Optimization Problems for Deep Neural Networks

by

Jérôme RONY

MANUSCRIPT-BASED THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE IN PARTIAL FULFILLMENT FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
Ph.D.

MONTREAL, APRIL 24, 2023

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Ismail Ben Ayed, Thesis supervisor
Department of System Engineering, École de technologie supérieure

Mr. Eric Granger, Thesis Co-Supervisor
Department of System Engineering, École de technologie supérieure

Mr. Alessandro Lameiras Koerich, Chair, Board of Examiners
Department of Software and Information Technology Engineering, École de technologie supérieure

Mr. Tony Wong, Member of the Jury
Department of System Engineering, École de technologie supérieure

Mr. Battista Biggio, External Examiner
Department of Electrical and Electronic Engineering, University of Cagliari, Italy

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON APRIL 17, 2023

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

Problèmes d'Optimisation pour les Réseaux de Neurones Profonds

Jérôme RONY

RÉSUMÉ

Les méthodes d'apprentissage profond reposent fortement sur la descente de gradient pour résoudre de nombreux problèmes de reconnaissance de forme. Étant donné les limitations connues de cette méthode d'optimisation, il est donc important de soigneusement formuler les objectifs pour résoudre efficacement et avec précision les tâches d'apprentissage et de vérification de ce domaine. En particulier, les chercheurs ont principalement utilisé des méthodes de pénalité pour gérer des contraintes et introduits de nombreux hyperparamètres pour faciliter l'optimisation.

Dans cette thèse, nous proposons de revisiter certains de ces problèmes, de les analyser à travers le prisme de l'optimisation et d'utiliser des outils connus de la littérature d'optimisation pour les résoudre de manière plus précise et efficace.

Notre première contribution est de prendre du recul par rapport à la littérature sur l'apprentissage profond de métrique et remarquer que la plupart des méthodes proposées ces dernières années et basées sur les paires ont des objectifs similaires. En effet, elles cherchent toutes à maximiser la même quantité : l'information mutuelle. En outre, la minimisation de la fonction de coût entropie croisée peut aussi être interprété comme la maximisation de l'information mutuelle. Cela suggère que l'utilisation de l'entropie croisée pour apprendre les paramètres d'un modèle profond d'apprentissage de métrique soit une solution viable. Nous confirmons cela expérimentalement, où la simplicité de l'entropie croisée permet d'obtenir des résultats de pointe sur tous les jeux de données habituellement utilisés.

En deuxième contribution, nous étudions plusieurs problèmes liés à la robustesse adverse, et en particulier, aux attaques adverses. Ces problèmes peuvent être formulés comme la minimisation d'une mesure de différence, combinée à une (ou plusieurs) contrainte(s) de classification erronée, avec des contraintes additionnelles sur les images. Nous développons un premier algorithme simple afin de générer des exemples adverses minimisant la norme ℓ_2 pour des modèles de classification. Comme de nombreuses attaques adverses publiées ultérieurement, cette méthode est efficace, mais manque de généralité, car elle est conçue spécialement pour une distance. Par conséquent, nous développons une deuxième attaque adverse pour les modèles de classification, basée sur une approche de Lagrangien augmenté. Cette attaque bénéficie de la généralité des méthodes de pénalité, qui peuvent accommoder de nombreuses mesures de différence lisses, et de l'efficacité des algorithmes spécifiques à une distance. Notre but est de fournir un cadre générique servant de point de départ aux futurs chercheurs lors de la conception d'attaques adverses spécifiques à de nouvelles mesures. Enfin, nous étudions les attaques dans le contexte d'une tâche de prédiction dense : la segmentation sémantique. Dans ce contexte, les attaques adverses peuvent être formulées comme un problème d'optimisation avec des millions de contraintes de classification erronée. Ainsi, nous tirons parti de notre méthode basée sur les Lagrangiens augmentés pour gérer une telle quantité de contraintes, et le combinons avec une

méthode de séparation proximale pour minimiser la norme ℓ_∞ non lisse. Cette attaque est, à notre connaissance, la première à résoudre précisément le problème des perturbations adverses minimales pour la segmentation sémantique.

Notre troisième contribution concerne l'étalonnage des réseaux de neurones profonds dans les tâches de classification. À la suite de travaux récents qui ont montré l'avantage d'utiliser des contraintes sur la sortie d'un modèle pour améliorer l'étalonnage, nous généralisons cette approche dans le cadre des Lagrangiens augmentés. En particulier, nous abordons les contraintes avec des pénalités adaptatives par classe. Cela permet d'obtenir une méthode extensible pour la classification et la segmentation, qui obtient des résultats de pointe en termes de classification et d'étalonnage.

Mots-clés: apprentissage profond, optimisation, apprentissage de métrique, attaques adverses, calibration

Optimization Problems for Deep Neural Networks

Jérôme RONY

ABSTRACT

Deep learning methods heavily rely on gradient descent to solve a variety of pattern recognition problems. Given the known limitations of this optimization method, it is of paramount importance to carefully craft objectives to accurately and efficiently solve learning and verification tasks arising in this domain. In particular, researchers have mostly relied on penalty methods to handle constraints and introduced many hyperparameters to ease the optimization process.

In this thesis, we propose to revisit some of these problems, analyze them through the lens of optimization, and leverage well-known tools from the optimization literature to more accurately and efficiently solve them.

Our first contribution is to take a step back from the deep metric learning literature, and notice that most pairwise methods proposed in recent years have similar objectives. In fact, they all correspond to maximizing the same quantity: the mutual information. Additionally, minimizing the well-known cross-entropy loss can also be viewed as maximizing the mutual information. This suggests that using the cross-entropy to learn the parameters of a deep metric learning model is a viable solution. This is confirmed experimentally, where the simplicity of the cross-entropy yields state-of-the-art results on all commonly used datasets.

As a second contribution, we investigate several problems related to adversarial robustness, and adversarial attacks in particular. These problems can be formulated as the minimization of a discrepancy measure under one (or several) misclassification constraint(s), with additional input space constraints. We develop a first simple algorithm to generate minimal ℓ_2-norms adversarial perturbations for classifications models. Like several of the later published adversarial attacks, this method is efficient, but lacks generality as it is customized to one particular distance. Therefore, we develop a second adversarial attack for classification models based on the augmented Lagrangian framework. This attack enjoys the generality of penalty based approaches, as it can handle many smooth discrepancy measures, and the computational efficiency of distance-specific algorithms. Our goal is to provide a general framework that can serve as a starting point to future researchers when designing adversarial attacks for new measures. Finally, we investigate attacks in the context of a dense prediction task: semantic segmentation. Adversarial attacks in this context can be formulated as optimization problems with millions of misclassification constraints. Therefore, we leverage our augmented Lagrangian based method to handle such large numbers of constraints, and combine it with a proximal splitting to minimize the non-smooth ℓ_∞-norm. This attack is, to the best of our knowledge, the first to accurately solve the minimal adversarial perturbation problem for semantic segmentation.

