an increasingly reduced dimensionality, and we assess the impact of this reduction on transferability.

We decompose the matrix of LGV weights deviations $\mathbf{P}$ into orthogonal directions, using principal component analysis (PCA) since the PCA coordinate transformation diagonalises this matrix. Following [IMK+19], we apply PCA based on exact full SVD[7] to obtain a new orthonormal basis of the LGV weight subspace. We exploit the orthogonality of the components to change the basis of each w_k with the PCA linear transformation and project onto the first C principal components. We then apply the inverse map, with w_{SWA} as shift vector, to obtain a new weight vector $w_{k,C}^{\text{proj}}$. We repeat the process with different value of C, which enables us to control the amount of explained weights variance and to build LGV ensembles with a reduced dimensionality.

The eigenvalues of the LGV weights deviation matrix equal the variance of the weights along the corresponding eigenvectors. We use the ratio of explained weights variance to measure the relative loss of information that would result from removing a given direction. From an information theory perspective, if a direction in the weight space is informative of transferability, we expect the success rate to

[7]As [IMK+19] we use the PCA implementation of sklearn[PVG+11], but here we select the full SVD solver instead of randomized SVD to keep all the singular vectors.



(a) ResNet-50 Target

(b) Inception v3 Target

Figure 5.24: Success rate of the LGV surrogate projected on an increasing number of dimensions with the corresponding ratio of explained variance in the weight space. Hypothetical average cases of proportionality to variance (*solid*) and equal contributions of all subspace dimensions (*dashed*). Scales not shared.

decrease with the loss of information due to dimensionality reduction. Note that the surrogate projected on the PCA zero space (i.e. $C = 0$) is LGV-SWA, whereas $C = K$ means we consider the full surrogate ensemble.

Figure 5.24 shows, for each dimensionality reduced LGV surrogates, the explained variance ratio of its lower dimensional weight subspace and the success rate that this ensemble achieves on the ResNet-50 and Inception v3 targets. To observe the trends, we add the hypothetical cases of proportionality to the variance (solid line) and equal contributions of all dimensions (dashed line).

For both targets in Figure 5.24, explained variance correlates positively with transferability. This means that our approach improves transferability more, as it samples along directions (from SWA) with higher variance. Especially in the intra-architecture case (Figure 5.24a), there is an almost-linear correlation between the importance of a direction in the weight space and its contribution to transferability. This conclusion can be loosely extended to the targets that belong to the same architecture family as the surrogate, i.e. ResNet-like models (Figure 5.25).

In some inter-architecture cases, we do not observe this linear trend, although the correlation remains positive. In Figure 5.24b, we see that the real variance ratio/transfer rate curve is close to the hypothetical case where each direction would equally improve transferability on the Inception v3 target. This means that, in this inter-architecture case, each direction contributes almost-equally to transferability regardless of their contribution to the subspace variance. In supplementary materials, we show other inter-architecture cases (e.g., DenseNet-

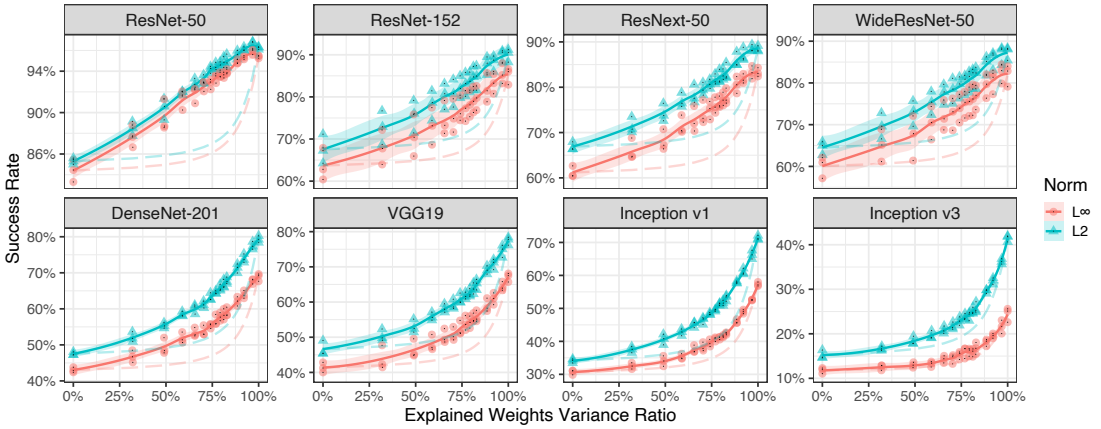


Figure 5.25: Transfer success rate of the LGV surrogate projected on an increasing number of dimensions with the corresponding ratio of explained variance in the weight space. The plain line is the smooth mean, and the area corresponds to one standard deviation. The dashed line is the hypothetical average case of equal contributions of all subspace dimensions. Ordinate scale not shared.

201 and VGG19) that are intermediate between linear correlation and almost-equal dimensional contributions (Figure 5.25).

In addition to Figure 5.24, we report in Figure 5.25 the success rate of surrogates projected into an increasing number of eigenvectors of the LGV weights deviations matrix $\mathbf{P}$, evaluated on all eight targets. The plain lines are smoothed means (local polynomial regression). We observe that the relationship between the weight space variance ratio and the transferability gets progressively less linear as the target architecture is different from the surrogate one (ResNet-50 here). Inception family targets are pretty close to the case of equal contribution of each dimension to transferability (dashed), independently of its variance. From an information theory perspective view of PCA, this would mean that the information contained in the weight space is directly relevant for intra-architecture transferability, and is not discriminant for dissimilar targets[8]. DenseNet-201 and VGG19 are intermediary cases. We show that the increase in transferability from the subspace $\mathcal{S}$ depends fundamentally on the functional similarity between the target and the surrogate architectures.

Taking together the above results, we explain the better transferability of LGV with the variance of the subspace it forms. However, this correlation is stronger as the surrogate and target architectures are more functionally similar.

[8]However, we recall that these directions remain more relevant than random directions in the full weight space, even for these target architectures.

Table 5.11: Transfer success rate of LGV deviations shifted to other independent solutions, for target architectures in the ResNet family.

		Target			
Norm	Surrogate	RN50	RN152	RNX50	WRN50
L∞	LGV-SWA + (LGV' - LGV-SWA')	94.3$_{\pm 0.5}$	81.5$_{\pm 2.3}$	79.1$_{\pm 1.4}$	78.1$_{\pm 2.4}$
L∞	LGV-SWA + RD	90.4$_{\pm 0.3}$	71.9$_{\pm 3.4}$	70.0$_{\pm 1.2}$	69.2$_{\pm 3.4}$
L∞	LGV (ours)	95.4$_{\pm 0.1}$	85.3$_{\pm 2.1}$	83.7$_{\pm 1.1}$	82.1$_{\pm 2.5}$
L∞	1 DNN + γ (LGV' - LGV-SWA')	73.3$_{\pm 2.0}$	52.8$_{\pm 2.9}$	52.6$_{\pm 1.6}$	56.6$_{\pm 2.8}$
L∞	1 DNN + RD	60.8$_{\pm 1.6}$	40.8$_{\pm 2.7}$	40.2$_{\pm 0.3}$	44.8$_{\pm 2.7}$
L2	LGV-SWA + (LGV' - LGV-SWA')	95.2$_{\pm 0.5}$	86.1$_{\pm 1.9}$	84.2$_{\pm 1.0}$	82.7$_{\pm 1.6}$
L2	LGV-SWA + RD	92.0$_{\pm 0.5}$	77.9$_{\pm 3.0}$	76.2$_{\pm 1.4}$	75.2$_{\pm 2.8}$
L2	LGV (ours)	96.3$_{\pm 0.1}$	90.2$_{\pm 1.1}$	88.6$_{\pm 0.6}$	87.6$_{\pm 1.7}$
L2	1 DNN + γ (LGV' - LGV-SWA')	84.2$_{\pm 0.8}$	68.7$_{\pm 2.6}$	70.0$_{\pm 1.3}$	72.4$_{\pm 1.5}$
L2	1 DNN + RD	74.6$_{\pm 0.5}$	55.8$_{\pm 3.1}$	56.1$_{\pm 0.6}$	59.9$_{\pm 3.2}$

5.6.3 Shift of LGV Subspace to Other Local Minima

This section demonstrates that the benefits of the LGV subspace geometry are shared across solutions in the weight space. We show that the LGV subspace significantly increases transferability when shifted both to another LGV-SWA solution and another regularly trained DNN. This indicates that the LGV subspace $\mathcal{S}$ embeds *generic* geometry properties that relate to transferability.

Shift to other LGV-SWA solution. We apply LGV to another independently trained DNN w'_0. We collect K new weights w'_k, which we average to obtain w'_{SWA}. We construct a new surrogate by adding the new deviations to w_{SWA},

$$\left\{ w_{\text{SWA}} + (w'_k - w'_{\text{SWA}}) \mid k \in [\![1, K]\!] \right\}, \tag{5.12}$$

and we call this new shifted surrogate "LGV-SWA + (LGV' - LGV-SWA')".

Shifting a LGV subspace to another flat solution (i.e., another LGV-SWA) yields a significantly better surrogate than sampling random directions from this solution. Tables 5.11 and 5.12 report the success rate of the surrogate described above. The difference between "LGV-SWA + (LGV' - LGV-SWA')" and "LGV-SWA + RD" varies from 3.27 to 12.32 percentage points, with a mean of 8.61. The fact that the subspace still improves transferability (compared to a random subspace) when applied to another vicinity reveals that subspace geometry has generic properties related to transferability.

Yet, we also find a degradation of success rate between this translated surrogate and our original LGV surrogate (-7.49 percentage points on average, with values

Table 5.12: Transfer success rate of LGV deviations shifted to other independent solutions, for non-ResNet targets.

		Target			
Norm	Surrogate	DN201	VGG19	IncV1	IncV3
L∞	LGV-SWA + (LGV' - LGV-SWA')	62.2±0.4	57.4±1.5	45.4±0.6	18.7±0.5
L∞	LGV-SWA + RD	50.0±1.0	47.5±1.9	34.9±0.4	13.4±0.7
L∞	LGV (ours)	69.7±1.0	67.5±1.1	58.6±0.8	25.4±1.5
L∞	1 DNN + γ (LGV' - LGV-SWA')	32.6±0.2	30.0±0.9	18.4±0.1	9.6±0.3
L∞	1 DNN + RD	23.1±0.8	22.8±0.4	14.1±0.1	6.8±0.6
L2	LGV-SWA + (LGV' - LGV-SWA')	69.8±0.6	65.7±0.7	55.0±1.0	27.4±0.5
L2	LGV-SWA + RD	58.1±0.3	55.6±1.9	42.7±0.6	20.2±0.5
L2	LGV (ours)	79.6±1.1	78.0±1.5	71.8±0.6	42.9±0.9
L2	1 DNN + γ (LGV' - LGV-SWA')	47.4±0.9	42.2±0.2	29.2±0.3	17.2±0.2
L2	1 DNN + RD	34.7±0.3	31.5±1.3	19.8±0.7	11.4±1.1

between -1.02 and -16.80). It indicates that, though the geometric properties are shared across vicinities, the subspace is optimal (w.r.t. transferability) when applied onto its original solution.

Shift to another initial DNN. The subspace is not solely relevant for solutions found by LGV: LGV deviations are also relevant when applied to regularly trained DNNs. For that, we build a new surrogate "1 DNN + γ (LGV' - LGV-SWA')" centered on the DNN w_0,

$$\left\{w_0 + \gamma(w'_k - w'_{\text{SWA}}) \mid k \in [\![1, K]\!]\right\}, \tag{5.13}$$

where the LGV deviations are scaled by a factor $\gamma \in \mathbb{R}$. Scaling is essential here because this new shift vector (another pretrained DNN) is sharper than LGV-SWA. A sharper shift vector means that deviations around it needs to be smaller to stay in the desirable vicinity.

We choose the γ hyperparameter by cross-validation (Figure 5.26). For computational efficiency, we randomly draw without replacement a subset of 10 LGV' deviations for each random seed. The original scale of LGV deviations is clearly not appropriate for a DNN. The optimal γ value is 0.5 for all eight targets. Unscaled LGV deviations exit the vicinity of low loss, which drops the success rate by 32.8 percentage points on average (6.52–59.47) compared to the optimal γ value of 0.5. When properly scaled and applied to an independently and regularly trained DNN, LGV deviations improve upon random directions by 10.0 percentage points in

average (2.87—13.88). Therefore, considering the flatness around the shift vector is of first importance to construct a good surrogate from weight deviations.

The optimal scale γ is consistent with the previously found optimal length along random directions in Section 5.6.1. The optimal Gaussian standard deviation for a DNN is also half of the one optimal for LGV-SWA. Flatness is consistent in that aspect between LGV and random subspaces. These observations also corroborate our observations that LGV-SWA is flatter than the initial DNN.

With all these results, we exhibit generic properties of the LGV subspace. It benefits solutions independently obtained. Applying LGV deviations on a solution of a different nature may require to scale them according to the new local flatness.

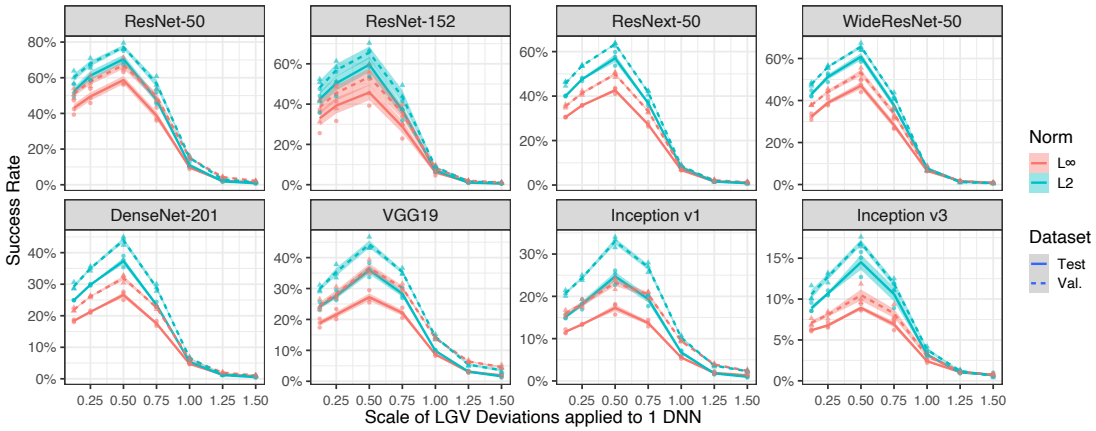


Figure 5.26: Transfer success rate with respect to the γ hyperparameter, the scale of the LGV' deviations applied to an independent DNN ("1 DNN + γ (LGV' - LGV-SWA')").

Section 5.6 – Conclusion. Taking together all our results, we conclude that the improved transferability of LGV comes from the geometry of the subspace formed by LGV weights in a flatter region of the loss. The LGV deviations spans a weight subspace whose geometry is densely and generically relevant for transferability. This subspace is key, as a single flat LGV model is not enough to succeed. This entire subspace enables to benefit from the flatness of this region, overcoming potential misalignment between the loss functions of the surrogate and that of the target model. That is, it increases the probability that adversarial examples maximizing the surrogate loss will also (near-)maximize the target loss – and thus successfully transfer.

5.7 Conclusion

We show that random directions in the weight space sampled at each attack iteration increase transferability, unlike random directions in the feature space. Based on this insight, we propose LGV, our approach to build a surrogate by collecting weights along the SGD trajectory with a high constant learning rate, starting from a regularly trained DNN. LGV alone beats all combinations of four state-of-the-art techniques. We analyse LGV extensively to conclude that (i) flatness in the weight space produces flatter adversarial examples which are more robust to surrogate-target misalignment; (ii) LGV weights spans a dense subspace whose geometry is intrinsically connected to transferability.

Overall, we open new directions to understand and improve transferability from the geometry of the loss in the weight space. We show that what matters to LGV is both the flatness of the loss and the weight subspace spanned by iterates. Additionally, LGV is a complementary technique to deep ensemble evaluated in Chapter 4: deep ensemble explores a new vicinity at every new DNN, and LGV can explore locally each of these vicinities starting from each independently trained DNN by deep ensemble. The established success of the local exploration of LGV closes the research question left open by the poor local exploration of cSGLD shown in Chapter 4. LGV also show the possibility of hybrid transferability techniques in-between training and attack times: LGV is a model augmentation transferability technique based on a few additional training epochs.

Furthermore, Mandt et al. [MHB17] give directions to connect the probabilistic perspective of Chapter 4 and the geometric perspective of this chapter. Mandt et al. [MHB17] show that SGD with constant learning rate can be used as an approximate Bayesian posterior inference algorithm. LGV may sample weights from the posterior distribution conditioned by the vicinity of the fully trained DNN where LGV starts. We left as future work the explicit development of this connection.

Two important directions are left open by this chapter. First, LGV starts from a regularly and fully trained DNN. The question of how to train a better base model follows naturally. Second, we want to ask if explicitly minimizing sharpness during training could improve transferability, since the flattening of the loss by LGV is a by-product of SGD. The next chapter tackles both directions by exploring ways to better train a single surrogate model, starting from early stopping, and then evaluating ways to minimize sharpness. Chapter 6 provides additional and complementary evidence in favour of our surrogate-target misalignment hypothesis.

6

Transferability From Representation in Flat Neighbourhood

This chapter confirms the key role of the flatness of the loss, by studying the training of a single surrogate model, i.e., a transferable representation. We provide evidence against a previous hypothesis regarding transferability from early stopping. Instead, we explain the success of early stopping in relation to the dynamics of the exploration of the loss landscape. SGD drives down the valley and progressively falls into deep, sharp holes, where the representations are too specific. Based on these insights, we propose RFN (Representation in Flat Neighboorhood) that searches for large flat neighbourhoods in the weight space to find transferable representations.

Contents

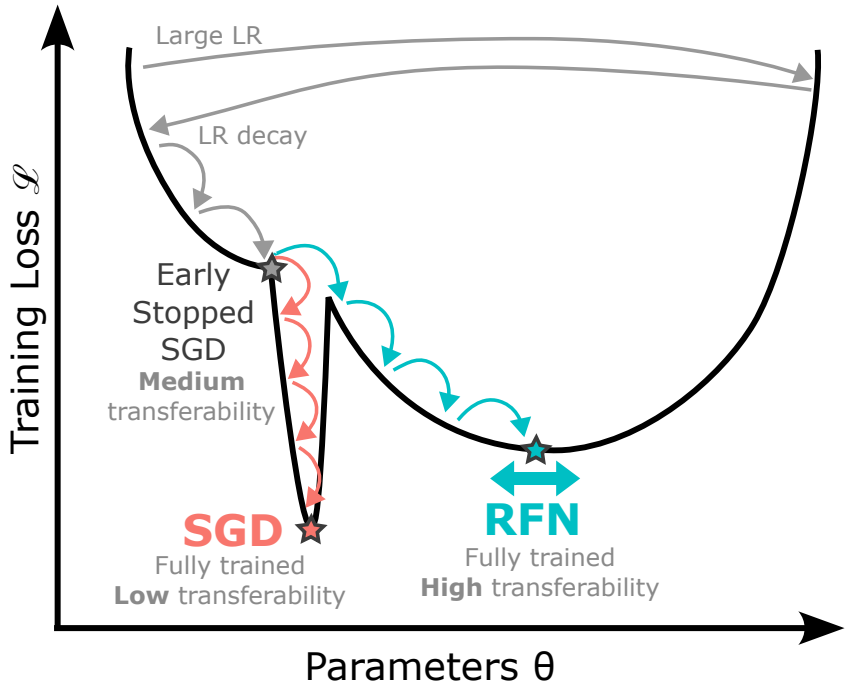


Figure 6.1: Illustration of the relation between the training dynamics of the surrogate model, sharpness, and transferability. Before the learning rate decays, training is in a "crossing the valley" phase for both SGD and RFN (gray) with plateauing transferability. A few iterations after the decay of the learning rate, early stopped SGD achieves its best transferability. In the following epochs, SGD falls progressively into deep, sharp holes in the parameter space with poor transferability (red). RFN (blue) avoids these holes by minimizing the maximum loss around an unusually large neighbourhood (thick blue arrow).