Our third contribution focuses on calibration of deep neural networks in classification tasks. Following recent work that showed the advantage of using constraints on the output of a model to improve calibration, we generalize the approach in an augmented Lagrangian framework. In

particular, we tackle the constraints with adaptive class-wise penalties. This results in a scalable method that can be applied to classification, as well as segmentation, and obtains state-of-the-art classification and calibration performances.

Keywords: deep learning, optimization, metric learning, adversarial attacks, calibration

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

ALM	Augmented Lagrangian Multiplier
CE	Cross-Entropy
DML	Deep Metric Learning
DNN	Deep Neural Network
ECE	Expected Calibration Error
EMA	Exponential Moving Average
KL	Kullback-Leibler divergence
LS	Label Smoothing

LIST OF SYMBOLS AND UNITS OF MEASUREMENTS

$\mathcal{X}$	Input space, usually $\mathcal{X} = [0, 1]^d$, with d the dimension
$\boldsymbol{x}$	Input sample in $\mathcal{X}$
y	Label
K	Number of classes
$\mathcal{D}$	Dataset, often composed of sample-label pairs: $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_i$
θ	Learnable parameters in a model
f_θ	Model parametrized by weights θ, often denoted f for brevity
$\boldsymbol{z}$	Logits produced by a model: $\boldsymbol{z} = (z_i)_{1 \leq i \leq K} = f_\theta(\boldsymbol{x}) \in \mathbb{R}^K$
Δ^{K-1}	Probability $(K-1)$-simplex: $\Delta^{K-1} \subseteq [0, 1]^K$ and $\forall \boldsymbol{s} \in \Delta^{K-1},\ \boldsymbol{s}^\top \mathbf{1}_K = 1$
$P(k\|\boldsymbol{x}, \theta)$	Probability of sample $\boldsymbol{x}$ to belong to class k: $P(k\|\boldsymbol{x}, \theta) = \text{softmax}_k(\boldsymbol{z})$
Λ	Space of admissible additive perturbations: $\Lambda = \mathcal{X} - \boldsymbol{x}$
$\boldsymbol{\delta}$	Perturbation in Λ
$\tilde{\boldsymbol{x}}$	Perturbed sample: $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\delta} \in \mathcal{X}$
D	Discrepancy measure
$\mathcal{L}$	Loss function
$\boldsymbol{g}$	Gradient of a loss function $\mathcal{L}$ w.r.t. the input: $\boldsymbol{g} = \nabla_{\boldsymbol{x}} \mathcal{L} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}}$

INTRODUCTION

Over the last decade, deep learning has revolutionized the machine learning field. Deep neural networks (DNNs) in particular, have proven to be well suited to solve supervised tasks, when a large amount of annotated data is available. These DNNs usually have $\sim 10^6$ to $\sim 10^9$ parameters, which need to be learned. In supervised tasks, the prevailing method to learn these parameters is to perform a gradient descent over a scalar loss function. Indeed, the sheer dimensionality of learning problems involving DNNs makes it impractical to use second-order methods (*e.g.* Newton's method), or even computing the Jacobian matrix for non-scalar loss functions. However, gradient descent is known to be extremely sensitive to ill-conditioned problems, where it can take an arbitrarily long time to converge to a satisfying solution. Thus, it becomes of paramount importance to correctly design the objective to solve deep learning related problems with gradient descent methods.

Deep Metric Learning The representative power of DNNs offers a significant advantage over hand-crafted features, since they are tailored to the underlying data. Therefore, one difficulty is to design an effective training method to learn the parameters of a model to extract features that are discriminative w.r.t. the semantic similarity of samples. This problem is known as *deep metric learning* and has attracted significant attention over the last few years (Kaya & Bilge, 2019), with the promise to train domain-specific but task-agnostic models.

Adversarial Attack Despite their successes in reaching unprecedented performance on many tasks, in particular classification of images, DNNs turned out not to be robust against input perturbations. These perturbations can be either environmental (*e.g.* fog, rain, snow) or come from malicious third parties. Therefore, accurately assessing the robustness of these models is necessary. In particular, empirical worst-case evaluations aim to provide lower-bounds on the expected performance of a model against *any* perturbation. In these evaluations, one needs to accurately solve a constrained optimization problem, involving a function that is costly to

evaluate, has high curvature, and whose derivative needs to be computed. Thus, designing efficient algorithms to solve these worst-case evaluation problems turns out to be difficult, but necessary to ensure the robustness of deployed solutions.

Calibration Another shortcoming of DNNs in the context of classification tasks is their poor calibration. Although they reach high accuracies, their predicted probabilities often do not offer reliable estimates of their correctness. This longstanding problem has a high impact for safety-critical applications such as medical diagnostic. As it turns out, this issue can also be tackled as a constrained optimization problem. However, satisfying constraints on the output of a model during its training is challenging and remains an open problem, with the most successful attempts resorting to penalty based methods.

Problems statements and challenges

Deep metric learning In deep metric learning, we consider a model f_θ parametrized by weights θ, which takes an input $\boldsymbol{x} \in \mathbb{R}^d$ and outputs a feature vector $\boldsymbol{z} = f_\theta(\boldsymbol{x}) \in \mathbb{R}^K$ where d is the dimension of the input space and K is the size of the feature representation. The objective is to train the model to output similar feature vectors, w.r.t. a discrepancy measure (*e.g.* ℓ_2-norm, cosine distance), for inputs that are semantically close, *e.g.* same object or same identity. However, this results in a trivial solution: the model outputs the same feature vector for all inputs, *i.e.* f_θ is constant. Therefore, the second objective is to obtain a model that also produces dissimilar feature vectors for inputs that are semantically different. In essence, the aim of deep metric learning is to find a model f_θ that produces output features with pairwise distances that reflect the semantic dissimilarity between the input samples. This is illustrated in Figure 0.1.

To train the model, we have access to pairwise labels, which encode the semantic similarity between samples. Usually, these labels are binary: two semantically close samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$

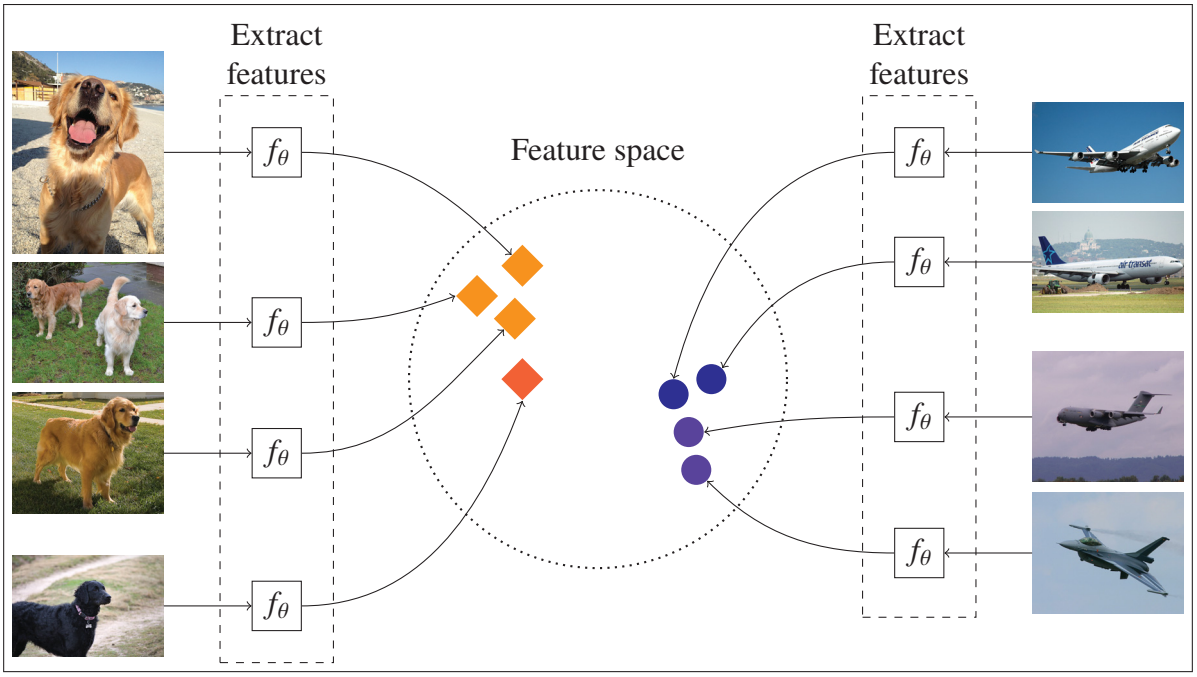


Figure 0.1 Goal of deep metric learning: extract features from samples that correlate with samples' semantic similarity. Images from the ImageNet validation set

have a pairwise label $y_{i,j} = 1$ and two semantically different samples have a pairwise label $y_{i,j} = 0$. Using a discrepancy measure $D : \mathbb{R}^K \times \mathbb{R}^K \to \mathbb{R}_+$, this can be formulated as the following training objective:

$$\underset{\theta}{\text{minimize}} \quad \sum_{i,j} \underbrace{y_{i,j} D(f(\boldsymbol{x}_i), f(\boldsymbol{x}_j))}_{\text{Pull similar samples together}} \underbrace{- (1 - y_{i,j}) D(f(\boldsymbol{x}_i), f(\boldsymbol{x}_j))}_{\text{Push dissimilar samples apart}} . \tag{0.1}$$

In practice, however, such a training objective does not result in good performance for the task. Indeed, in most applications, the number of positive pairs ($y_{i,j} = 1$) is much smaller than the number of negative pairs ($y_{i,j} = 0$). In other words, the pairwise label matrix $Y = (y_{i,j})$ is sparse. This creates an imbalance between the two terms of the objective. Additionally, some pairs of samples are more difficult to discriminate than others, and are therefore, more or less "useful" than other pairs at different stages of training. As such, designing an effective method to

train DNNs for deep metric learning requires solving several challenges (Kaya & Bilge, 2019): handling the imbalance between positive and negative pairs of samples, sampling the best pairs of samples to use in training, managing the numerical instability that can come from quadratic terms often used in the training objective.

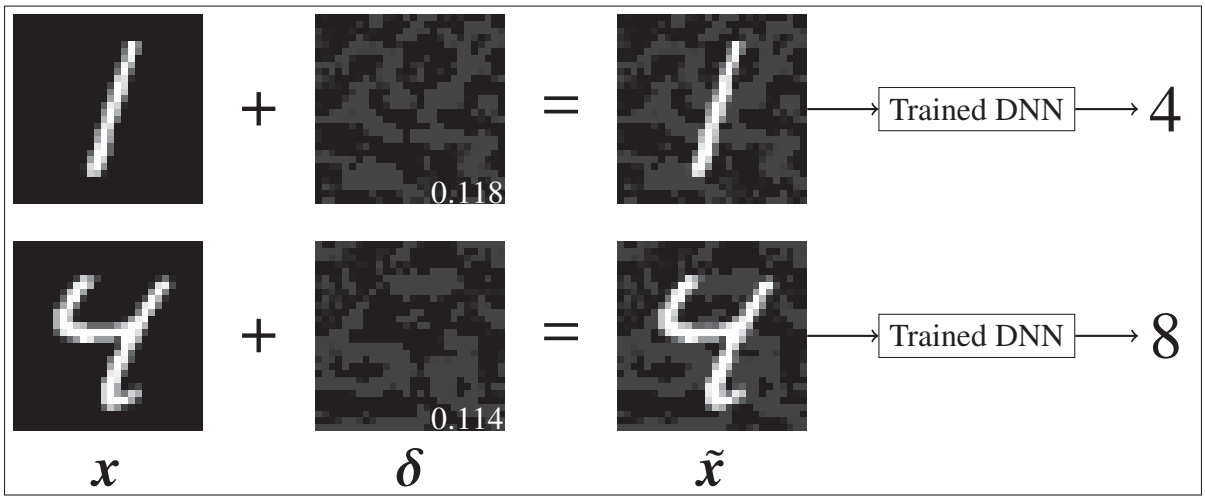


Figure 0.2 Adversarial examples $\tilde{\boldsymbol{x}}$ on MNIST for a SmallCNN reaching 99.44% accuracy (see chapter 3). Both clean samples $\boldsymbol{x}$ are correctly classified. Perturbations $\boldsymbol{\delta}$ are non-negative and their ℓ_∞-norms are reported in the lower right corners

Adversarial robustness The most studied task in the context of adversarial examples is classification. In this task, the model $f : \mathbb{R}^d \to \mathbb{R}^K$ predicts the label $\hat{y}$ for a sample $\boldsymbol{x}$: $\hat{y} = \arg\max_k f_k(\boldsymbol{x})$. While deep learning based models (*i.e.* deep neural networks) have achieved impressive performance on several classification tasks, their predictions are highly susceptible to input perturbations (Biggio & Roli, 2018; Hendrycks & Dietterich, 2019). Figure 0.2 shows adversarial examples on two MNIST samples for a model that reaches 99.44% accuracy. The field of adversarial robustness studies the effect of perturbations on the performance of models.