This chapter is based on the following paper:

- Martin Gubri et al. Going Further: Flatness at the Rescue of Early Stopping for Adversarial Example Transferability, April 2023. URL: `https://gubri.eu/publication/rfn_flatness_transferability/`

6.1 Introduction

In Chapter 5, we shed light on the importance of the local exploration of a vicinity around a base surrogate model, to improve the success rate of transfer-based black-box attacks. LGV, our proposed transferability technique, has proven superior to all combinations of four transferability techniques. The comprehensive analysis of LGV suggested that the flatness in the weight space contributes to the generation of flatter adversarial examples, which are more robust to surrogate-target misalignment. Additionally, we established that the weights of LGV span a dense

subspace, intrinsically tied to transferability. The success of the local exploration exhibited by LGV resolved the research gap left open by Chapter 4.

Nevertheless, this exploration has left us with two open directions that call for further investigation. Firstly, LGV operates from a regularly and fully trained DNN. This brings us to the question of how we can train a better base model. Secondly, we are intrigued by whether *explicitly* minimizing sharpness during training could enhance transferability, given that the flattening of the loss by LGV is a by-product of SGD with large learning rates. Establishing that explicitly minimizing sharpness improves transferability would offer additional and complementary evidence in favour of our surrogate-target misalignment hypothesis.

This chapter is dedicated to evaluating and analysing the transferability of single surrogate models. This topic is of particular importance, since understanding the underlining characteristics that drive transferability of a single DNN provides insights into how DNNs learn generic representations. This chapter strives to further our understanding of the training procedure of base surrogate models and the impact of sharpness minimization on transferability.

Early stopping is a common practice to improve natural generalization of DNNs by avoiding overfitting. Benz et al. [BZK21], Zhang et al. [ZCB+21], and Nitin [Nit21] propose to use early stopping to train better surrogate models. Despite an important amount of work on transferability, little attention was given to the selection of the surrogate model training procedure. Early stopping is probably the most discussed. The commonly accepted hypothesis is that an early stopped DNN is composed of more robust features than its fully trained counterpart, which has more brittle non-robust features [BZK21; ZCB+21; Nit21]. We provide in Section 6.3 some observations that contradict this hypothesis: early stopping improves transferability from and to models composed of non-robust features. Instead, our hypothesis is that the success of early stopping is closely related to the dynamics of the exploration of the loss surface. In Section 6.4, we describe this dynamic in relation to transferability. In particular, transferability peaks a few iterations of SGD after the decay of the learning rate. At the same time, sharpness in the parameter space drops. Later, the transferability slowly decreases and the sharpness slowly increases. On the basis of these insights, we propose RFN, a new approach to train surrogate models. By explicitly minimizing sharpness on unusually large neighbourhoods, we significantly improve transferability over SGD. Section 6.5 shows that this improvement is specific to transferability, since RFN and SGD have a similar natural generalization. We conclude that RFN alters the exploration of the loss landscape by *avoiding deep, sharp holes where the learned representation is too specific*. Finally, Section 6.6 evaluates RFN competitively against other training procedures and complementarily to other categories of transferability techniques.

Figure 6.1 summarizes the contributions of this chapter:

- The learning rate decay allows the exploration of the loss landscape to go down the valley. After a few iterations, SGD reaches its best transferability ("early stopped SGD", gray star). The sharpness is temporarily contained.
- As training with SGD continues, sharpness increases and transferability decreases. The fully trained model (red star) is a suboptimal surrogate. SGD falls into deep, sharp holes where the representation is too specific.
- RFN explicitly minimizes sharpness over a large neighbourhood (thick blue arrow) and avoids undesirable holes. Transferability is maximum after a full training (blue star), and early stopping is not needed.

6.2 Experimental Settings

This section describes the experimental settings used in this article. The experimental setup is standard for transfer-based attacks.

- Our source code used to train and evaluate models is publicly available on GitHub at this URL: `https://github.com/Framartin/rfn-flatness-transferability`.
- Our trained models on both CIFAR-10 and ImageNet are publicly distributed through HuggingFace at this URL: `https://huggingface.co/mgubri/rfn-flatness-transferability`.

Target models. All our target models on CIFAR-10 are fully trained for 150 epochs with SGD using the hyperparameters reported in Table 6.1. For a fair comparison, the baseline surrogate is trained with SGD using the same hyperparameters as the targets. On ImageNet, the target models are the pretrained models distributed by PyTorch. On CIFAR-10, we target the following nine architectures: ResNet-50 (the surrogate with the same architecture is an independently trained model), ResNet-18, ResNet-101, DenseNet-161, DenseNet-201, WideResNet-28-10, VGG13, VGG19, and Inception v3. The ten target architectures on ImageNet are the following: ResNet-50, ResNet-152, ResNeXt-50 32X4D, WideResNet-50-2, DenseNet-201, VGG19, GoogLeNet (Inception v1), Inception v3, ViT B 16 and Swin S. Additionally, we train a "validation" set of architectures on CIFAR-10 to select hyperparameters independently of reported results. This set is composed of: ResNet-50 (another independently trained model), ResNet-34, ResNet-152, DenseNet-121, DenseNet-169, WideResNet-16-8, VGG11, VGG16, and GoogLeNet (Inception v1). This validation set of target models on ImageNet is composed of the following architectures: ResNet-50 (another independently trained model), ResNet-101, ResNeXt-101 64X4D, WideResNet101-2, VGG16, DenseNet121, ViT B 32 and Swin B.

Table 6.1: Hyperparameters used to train surrogate models.

Training	Hyperparameter	Dataset	Value
All	Number of epochs	CIFAR-10	150
		ImageNet	90
	Initial learning rate	All	0.1
	Learning rate decay	CIFAR-10	Step-wise /10 each 50 epochs
		ImageNet	Step-wise /10 each 30 epochs
	Momentum	All	0.9
	Batch-size	CIFAR-10	256
		ImageNet	128
	Weight decay	CIFAR-10	0.0005
		ImageNet	0.0001
SAM	ρ	All	0.05 for SAM, 0.4 for RFN

Surrogate models trained with SGD. We train the surrogate models on CIFAR-10 and ImageNet using SGD with the standard hyperparameters of the robustness library [EIS+19] (Table 6.1). Due to computational limitations on ImageNet, we limit the number of epochs to 90, reusing the same hyperparameters as [ALM+20].

Surrogate models trained with SAM/RFN. We train surrogate models with SAM using the same hyperparameters as the models trained with SGD for both datasets. We integrate the SAM optimizer into the robustness library [EIS+19]. The unique hyperparameter of SAM is ρ which is set to 0.05 as the original paper for both datasets for the original SAM surrogate. The RFN surrogate is trained with SAM and ρ equal to 0.4.

Surrogate models of competitive techniques (Section 6.6). To compare with competitive training techniques on ImageNet, we reuse the original models of LGV-SWA (Chapter 5), weights averaged over 10 additional epochs with a high learning rate of 0.1, and SAT [SMK21], an adversarially trained model with a small maximum L_2 norm perturbation ε of 0.1 and with the PGD attack applied with 3 steps and a step size equals to $2\varepsilon/3$. On CIFAR-10, we reuse the best hyperparameters of [SMK21] to adversarially train the SAT surrogate model with a maximum L_2 norm ε of 0.025 and PGD with 7 steps and a step size of 0.3ε. For a fair comparison, we choose the best checkpoint of the early stopped SGD surrogate

Table 6.2: Hyperparameters of transferability techniques.

Dataset	**Technique**	**Hyperparameter**	Value
Early Stopping	ImageNet	Epoch	66
	CIFAR-10	Epoch	54
GN	ImageNet	Random range	$[1-0.3, 1+0.3]$
SGM	ImageNet	γ	0.2
LGV	ImageNet	Epochs	5
		Learning rate	0.05
DI	ImageNet	Minimum resize ratio	85%
		Probability transformation	80%
SI	ImageNet	Number of copies m	4
VT	ImageNet	β	1.8
		Number of copies N	20
MI	ImageNet	Decay	1.2
NI	ImageNet	Decay	0.6

by evaluating the transferability of every training epoch. For each epoch, we craft 1,000 adversarial examples from a distinct validation set of original examples and compute their success rate over a distinct set of validation target architectures. On CIFAR-10, the selected epoch is 54, and 66 on ImageNet. All the other hyperparameters not mentioned in this paragraph are the same as those used to train the surrogates with SGD.

Transferability Techniques. For a fair comparison with existing transferability techniques, we select their hyperparameter by cross-validation: we compute the success rate on a validation set of targets from a distinct set of natural examples. Table 6.2 reports the hyperparameters of the eight transferability techniques considered, i.e., early stopping, GN, SGM, LGV, DI, SI, VT, MI, and NI.

Attack. Unless specified otherwise, we use the BIM (Basic Iterative Method) [KGB17] which is the standard attack for transferability [BZK21; DLP+18; LBZ+18; LSH+20; SMK21; WWX+20; XZZ+19; ZZL+22]. By default, the maximum L_∞ perturbation norm ε is set to 4/255. We use the BIM hyperparameters tuned in Chapters 4 and 5 on a distinct set of validation target models: BIM performs 50 iterations with a step size equal to $\varepsilon/10$. Unless specified otherwise, we craft adversarial examples from a subset of 1,000 natural test examples that are correctly

predicted by all the target models. The success rate is the misclassification rate of these adversarial examples evaluated on one target model.

Threat model. We study the threat model of untargeted adversarial examples: the goal of the adversary is misclassification. We consider the standard adversary capability for transfer-based black-box attacks, where the adversary does not have query access to the target model. Query-based attacks are another distinct family of attacks.

Implementation. The source code of every experiment is available on GitHub. Our models are distributed through HuggingFace. We use the torchattacks library [Kim20] to craft adversarial examples with the BIM attacks and four transferability techniques, namely LGV, DI, SI, VT, MI and NI. We reuse the original implementations of GN and SGM to "patch" the surrogate architecture, and use the TorchAttacks implementation of BIM on top. The software versions are the following: Python 3.10.8, PyTorch 1.12.1, Torchvision 0.13.1, and TorchAttacks 3.3.0.

Infrastructure. For all experiments, we use Tesla V100-DGXS-32GB GPUs on a server with 256GB of RAM, CUDA 11.4, and the Ubuntu operating system.

6.3 Preliminaries: Another Look at the Non-Robust Features Hypothesis About Early Stopping

In this section, we point the flaws of the robust and non-robust features (RFs/NRFs) hypothesis [BZK21; ZCB+21; Nit21] to explain the success of early stopping for transferability. According to this hypothesis, earlier representations are more transferable than their fully trained counterparts because they contain more slightly RFs than NRFs. Slightly RFs are features that are robust to tiny worst-case perturbations, and NRFs are features that are not. See Section 3.1.4 for more details.

Early stopping indeed increases transferability. First, we check that a fully trained surrogate model is not optimal for transferability. We train two ResNet-50 surrogate models on the CIFAR-10 and ImageNet datasets using standard settings. Figures 6.2 and Figure 6.3 report the success rates respectively on CIFAR-10 and ImageNet, of the I-FGSM attack applied at every epoch and evaluated on a total of 19 fully trained target models. For both datasets and a variety of targeted architectures, the optimal epoch for transferability occurs around one or two thirds of training. Transferability decreases along epochs, except for the two vision transformers targets on ImageNet where the transferability plateaus at the end of training. Therefore, we confirm that early stopping increases transferability.

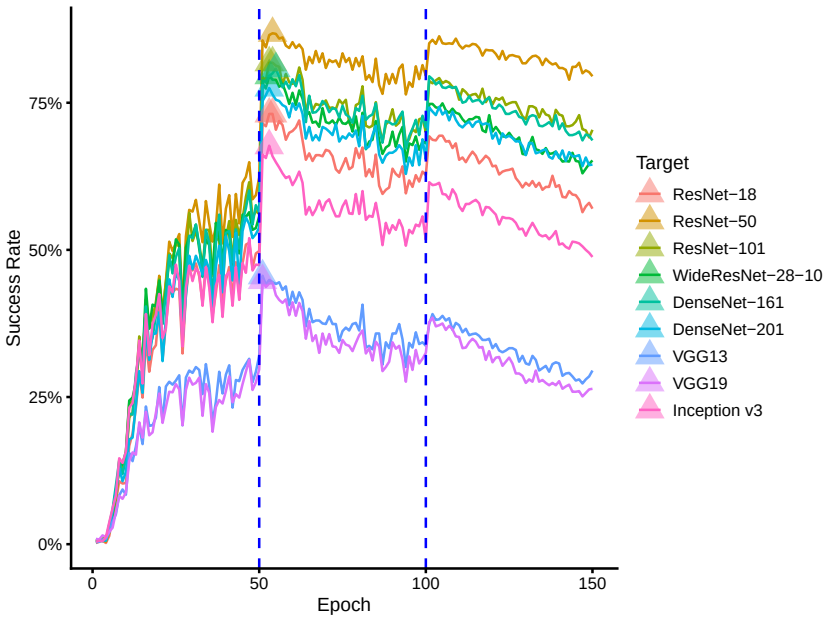


Figure 6.2: Early stopping improves transferability consistently across target models on CIFAR-10. Success rate evaluated on nine target models (colour) from a ResNet-50 surrogate model trained for a number of epochs (x-axis) on the CIFAR-10 dataset. Vertical bars indicate the step decays of the learning rate. Triangles indicate the epochs corresponding to the highest success rate per target.

Early stopping improves transferability from both surrogates trained on robust and non-robust datasets. We show that early stopping works *similarly* well on surrogate models trained on robust and non-robust datasets. We retrieve the robust and non-robust datasets from [IST+19], that are altered from CIFAR-10 to contain mostly RFs and, respectively, NRFs. We train two ResNet-50 models on both datasets with SGD and the same hyperparameters (reported in Section 6.2). Figure 6.4 shows the transferability across training epochs, averaged over the nine targets. The success rates of both robust and non-robust surrogate models evolve similarly (scaled by factor) to the model trained on the original dataset: transferability peaks around the epochs 50 and 100 and decreases during the following epochs. This observation is valid for all nine targets (Figure 6.5). According to the RFs/NRFs hypothesis, we expected "X-shaped" transferability curves: increasing transferability from NRFs and strictly decreasing transferability from RFs (after initial convergence). The RFs/NRFs hypothesis does not describe why early learned NRFs are better for transferability than fully learned NRFs.

Early stopping improves transferability to both targets trained on robust and non-robust datasets. We observe that an early stopped surrogate model

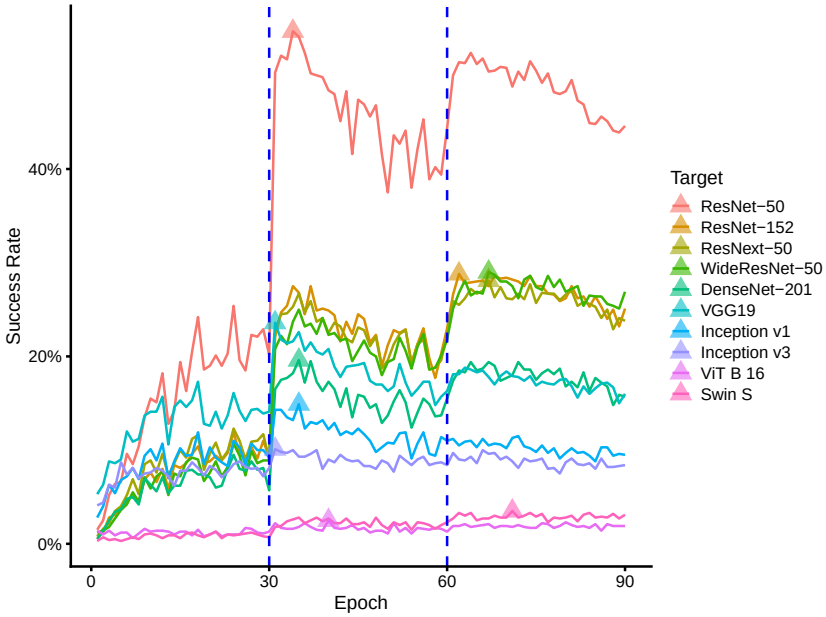


Figure 6.3: Early stopping improves transferability to various target models on ImageNet, except to vision transformers (ViT-B-16 and Swin-S) against which the success rate plateaus at the end of training. Success rate evaluated on ten target models (colour) from a ResNet-50 surrogate model trained for a number of epochs (x-axis) on the ImageNet dataset. Vertical bars indicate the step decays of the learning rate. Triangles indicate the epochs corresponding to the highest success rate per target.

trained on the original dataset is best to target both targets composed of RFs and NRFs. Here, we keep the original CIFAR-10 dataset to train the surrogate model. We target four ResNet-50 models trained on the robust and non-robust datasets of [IST$^+$19][1]. Figure 6.6 shows the same early stopped surrogate model is optimal for targeting both models composed of RFs and NRFs. Since the higher the transferability, the more similar the representations are, we conclude that the early trained representations are more similar to *both* RFs and NRFs than their fully trained counterparts.

Overall, we provide new evidence that early stopping for transferability acts similarly on robust and non-robust features. We do not observe an inherent trade-off between RFs and NRFs. Therefore, the hypothesis that early stopping favours RFs over NRFs does not hold. We conjecture that a phenomenon orthogonal to

[1]In this experiment, we include two other non-robust datasets D_{rand} and D_{det} from [IST$^+$19]. By construction, the *only* useful features for classification are NRFs. We did not include them in the previous experiment because training on them is too unstable.

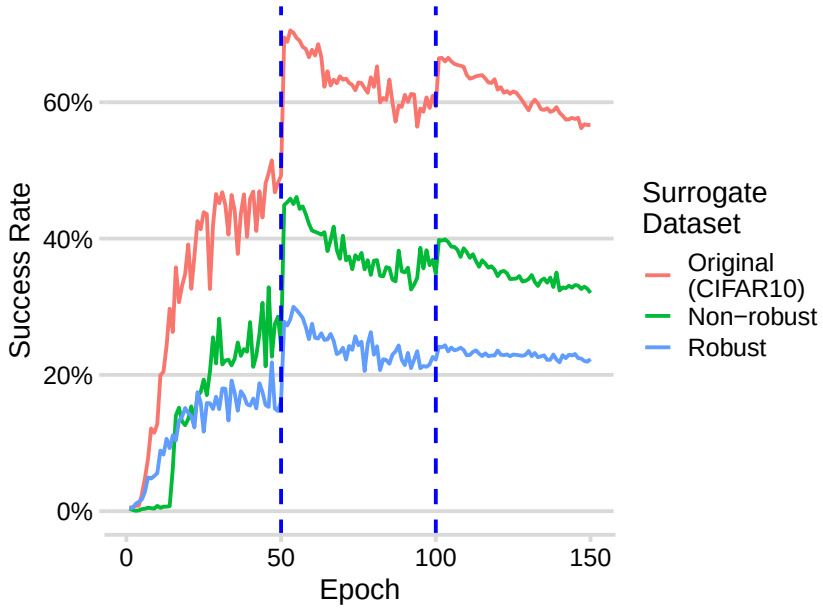


Figure 6.4: Early stopping improves transferability of surrogate models trained on both robust and non-robust datasets. Average success rate evaluated over nine target models from a ResNet-50 surrogate model trained for a number of epochs (x-axis) on the datasets D_R (blue) and D_{NR} (green) of [IST+19] modified from CIFAR-10 (red). We craft all adversarial examples from the same subset of the original CIFAR-10 test set.

RFs/NRFs explains why fully trained surrogates are not optimal.

6.4 Transferability and Training Dynamics

This section explores the relationship between the training dynamics of the surrogate model and its transferability. In particular, we observe several phenomena following the learning rate step decays: transferability peaks, sharpness drops, and the exploration of the loss surface transitions phases.