In particular, the *white-box* scenario is useful to study the *worst-case* performance of a model against a certain type of perturbations. The goal of studying the worst-case performance is to obtain a lower bound on the performance of a model against *any* adversary: from camera noise to malicious user. In a white-box scenario, an attacker has access to all the information about the

model: its architecture, its weights and any additional secret such as an internal state or the seed for a pseudorandom number generator. This scenario is also called *full knowledge*, as opposed to *limited knowledge* scenarios, commonly referred to as *black-box*, or more accurately *gray-box*[1].

In this thesis, we focus on the white-box scenario, used to assess worst-case robustness, and study the corresponding optimization problems. Algorithms designed to solve these problems are called *adversarial attacks*. Their counterparts are called *adversarial defenses*, whose goal is to make a model robust against certain types of adversarial perturbations. For a given discrepancy measure D and a *budget* ϵ, we can formulate the optimization problem related to finding a perturbation that fools a model as:

$$\begin{aligned} \text{find} \quad \boldsymbol{\delta} \quad \text{subject to} \quad & \arg\max_k f_k(\boldsymbol{x}+\boldsymbol{\delta}) \neq y, \\ & D(\boldsymbol{x}+\boldsymbol{\delta}, \boldsymbol{x}) \leq \epsilon, \\ & \boldsymbol{x}+\boldsymbol{\delta} \in \mathcal{X}. \end{aligned} \tag{0.2}$$

This corresponds to an *untargeted* attack, where the goal is to induce misclassification. In contrast, one can perform *targeted* attack by replacing the first constraint with $\arg\max_k f_k(\boldsymbol{x}+\boldsymbol{\delta}) = t$ where t is a target label. Problem (0.2) is a feasibility problem and does not reflect well the goal of a worst-case evaluation. Therefore, a more commonly studied problem is:

$$\begin{aligned} \underset{\boldsymbol{\delta}}{\text{maximize}} \quad \mathcal{L}(\boldsymbol{x}+\boldsymbol{\delta}, y) \quad \text{subject to} \quad & D(\boldsymbol{x}+\boldsymbol{\delta}, \boldsymbol{x}) \leq \epsilon, \\ & \boldsymbol{x}+\boldsymbol{\delta} \in \mathcal{X}, \end{aligned} \tag{0.3}$$

where $\mathcal{L}$ is a loss function, typically the one minimized during training, *e.g.* cross-entropy. This formulation can also be used for targeted attacks with target t by minimizing $\mathcal{L}(\boldsymbol{x}+\boldsymbol{\delta}, t)$ subject to the same constraints.

[1] The "black-box" terminology, referring to a zero knowledge scenario in computer security, has been alienated to refer to a limited knowledge scenario in the adversarial robustness literature. Indeed, most of the time, an attacker has at least some knowledge of the task that the model is trying to solve.

Note that problem (0.3) does not exactly correspond to finding a perturbation that causes misclassification, and problem (0.2) might not always be feasible for sufficiently small ϵ. However, for any useful model (*i.e.* that can correctly classify benign samples), there exists an arbitrarily large perturbation that can induce misclassification. For two correctly classified samples $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ with different labels, a valid perturbation to fool the model on $\boldsymbol{x}_1$ is $\boldsymbol{\delta} = \boldsymbol{x}_2 - \boldsymbol{x}_1$. Therefore, a more general problem is to find the minimal perturbation w.r.t. a discrepancy measure such that the model misclassifies the sample $\boldsymbol{x}$. This problem can be formalized as:

$$\begin{aligned} \underset{\boldsymbol{\delta}}{\text{minimize}} \quad D(\boldsymbol{x}+\boldsymbol{\delta}, \boldsymbol{x}) \quad \text{subject to} \quad & \arg\max_k f_k(\boldsymbol{x}+\boldsymbol{\delta}) \neq y, \\ & \boldsymbol{x}+\boldsymbol{\delta} \in \mathcal{X}. \end{aligned} \tag{0.4}$$

Solving this problem is, in fact, equivalent to solving (0.2) for every ϵ. As such, it constitutes a more general – but also more difficult – problem. Solving it efficiently, accurately and for different choices of discrepancy measures D remains a difficult problem.

Calibration In the context of classification (including segmentation, which is a per-pixel classification problem), a desirable property for a model is to be *calibrated*: the probabilities produced by the model should match the accuracy. For instance, if 100 samples are predicted with a probability of 0.9, then 90% of these samples should be correctly classified. In practice, deep neural networks achieve high accuracy on several classification tasks, but suffer from poor calibration. Their predictions tend to be over-confident, *i.e.* the probability $P(\hat{y}|\boldsymbol{x}, \theta) = \text{softmax}_{\hat{y}}(f(\boldsymbol{x}))$ for the predicted class $\hat{y}$ is often close to 1. This phenomenon is a well-known issue (Guo *et al.*, 2017), particularly for safety-critical applications such as medical diagnosis. Figure 0.3 shows reliability diagrams for two models on CIFAR100 and ImageNet, with their corresponding Expected Calibration Error (ECE). These diagrams depict the calibration of a model: each bar corresponds to the accuracy for predictions with a confidence in the bin interval.

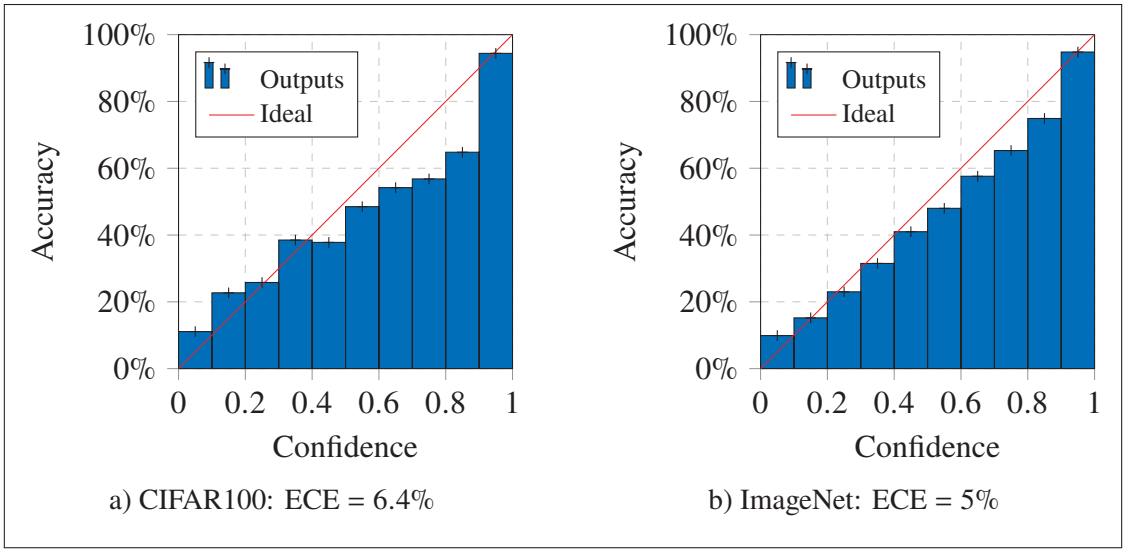


Figure 0.3 Reliability diagrams for a WideResNet 28-10 on CIFAR100 with 81.85% accuracy and a ResNet-152 on ImageNet with 78.32% accuracy. For a perfectly calibrated model, the middle of the bars would be on the red line

Formulating this calibration problem as a tractable optimization problem turns out to be difficult. Indeed, perfect calibration can be defined as:

$$P(\hat{y} = y | P(\hat{y}|\boldsymbol{x}, \theta) = p) = p, \quad \forall p \in [0, 1], \tag{0.5}$$

where $\hat{y} = \arg\max_k f_k(\boldsymbol{x})$ is the predicted label. The probability in (0.5) cannot be calculated with a finite dataset $\mathcal{D}$ as it is a continuous random variable (Guo *et al.*, 2017). Therefore, tackling this problem requires finding a surrogate objective that is compatible with the training of deep neural networks.

Research Objectives and Contributions

In this thesis, our goal is to understand the properties of several optimization problems related to deep neural networks, and provide new methods to solve them more accurately and efficiently. Our contributions are mostly validated on computer vision tasks such as image classification and

segmentation, since they are the prevalent application in the corresponding literature. Note that, theoretically, most of our contributions can be applied, with minimal effort, to other domains that have continuous inputs[2], such as audio signals.

The core contributions of this thesis are:

- **Deep Metric Learning** In chapter 2, we analyze several previously proposed deep metric learning losses to better understand their effect on feature learning. Observing that they perform similarly when fairly evaluated against each other, we uncover an equivalence between the minimization of the cross-entropy and the maximization of the mutual information, which is an upper bound on these previous losses. Hence, training a DNN with the cross-entropy loss turns out to be simpler and achieve better performance on several deep metric learning tasks.

  Related publication:

  A unifying mutual information view of metric learning: cross-entropy vs. pairwise losses, M Boudiaf*, J Rony*, IM Ziko*, E Granger, M Pedersoli, P Piantanida, I Ben Ayed, published at the European Conference on Computer Vision (ECCV), 2020.

- **Adversarial Attacks** In chapter 3, we observe that the current literature on adversarial attacks does not provide an efficient solution to craft adversarial examples with small ℓ_2-norms. Therefore, we design a simple attack algorithm to overcome this shortcoming, called Decoupling Direction and Norm (DDN). This algorithm was used to win one of the competitions of the NeurIPS 2018 Adversarial Vision Challenge (Brendel *et al.*, 2020).

  Related publication:

  Decoupling Direction and Norm for Efficient Gradient-Based ℓ_2 Adversarial Attacks and Defenses, J Rony*, LG Hafemann*, LS Oliveira, I Ben Ayed, R Sabourin, E Granger,

[2] Although images are quantized, their domain is usually considered to be $[0, 1]^d$ in the relevant literature.

published at the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

- **Adversarial Attacks** In chapter 4, we distinguish two families of methods to solve the minimal problem (0.4) in adversarial attacks: distance customized approaches, which tend to be efficient but lack generality, and penalty based approaches which are general but lack efficiency. We propose a general attack algorithm, based on an augmented Lagrangian method, that is efficient and can accommodate a large family of discrepancy measures

  Related publication:

  Augmented Lagrangian Adversarial Attacks, J Rony, E Granger, M Pedersoli, I Ben Ayed, published at the IEEE/CVF International Conference on Computer Vision (ICCV), 2021.

- **Adversarial Attacks** In chapter 5 , we identify a shortcoming of the adversarial robustness literature: attack algorithms are proposed mostly for classification tasks, and are difficult to apply to denser predictions tasks such as semantic segmentation. Therefore, we devise an attack based on the same augmented Lagrangian method, combined with a proximal splitting to produce minimal ℓ_∞-norm perturbations while satisfying millions of constraints.

  Related publication:

  Proximal Splitting Adversarial Attack for Semantic Segmentation, J Rony, JC Pesquet, I Ben Ayed, accepted to the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023.

- **Calibration** In chapter 6, we build upon the recent work of (Liu *et al.*, 2022a), which showed that combining the cross-entropy with constraints on the logit distance significantly improves calibration. In particular, we use an augmented Lagrangian method to learn class-wise penalty weights during training, which further improves the calibration performance of DNNs.

  Related publication:

Class Adaptive Network Calibration, B Liu*, J Rony*, A Galdran, J Dolz, I Ben Ayed, accepted to the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023.

Organization of the thesis

This is a thesis by articles. Chapter 1 presents a broad overview of the literature in deep metric learning, adversarial robustness and calibration in the context of deep learning.

The second chapter presents a detailed analysis of several deep metric learning losses, and how they relate to the cross-entropy. In particular, all of these methods can be viewed as a maximization of the mutual information between the features representing the samples and their pairwise labels. As such, minimizing the cross-entropy can be viewed as an approximate bound-optimization algorithm for minimizing the pairwise losses. We confirm these findings experimentally, by comparing models trained with state-of-the-art deep metric learning losses and a model trained with cross-entropy directly on the class labels. We achieve similar or better metric learning performance on all datasets, while having a simpler method with fewer hyperparameters to tune. This work was presented as a spotlight at the ECCV 2020 conference (Boudiaf *et al.*, 2020).

The third chapter presents a simple, but efficient and robust algorithm to generate adversarial perturbations with smaller ℓ_2-norms than the state-of-the-art attack at the time: Carlini & Wagner (2017). This attack, which was created in the context of a competition on adversarial robustness, requires only $\sim 10^2$ to $\sim 10^3$ iterations compared to $\sim 10^4$ for (Carlini & Wagner, 2017), and is also used to adversarially train robust models against ℓ_2 attacks. This work was presented as an oral at the CVPR 2019 conference (Rony *et al.*, 2019).