6.4.1 Transferability Peaks When the Learning Rate Decays

We point out the key role of the learning rate decay in the success of early stopping for transferability. The optimal number of surrogate training epochs for transferability occurs a couple of epochs after the decay of the learning rate. We train a ResNet-50 surrogate model for 150 epochs on CIFAR-10, using the standard learning rate schedule of [EIS+19] which divides the learning rate by 10 at epochs 50 and 100. For all nine targets considered individually, the highest transferability

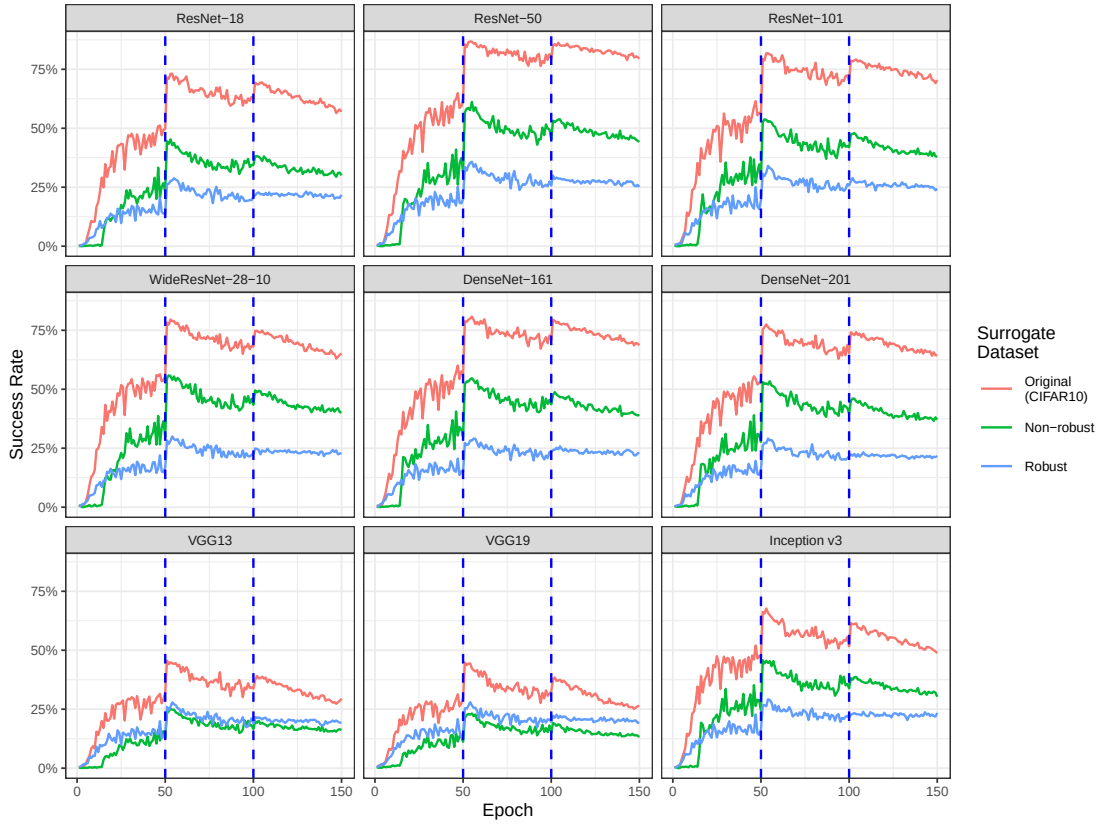


Figure 6.5: Early stopping improves transferability of surrogate models trained on both robust and non-robust datasets. Success rate evaluated over nine target models (title subfigure) from a ResNet-50 surrogate model trained for a number of epochs (x-axis) on the datasets D_R (blue) and D_{NR} (green) of [IST+19] modified from CIFAR-10 (red).

is between epochs 51 and 55 (Figure 6.3). Figure 6.7 shows that transferability suddenly peaks after the learning rate decay (red line). We train on ImageNet a ResNet-50 surrogate model for 90 epochs with learning rate decay at epochs 30 and 60. The highest transferability per target occurs either after the first decay (epochs 31 or 35) or after the second one (epochs 62 or 67), except for both vision transformer targets, where transferability plateaus at a low success rate after the second decay. Overall, the success of early stopping appears to be related to the exploration of the loss landscape, which is governed by the learning rate.

Consistency of the peak of transferability across training. The peak of transferability described above can be consistently observed at any point of training (after initial convergence). Here, we modify the standard double decay learning

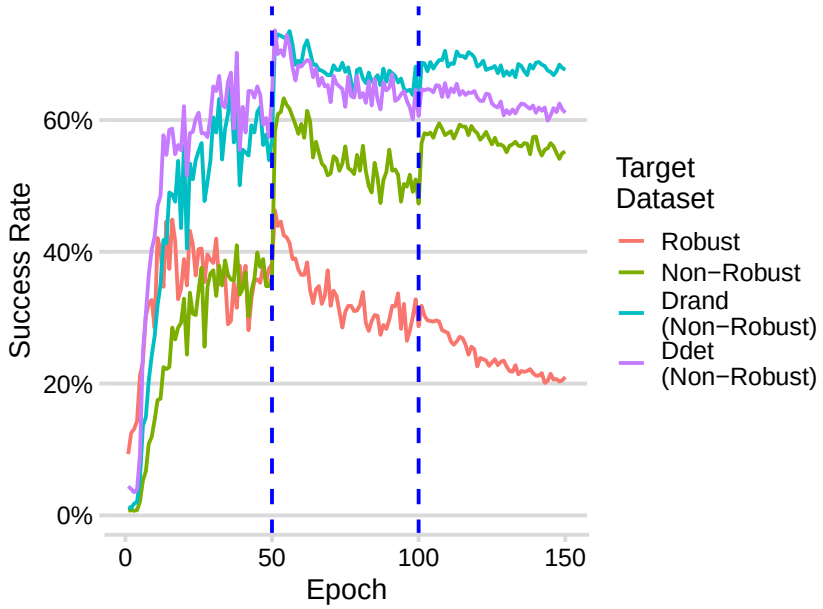


Figure 6.6: Early stopping improves transferability to target models trained on both robust and non-robust datasets. Success rate from a ResNet-50 surrogate model trained for a number of epochs (x-axis) on the original CIFAR-10 dataset, to ResNet-50 target models trained on the robust dataset D_R (red), and the three non-robust datasets D_{NR} (green), D_{rand} (green) and D_{det} (purple) of [IST+19] modified from CIFAR-10. The size of perturbation ε is 16/255 for the D_R target, 2/255 for the D_{NR} target and 1/255 for the D_{rand} and D_{det} targets to adapt to the vulnerability of target models (the order of curves cannot be compared). We craft all adversarial examples from the same subset of the original CIFAR-10 test set.

rate schedule to perform a single decay at a specified epoch. The learning rate is constant (0.1) until the specified epoch, where it will be 10 times lower (0.01) for the rest of the training. We evaluate the transferability of five surrogates with a decay at, respectively, epoch 25, 50, 75, 100 and 125. In Figure 6.7, we observe a similar transferability peak for all these surrogates, except for the decay at epoch 25 where the decay occurs before the end of the initial convergence. Figure 6.8 contains the transferability per target of the surrogate models trained with a single learning rate decay at a varying epoch. The consistency of the peak of transferability across training epochs is valid for all nine targets. We add as baseline the constant learning rate (at 0.1). Without learning rate decay, the transferability plateaus after initial convergence. Therefore, we conclude that the step decay of the learning rate enables early stopping to improve transferability.

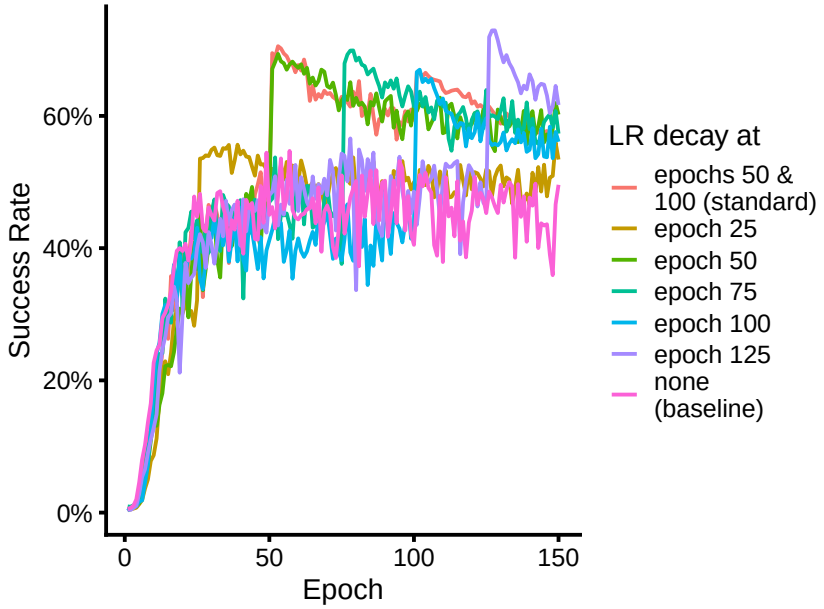


Figure 6.7: Step learning rate decay benefits transferability at any epochs after initial convergence. Average success rate evaluated over nine target models from a ResNet-50 surrogate model trained for a number of epochs (x-axis) on the CIFAR-10. The learning rate is divided by 10 a single time during training at an epoch indicated by the colour. Figure 6.8 contains the details per target.

6.4.2 Sharpness Drops When the Learning Rate Decays

When the learning rate decays, the sharpness in the parameter space drops. We observe a drop of both worst-case sharpness and average sharpness, measures respectively by the largest eigenvalue and the trace of the Hessian. We compute both sharpness metrics at every epoch using the PyHessian library [YGK+19] on a random subset of a thousand examples from the CIFAR-10 train dataset. Figure 6.9 reproduces the largest Hessian eigenvalue and the Hessian trace per epoch of our standard CIFAR-10 surrogate. For both metrics, we observe that sharpness decreases abruptly and significantly immediately after both learning rate decays at epochs 50 and 100.

6.4.3 Crossing the Valley Before Exploring the Valley

Before the learning rate decays, the exploration tends to behave more like "crossing the valley" than after decay, when it is more likely to "crawl down to the valley", as described in [SDH21]. [SDH21] proposes the α-quantity, a metric computed at the level of SGD iterations to disentangle whether the iteration understeps or overshoots the minimum along the current step direction. Based

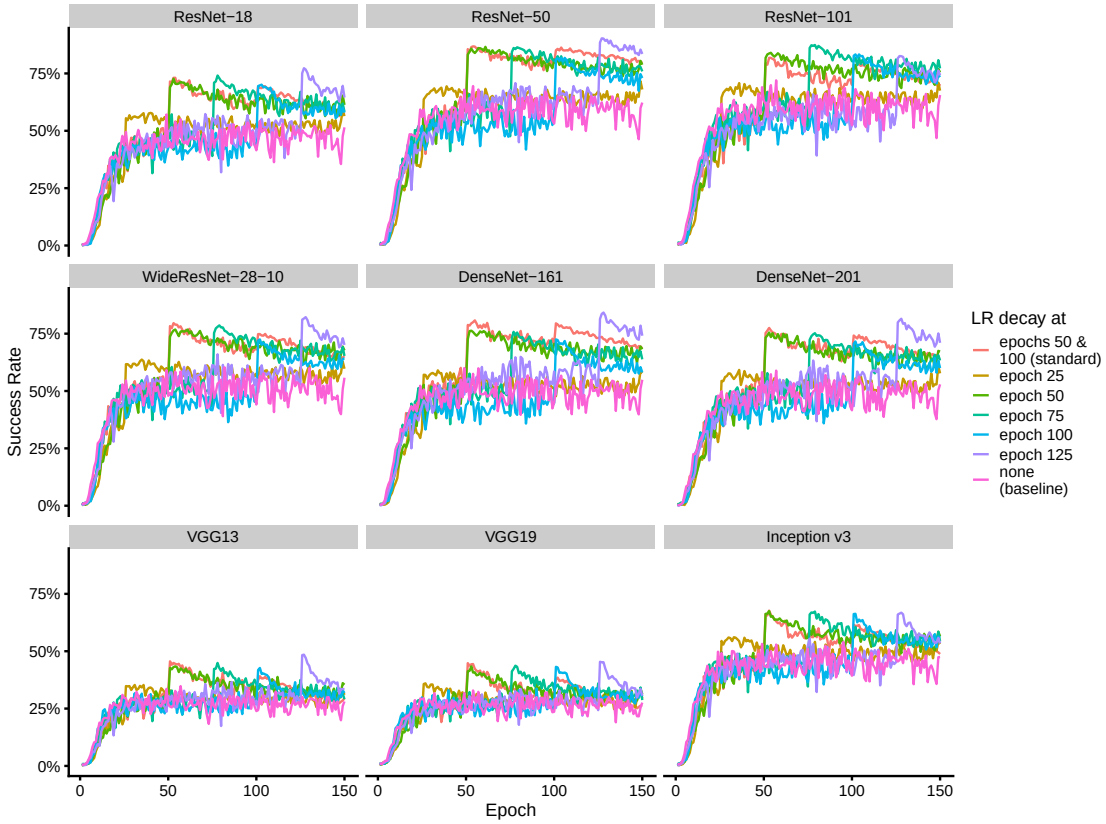


Figure 6.8: Step learning rate decay benefits transferability at any epochs after initial convergence. Success rate evaluated over nine target models from a ResNet-50 surrogate model trained for a number of epochs (x-axis) on the CIFAR-10. The learning rate is divided by 10 a single time during training at an epoch indicated by the colour.

on a noise-informed quadratic fit, $\alpha \approx 0$ indicates an appropriate learning rate that minimizes the loss in the direction of the gradient at this iteration ("going down to the valley"). $\alpha > 0$ indicates that the current learning rate overshoots this minimum ("crossing the valley"). We compute the α-quantity every four SGD iterations during the best five epochs for transferability on CIFAR-10 ("after learning rate decay", epochs 50–54) and during the five preceding epochs ("before learning rate decay", epochs 45–49). The one-sided Welch Two Sample t-test has a p-value inferior to $2.2e^{-16}$. We reject the null hypothesis in favour of the alternative hypothesis that the true difference of α-quantity in means between the group "before learning rate decay" and the group "after learning rate decay" is strictly greater than 0. We also perform a one-sided Welch Two Sample t-test on the 5 epochs before and after the second learning rate decay (epochs 95–99 vs. epochs

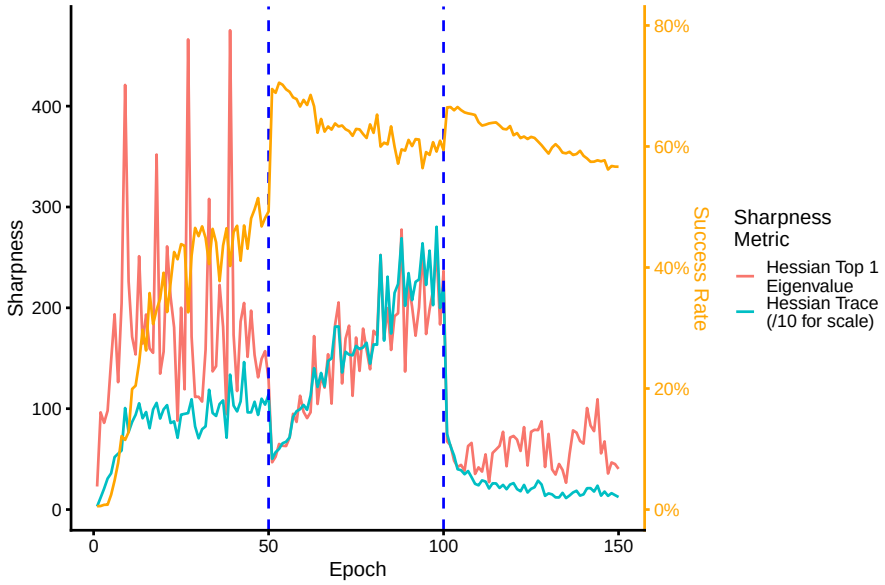


Figure 6.9: Sharpness drops when the learning rate decays. Largest eigenvalue of the Hessian (red) and trace of the Hessian (blue) for all training epochs (x-axis) on CIFAR-10. The largest eigenvalue of the Hessian represents the worst-case sharpness, i.e., the sharpness of the sharpest direction in the weight space. The trace of the Hessian represents the average sharpness in weight space. Average success rate on nine targets for all training epochs (orange, right axis). Vertical bars indicate the learning rate step decays.

100-105). Its p-value is equal to 0.004387. Using the Bonferroni correction, we compare the p-values of both individual tests with a significance threshold of 0.5%. We reject the null hypothesis for both learning rate decays with a significance level of 1%.

Figure 6.10 is the density plot of the α-quantities for both groups. Our results suggest that before the learning rate decay, training is slow due to a "crossing the valley" pattern. The best early stopped surrogate occurs a few training epochs after the learning rate decay when the SGD starts exploring the bottom of the valley.

We conclude that *the effect of early stopping on transferability is tightly related to the dynamics of the exploration of the loss surface*, governed by the learning rate. Overall, Figure 6.1 illustrates our observations:

1. Before the learning rate decays, the training bounces back and forth crossing the valley from above (top gray arrows).

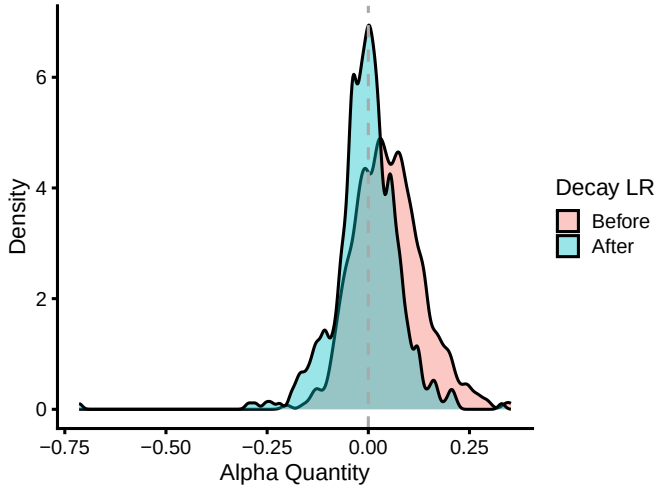


Figure 6.10: When the learning rate decays, the exploration tends to cross the valley less, and to crawl down to the valley more. Density plot of the α-quantity values computed each four SGD iterations during the best five epochs for transferability on CIFAR-10 (epochs 50–54, "After" group, blue) and the five preceding epochs (epochs 45–49, "Before" group, red).

2. After the learning rate decays, training goes down the valley. Soon after, SGD has its best transferability ("early stopped SGD" gray star). Sharpness is reduced.

3. When learning continues, the training loss decreases and sharpness slowly increases. SGD finds a "deep hole" of the loss landscape, corresponding to a specific representation that has poor transferability ("fully trained SGD" red star).

6.5 Going Further: Flatness at the Rescue of SGD

Since transferability peaks to its higher value when sharpness drops, we explore in this section how to improve transferability by explicitly minimizing the sharpness of the surrogate model. First, we show that SAM, sharpness-aware minimizer, finds surrogate models with high transferability at every epoch. Second, we propose a new transferability technique called **RFN** (Representation from Flat Neighbourhood), based on unusually large flat neighbourhoods that avoid large holes on the surface of the loss landscape. In practice, RFN is simply SAM with an unusually high hyperparameter ρ that benefits transferability specifically, since its value degrades natural accuracy compared to the original SAM.

Sharpness-Aware Minimizer (SAM, [FKM+20]) minimizes the maximum loss around a neighbourhood by performing a gradient ascent step followed by a gradient descent step. At the cost of one additional forward-backward pass per iteration, SAM avoids deep, sharp holes on the surface of the loss landscape [KLS+22]. We explore the use of SAM to train a surrogate model and show that:

1. Explicitly minimizing sharpness improves transferability over SGD.

2. The large flat neighbourhoods used by RFN benefit specifically transferability (not natural generalization).

3. Contrary to SGD, SAM and RFN benefit from a full-training. Thus, RFN does not need early stopping to train generic representations.

6.5.1 Minimizing Sharpness Improves Transferability

By explicitly minimizing sharpness in the parameter space, SAM trains better surrogate models than SGD. On CIFAR-10, we train a ResNet-50 model *without any hyperparameter tuning*, using the original SAM hyperparameter ($\rho = 0.05$). The success rate averages over the nine targets at 79.71%, compared to 56.66% for full training with SGD, and 70.54% for SGD at its best (epoch 53)[2]. Similarly, on ImageNet, we train a ResNet-18 with SAM ($\rho = 0.05$) and obtain an average success rate on ten targets of 18.81%. A full training with SGD averages to 13.23%, and an early stopped one reaches 14.48% at its best (epoch 68). For both datasets, SAM consistently improves over fully trained and early stopped SGD for all our 19 targets (Figures 6.11, 6.14 and 6.15).

6.5.2 Large Flat Neighbourhoods Are Specific To Transferability

RFN significantly increases the transferability over the original SAM by minimizing sharpness over uncommonly large neighbourhoods (defined by the ρ hyperparameter). The unique hyperparameter of SAM, ρ, controls the size of the neighbourhood where SAM minimizes sharpness. We evaluate the sensibility of transferability regarding this hyperparameter on CIFAR-10. We show that large flat neighbourhoods are optimal for transferability, but degrade slightly the natural accuracy, indicating a specificity of transferability. Without any additional tuning, we show that similarly large neighbourhood benefits transferability on ImageNet. Finally, we show that such large neighbourhoods are also beneficial to three SAM variants, but none of the variants strictly dominate RFN.

[2]Note that this original SAM surrogate is trained without tuning hyperparameter for transferability. We show below that larger ρ significantly improves it.

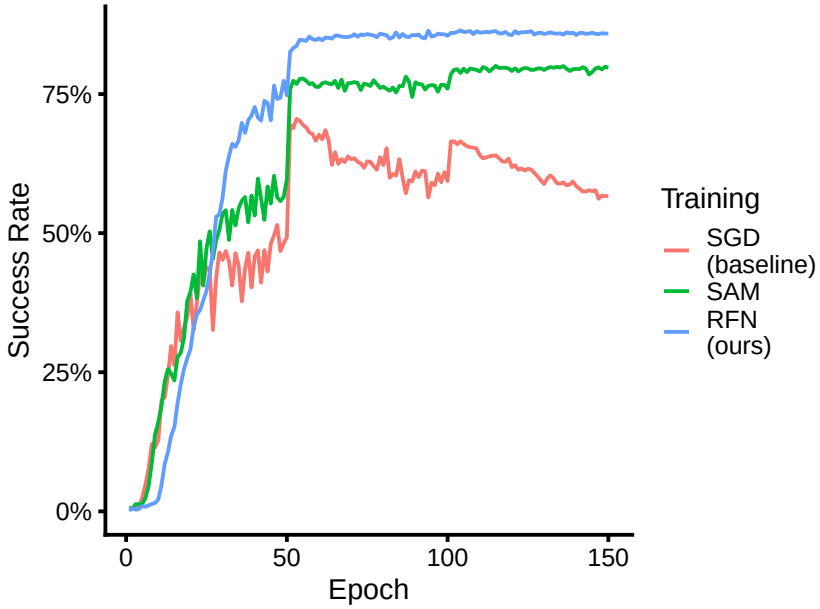


Figure 6.11: RFN improves transferability over SGD and the original SAM: RFN and SAM have higher transferability than both fully trained and early stopped SGD, RFN and SAM do not need early stopping, RFN is better for transferability than the original SAM. Transferability from three surrogate models trained respectively with SGD (baseline, red), SAM with its original ρ hyperparameter ($\rho = 0.05$, green) and RFN, i.e., SAM with large neighbourhood optimal for transferability ($\rho = 0.4$, blue). Average success rate evaluated over nine target models from a ResNet-50 surrogate model trained for a number of epochs (x-axis) on CIFAR-10. Transferability from ResNet-18 and ResNet-50 trained on ImageNet evolves similarly (Figures 6.14 and 6.15).

The size of flat neighbourhoods: the choice of the ρ hyperparameter. SAM has similar transferability for a wide range of its unique ρ hyperparameter. Table 6.3 shows that the success rates on our test targets vary less than a percentage point for ρ between 0.3 and 0.5. Figure 6.12 shows that the improvements from SAM holds for all targets in this same range. Moreover, SAM with any ρ value clearly dominates the SGD baseline. SAM is robust to a badly selected hyperparameter. Therefore, the risk of worsening transferability by switching from SGD to SAM to train the surrogate model is tiny.

For RFN, we select $\rho = 0.4$ based on a separate set of architectures for the target models and a disjoint subset of original examples (Table 6.3, see Section 6.2 for more details about the experimental setting). We use the same $\rho = 0.4$ for RFN on ImageNet without an additional hyperparameter selection.

Table 6.3: SAM improves transferability for a wide range of its unique ρ hyperparameter. Average success rate on nine targets of surrogates trained using SAM with various values of its ρ hyperparameter on CIFAR-10. The "Validation" and "Test" columns use distinct sets of targeted architectures and distinct subsets of original examples. All surrogate models are trained for 150 epochs on CIFAR-10. $\rho = 0$ corresponds to the baseline of the fully trained standard SGD surrogate. In %.

Value of ρ	**Validation Success Rate**	**Test Success Rate**
0.00 (SGD baseline)	55.59	56.61
0.05 (original SAM)	78.06	79.70
0.10	80.08	83.10
0.20	78.62	81.11
0.30	81.65	85.36
0.40 (RFN, best for transferability)	**81.68**	**85.88**
0.50	81.64	85.14
0.60	77.36	80.38
0.70	76.56	80.90
0.80	59.47	64.23
1.00	58.63	64.39

Table 6.3 and Figure 6.12 show that the unusually large flat neighbourhood ($\rho = 0.4$) used by RFN benefits all targets compared to the original SAM ($\rho = 0.05$).

Large flat neighbourhoods are specific to transferability. Figure 6.13 shows the natural accuracy of SGD, SAM and RFN. As reported by Foret et al. [FKM+20], SAM improves over SGD. Nevertheless, we observe that the large neighbourhoods used by RFN degrade natural accuracy compared to SAM and to the SGD baseline, suggesting that *such large flat neighbourhoods are specifically related to transferability*. This observation echoes the finding in Chapter 5 that the large learning rate used by LGV is best for transferability and degrades natural accuracy.

Kaddour et al. [KLS+22] show that SAM with different ρ values ends up in different basins. Therefore, RFN, i.e., SAM with high ρ, finds neighbourhoods that are specifically beneficial for transferability. It avoids large sharp holes on top of the loss surface.

Generalization of large flat neighbourhoods: extension to ImageNet. The ρ value selected on CIFAR-10 is also significantly beneficial on ImageNet. The conclusions about RFN, SAM and SGD made on CIFAR-10 hold on ImageNet:

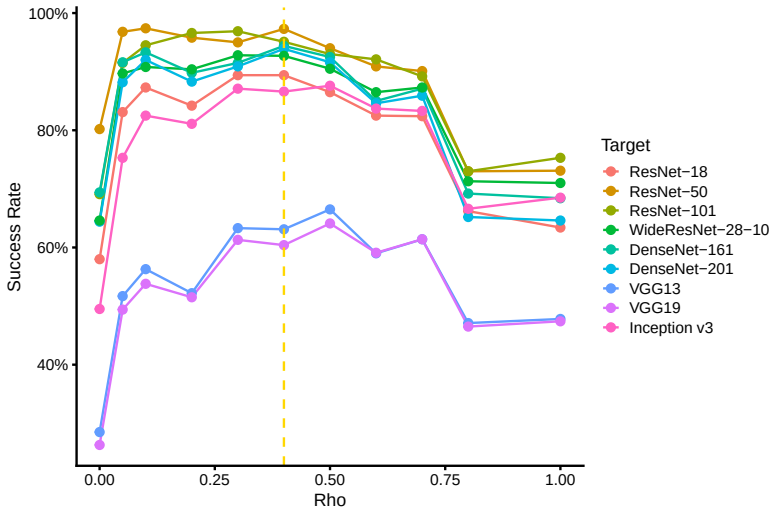


Figure 6.12: All targeted architectures benefits from SAM with a large flat neighbourhood. Success rate evaluated on nine target models (colour) from a ResNet-50 surrogate model trained using SAM with various ρ hyperparameters (x-axis) on the CIFAR-10 dataset. The vertical yellow bar indicates the ρ selected for RFN. Adversarial examples are crafted from a subset of one thousand original examples from the test set. $\rho = 0$ is the SGD fully trained baseline.

RFN is better for transferability than both the original SAM and SGD, and RFN degrades natural accuracy compared to SAM. Figures 6.15 and 6.14 report the success rates along epochs on ImageNet, respectively, from a ResNet-50 and ResNet-18 surrogate. Due to computational limitations, we only train the original SAM with the ResNet-18 architecture. We apply RFN on ImageNet without any hyperparameter selection: we reuse directly the γ value selected on CIFAR-10. Despite this disadvantage, *the same size of flat neighbourhood extends to ImageNet* for both architectures. Figures 6.16 and 6.17 report the natural accuracy along epochs of the above-mentioned surrogate models respectively for the ResNet-18 and ResNet-50 architectures. The same conclusion on CIFAR-10 extends to ResNet-18 on ImageNet: the large flat neighbourhoods used in RFN degrade are specific to transferability, i.e., they degrade natural accuracy compared to SAM[3].

The transferability of SAM variants. On a side note, we also train three variants of SAM (ASAM, GSAM, AGSAM) proposed by Kwon et al. [KKP+21] and Zhuang et al. [ZGY+22]. We show that the large neighbourhoods identified above are also beneficial to these variants, but none of the variants provide a significant

[3]Due to computational limitations, we did not train a ResNet-50 with the regular SAM on ImageNet, and therefore cannot compare SAM and RFN with ResNet-50 on ImageNet.

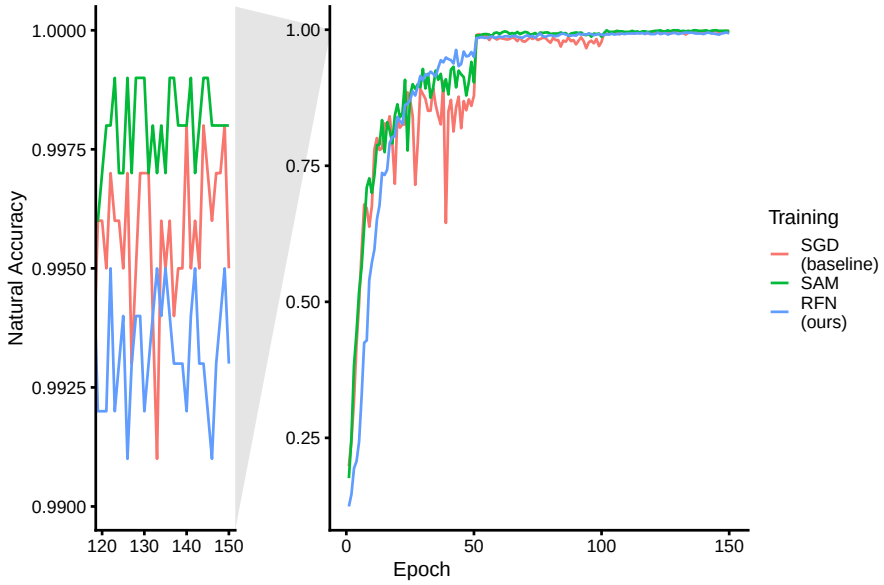


Figure 6.13: The original SAM is best for natural accuracy, and RFN worsens natural accuracy compared to SGD and the original SAM. Natural test accuracy of the ResNet-50 surrogate model trained for a number of epochs (x-axis) on the CIFAR-10 dataset. Evaluated on the test subset used to craft adversarial examples in Figure 6.12.

improvement over RFN.

We try GSAM for various values of the ρ hyperparameter. Figure 6.18 shows that GSAM does not significantly improve transferability over SAM. GSAM is more sensitive to very large neighbourhoods ($\rho = 1.0$) where the SGD surrogate baseline is significantly better for all targets. Therefore, we do not consider GSAM for RFN.

Additionally, we train two additional variants ASAM, an adaptive variant of SAM, and AGSAM, an adaptive variant of GSAM. We follow the original paper [KKP+21] to select the ρ hyperparameter: the authors recommend multiplying ρ by 10 when switching to an adaptive variant. We use $\rho = 4$ for ASAM and AGSAM. Table 6.4 shows that ASAM has a slightly higher average success rate than RFN (0.29 percentage point). But ASAM improves over RFN only for four of our nine target models. The AGSAM surrogate is better than RFN only for two of our nine targets, and decreases the average success rate by more than one percentage point compared to RFN. We decided to base RFN on top of SAM, but ASAM might be a promising direction for future work.

Overall, we found that unusually large flat neighbourhoods are best for trans-

Table 6.4: None of the three SAM variants strictly dominates RFN. Average success rate on the nine targets of a ResNet-50 surrogate model trained using SAM variants on CIFAR-10. The "better than RFN" column reports the number of targets where the SAM variant in the corresponding row has higher transferability than RFN. Adversarial examples are crafted from a subset of 1,000 original examples from the test set. We train all surrogates for 150 epochs.

Training	**Success Rate (%)**	**Better than RFN**
SGD (baseline)	56.66	0 / 9
SAM (original)	79.71	0 / 9
RFN (ours)	85.88	
GSAM (rho=0.4)	83.52	0 / 9
ASAM (rho=4)	**86.17**	4 / 9
AGSAM (rho=4)	84.57	2 / 9

ferability. The original SAM [FKM+20] uses a $\rho = 0.05$, Kaddour et al. [KLS+22] tunes it with a maximum of 0.2. On both CIFAR-10 and ImageNet, $\rho = 0.4$ is best for transferability, i.e., a value twice larger than the maximum generally considered for natural generalization. Indeed, we observe that such large neighbourhoods degrade natural accuracy. [KLS+22] shows that changing ρ ends up in different basins. Therefore, RFN, i.e., SAM with high ρ, finds neighbourhoods that are specifically beneficial for transferability. It avoids large sharp holes on top of the loss surface.

6.5.3 RFN Does Not Need Early Stopping

Training longer with RFN or SAM is both more stable than SGD and beneficial for transferability. In Figure 6.11, we report the success rate per training epoch for both SAM (green), RFN (blue), and SGD (red) on CIFAR-10. Contrary to SGD which needs to be carefully stopped to not train a suboptimal surrogate, the transferability of RFN and SAM increases or plateaus. These observations are also valid for ResNet-18 and ResNet-50 surrogates trained on ImageNet (Figures 6.14 and 6.15). RFN and SAM trainings are more stable, benefit from a full training, and do not require an error-prone stopping criterion.

Overall, RFN can avoid deep sharp minima in favour of unusually large flat neighbourhoods containing more generic representations.

6.6 Evaluation of RFN

In this section, we show that RFN is an alternative to competitive techniques and complements other techniques well. To compare our technique to related work, we follow the good practices recommended by [ZZL+22]: we evaluate each technique on its own with other comparable techniques that belong to the same category. First, we show that RFN is a competitive alternative to existing training techniques. Second, we show that RFN complements nicely transferability techniques. All our code and models are available on GitHub[4].

6.6.1 RFN Improves Over Competitive Techniques

We compare our RFN technique to competitive techniques that train a single representation. We aim to find training methods that lead to generic representations for transferability. We evaluate the model-augmentation transferability techniques in the next paragraph because avoiding the attack to overfit to a single model is an orthogonal objective. For a fair comparison, we choose the epoch of the early stopped SGD surrogate by evaluating a validation transferability at every training epoch. We craft one thousand adversarial examples from images of a validation set and evaluate them against a distinct set of target models. We retrieve the SAT (Slight Adversarial Training) ImageNet pretrained model used in [SMK21], and we train it on CIFAR-10 using the same hyperparameters of adversarial training. LGV-SWA (Chapter 1) is a single model defined by the weight average of the models collected by LGV. RFN uses $\rho = 0.4$ for both datasets.

Tables 6.5 and 6.8 report the success rates of various techniques. On CIFAR-10, RFN strictly dominates the other methods for every target. RFN requires twice as many computations as SGD, but four times less than SAT[5]. Tables 6.6 and 6.7 evaluate competitive techniques of RFN on CIFAR-10 with, respectively, maximum perturbations L_∞ norm ε of 2/255 and 8/255. The same conclusions made with perturbations of size 4/255 hold for these two norms: RFN clearly improves transferability. RFN beats other competitive techniques for all nine targets and both norms. On ImageNet, RFN beats the other techniques for 5 of the 10 targets. For the other targets, RFN is second after SAT. The same observations extend to other maximum perturbations L_∞ norms. Tables 6.9 and 6.10 show respectively that RFN beats the other techniques in 6 out of 10 targets for ε equal to 2/255, and in 5 out of 10 targets for ε equal to 8/255. Still, RFN is an interesting alternative since SAT doubles the training computational budget compared to RFN. We leave as future work the combination of SAT and RFN by replacing SGD by SAM to update parameters during adversarial training.

[4] `https://github.com/Framartin/rfn-flatness-transferability`

[5] SAT [SMK21] uses adversarial training with seven PGD steps on CIFAR-10. It needs a total of 8 backward-forward passes per training iteration. RFN always requires two.

6.6.2 RFN Is a Better Base Model for Complementary Techniques

We show that RFN is a good base model to combine with existing model augmentation, data augmentation, and attack optimization transferability techniques. These categories tackle complementary objectives: model and data augmentations aim at reducing the tendency of the attack to overfit the base model by adding randomness to gradients. Attack optimizers intend to smooth the gradient updates. Table 6.11 presents the success rates of eight transferability techniques combined with our RFN base model on ImageNet. For every target, RFN provides a base model that improves every eight techniques, compared to the standard fully trained SGD surrogate. The only exception (underlined) is the combination with LGV for three of the ten targets. Since LGV collects models with SGD and a high learning rate, a conflict might occur when LGV continues training with SGD from a checkpoint trained with SAM. We leave for future work the evaluation of an LGV variant where models are collected with SAM and a high constant learning rate.

These conclusions extend to other maximum perturbation norms ε. Tables 6.12 and 6.13 evaluate the complementary transferability techniques on ImageNet with, respectively, 2/255 and 8/255 norms. RFN increases the transferability of every eight techniques against every ten targets when combined, except for LGV on 4 targets using ε equals 2/255, and LGV on 3 targets with ε equals 2/255. Again, since LGV collects models with SGD and a high learning rate, a conflict might occur when LGV continues training with SGD from a checkpoint trained with SAM. Future work may explore the adaptation of the LGV model collection to SAM.