In the fourth chapter, we present the adversarial attack literature in a as two algorithmic families. The first contains attacks that are tailored to a particular distance and takes advantage of the properties of that distance to efficiently solve the adversarial perturbation problem. The second contains attacks based on penalties which are general enough to accommodate for many discrepancy measures – as long as they can be minimized by gradient descent – but lack the efficiency of the first family. From there, we design a new attack based on an augmented Lagrangian multiplier, which combines the advantages of the two families: efficiency and generality. We perform an extensive evaluation of this attack on a large collection of models and for several measures, and consistently observe state-of-the-art performances. This work was presented at the ICCV 2021 conference (Rony, Granger, Pedersoli & Ben Ayed, 2021).

In chapter five, we observe that the literature on adversarial attacks for dense predictions tasks, which are often more useful in practical applications, is scarce compared to the classification scenario. Indeed, no attack exists to find small ℓ_∞-norm perturbations for dense classification tasks, *e.g*. segmentation. We build upon the work presented in chapter four, and combine the augmented Lagrangian approach to a proximal splitting, which allows minimizing the non-smooth ℓ_∞-norm. In particular, we design a fast algorithm to compute the proximity operator efficiently. This results in an attack that can be used to more accurately evaluate the robustness of segmentation models, which was largely overestimated until now. This work is accepted to CVPR 2023.

In the final chapter, we notice that the work of Liu *et al.* (2022a) opened an avenue for constrained optimization in the context of calibration. By constraining the logit distance during training, one can improve the calibration performance of DNNs. However, this needs to be carefully incorporated in the training procedure, as we wish to keep a high classification accuracy. We resort to an augmented Lagrangian multiplier scheme to adaptively tune class dependent penalty multipliers at train time. This results in a scalable method to improve calibration, that can be

applied to classification and segmentation, at negligible additional computational cost. The work is accepted to CVPR 2023.

Appendices I to V correspond to the supplementary materials accompanying chapters two to six. They include the proofs of propositions, additional technical details, complete algorithms and detailed results.

CHAPTER 1

LITERATURE REVIEW

In this chapter, we first review the main losses proposed for deep metric learning. We then describe in details the main developments in the adversarial robustness literature, focusing on the algorithmic formulations of attacks. Finally, we present the main works on deep models calibration, which mostly investigated the effects of the training loss on the final calibration performance.

1.1 Deep metric learning

In this section, we present the main methods proposed to tackle the deep metric learning (DML) problem laid out in Equation 0.1. In the general context of DML, these methods mostly consist in finding better suited training objectives. They are often combined with training strategies, *e.g.* pre-training, augmentations or batch normalization freezing, which are not specific to DML. Additionally, in closely related domain-specific applications (*e.g.* face verification, which can be formulated as a metric learning problem), several works also investigate *mining* strategies to find the best pairs of samples to train the most discriminative model. Our work studied the impact of the loss used during training, so we focus on these different training losses proposed for general DML problems.

Most of the losses presented in this section are defined over collections of samples. We denote a collection of samples (*e.g.* a mini-batch) as $\mathcal{D} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$, and the corresponding pairwise labels as $y_{i,j} \in \{0, 1\}$, with $y_{i,j} = 1$ if samples i and j are semantically similar and $y_{i,j} = 0$ if they are dissimilar. Often, these losses also use a discrepancy measure between two embeddings, *e.g.* a squared ℓ_2-norm. We generically denote such a function as $D : \mathbb{R}^K \times \mathbb{R}^K \to \mathbb{R}_+$, and the applications of this function to two samples i and j embeddings as:

$$D_{i,j} = D\left(f_\theta(\boldsymbol{x}^{(i)}), f_\theta(\boldsymbol{x}^{(j)})\right) = D\left(\boldsymbol{z}^{(i)}, \boldsymbol{z}^{(j)}\right). \tag{1.1}$$

1.1.1 Contrastive loss

In DML, the goal is to learn an embedding function f_θ that matches semantic similarity. Early methods were not designed specifically with DML in mind, but tackled the dimensionality reduction problem, which generalizes DML. One such method is the contrastive loss (Hadsell, Chopra & LeCun, 2006). This loss is defined as:

$$\mathcal{L}_{\text{contrastive}}(\mathcal{D}) = \frac{1}{N}\sum_{i,j}^{N} y_{i,j} D_{i,j}^2 + (1 - y_{i,j}) \max\{0, D_{i,j} - m\}^2 \tag{1.2}$$

where $m \in \mathbb{R}_+$ is a margin hyperparameter. Here the discrepancy measure is the ℓ_2-norm: $D_{i,j} = \left\| z^{(i)} - z^{(j)} \right\|_2$. Therefore, the goal of the contrastive loss is to have a zero discrepancy between similar samples, and a larger than m discrepancy for dissimilar samples. A recent approach revisited the contrastive framework by improving the pair mining and weighting strategies (Wang, Han, Huang, Dong & Scott, 2019b).

1.1.2 Triplet loss

Another well-known similar approach is the triplet loss (Weinberger & Saul, 2009). Instead of being defined on pairs of samples, the triplet loss is defined over a triplet: an anchor $\boldsymbol{x}^{(\text{a})}$, a positive $\boldsymbol{x}^{(\text{p})}$ which is semantically similar to the anchor and a negative $\boldsymbol{x}^{(\text{n})}$ which is semantically dissimilar. The loss is defined over each triplet as:

$$\mathcal{L}_{\text{triplet}}\left(\boldsymbol{x}^{(\text{a})}, \boldsymbol{x}^{(\text{p})}, \boldsymbol{x}^{(\text{n})}\right) = \max\{0, D_{a,p}^2 - D_{a,n}^2 + m\}. \tag{1.3}$$

where m is the margin and D is the ℓ_2-norm of the difference. The goal is slightly different compared to the contrastive loss: the discrepancy between similar samples does not need to be 0, but only smaller by a margin m than the discrepancy between dissimilar samples. Several works extended this approach, by adding better triplet mining strategies (Hermans, Beyer & Leibe, 2017; Sohn, 2016), considering additional triplets in a mini-batch (Song, Xiang,

Jegelka & Savarese, 2016), creating hierarchical triplet structures (Ge, 2018) and re-weighting the triplets w.r.t. their *hardness* (Zheng, Chen, Lu & Zhou, 2019).