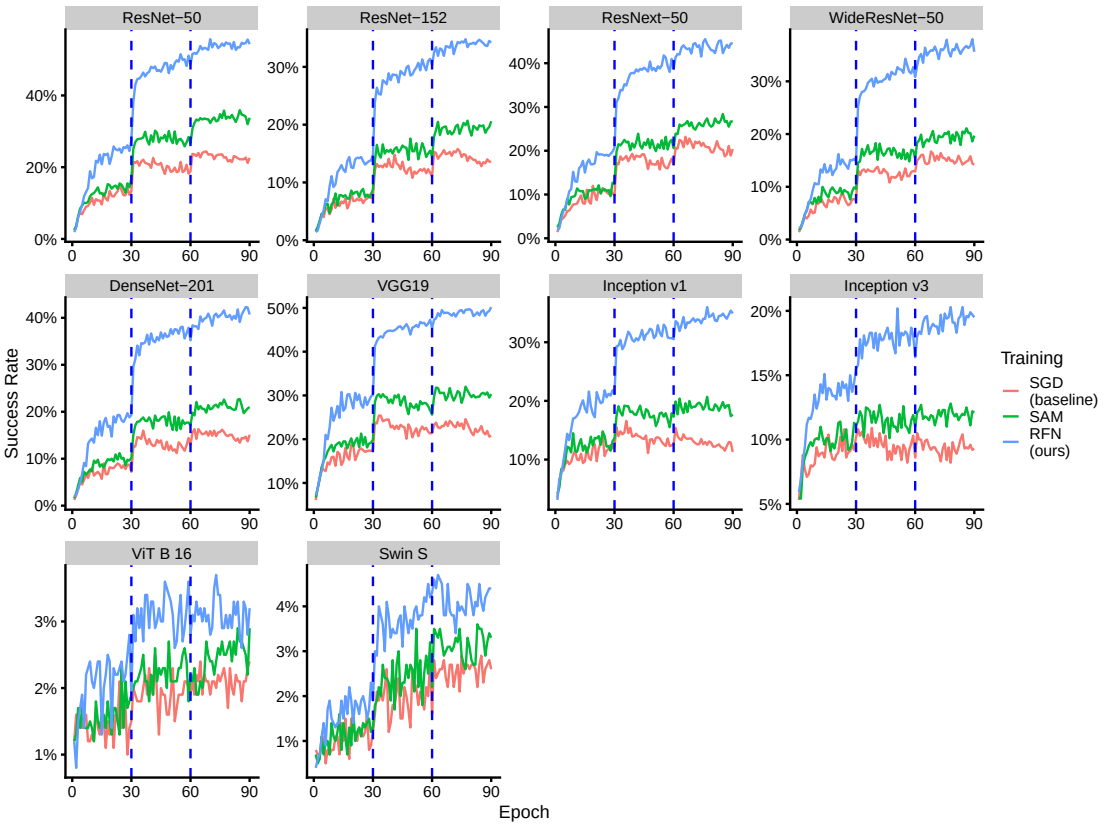


Figure 6.14: RFN improves transferability over SGD and the original SAM on ImageNet: RFN and SAM have higher transferability than both fully trained and early stopped SGD, RFN and SAM do not need early stopping, RFN is better for transferability than the original SAM. Transferability from three surrogate models trained respectively with SGD (baseline, red), SAM with its original ρ hyperparameter ($\rho = 0.05$, green) and RFN, i.e., SAM with large neighbourhood optimal for transferability ($\rho = 0.4$, blue). Success rate evaluated over ten target models from a **ResNet-18** surrogate model trained for a number of epochs (x-axis) on CIFAR-10.

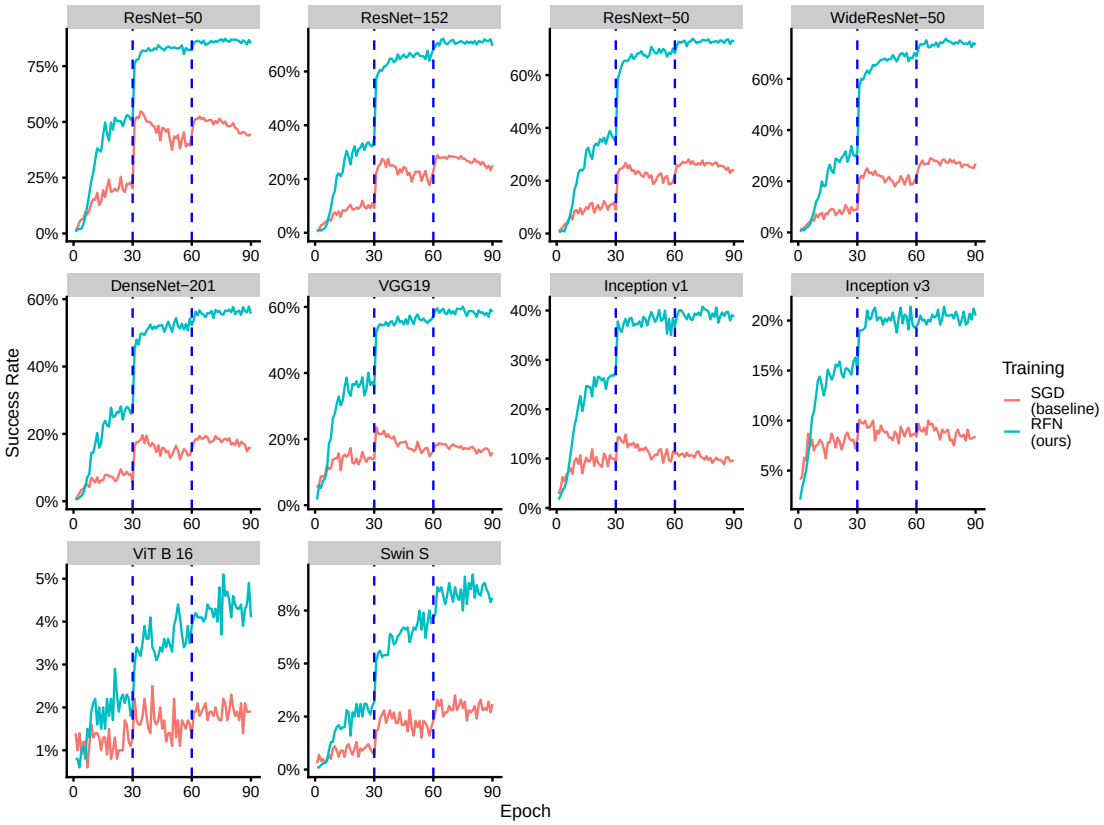


Figure 6.15: RFN improves transferability over SGD on ImageNet: RFN has higher transferability than both fully trained and early stopped SGD, RFN does not need early stopping. Transferability from two surrogate models trained respectively with SGD (baseline, red), RFN, i.e., SAM with large neighbourhood optimal for transferability ($\rho = 0.4$, blue). Success rate evaluated over ten target models (subfigure title) from a ResNet-50 surrogate model trained for a number of epochs (x-axis) on ImageNet.

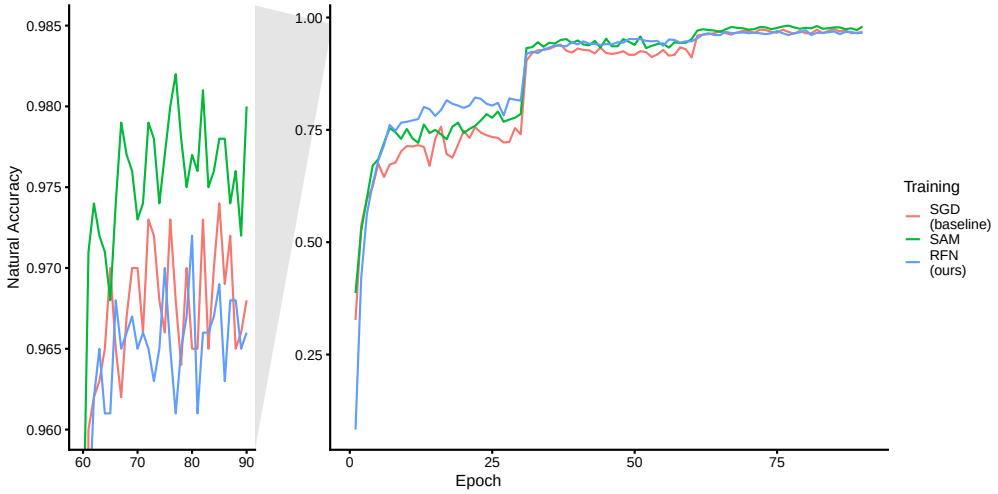


Figure 6.16: The original SAM is best for natural accuracy, and RFN worsens natural accuracy compared to SGD and the original SAM. Natural test accuracy of the **ResNet-18** surrogate model trained for a number of epochs (x-axis) on the ImageNet dataset. Evaluated on the test subset used to craft adversarial examples in Figure 6.14.

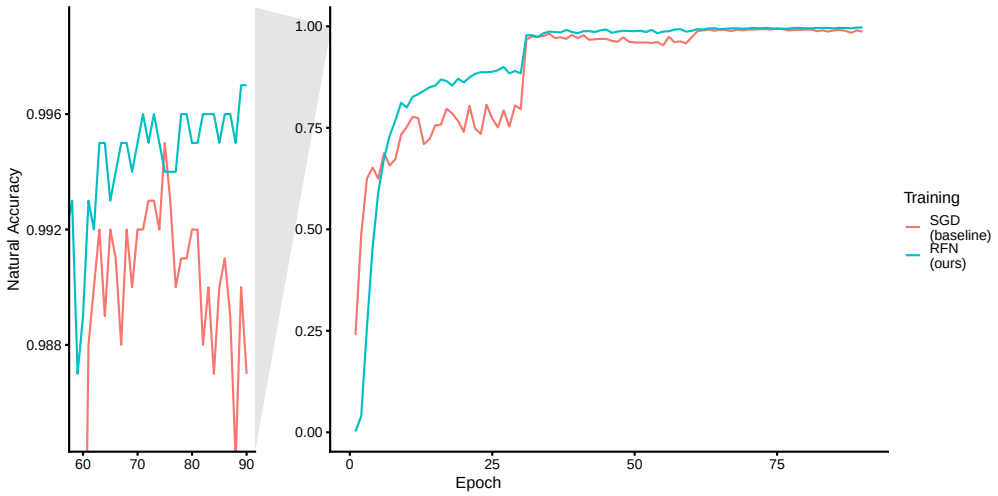


Figure 6.17: Natural test accuracy of the ResNet-50 surrogate model trained for a number of epochs (x-axis) on the ImageNet dataset. Evaluated on the test subset used to craft adversarial examples in Figure 6.15.

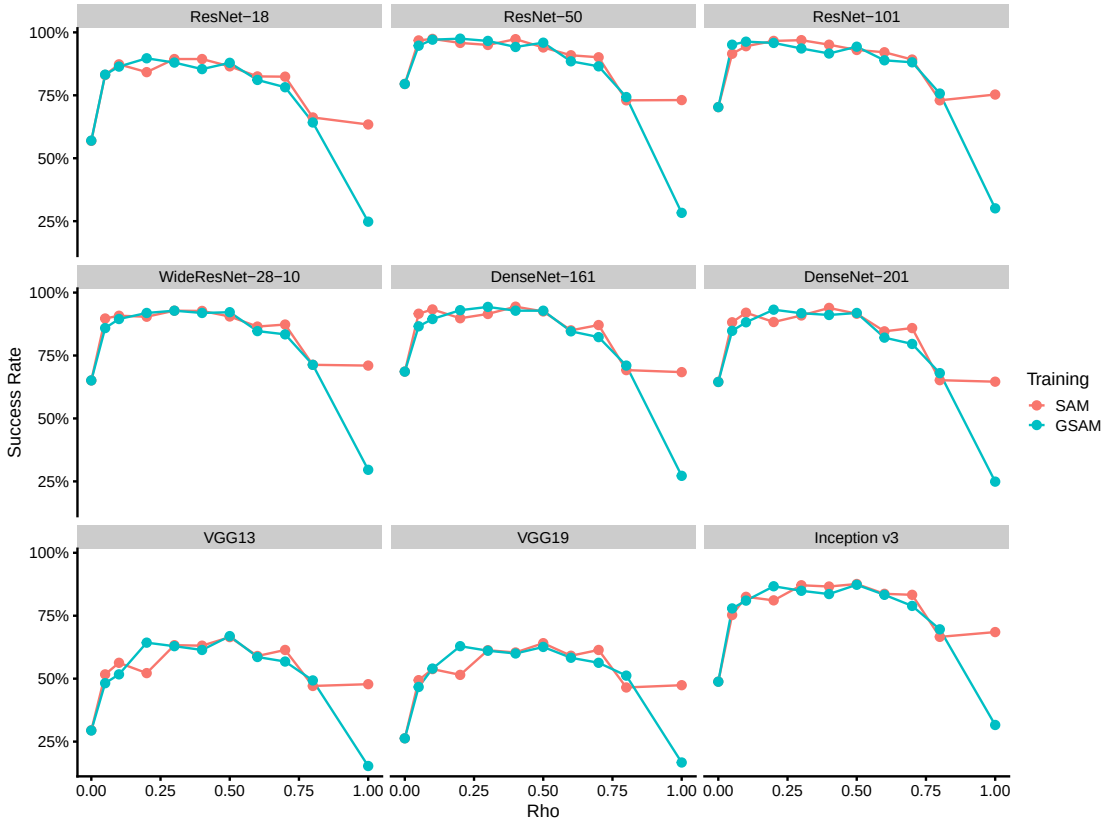


Figure 6.18: GSAM does not improve transferability compated to SAM. Success rate evaluated on nine target models (colour) from a ResNet-50 surrogate model trained using SAM (red) or GSAM (blue) with various ρ hyperparameters (x-axis) on the CIFAR-10 dataset. Adversarial examples are crafted from a subset of one thousand original examples from the test set. $\rho = 0$ is the SGD fully trained baseline.

Table 6.5: Success rate on CIFAR-10 of competitive techniques to train a single surrogate model. Adversarial examples evaluated on nine targets with a maximum perturbation L_∞ norm ε of 4/255. Bold is best. In %.

	Target								
Surrogate	RN18	RN50	RN101	DN161	DN201	VGG13	VGG19	IncV3	WRN28
Fully Trained SGD	57.9	81.2	70.6	70.8	66.1	27.8	26.3	49.4	66.5
Early Stopped SGD	73.3	87.8	82.1	81.4	78.3	45.5	44.3	66.8	79.5
SAT	66.3	76.2	73.6	66.9	66.1	49.8	48.5	57.9	67.8
RFN **(ours)**	**89.7**	**97.3**	**95.5**	**95.7**	**94.0**	**63.6**	**60.6**	**87.3**	**93.0**

Table 6.6: Success rate on CIFAR-10 of competitive techniques to train a single surrogate model. Adversarial examples evaluated on nine targets with a maximum perturbation L_∞ norm ε of 2/255. Bold is best. In %.

	Target								
Surrogate	RN18	RN50	RN101	DN161	DN201	VGG13	VGG19	IncV3	WRN28
Fully Trained SGD	24.2	44.7	35.6	33.3	31.4	9.6	9.2	22.6	30.8
Early Stopped SGD	28.6	46.1	38.6	36.3	34.6	12.7	13.0	27.1	34.9
SAT	19.7	27.3	25.4	20.1	20.3	13.4	13.5	17.6	20.5
RFN (ours)	**45.4**	**67.1**	**60.6**	**58.9**	**55.8**	**20.5**	**19.8**	**45.0**	**54.1**

Table 6.7: Success rate on CIFAR-10 of competitive techniques to train a single surrogate model. Adversarial examples evaluated on nine targets with a maximum perturbation L_∞ norm ε of 8/255. Bold is best. In %.

	Target								
Surrogate	RN18	RN50	RN101	DN161	DN201	VGG13	VGG19	IncV3	WRN28
Fully Trained SGD	88.3	97.4	92.4	93.9	91.4	64.2	60.5	79.3	91.9
Early Stopped SGD	97.8	99.6	98.8	98.9	98.4	89.1	87.5	95.6	98.8
SAT	97.0	98.7	98.0	97.1	96.4	90.2	89.2	93.2	97.1
RFN (ours)	**99.7**	**100.0**	**100.0**	**100.0**	**99.9**	**96.6**	**95.6**	**99.6**	**99.9**

Table 6.8: Success rate on ImageNet of competitive techniques to train a single surrogate model. Adversarial examples evaluated on ten targets with a maximum perturbation L_∞ norm ε of 4/255. Bold is best. In %.

	Target									
Surrogate	RN50	RN152	RNX50	WRN50	VGG19	DN201	IncV1	IncV3	ViT B	SwinS
Fully Trained SGD	44.5	25.2	24.8	27.1	16.2	16.4	9.8	8.0	1.8	3.3
Early Stopped SGD	51.5	27.4	27.7	28.0	18.4	18.7	10.8	10.4	2.2	2.7
LGV-SWA	82.5	56.8	58.5	54.0	40.9	42.4	28.3	15.1	3.1	5.7
SAT	76.3	62.5	66.8	63.4	48.1	**59.0**	**47.9**	**40.8**	**17.4**	**16.8**
RFN **(ours)**	**85.7**	**70.3**	**73.3**	**73.2**	**58.2**	55.6	37.9	20.5	4.0	8.2

Table 6.9: Success rate on ImageNet of competitive techniques to train a single surrogate model. Adversarial examples evaluated on ten targets with a maximum perturbation L_∞ norm ε of 2/255. Bold is best. In %.

	Target									
Surrogate	RN50	RN152	RNX50	WRN50	VGG19	DN201	IncV1	IncV3	ViT B	SwinS
Fully Trained SGD	18.7	9.4	10.0	9.3	7.6	5.8	4.8	5.2	1.1	1.4
Early Stopped SGD	23.8	10.7	10.6	10.6	8.7	6.8	5.6	6.1	1.1	1.5
LGV-SWA	49.3	24.8	25.0	21.7	18.5	16.8	11.6	7.9	1.4	1.5
SAT	30.0	19.2	24.4	20.6	18.4	20.2	**20.0**	**16.6**	**4.9**	**4.4**
RFN (ours)	**53.3**	**34.3**	**37.5**	**38.3**	**30.7**	**25.0**	16.6	10.8	1.7	3.8

Table 6.10: Success rate on ImageNet of competitive techniques to train a single surrogate model. Adversarial examples evaluated on ten targets with a maximum perturbation L_∞ norm ε of 8/255. Bold is best. In %.

	Target									
Surrogate	RN50	RN152	RNX50	WRN50	VGG19	DN201	IncV1	IncV3	ViT B	SwinS
Fully Trained SGD	77.5	52.9	51.1	55.0	33.4	36.9	21.1	15.2	3.7	6.7
Early Stopped SGD	82.0	56.8	54.6	59.2	35.9	41.1	24.8	18.3	3.6	5.9
LGV-SWA	96.9	87.7	87.1	84.9	65.4	72.8	56.8	31.2	7.0	12.3
SAT	95.4	92.6	93.0	92.8	79.0	**90.1**	**79.1**	**66.3**	**38.5**	**39.1**
RFN (ours)	**97.6**	**92.8**	**93.8**	**95.3**	**83.2**	85.5	71.2	42.3	9.1	19.0

Table 6.11: Success rate on ImageNet of three complementary categories of transferability techniques evaluated on ten targets with a maximum perturbation L_∞ norm ε of 4/255. Dagger (†) is worse when combined with RFN. In %.

	Target									
Attack	RN50	RN152	RNX50	WRN50	VGG19	DN201	IncV1	IncV3	ViT B	SwinS
Model Augmentation Techniques										
GN	68.0	43.1	41.3	44.1	24.8	27.2	14.3	9.9	1.9	3.8
GN+RFN	89.6	76.6	79.4	79.9	65.7	60.3	42.2	22.4	3.8	7.8
SGM	62.8	40.6	41.5	43.5	31.9	28.0	19.3	13.2	4.1	7.9
SGM+RFN	83.2	68.7	71.5	73.0	67.0	56.2	48.9	26.6	6.2	13.6
LGV	93.3	78.1	75.3	73.1	64.4	61.6	49.3	28.8	5.0	6.5
LGV+RFN	†88.7	†74.3	75.7	75.7	70.3	61.9	56.8	31.5	†4.5	7.3
Data Augmentation Techniques										
DI	83.1	60.5	68.1	67.3	45.4	57.9	41.4	30.7	5.7	9.9
DI+RFN	95.0	89.7	90.7	91.6	85.3	87.8	87.5	64.2	14.2	19.0
SI	60.0	37.9	37.3	40.0	23.9	30.0	19.6	13.5	2.6	3.8
SI+RFN	89.2	76.6	80.1	79.1	65.2	69.8	58.0	35.8	5.0	8.5
VT	58.6	35.0	35.2	38.5	23.9	24.7	14.9	11.0	2.3	4.9
VT+RFN	92.0	81.2	82.4	82.9	72.3	72.3	56.7	33.6	7.0	13.5
Attack Optimizers										
MI	56.8	37.4	37.5	38.9	27.0	29.3	18.4	14.6	3.5	4.8
MI+RFN	89.4	79.3	80.4	80.8	71.5	71.1	60.1	39.3	8.5	15.2
NI	53.7	33.1	32.9	35.1	20.5	20.8	12.2	9.4	1.8	3.9
NI+RFN	83.9	67.3	69.8	71.4	56.1	52.5	35.6	17.6	3.8	7.0

Table 6.12: Success rate on ImageNet of three complementary categories of transferability techniques evaluated on ten targets with a maximum perturbation L_∞ norm ε of 2/255. Dagger (†) is worse when combined with RFN. In %.