1.1.3 Neighborhood component analysis

In the neighborhood component analysis (NCA) (Goldberger, Hinton, Roweis & Salakhutdinov, 2005), the goal is to directly maximize the cosine similarity between the embeddings of similar samples, while minimizing it for dissimilar samples. The resulting loss on a whole dataset $\mathcal{D}$ is the following:

$$\mathcal{L}_{\text{NCA}}(\mathcal{D}) = -\sum_{i,j:y_{i,j}=1} \log \frac{\exp(s_{i,j}/\tau)}{\sum_{k\neq i} \exp(s_{i,k}/\tau)} \quad \text{with} \quad s_{i,j} = \frac{{\boldsymbol{z}^{(i)}}^\top \boldsymbol{z}^{(j)}}{\left\|\boldsymbol{z}^{(i)}\right\| \left\|\boldsymbol{z}^{(j)}\right\|}, \tag{1.4}$$

where τ controls the scale of the neighborhood.

The main issue with NCA is that it requires computing a similarity matrix $S = (s_{i,j})_{1\leq i,j\leq N} \in \mathbb{R}^{N\times N}$ on the whole dataset, which can be memory and computationally impractical. A first solution is to use proxies to reduce the size of S by only computing the similarity with a subset of representative negative samples (Movshovitz-Attias, Toshev, Leung, Ioffe & Singh, 2017). Wu, Efros & Yu also proposed a scalable alternative by designing a mini-batch strategy: the embeddings for the whole dataset are kept in memory, and the loss is computed only for the samples in a mini-batch, which updates the memory for these samples (Wu *et al.*, 2018). Both methods aimed at reducing the number of entries to compute in S to reduce the memory and computational requirements.

1.1.4 Center loss

Instead of relying on the pairwise distances as in NCA, one can directly learn a *prototype* for each class of the problem and minimize the distance of the embedding to its class prototype. This is what is done with the center loss presented in (Wen, Zhang, Li & Qiao, 2016), where the author also add a cross-entropy term. The total resulting loss for each sample $\boldsymbol{x}$ with label y is

the following:

$$\mathcal{L}_{\text{Center}}(\boldsymbol{x}, y) = -\log p_y + \frac{\lambda}{2} \left\| \boldsymbol{z} - \boldsymbol{c}^{(y)} \right\|_2^2 \tag{1.5}$$

where $p_y = \text{softmax}(W\boldsymbol{z} + \boldsymbol{b})$ is the probability of the embedding $\boldsymbol{z} = f_\theta(\boldsymbol{x})$ to belong to class y using a linear classifier parametrized by W and $\boldsymbol{b}$, and $\boldsymbol{c}^{(y)} \in$ is the prototype for class y in the embedding space. Note that this loss does not rely on pairwise labels, as opposed to the previous ones. However, as we will see in chapter 2, the squared norm term is redundant with the cross-entropy.

1.1.5 Evaluation fairness

As previously mentioned, most DML methods proposed a new loss and / or a mining strategy to improve the training. Additionally, they often combine it with training tricks, which can greatly influence the final performance. Therefore, one needs to be careful in correctly evaluating a new method and comparing with up-to-date baselines. Indeed, if a new trick is proposed, independently of the loss and mining strategy used, it should also be included in previous baselines. Such tricks include architectural improvements, improved augmentation schemes, and more effective implicit regularization techniques (*e.g.* batch normalization, dropout). Concurrently to our work (see chapter 2), Musgrave, Belongie & Lim (2020) showed that most of the developments in the last 15 years of DML research provide, at best, marginal improvements, when evaluated fairly, *i.e.* using the same training tricks and hyperparameter tuning budgets. This confirms the need to fairly evaluate methods w.r.t. to the baselines.

1.2 Adversarial robustness

In this section, we focus on presenting the main adversarial attacks proposed in the literature. We provide the complete algorithms of several of these attacks with a unified notation and provide insights on their usage and efficiency. Additionally, we give a brief overview of the main defense mechanisms: adversarial training, provable defenses and certified defenses using randomized smoothing.

Most of these have been applied to image classification, but they usually do not make any assumption about the input, except that the input domain is bounded (*e.g.* $\boldsymbol{x} \in [0,1]^d$). Therefore, except for the PerC-AL attack (subsubsection 1.2.1.9), they can be applied to other domain in a straightforward manner.

1.2.1 Adversarial attacks

Table 1.1 Summary of the main white-box adversarial attacks. Complexity refers to the number of model forward and backward propagation needed to perform the attack

Attack	Year	Objective	Distance(s)	Complexity
FGSM	2015	max loss	$\ell_1, \ell_2, \ell_\infty$	1
DeepFool	2016	min norm	ℓ_2, ℓ_∞	$\sim 10^1$
JSMA	2016	min norm	ℓ_0	$\sim 10^1$
PGD	2017	max loss	$\ell_1, \ell_2, \ell_\infty$	$\sim 10^1$ to $\sim 10^2$
C&W	2017	min norm	$\ell_0, \ell_2, \ell_\infty$	$\sim 10^4$
EAD	2018	min norm	ℓ_1	$\sim 10^4$
TR	2019	min norm	ℓ_2, ℓ_∞	$\sim 10^3$
Wasserstein Attack	2019	max loss	Wasserstein distance	$\sim 10^2$
PerC-AL	2020	min norm	CIEDE2000	$\sim 10^4$
FAB	2020	min norm	$\ell_1, \ell_2, \ell_\infty$	$\sim 10^3$ to $\sim 10^4$
APGD	2020	max loss	ℓ_2, ℓ_∞	$\sim 10^2$ to $\sim 10^4$
FMN	2021	min norm	$\ell_0, \ell_1, \ell_2, \ell_\infty$	$\sim 10^2$ to $\sim 10^3$

Here, we present the main white-box gradient-based adversarial attacks for DNNs. It is important to note that attacks are a combination of both a formulation of the optimization problem and a strategy to solve it, which usually include heuristics. Table 1.1 presents a summary of the main attacks proposed in the literature, as well as their objective, distance considered and complexity. The complexity is measured as the number of forward (*i.e.* computing the output of a model) and backward (*i.e.* computing the derivative of the output w.r.t. to the input of a model) propagation performed, which is the main computational cost when generating adversarial attacks.