	Target									
Attack	RN50	RN152	RNX50	WRN50	VGG19	DN201	IncV1	IncV3	ViT B	SwinS
Model Augmentation Techniques										
GN	34.6	17.9	17.4	18.0	12.7	10.4	8.1	6.3	1.3	2.0
GN+RFN	59.7	42.2	42.8	45.4	35.4	29.1	18.3	11.0	1.8	2.4
SGM	26.9	14.9	15.2	15.8	15.5	9.7	7.4	6.6	1.6	3.6
SGM+RFN	46.3	32.0	33.8	35.5	33.4	21.8	20.3	11.9	2.7	5.6
LGV	59.8	33.0	32.9	28.4	31.1	24.2	21.3	12.5	2.4	2.6
RFN+LGV	†50.7	†31.3	32.9	31.4	33.5	27.9	25.0	14.3	†2.1	†2.5
Data Augmentation Techniques										
DI	46.1	27.2	30.9	30.3	22.4	24.8	17.8	15.0	2.5	4.1
DI+RFN	66.6	49.5	57.1	52.3	54.1	49.3	47.5	31.8	4.4	6.9
SI	26.2	14.2	14.3	13.3	10.4	11.3	8.4	7.3	0.9	1.4
SI+RFN	56.5	37.9	42.9	41.2	33.0	31.4	25.0	14.7	2.1	2.9
VT	26.5	14.4	14.1	13.6	10.8	10.1	6.1	6.3	1.3	2.2
VT+RFN	61.5	43.0	47.0	47.4	39.4	35.9	24.3	13.3	2.1	4.9
Attack Optimizers										
MI	29.8	15.9	16.4	16.2	12.6	11.5	7.7	8.0	1.9	2.7
MI+RFN	58.2	41.5	45.4	44.6	39.8	35.8	28.9	17.4	2.8	5.4
NI	21.1	11.0	10.9	11.2	8.4	6.9	5.0	5.2	1.3	1.7
NI+RFN	44.1	28.5	30.7	32.0	25.9	19.6	11.9	9.1	1.3	2.4

Table 6.13: Success rate on ImageNet of three complementary categories of transferability techniques evaluated on ten targets with a maximum perturbation L_∞ norm ε of 8/255. Dagger (†) is worse when combined with RFN. In %.

	Target									
Attack	RN50	RN152	RNX50	WRN50	VGG19	DN201	IncV1	IncV3	ViT B	SwinS
Model Augmentation Techniques										
GN	92.0	73.3	69.7	74.5	45.8	50.4	29.8	19.2	3.2	7.1
GN+RFN	98.2	96.5	96.5	97.4	87.3	88.3	74.4	42.9	9.0	19.4
SGM	91.2	78.4	76.2	79.2	65.1	59.7	48.2	29.1	8.9	19.6
SGM+RFN	97.3	95.1	96.4	96.5	91.5	88.7	84.8	59.8	18.9	32.8
LGV	99.6	97.4	95.9	95.7	87.7	91.7	79.9	47.9	8.9	16.4
LGV+RFN	†99.0	†96.5	96.2	96.7	90.7	†91.0	85.7	53.8	9.5	17.7
Data Augmentation Techniques										
DI	96.1	90.7	91.8	91.4	74.1	88.1	72.4	55.0	14.2	20.4
DI+RFN	99.8	99.6	99.5	99.7	98.6	99.3	98.4	90.4	34.7	48.7
SI	90.4	70.0	69.9	71.9	47.8	60.2	42.6	29.3	6.5	10.3
SI+RFN	98.9	97.3	97.3	98.0	89.9	94.8	90.1	67.2	15.0	23.6
VT	79.6	62.8	61.1	63.5	41.9	48.4	32.2	23.3	6.3	10.5
VT+RFN	98.0	96.7	96.1	97.3	92.9	93.1	87.1	64.2	20.0	39.2
Attack Optimizers										
MI	83.3	60.9	63.3	64.3	48.7	53.8	39.2	30.4	7.4	11.7
MI+RFN	98.5	96.3	96.7	97.1	91.9	92.7	88.1	68.6	21.3	31.5
NI	86.2	65.3	65.1	70.3	43.6	47.1	28.7	19.6	4.7	8.3
NI+RFN	97.9	94.0	95.0	96.0	87.3	86.2	74.2	42.6	10.7	21.0

6.7 Conclusion

Overall, our insights into the behaviour of SGD through the lens of transferability drive us to a new successful approach to train better surrogate representations with limited computational overhead. Our observations lead us to reject the hypothesis that early stopping benefits transferability due to an inherent trade-off between robust and non-robust features. Instead, we explain the success of early stopping in relation to the dynamics of the exploration of the loss landscape. After the learning rate decays, SGD drives down the valley and progressively falls into deep, sharp holes. These fully trained representations are too specific to generate highly transferable adversarial examples. We remediate this issue by explicitly minimizing sharpness in an unusually large neighbourhood. Avoiding those large sharp holes proves to be useful in improving transferability. Finally, we propose RFN, a competitive technique to train a surrogate model that nicely complements other existing transferability techniques.

This chapter provides new evidence in favour of the surrogate-target misalignment hypothesis developed in Chapter 5. LGV explores flatter regions of the loss than where it starts. This flattening is not explicit in LGV, since it is a by-product of SGD. In Chapter 5, we apply LGV from a regularly trained model. Therefore, this chapter closes the question left open by Chapter 5 about finding a better base model by *explicitly* minimizing sharpness. This chapter shows that we can indeed find a good surrogate representation, i.e, a single model, by minimizing sharpness over large neighbourhoods.

7

Conclusion

This chapter presents the overall conclusion of the dissertation, its limitations and proposes potential research directions.

Contents

Table 7.1: Summary of the contributions of this dissertation.

Chapter	Transferability from...	Weight Space Exploration	What Matters	Analysis
4	Deep Ensemble & Bayesian Neural Networks	*Multimodal*: multiple models from different vicinities	Distribution of the target model from the *training noise*	Probabilistic
5	Large Geometric Vicinity (LGV)	*Local*: multiple models from the same vicinity	Weight subspace from the *training noise* & *loss flatness*	Geometric
6	Representation in Flat Neighbourhood (RFN)	*Point*: single model	*Loss flatness*	Geometric

7.1 Summary of Contributions

This dissertation extensively studies how to leverage training techniques to obtain effective surrogate models for black-box attacks. We propose and analyse in-depth new techniques to better understand the relationship between the surrogate weight space and the transferability of the crafted adversarial examples. First, we identify five challenges in the scientific literature about transferability, including the blind spot on how to train surrogate models. To address these challenges, we explore three complementary ways to explore the surrogate weight space: the multimodal exploration to obtain multiple models from different vicinities, the local exploration to obtain multiple models in the same vicinity, and the point selection to obtain a single transferable representation. Our combined techniques show how to obtain a surrogate composed of better individual representations from different vicinities, which are well characterised locally.

The primary goal of this dissertation is to provide novel insights, which represents a significant portion of our research effort. Our probabilistic and geometric perspectives exhibit the importance of the training noise and of the flatness of the loss, to improve the transferability of adversarial examples from the weight space. Table 7.1 summarises the contributions of this dissertation.

Our first contribution develops a probabilistic perspective of transferability and shows that transferability fundamentally relates to uncertainty. As the weights of the target DNN are unknown, they can be treated as random variables. Under a specified threat model, the randomness of the target DNN comes from the training

noise. Deep ensemble can generate an effective surrogate model, since this technique samples weights from the distribution of the target model. Unfortunately, deep ensemble is computationally expensive. We propose an efficient alternative by sampling surrogate models from the posterior distribution using cSGLD, a state-of-the-art Bayesian deep learning technique. Our extensive experiments on ImageNet, CIFAR-10 and MNIST show that our approach improves the success rates of four state-of-the-art attacks significantly (up to 83.2 percentage points), in both intra-architecture and inter-architecture transferability. On ImageNet, our approach can reach 94% of success rate while reducing training computations from 11.6 to 2.4 exaflops, compared to deep ensemble. Our vanilla surrogate achieves 87.5% of the time higher transferability than three transferability techniques designed for this purpose. We evaluate seven training methods in total to train a surrogate model. This contribution was the first to investigate how to generate a surrogate from the weight space of a single architecture. Our experiments also reveal that deep ensemble and cSGLD poorly characterise the local variations in the weight space. We identify a promising direction to address this shortfall using a training technique that builds an ensemble from fine-tuning a trained DNN. Our work suggests new hybrid transferability techniques in-between training time and attack time by augmenting a surrogate model using on training. Our next contribution follows this direction.

Second, we start by establishing the relevance of the local exploration of the weight space for transferability by showing that Gaussian noise in the input space does not improve transferability, while Gaussian noise in the weight space does. Both noises differ in the structure of the covariance matrix of the induced Gaussian distribution of input gradients. To improve over random noise in the weight space, we propose transferability from Large Geometric Vicinity (LGV), a new model augmentation technique based on the local exploration of the weight space with SGD. LGV starts from a pretrained model and collects multiple weights in a few additional training epochs with a constant and high learning rate. LGV exploits two geometric properties that we relate to transferability. First, we show that LGV explores a flatter region of the weight space and generates flatter adversarial examples in the input space. We present the surrogate-target misalignment hypothesis to explain why flatness could increase transferability: if the surrogate loss is shifted with respect to the target loss in the input space, wide adversarial examples are desirable to keep the difference between the surrogate and the target losses small. Second, we show that the LGV weights span a dense weight subspace whose geometry is intrinsically connected to transferability. This small subspace is particularly effective because the training noise of SGD concentrates in a tiny weight subspace. Through extensive experiments, we show that LGV alone outperforms all (combinations of) four established transferability techniques by 1.8 to 59.9

percentage points. Although LGV addresses how to augment a regularly trained DNN, training a single base model for transferability is an open challenge that requires further investigation. As suggested by our analysis of the flatness of the loss, explicitly minimizing sharpness during the training of the surrogate model might solve both this issue and give further additional evidence in favour of our surrogate-target misalignment hypothesis. Our next contribution explores this direction in-depth.

Third, we investigate how to train a transferable representation, that is, a single model for transferability. First, we refute a common hypothesis from previous research to explain why early stopping improves transferability. Previous work proposes the hypothesis that DNN first learns robust features and then non-robust features, explaining why early stopped models are better surrogates, since non-robust features are brittle. However, we provide evidence against this hypothesis, showing that early stopping improves transferability from and to non-robust features. We hypothesize that robust and non-robust features are learnt conjointly during training, since the transferability of adversarial examples can provide insights on the closeness of two learnt representations. We propose an alternative explanation to the transferability of early stopping by establishing links between transferability and the exploration dynamics of the weight space, in which early stopping has an inherent effect. More precisely, we observe that transferability peaks when the learning rate decays, which is also the time at which the sharpness of the loss significantly drops. SGD drives down the valley and progressively falls into deep and sharp holes, where the representations are too specific. This leads us to propose RFN, a new approach to transferability that minimises the sharpness of the loss during training. We show that, by searching for large flat neighbourhoods, RFN always improves over early stopping (by up to 47 points of success rate) and is competitive with (if not better than) strong state-of-the-art baselines. RFN also complements nicely complementary transferability techniques of other categories, established by our taxonomy of transferability techniques.

Overall, our three complementary techniques provide an extensive and practical method to obtain highly transferable adversarial examples from the multimodal and local exploration of flatter vicinities in the weight space. Our probabilistic and geometric approaches demonstrate that the way to train the surrogate model has been overlooked, although both the training noise and the flatness of the loss landscape are important elements of transfer-based attacks. We hope that these new insights would help to better understand the phenomenon of the transferability of adversarial examples.

7.2 Limitations and Perspectives

Resulting from our contributions, we identify the following limitations and directions for future work.

Transferability and epistemic or aleatoric uncertainties. A promising venue is to explore how different threat models affect the type of uncertainty. The threat model studied in Chapter 4 is mainly related to uncertainty in parameter estimation, originating from the randomness of SGD. Numerous variations in the experimental settings can impact the uncertainty. The ignorance of the training dataset would increase the aleatoric uncertainty. In the case of an unknown dataset, distribution shift may appear when the attacker collects a surrogate training dataset. Some categories of distributional shifts may increase the epistemic uncertainty, while other categories may increase the aleatoric uncertainty. Adding a defence such as random input transformation [XZY+18] would increase the epistemic uncertainty if its presence is unknown, and the aleatoric uncertainty through its randomness.

Bayesian perspectives on our geometrical approaches. We believe that our geometrical approaches developed in Chapters 5 and 6 could be extended to connect to our probabilistic approach developed in Chapter 4. Mandt et al. [MHB17] show that SGD with a constant learning rate can be used as an approximate Bayesian posterior inference algorithm. Therefore, LGV may sample weights from the posterior distribution conditioned by the vicinity of the fully trained DNN where LGV starts. A particularly interesting application would be to automatically choose or adapt the hyperparameters of LGV. Mandt et al. [MHB17] derive analytically the stationary distribution of the Markov chain of SGD with constant learning rate. The analytical expression of the stationary distribution is parameterized by the learning rate, the batch size, the momentum decay factor, and a preconditioning matrix. Therefore, dependences between LGV hyperparameters could be derived from this stationary distribution and used, for example, to adapt the learning rate or the momentum decay factor when the batch size has to be lowered due to memory limitations. Similarly, for Chapter 6, a Bayesian interpretation of RFN could be developed, based on the recent work of Möllenhoff and Khan [MK22] who show that SAM is an optimal relaxation of the Bayes objective. We left for future work the explicit development of these two connections.

Controversy about the loss flatness. As presented in detail in Section 3.2.4, the link between the flatness of the loss and natural generalisation is subject to a scientific controversy. Our Chapters 5 and 6 show strong evidence that better surrogate models can be obtained from flatter regions of the loss landscape. Despite our best attempt at finding alternative hypotheses from the other side of the controversy, we found no consistent experimental evidence in favour of an alternative hypothesis to our surrogate-target misalignment hypothesis. Section 3.2.4 describes

in detail our preliminary experiments that followed the contributions of Chapter 5 to prepare for Chapter 6, exploring an alternative hypothesis about transferability and the surrogate function complexity. Future research may consider further studying the relationship between regularization and transferability to better understand why some explicit or implicit regularization schemes improve transferability while others do not. Given the speed of deep learning research, we strongly believe that important new insights about transferability from the weight space will emerge from the evolution of this controversy. Therefore, we recommend follow-up research to consider new developments of hypotheses related to flatness, and to not take our surrogate-target misalignment hypothesis for granted.

Extensions to unusual experimental settings. Numerous new directions might emerge from reconsidering the elements of the experimental settings that are usually considered to study transferability. Future research may evaluate transferability in the physical domain to find to what extent the improvement of success rate from our exploration of the weight space survives more destructive threat models. Evaluation in the physical domain may consist, for example, of printing adversarial examples and then capturing with a camera [KGB17]. Moreover, we evaluate transferability to some ViT targets (from CNN surrogates), and observe that the success rates of our techniques are higher than baselines, but still quite low. We think that the gap of transferability between CNNs and ViT targets are related to the important differences between the surrogate and the target architectures in that case. The evaluation of ViT surrogates would be interesting to study, in order to confirm that our conclusions about the surrogate weight space of CNNs hold for the surrogate weight space of ViT architectures, which may behave differently. Another element of the experimental settings that is worth revisiting is the domain of application: for example, the transferability of adversarial examples against a natural language processing classifiers is mostly unexplored. A major advantage of the approach developed in this dissertation, i.e., improving transferability from the weight space, is its domain invariance, unlike other types of transferability techniques such as input augmentations. Finally, interesting new insights might arise from studying the transferability of non-L_p attacks. A growing number of attacks have been developed where the adversarial perturbation is not bounded by a L_p norm. Studying the transferability of these attacks might lead to more realistic threat models and broader applications.

Abbreviations

AI Artificial Intelligence.

BNN Bayesian Neural Network.

CNN Convolutional Neural Network.
cSGLD Cyclical Stochastic Gradient Langevin Dynamics.

DI Input Diversity.
DL Deep Learning.
DNN Deep Neural Network.

ERM Empirical Risk Minimization.

FC Fully Connected.
FGE Fast Geometric Ensembling.
FGSM Fast Gradient Sign Method.

GN Ghost Networks.

HMC Hamiltonian Monte Carlo.

I-FGSM Iterative Fast Gradient Sign Method.

LGV Large Geometric Vicinity.

MCMC Markov Chain Monte Carlo.
MI Momentum Iterative attack.
ML Machine Learning.

NI Nesterov Iterative attack.

PGD Projected Gradient Descent.

RFN Representation from Flat Neighbourhood.

SAM Sharpness-aware Minimizer.

SG-MCMC Stochastic Gradient-Markov Chain Monte Carlo.
SGD Stochastic Gradient Descent.
SGLD Stochastic Gradient Langevin Dynamics.
SGM Skip Gradient Method.
SI Subspace Inference.
SSE Snapshot Ensembles.
SWA Stochastic Weight Averaging.
SWAG Stochastic Weight Averaging-Gaussian.

TI Translation Invariance.

VI Variational Inference.
ViT Vision Transformer.

List of Publications, Tools and Services

Papers included in the dissertation.

- Martin Gubri et al. Efficient and Transferable Adversarial Examples from Bayesian Neural Networks. In *UAI 2022*, 2022. URL: `https://gubri.eu/publication/transferable_adv_ex_from_bnn/`
- Martin Gubri et al. LGV: Boosting Adversarial Example Transferability from Large Geometric Vicinity. In *ECCV 2022*, 2022. URL: `https://gubri.eu/publication/lgv_eccv22/`
- Martin Gubri et al. Going Further: Flatness at the Rescue of Early Stopping for Adversarial Example Transferability, April 2023. URL: `https://gubri.eu/publication/rfn_flatness_transferability/`

The following sections are *new or updated materials* which were not included as such in the published papers.

- The entire Chapter 1 is new, including the evidence represented by the Figure 1.1, and the new illustration of Figure 1.3 for clarity.
- The entire Chapter 2 is also new.
- A large portion of Chapter 3 is new to update the related work. We added the discussion of several related papers published after or in parallel to ours, for examples Qin et al. [QFL+22], Li et al. [LGZ+23], and Möllenhoff and Khan [MK22]. We also added some missing related papers, in particular the discussions of the technique, hypothesis, and analysis of Wu et al. [WZT+18] and of Fort et al. [FHL19].
- We added an extensive discussion of the scientific controversy regarding the flatness of the loss in Section 3.2.4.
- We integrated in the main text all the content originally in the appendices when relevant, with additional new comments when needed (Chapters 4 to 6).
- We rewrote the introductions and conclusions of Chapters 4 to 6 to have a coherent, logical flow between chapters.
- We added the entire Section 4.4.1 and a paragraph in Section 4.2 to clarify our contribution regarding deep ensemble.
- We added the entire Section 4.4.5 to discuss the multimodal exploration of

the loss landscape and make the transition between Chapter 4 and Chapter 5 more explicit.
- We updated the Section 5.3.1 to discuss the related work (Wu et al. [WZT+18]).
- We added the Section 5.4.4 to comment the settings of LGV.
- We added the Section 5.4.3 to show the complementarity of deep ensemble (Chapter 4) and LGV (Chapter 5).
- We added the new paragraph titled "Optimal lengths of random and LGV deviations vectors" in Section 5.5, and added more structure to this section.
- We restructured and partially rewrote Section 5.6 to clarify.
- We added the Table 5.5 to ease the comparison between noise in the input space and noise in the weight space.
- Finally, the entire Chapter 7 is new.

Papers not included in the dissertation.

- Martin Gubri. Adversarial Perturbation Intensity Achieving Chosen Intra-Technique Transferability Level for Logistic Regression, January 2018. URL: http://arxiv.org/abs/1801.01953
- Salah Ghamizi et al. Search-Based Adversarial Testing and Improvement of Constrained Credit Scoring Systems. In *FSE 2020*, ESEC/FSE 2020, 2020. ISBN: 9781450370431. DOI: 10.1145/3368089.3409739
- Adriano Franci et al. Influence-Driven Data Poisoning in Graph-Based Semi-Supervised Classifiers. *CAIN 2022*:77–87, 2022. DOI: 10.1145/3522664.3528606. URL: https://doi.org/10.1145/3522664.3528606

Tools.

- Torchattacks library: Refactor, merge and showcase LGV code (method proposed in Chapter 5).
  - Refactoring of LGV code to distribute a clean and easy-to-use implementation in the widely used library torchattacks library. The LGV class is a wrapper that is compatible with any torchattacks attack to be applied on top.
  - A complete and runnable notebook to showcase the usage, using our LGV models which we publicly distribute.
  - Pull-request: https://github.com/Harry24k/adversarial-attacks-pytorch/pull/91
  - Demonstration notebook: https://github.com/Harry24k/adversarial-attacks-pytorch/blob/c4da6a95546283992a3d1816ae76a0cd4dfc2d8b/demo/Transfer%20Attack%20combined%20with%20LGV%20on%20ImageNet.ipynb
  - Diverse contributions and bug reports: https://github.com/Harry24k/adversarial-attacks-pytorch/issues?q=author%3AFramartin
- Source code of experiments in Chapter 4.
  - https://github.com/Framartin/transferable-bnn-adv-ex

- Source code of experiments in Chapter 5.
  - `https://github.com/Framartin/lgv-geometric-transferability`
- Source code of experiments in Chapter 6.
  - `https://github.com/Framartin/rfn-flatness-transferability`
- ART library: Fix uniform random sampling in the L_1 ball.
  - Implement and merge a new sampling method uniform at random in the L1 ball to fix the "art.utils.random_sphere()" function.
  - Add new unit tests for all supported norms to check that the norm of the point in high dimension is close to radius, check a smaller radius than default, and check the shape of all norms.
  - Fix the docstring of "art.utils.random_sphere()" to specify that the function samples uniformly at random in the p-norm ball, and not on the surface of the sphere.
  - Pull-request: `https://github.com/Trusted-AI/adversarial-robustness-toolbox/pull/1802`
  - Bug report: `https://github.com/Trusted-AI/adversarial-robustness-toolbox/issues/1799`

Reviewing. I served as a (co)-reviewer for the following conferences and journals.

- Machine Learning Venues: AAAI 2021, AAAI 2022, NeurIPS 2022 (Datasets and Benchmarks Track), IEEE Transactions on Image Processing (journal), CVPR 2023, UAI 2023, IJCAI 2023, SiMLA 2023 (workshop)
- Software Engineering Venues: ICSE 2021, ICSE 2022, FSE 2020, FSE 2022, ICST 2020, ICST 2021, QRS 2020, QRS 2022, SANER 2023

Teaching. I taught the following courses at the University of Luxembourg.

- Advanced topics in Applied Machine Learning (Master, 2022): 2nd year of Master in Computer Science. Two lectures, design and correction of the project, planning.
  1. Recalls of machine learning, machine learning frameworks, first part project
  2. Overview of adversarial machine learning, model calibration
- Introduction to Machine Learning (Master, 2022): 2^{nd} year of Master in Space Science. Six sessions, exam writing.
  1. Recalls of linear algebra
  2. Zero-order optimization
  3. First-order optimization
  4. Linear regression and linear classification
  5. ML project lifecycle: Data preparation, feature engineering, overfitting & underfitting, model evaluation

6. Neural Networks, Keras & Convolutional Neural Nets, and Advanced Neural Networks
7. Written examination

- Introduction to Machine Learning (Master, 2021): 2nd year of Master in Space Science. Two introductory lectures on Machine Learning.
- Software engineering 2 (Bachelor, 2020): 3rd year of Bachelor in Computer Science. Four introductory lectures on Machine Learning Engineering. Online sessions given online during lockdown. Quizzes on Moodle.

1. Introduction to Machine Learning: Useful Definitions, Types of Tasks in Machine Learning
2. Introduction to Machine Learning: Recalls of Statistics, Models elements, Elements of Statistical Learning Theory
3. Machine Learning Project Lifecycle: When to (not) use Machine Learning, Goal Definition, Data Collection & Preparation
4. Machine Learning Project Lifecycle: Feature Engineering, Choosing and Training a model, Model Evaluation, Feedback loop

Reading Group. I organized and animated the weekly Reading Group on Machine Learning at the SerVal research group, since February 2021, including several invited talks.

List of Figures

List of Tables

Bibliography

[ACM+23] Maksym Andriushchenko, Francesco Croce, Maximilian Müller, Matthias Hein, and Nicolas Flammarion. A modern look at the relationship between sharpness and generalization, February 2023. URL: https://arxiv.org/abs/2302.07011v1 (cited on page 50).

[ACW18] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. *ICML*, February 2018 (cited on pages 28, 73).

[AF22] Maksym Andriushchenko and Nicolas Flammarion. Towards Understanding Sharpness-Aware Minimization, June 2022. URL: https://arxiv.org/abs/2206.06232v1 (cited on page 50).

[AJS+22] Sravanti Addepalli, Samyak Jain, Gaurang Sriramanan, and R. Venkatesh Babu. Scaling Adversarial Training toăLarge Perturbation Bounds. In *ECCV 2022*. Springer Science and Business Media Deutschland GmbH, October 2022 (cited on page 7).

[ALM+20] Arsenii Ashukha, Alexander Lyzhov, Dmitry Molchanov, and Dmitry Vetrov. Pitfalls of In-Domain Uncertainty Estimation and Ensembling in Deep Learning. *ICLR*, February 2020. ISSN: 2331-8422. URL: http://arxiv.org/abs/2002.06470 (cited on pages 52, 53, 66–68, 73, 95, 96, 141).

[AVP+22] Maksym Andriushchenko, Aditya Varre, Loucas Pillaud, Vivien Epfl, and Nicolas Flammarion. SGD with large step sizes learns sparse features, October 2022. DOI: 10.48550/arxiv.2210.05337. URL: https://arxiv.org/abs/2210.05337v1 (cited on pages 50, 51).

[BCM+13] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim rndi, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Lecture Notes in Computer Science*, number PART 3, pages 387–402, August 2013. DOI: 10.1007/978-3-642-40994-3{_}25 (cited on pages 2, 3, 28, 32).

[BCN+14] Battista Biggio, Igino Corona, Blaine Nelson, Benjamin I. P. Rubinstein, Davide Maiorca, Giorgio Fumera, Giorgio Giacinto, Roli, and Fabio. Security Evaluation of Support Vector Machines in Adversarial Environments, January 2014 (cited on pages 31, 32).

[BNJ+10] Marco Barreno, Blaine Nelson, Anthony D. Joseph, and J. D. Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, November 2010. ISSN: 08856125. DOI: 10.1007/S10994-010-5188-5/METRICS. URL: https://link.springer.com/article/10.1007/s10994-010-5188-5 (cited on page 31).

[BNL12] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning Attacks against Support Vector Machines. *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, 2:1807–1814, June 2012. URL: https://arxiv.org/abs/1206.6389v3 (cited on page 31).

[BNS+06] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D. Joseph, and J. D. Tygar. Can machine learning be secure? *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security, ASIACCS '06*, 2006:16–25, 2006. DOI: 10.1145/1128817.1128824. URL: https://dl.acm.org/doi/10.1145/1128817.1128824 (cited on page 31).

[BR18] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, December 2018. ISSN: 00313203. DOI: 10.1016/j.patcog.2018.07.023. URL: http://arxiv.org/abs/1712.03141%20http://dx.doi.org/10.1016/j.patcog.2018.07.023 (cited on pages 28, 29).

[BZK21] Philipp Benz, Chaoning Zhang, and In So Kweon. Batch Normalization Increases Adversarial Vulnerability and Decreases Adversarial Transferability: A Non-Robust Feature Perspective. In *ICCV 2021*, October 2021. DOI: 10.1109/ICCV48922.2021.00772. URL: http://arxiv.org/abs/2010.03316 (cited on pages 6, 38, 44–46, 139, 142, 143).

[CAP+19] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On Evaluating Adversarial Robustness, February 2019. URL: https://arxiv.org/abs/1902.06705v2 (cited on pages 7, 28, 35).

[CCS+16] Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-SGD: Biasing Gradient Descent Into Wide Valleys. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, November 2016. DOI: 10.48550/arxiv.1611.01838. URL: https://arxiv.org/abs/1611.01838v5 (cited on page 49).

[CH20] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, volume PartF16814, pages 2184–2194, March 2020. ISBN: 9781713821120 (cited on page 68).

[CHM+14] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The Loss Surfaces of Multilayer Networks. *Journal of Machine Learning Research*, 38:192–204, November 2014. ISSN: 15337928. URL: https://arxiv.org/abs/1412.0233v3 (cited on page 9).

[CLL+17] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning, December 2017. URL: https://arxiv.org/abs/1712.05526v1 (cited on page 31).

[CRP20] Zachary Charles, Harrison Rosenberg, and Dimitris Papailiopoulos. A geometric perspective on the transferability of adversarial directions. In *AISTATS 2019*, November 2020. URL: http://arxiv.org/abs/1811.03531 (cited on pages 46, 95).

[CWL+20] Ginevra Carbone, Matthew Wicker, Luca Laurenti, Andrea Patane, Luca Bortolussi, and Guido Sanguinetti. Robustness of Bayesian neural networks to gradient-based attacks. In *NeurIPS*. arXiv, February 2020 (cited on pages 52–54).

[DKA+19] Shaveta Dargan, Munish Kumar, Maruthi Rohit Ayyagari, and Gulshan Kumar. A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning. *Archives of Computational Methods in Engineering*, 27(4):1071–1092, September 2019 (cited on page 2).

[DLP+18] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting Adversarial Attacks with Momentum. In *CVPR*, pages 9185–9193, October 2018. ISBN: 9781538664209. DOI: 10.1109/CVPR.2018.00957. URL: http://arxiv.org/abs/1710.06081 (cited on pages 6, 39, 75, 80, 104, 106, 142).

[DPB+17] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp Minima Can Generalize For Deep Nets. *34th International Conference on Machine Learning, ICML 2017*, 3:1705–1714, March 2017. URL: https://arxiv.org/abs/1703.04933v2 (cited on page 50).

[DPS+19] Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *CVPR*, pages 4307–4316, April 2019 (cited on pages 6, 8, 37).

[EIS+19] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, and Dimitris Tsipras. Robustness (Python Library), 2019. URL: https://github.com/MadryLab/robustness (cited on pages 141, 146).

[FC19] Jonathan Frankle and Michael Carbin. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. *ICLR*, 2019 (cited on page 61).

[FCG+22] Adriano Franci, Maxime Cordy, Martin Gubri, Mike Papadakis, and Yves Le Traon. Influence-Driven Data Poisoning in Graph-Based Semi-Supervised Classifiers. *CAIN 2022*:77–87, 2022. DOI: 10.1145/3522664.3528606. URL: https://doi.org/10.1145/3522664.3528606 (cited on page iv).

[FHL19] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep Ensembles: A Loss Landscape Perspective, December 2019. URL: https://arxiv.org/abs/1912.02757v2 (cited on pages 10, 43, 44, 47, 91, iii).

[FJR15] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 1322–1333, New York, NY, USA. Association for Computing Machinery, 2015. ISBN: 9781450338325. DOI: 10.1145/2810103.2813677. URL: https://doi.org/10.1145/2810103.2813677 (cited on page 31).

[FKM+20] Pierre Foret, Ariel Kleiner Google Research, Hossein Mobahi Google Research, and Behnam Neyshabur Blueshift. Sharpness-Aware Minimization for Efficiently Improving Generalization, October 2020. DOI: 10.48550/arxiv.2010.01412. URL: https://arxiv.org/abs/2010.01412v3 (cited on pages 49, 50, 95, 153, 155, 158).

[FLL+22] Shuman Fang, Jie Li, Xianming Lin, and Rongrong Ji. Learning to Learn Transferable Attack. *Proceedings of the 36th AAAI Conference on Artificial Intelligence, AAAI 2022*, 36:571–579, June 2022. ISSN: 2159-5399. DOI: 10.1609/AAAI.V36I1.19936. URL: https://arxiv.org/abs/2112.06658v1 (cited on page 40).

[FW12] Nial Friel and Jason Wyse. Estimating the evidence - a review. *Statistica Neerlandica*, 66(3):288–308, August 2012. ISSN: 00390402 (cited on page 64).

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016 (cited on page 50).

[GCG+20] Salah Ghamizi, Maxime Cordy, Martin Gubri, Mike Papadakis, Andrey Boystov, Yves Le Traon, and Anne Goujon. Search-Based Adversarial Testing and Improvement of Constrained Credit Scoring Systems. In *FSE 2020*, ESEC/FSE 2020, 2020. ISBN: 9781450370431. DOI: 10.1145/3368089.3409739 (cited on page iv).

[GCP+22a] Martin Gubri, Maxime Cordy, Mike Papadakis, and Yves Le Traon. Efficient and Transferable Adversarial Examples from Bayesian Neural Networks. In *UAI 2022*, 2022. URL: https://gubri.eu/publication/transferable_adv_ex_from_bnn/ (cited on pages 58, iii).

[GCP+22b] Martin Gubri, Maxime Cordy, Mike Papadakis, Yves Le Traon, and Koushik Sen. LGV: Boosting Adversarial Example Transferability from Large Geometric Vicinity. In *ECCV 2022*, 2022. URL: https://gubri.eu/publication/lgv_eccv22/ (cited on pages 94, iii).

[GCT23] Martin Gubri, Maxime Cordy, and Yves Le Traon. Going Further: Flatness at the Rescue of Early Stopping for Adversarial Example Transferability, April 2023. URL: https://gubri.eu/publication/rfn_flatness_transferability/ (cited on pages 138, iii).

[GDG17] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain, August 2017. URL: https://arxiv.org/abs/1708.06733v2 (cited on page 31).

[GIP+18] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs. *NeurIPS*:8789–8798, February 2018 (cited on pages 10, 52, 54).

[GLC20] Yiwen Guo, Qizhang Li, and Hao Chen. Backpropagating Linearly Improves Transferability of Adversarial Examples. Technical report, 2020. URL: https://github.com/qizhangli/linbp-attack. (cited on pages 6, 8, 38, 51, 94).

[GPS+18] Kathrin Grosse, David Pfaff, Michael Thomas Smith, and Michael Backes. The Limitations of Model Uncertainty in Adversarial Settings. *NeurIPS Workshop*, December 2018 (cited on page 52).

[GRD18] Guy Gur-Ari, Daniel A. Roberts, and Ethan Dyer. Gradient Descent Happens in a Tiny Subspace, December 2018. URL: http://arxiv.org/abs/1812.04754 (cited on pages 10, 46, 95, 128).

[GSS14] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples, December 2014 (cited on pages 31, 33, 35, 59, 60).

[Gub18] Martin Gubri. Adversarial Perturbation Intensity Achieving Chosen Intra-Technique Transferability Level for Logistic Regression, January 2018. URL: http://arxiv.org/abs/1801.01953 (cited on page iv).

[GV14] Ian J. Goodfellow and Oriol Vinyals. Qualitatively characterizing neural network optimization problems. *CoRR*, 2014 (cited on page 9).

[HKG+19] Qian Huang, Isay Katsman, Zeqi Gu, Horace He, Serge Belongie, and Ser Nam Lim. Enhancing adversarial example transferability with an intermediate level attack. In *ICCV*, pages 4732–4741, July 2019. DOI: 10.1109/ICCV.2019.00483. URL: http://arxiv.org/abs/1907.10823 (cited on pages 6, 39).

[HLP+17] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. Snapshot ensembles: Train 1, get M for free. In *ICLR*. International Conference on Learning Representations, ICLR, March 2017. URL: http://arxiv.org/abs/1704.00109 (cited on pages 52, 54).

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Flat Minima. *Neural Computation*, 9(1):1–42, January 1997. ISSN: 0899-7667. DOI: 10.1162/NECO.1997.9.1.1 (cited on pages 48, 116).

[HZR+16a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *ICCV*, pages 770–778. IEEE Computer Society, December 2016. ISBN: 9781467388504. DOI: 10.1109/CVPR.2016.90 (cited on page 66).

[HZR+16b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity Mappings in Deep Residual Networks. *Lecture Notes in Computer Science*, 9908 LNCS:630–645, March 2016 (cited on page 66).

[IMK+19] Pavel Izmailov, Wesley J. Maddox, Polina Kirichenko, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Subspace Inference for Bayesian Deep Learning. *UAI 2019*, July 2019. URL: http://arxiv.org/abs/1907.07504 (cited on pages 54, 97, 98, 128, 130).

[IPG+18] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, volume 2, pages 876–885. Association For Uncertainty in Artificial Intelligence (AUAI), March 2018. ISBN: 9781510871601. URL: http://arxiv.org/abs/1803.05407 (cited on pages 49, 50, 54, 87, 95, 112, 116, 117, 119).

[IST+19] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial Examples Are Not Bugs, They Are Features, May 2019. URL: http://arxiv.org/abs/1905.02175 (cited on pages 46, 144–148).

[IWL+19] Nathan Inkawhich, Wei Wen, Hai Helen Li, and Yiran Chen. Feature space perturbations yield more transferable adversarial examples. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:7059–7067, June 2019. ISSN: 10636919. DOI: 10.1109/CVPR.2019.00723 (cited on page 39).

[KCL22] Simran Kaur, Jeremy Cohen, and Zachary C. Lipton. On the Maximum Hessian Eigenvalue and Generalization, June 2022. URL: https://arxiv.org/abs/2206.10654v2 (cited on page 50).

[KGB17] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *ICLR*, July 2017 (cited on pages 3, 6, 36, 59, 142, 178).

[Kim20] Hoki Kim. Torchattacks: A pytorch repository for adversarial attacks. *arXiv preprint arXiv:2010.01950*, 2020 (cited on page 143).

[KKP+21] Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. ASAM: Adaptive Sharpness-Aware Minimization for Scale-Invariant Learning of Deep Neural Networks, February 2021. DOI: 10.48550/arxiv.2102.11600. URL: https://arxiv.org/abs/2102.11600v3 (cited on pages 49, 50, 156, 157).

[KLS+22] Jean Kaddour, Linqing Liu, Ricardo Silva, and Matt J. Kusner. When Do Flat Minima Optimizers Work? In *NeurIPS 2022*, February 2022. DOI: 10.48550/arxiv.2202.00661. URL: https://arxiv.org/abs/2202.00661v5 (cited on pages 49, 153, 155, 158).

[KNT+16] Nitish Shirish Keskar, Jorge Nocedal, Ping Tak Peter Tang, Dheevatsa Mudigere, and Mikhail Smelyanskiy. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. *ICLR 2017*, September 2016. URL: https://arxiv.org/abs/1609.04836v2 (cited on pages 49, 50, 95, 116).

[Kri09] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images, 2009. ISSN: 1098-6596. DOI: 10.1.1.222.9220 (cited on page 66).

[LBZ+18] Yingwei Li, Song Bai, Yuyin Zhou, Cihang Xie, Zhishuai Zhang, and Alan Yuille. Learning Transferable Adversarial Examples via Ghost Networks. *AAAI*, 34(07), December 2018. ISSN: 2374-3468. DOI: 10.1609/aaai.v34i07.6810. URL: http://arxiv.org/abs/1812.03413 (cited on pages 6, 8, 38, 42–44, 51, 63, 68, 75, 80, 85, 94, 104, 106, 142).

[LCL+17] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *ICLR*, November 2017. URL: https://arxiv.org/abs/1611.02770 (cited on pages 6, 38, 44, 51, 59, 63, 75).

[LFL+18] Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the Intrinsic Dimension of Objective Landscapes. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, April 2018. URL: https://arxiv.org/abs/1804.08838v1 (cited on pages 46, 95).

[LGZ+23] Qizhang Li, Yiwen Guo, Wangmeng Zuo, and Hao Chen. Making Substitute Models More Bayesian Can Enhance Transferability of Adversarial Examples. In *ICLR*, February 2023. URL: http://arxiv.org/abs/2302.05086 (cited on pages 6, 45, iii).

[LPB16] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. *NeurIPS*:6403–6414, 2016 (cited on pages 52, 61, 63).

[LSH+20] Jiadong Lin, Chuanbiao Song, Kun He, Liwei Wang, and John E. Hopcroft. Nesterov Accelerated Gradient and Scale Invariance for Adversarial Attacks. *ICLR*, August 2020. URL: http://arxiv.org/abs/1908.06281 (cited on pages 6, 8, 37, 39, 142).

[LWM19] Yuanzhi Li, Colin Wei, and Tengyu Ma. Towards Explaining the Regularization Effect of Initial Large Learning Rate in Training Neural Networks. *Advances in Neural Information Processing Systems*, 32,

July 2019. ISSN: 10495258. DOI: 10.48550/arxiv.1907.04595. URL: https://arxiv.org/abs/1907.04595v2 (cited on pages 50, 51).

[MBD+17] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C. Lupu, and Fabio Roli. Towards Poisoning of Deep Learning Algorithms with Back-gradient Optimization, August 2017 (cited on page 31).

[MFU+19] Seyed Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Jonathan Uesato, and Pascal Frossard. Robustness via curvature regularization, and vice versa. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:9070–9078, June 2019. ISSN: 10636919. DOI: 10.1109/CVPR.2019.00929. URL: https://arxiv.org/abs/1811.09716v1 (cited on page 51).

[MGI+19a] Wesley J. Maddox, Timur Garipov, Izmailov, Dmitry Vetrov, and Andrew Gordon Wilson. A simple baseline for Bayesian uncertainty in deep learning. In *NeurIPS*, volume 32. arXiv, February 2019. URL: http://arxiv.org/abs/1902.02476 (cited on pages 52–54, 68, 86, 97).

[MGI+19b] Wesley J. Maddox, Timur Garipov, Izmailov, Dmitry Vetrov, and Andrew Gordon Wilson. A simple baseline for Bayesian uncertainty in deep learning. In *NeurIPS*, volume 32. arXiv, February 2019. URL: http://arxiv.org/abs/1902.02476 (cited on page 106).

[MHB17] Stephan Mandt, Matthew D. Hof Fman, and David M. Blei. Stochastic Gradient Descent as Approximate Bayesian Inference. *Journal of Machine Learning Research*, 18:1–35, April 2017. ISSN: 15337928. URL: https://arxiv.org/abs/1704.04289v2 (cited on pages 54, 61, 104, 136, 177).

[Min02] Minka Thomas P. Bayesian model averaging is not model combination. Technical report, 2002 (cited on page 63).

[MK22] Thomas Möllenhoff and Mohammad Emtiyaz Khan. SAM as an Optimal Relaxation of Bayes. In *ICLR 2023*, October 2022. URL: https://arxiv.org/abs/2210.01620v2 (cited on pages 50, 177, iii).

[MMS+18] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, June 2018. URL: http://arxiv.org/abs/1706.06083 (cited on pages 3, 36).

[MSV+19] Chris Mingard, Joar Skalse, Guillermo Valle-Pérez, David Martínez-Rubio, Vladimir Mikulik, and Ard A. Louis. Neural networks are a priori biased towards Boolean functions with low entropy, September 2019. URL: https://arxiv.org/abs/1909.11522v3 (cited on page 50).

[MVS+20] Chris Mingard, Guillermo Valle-Pérez, Joar Skalse, and Ard A. Louis. Is SGD a Bayesian sampler? Well, almost. *Journal of Machine Learning Research*, 22, June 2020. ISSN: 15337928 (cited on pages 11, 61, 88).

[MZ15] Shike Mei and Xiaojin Zhu. Using Machine Teaching to Identify Optimal Training-Set Attacks on Machine Learners. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2871–2877. AAAI Press, 2015. ISBN: 0262511290 (cited on page 31).

[Nea96] Radford M Neal. *Bayesian Learning for Neural Networks*. Springer New York, 1996, page 341. ISBN: 9783642124648 (cited on page 23).

[Nit21] Vikram Nitin. SGD on Neural Networks learns Robust Features before Non-Robust, March 2021 (cited on pages 44–46, 139, 143).

[NKH+21] Muzammal Naseer, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Fatih Porikli. On Generating Transferable Targeted Perturbations. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*:7688–7697, 2021. ISSN: 15505499. DOI: 10.1109/ICCV48922.2021.00761 (cited on pages 6, 40).

[NKK+19] Muzammal Naseer, Salman Khan, Muhammad Haris Khan, Fahad Shahbaz Khan, and Fatih Porikli. Cross-Domain Transferability of Adversarial Perturbations. *Advances in Neural Information Processing Systems*, 32, May 2019. ISSN: 10495258. URL: https://arxiv.org/abs/1905.11736v5 (cited on page 40).

[NRK+22] Muzammal Naseer, Kanchana Ranasinghe, Salman Khan, Fahad Shahbaz Khan, and Fatih Porikli. On Improving Adversarial Transferability of Vision Transformers. In *ICLR (spotlight)*, March 2022 (cited on pages 8, 38).

[NST+18] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, Ian Molloy, and Ben Edwards. Adversarial Robustness Toolbox v1.2.0. *CoRR*, 1807.01069, 2018 (cited on pages 29, 68).

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, pages 8024–8035. Curran Associates, Inc., 2019 (cited on pages 66, 68, 95, 96).

[PH18] Henri Palacci and Henry Hess. Scalable Natural Gradient Langevin Dynamics in Practice, June 2018. URL: http://arxiv.org/abs/1806.02855 (cited on page 52).

[PKG+17] Omid Poursaeed, Isay Katsman, Bicheng Gao, and Serge Belongie. Generative Adversarial Perturbations. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*:4422–4431, December 2017. ISSN: 10636919. DOI: 10.1109/CVPR.2018.00465. URL: https://arxiv.org/abs/1712.02328v3 (cited on page 40).

[PLC18] Guillermo Valle Pérez, Ard A. Louis, and Chico Q. Camargo. Deep learning generalizes because the parameter-function map is biased towards simple functions. *7th International Conference on Learning Representations, ICLR 2019*, May 2018. URL: https://arxiv.org/abs/1805.08522v5 (cited on page 50).

[PMG16] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples, May 2016. URL: http://arxiv.org/abs/1605.07277 (cited on pages 4, 33, 60).

[PVG+11] F Pedregosa, G Varoquaux, A Gramfort, V Michel, B Thirion, O Grisel, M Blondel, P Prettenhofer, R Weiss, V Dubourg, J Vanderplas, A Passos, D Cournapeau, M Brucher, M Perrot, and E Duchesnay. Scikit-learn: Machine Learning in {P}ython. *Journal of Machine Learning Research*, 12:2825–2830, 2011 (cited on page 130).

[QFL+22] Zeyu Qin, Yanbo Fan, Yi Liu, Li Shen, Yong Zhang, Jue Wang, and Baoyuan Wu. Boosting the Transferability of Adversarial Attacks with Reverse Adversarial Perturbation. In *NeurIPS*, October 2022. DOI: 10.48550/arxiv.2210.05968. URL: https://arxiv.org/abs/2210.05968v1 (cited on pages 6, 39, 47, 48, iii).

[RBH+09] Benjamin I. P. Rubinstein, Peter L. Bartlett, Ling Huang, and Nina Taft. Learning in a Large Function Space: Privacy-Preserving Mechanisms for SVM Learning, November 2009 (cited on page 32).

[RDS+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015 (cited on page 66).

[Ris83] Jorma Rissanen. A Universal Data Compression System. *IEEE Transactions on Information Theory*, 29(5):656–664, 1983. ISSN: 15579654. DOI: 10.1109/TIT.1983.1056741 (cited on page 49).

[SDH21] Frank Schneider, Felix Dangel, and Philipp Hennig. Cockpit: A Practical Debugging Tool for the Training of Deep Neural Networks, February 2021. DOI: 10.48550/arxiv.2102.06604. URL: https://arxiv.org/abs/2102.06604v2%20http://arxiv.org/abs/2102.06604 (cited on page 149).

[SHS21] David Stutz, Matthias Hein, and Bernt Schiele. Relating Adversarially Robust Generalization to Flat Minima. *Proceedings of the IEEE International Conference on Computer Vision*:7787–7797, April 2021. ISSN: 15505499. DOI: 10.1109/ICCV48922.2021.00771. URL: https://arxiv.org/abs/2104.04448v2 (cited on page 45).

[SMK21] Jacob M. Springer, Melanie Mitchell, and Garrett T. Kenyon. A Little Robustness Goes a Long Way: Leveraging Robust Features for Targeted Transfer Attacks. *Advances in Neural Information Processing Systems*, 12:9759–9773, June 2021. ISSN: 10495258. DOI: 10.48550/arxiv.2106.02105. URL: https://arxiv.org/abs/2106.02105v2 (cited on pages 6, 38, 44–46, 51, 141, 142, 159).

[SSS+16] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership Inference Attacks against Machine Learning Models, October 2016 (cited on page 32).

[SZ15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015 (cited on page 66).

[SZS+13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, December 2013 (cited on pages 2, 3, 32).

[TCP+18] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. MnasNet: Platform-Aware Neural Architecture Search for Mobile. *ICCV*, 2018 (cited on page 66).

[TKP+18] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, May 2018. URL: http://arxiv.org/abs/1705.07204 (cited on page 99).

[TL19] Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *ICML*:10691–10700, May 2019 (cited on page 66).

[TPG+17] Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The Space of Transferable Adversarial Examples, April 2017. URL: http://arxiv.org/abs/1704.03453 (cited on pages 46, 95).

[TZJ+16] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing Machine Learning Models via Prediction APIs, September 2016 (cited on page 31).

[WGZ+21] Zhibo Wang, Hengchang Guo, Zhifei Zhang, Wenxin Liu, Zhan Qin, and Kui Ren. Feature Importance-aware Transferable Adversarial Attacks. *Proceedings of the IEEE International Conference on Computer Vision*:7619–7628, July 2021. ISSN: 15505499. DOI: 10.1109/ICCV48922.2021.00754. URL: https://arxiv.org/abs/2107.14185v3 (cited on page 39).

[WH21] Xiaosen Wang and Kun He. Enhancing the Transferability of Adversarial Attacks through Variance Tuning. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*:1924–1933, March 2021. ISSN: 10636919. DOI: 10.1109/CVPR46437.2021.00196. URL: https://arxiv.org/abs/2103.15571v3 (cited on pages 6, 8, 37, 39).

[WHW+21] Xiaosen Wang, Xuanran He, Jingdong Wang, and Kun He. Admix: Enhancing the Transferability of Adversarial Attacks. *Proceedings of the IEEE International Conference on Computer Vision*:16138–16147, January 2021. ISSN: 15505499. DOI: 10.1109/ICCV48922.2021.01585. URL: https://arxiv.org/abs/2102.00436v3 (cited on pages 6, 8, 37).

[WLH+21] Xiaosen Wang, Jiadong Lin, Han Hu, Jingdong Wang, and Kun He. Boosting Adversarial Transferability through Enhanced Momentum, March 2021. URL: https://arxiv.org/abs/2103.10609v1 (cited on page 39).

[WRL+21] Xin Wang, Jie Ren, Shuyun Lin, Xiangming Zhu, Yisen Wang, Quanshi Zhang, Shanghai Jiao, and Tong University. A Unified Approach to Interpreting and Boosting Adversarial Transferability. In *ICLR 2021*, October 2021. DOI: 10.48550/arxiv.2010.04055. URL: https://arxiv.org/abs/2010.04055v1 (cited on pages 6, 39).

[WT11] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *ICML 2011*, pages 681–688, 2011. ISBN: 9781450306195 (cited on pages 24, 62).

[WVL+18] Kuan Chieh Wang, Paul Vicol, James Lucas, Li Gu, Roger Grosse, and Richard Zemel. Adversarial distillation of Bayesian neural network posteriors. In *ICML*, volume 12, pages 8239–8248, June 2018. ISBN: 9781510867963. URL: http://arxiv.org/abs/1806.10317 (cited on page 52).

[WWX+20] Dongxian Wu, Yisen Wang, Shu-Tao Xia, James Bailey, and Xingjun Ma. Skip Connections Matter: On the Transferability of Adversarial Examples Generated with ResNets. *ICLR*, February 2020 (cited on pages 6, 8, 38, 44, 51, 68, 80, 85, 94, 104, 106, 142).

[WWY] Guoqiu Wang, Xingxing Wei, and Huanqian Yan. Improving Adversarial Transferability with Spatial Momentum. *PRCV 2022*. DOI: 10.48550/arxiv.2203.13479. URL: https://arxiv.org/abs/2203.13479v1 (cited on page 39).

[WXW20] Dongxian Wu, Shu Tao Xia, and Yisen Wang. Adversarial weight perturbation helps robust generalization. In *Advances in Neural Information Processing Systems*. Neural information processing systems foundation, April 2020. URL: https://arxiv.org/abs/2004.05884v2 (cited on pages 49, 95).

[WZT+18] Lei Wu, Zhanxing Zhu, Cheng Tai, and Weinan E. Understanding and Enhancing the Transferability of Adversarial Examples, February 2018. URL: http://arxiv.org/abs/1802.09707 (cited on pages 6, 48, 49, 51, 100, iii, iv).

[XAT+18] Chen Xing, Devansh Arpit, Christos Tsirigotis, and Yoshua Bengio. A Walk with SGD, February 2018. URL: https://arxiv.org/abs/1802.08770v4 (cited on pages 49, 50).

[XBB+15] Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. Is feature selection secure against training data poisoning? *32nd International Conference on Machine Learning, ICML 2015*, 2:1689–1698, April 2015. URL: https://arxiv.org/abs/1804.07933v1 (cited on page 31).

[XGD+17] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, pages 5987–5995, November 2017 (cited on page 66).

[XZY+18] Cihang Xie, Zhishuai Zhang, Alan L. Yuille, Jianyu Wang, and Zhou Ren. Mitigating adversarial effects through randomization. In *ICLR*. arXiv, November 2018 (cited on page 177).

[XZZ+19] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L. Yuille. Improving transferability of adversarial examples with input diversity. In *ICCV*, March 2019. ISBN: 9781728132938. DOI: 10.1109/CVPR.2019.00284. URL: http://arxiv.org/abs/1803.06978 (cited on pages 6, 8, 37, 42, 63, 68, 75, 80, 85, 104, 106, 142).

[YGK+19] Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael W. Mahoney. PyHessian: Neural Networks Through the Lens of the Hessian. *Proceedings - 2020 IEEE International Conference on Big Data, Big Data 2020*:581–590, December 2019. ISSN: 2331-8422. DOI: 10.1109/BigData50022.2020.9378171. URL: https://arxiv.org/abs/1912.07145v3 (cited on pages 44, 50, 149).

[YZJ+21] Zheng Yuan, Jie Zhang, Yunpei Jia, Chuanqi Tan, Tao Xue, and Shiguang Shan. Meta Gradient Adversarial Attack. In *ICCV*, August 2021. URL: https://arxiv.org/abs/2108.04204v2 (cited on pages 6, 38).

[ZBC+21] Chaoning Zhang, Philipp Benz, Gyusang Cho, Adil Karjauv, and Chan-Hyun Youn. Backpropagating Smoothly Improves Transferability of Adversarial Examples. In *Workshop on Adversarial Machine Learning in Real-World Computer Vision Systems and Online Challenges (AML-CV) at CVPR*, 2021 (cited on page 38).

[ZCB+21] Chaoning Zhang, Gyusang Cho, Philipp Benz, Kang Zhang, Chenshuang Zhang, Chan-Hyun Youn, and In So Kweon. Early Stop And Adversarial Training Yield Better surrogate Model: Very Non-Robust Features Harm Adversarial Transferability, 2021 (cited on pages 38, 44–46, 139, 143).

[ZGY+22] Juntang Zhuang, Boqing Gong, Liangzhe Yuan, Yin Cui, Hartwig Adam, Nicha C Dvornek, sekhar tatikonda, James s Duncan, and Ting Liu. Surrogate Gap Minimization Improves Sharpness-Aware Training. In *International Conference on Learning Representations*, 2022. URL: https://openreview.net/forum?id=edONMAnhLu- (cited on pages 49, 50, 156).

[ZHC+18] Wen Zhou, Xin Hou, Yongjun Chen, Mengyun Tang, Xiangqi Huang, Xiang Gan, and Yong Yang. Transferable adversarial perturbations. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11218 LNCS:471–486, 2018. ISSN: 16113349. DOI: 10.1007/978-3-030-01264-9{_}28/FIGURES/7. URL: https://link.springer.com/chapter/10.1007/978-3-030-01264-9_28 (cited on pages 6, 8, 37, 39).

[ZK16] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. *British Machine Vision Conference 2016, BMVC 2016*, 2016-Septe:1–87, May 2016 (cited on page 66).

[ZLC+22] Qilong Zhang, Xiaodan Li, Yuefeng Chen, Jingkuan Song, Lianli Gao, Yuan He, and Hui Xue. Beyond ImageNet Attack: Towards Crafting Adversarial Examples for Black-box Domains, January 2022. URL: https://arxiv.org/abs/2201.11528v4 (cited on page 40).

[ZLL] Zhengyu Zhao, Zhuoran Liu, and Martha Larson. On Success and Simplicity: A Second Look at Transferable Targeted Attacks. In *NeurIPS 2021*, volume 8, pages 6115–6128. Neural information processing systems foundation. ISBN: 9781713845393. DOI: 10.48550/arxiv.2012.11207. URL: https://arxiv.org/abs/2012.11207v4 (cited on page 39).

[ZLZ+20] Ruqi Zhang, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson. Cyclical Stochastic Gradient MCMC for Bayesian Deep Learning. *ICLR*, February 2020. URL: http://arxiv.org/abs/1902.03932 (cited on pages 10, 52, 59, 62, 63, 67, 85).

[ZRP+21] Shuofeng Zhang, Isaac Reid, Guillermo Valle Pérez, and Ard Louis. Why flatness does and does not correlate with generalization for deep neural networks, March 2021. URL: http://arxiv.org/abs/2103.06219 (cited on page 50).

[ZWH+22] Jianping Zhang, Weibin Wu, Jen-tse Huang, Yizhan Huang, Wenxuan Wang, Yuxin Su, and Michael R. Lyu. Improving Adversarial Transferability via Neuron Attribution-Based Attacks, March 2022. URL: https://arxiv.org/abs/2204.00008v1 (cited on page 39).

[ZZL+22] Zhengyu Zhao, Hanwei Zhang, Renjue Li, Ronan Sicre, Laurent Amsaleg, and Michael Backes. Towards Good Practices in Evaluating Transfer Adversarial Attacks, November 2022. DOI: 10.48550/arxiv.2211.09565. URL: https://arxiv.org/abs/2211.09565v1 (cited on pages 6, 36, 37, 40, 42, 44, 45, 106, 109, 142, 159).