1.2.1.1 Fast Gradient Sign Method

The Fast Gradient Sign Method (FGSM) is one of the first works on adversarial attacks in the DL community (Goodfellow, Shlens & Szegedy, 2015). This attack is a one-step method that can generate adversarial examples and is arguably the most straightforward method. In this attack:

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} + \epsilon \operatorname{sign}[\nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, y, \theta)] \tag{1.6}$$

where sign is a component-wise function which returns 1 for positive values and -1 for negative values. This method was originally proposed for ℓ_∞ but can be easily adapted to any ℓ_p-norm for $p \in [1, \infty[$ as such:

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} + \epsilon \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|} \quad \text{where} \quad \boldsymbol{g} = \nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, y, \theta) \tag{1.7}$$

In this case, it is called the Fast Gradient Method (FGM). For the ℓ_0-norm, this attack maximally modifies the values of the ϵ pixels that have the largest absolute gradient, but is rarely used in that context. Since it is a one-step attack with no verification of misclassification, this attack aims at solving Equation 0.3.

It can also be adapted to perform targeted attacks by performing a gradient descent instead of a gradient ascent:

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} - \epsilon \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|} \quad \text{where} \quad \boldsymbol{g} = \nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{t}, \theta) \tag{1.8}$$

1.2.1.2 Basic Iterative Method and Projected Gradient Descent

In (Kurakin, Goodfellow & Bengio, 2017a), the authors introduce a straightforward extension of the FSGM attack by performing several iterations of Equation 1.6, and projecting the perturbation such that it lies in an ϵ-neighborhood of $\boldsymbol{x}$:

$$\tilde{\boldsymbol{x}}^{(0)} = \boldsymbol{x} \quad \text{and} \quad \tilde{\boldsymbol{x}}^{(i+1)} = \mathcal{P}_{\Lambda \cap [-\epsilon, \epsilon]} \left(\tilde{\boldsymbol{x}}^{(i)} + \alpha \operatorname{sign}[\nabla_x \mathcal{L}(\tilde{\boldsymbol{x}}^{(i)}, y, \theta)] \right) \tag{1.9}$$

where α is the step-size, and $\mathcal{P}_{\Lambda\cap[-\epsilon,\epsilon]}$ projects on the intersection of the admissible set of perturbation Λ and the ℓ_∞-ball around $\boldsymbol{x}$ of radius ϵ. Note that this intersection is a hypercube, making the projection trivial, as it is separable in each component. This attack, originally called Basic Iterative Method (BIM) (Kurakin *et al.*, 2017a), is now more generally referred to as Projected Gradient Descent (PGD) (Madry, Makelov, Schmidt, Tsipras & Vladu, 2018) and is widely used in adversarial examples research. It is also known to be extremely strong (*i.e.* being able to find small adversarial perturbations) even against several defense methods for the ℓ_∞-norm in a few steps (*e.g.* $\sim 10^1$ to $\sim 10^2$). In the same way as FGSM, the PGD attack aims at solving Equation 0.3, and often uses random restarts of the optimization (*i.e.* setting $\tilde{\boldsymbol{x}}_0 = \boldsymbol{x} + \mathcal{N}(0, \sigma^2 I)$ several times) to obtain a better adversarial example (*i.e.* with larger loss).

While originally proposed for the ℓ_∞-norm, this method is also easily extendable to other ℓ_p-norms, but often fails to find adversarial examples with small ℓ_1 and ℓ_2 norms compared to other attacks. Similarly to FGM, it can also be adapted to produce targeted adversarial examples.

1.2.1.3 Jacobian Saliency Map Attack

Algorithm 1.1 Jacobian Saliency Map Attack (JSMA)

Input: Classifier f, original image $\boldsymbol{x}$ and label y, target label $\boldsymbol{t}$

Initialize $\tilde{\boldsymbol{x}} \leftarrow \boldsymbol{x}$, $\Gamma \leftarrow \{1, \dots, |\boldsymbol{x}|\}$; // Γ is the set of modifiable pixels

while $\arg\max_k f_k(\tilde{\boldsymbol{x}}) \neq \boldsymbol{t}$ **do**

  $\boldsymbol{p} \leftarrow \text{softmax}(f(\tilde{\boldsymbol{x}}))$

  $\boldsymbol{s} \leftarrow \max(\boldsymbol{0}, \nabla_{\boldsymbol{x}} \boldsymbol{p}_{\boldsymbol{t}}) \odot (-\sum_{k \neq \boldsymbol{t}} \nabla_{\boldsymbol{x}} \boldsymbol{p}_k)$; // Computing saliency map

  $j_{\max} \leftarrow \arg\max_{j \in \Gamma} \boldsymbol{s}_j$

  $\tilde{\boldsymbol{x}}_{j_{\max}} \leftarrow 1$; // Set the $j_{\max}$-th pixel to max value

  $\Gamma \leftarrow \Gamma \setminus j_{\max}$; // Remove $j_{\max}$ from modifiable pixels

end while

return $\tilde{\boldsymbol{x}}$

In (Papernot *et al.*, 2016a), Papernot *et al*. propose a greedy approach, called Jacobian Saliency Map Attack (JSMA), to generate targeted adversarial examples minimizing the ℓ_0-norm. The algorithm modifies pixels, one at a time, based on the derivative of the target probability w.r.t. to

the input (see Algorithm 1.1). To choose which pixel to modify, a saliency map is computed as the product of the derivative of the target probability w.r.t. the input and the negative sum of the derivatives of the other classes probabilities w.r.t. to the input. With this saliency map, we can find which pixel will most heavily increase the target probability while reducing the probabilities of the other classes. This pixel is modified to its maximum value, which increases the ℓ_0-norm by 1.

This attack is effective to modify only a few pixels in an image to make it adversarial, but the perturbation becomes visible, as the modified pixels are changed to extremal values.

1.2.1.4 DeepFool

Algorithm 1.2 DeepFool ℓ_2 Attack

Input: Classifier f, original image $\boldsymbol{x}$ and label y
1 Initialize $\boldsymbol{\delta}^{(0)} \leftarrow \mathbf{0}$, $\tilde{\boldsymbol{x}}^{(0)} \leftarrow x$, $i \leftarrow 0$
2 **while** $\arg\max_k f_k(\tilde{\boldsymbol{x}}^{(i)}) = y$ **do**
3 **for** $k \leftarrow 1$ *to* K **do**
4 $\Delta \boldsymbol{g}_k \leftarrow \nabla_{\boldsymbol{x}} f_k(\tilde{\boldsymbol{x}}^{(i)}) - \nabla_{\boldsymbol{x}} f_y(\tilde{\boldsymbol{x}}^{(i)})$
5 $\Delta f_k \leftarrow f_k(\tilde{\boldsymbol{x}}^{(i)}) - f_y(\tilde{\boldsymbol{x}}^{(i)})$
6 **end for**
7 $c \leftarrow \arg\min_{k \neq y} \frac{\Delta f_k}{\|\Delta \boldsymbol{g}_k\|_2}$; `// Find approximately closest class`
8 $\boldsymbol{\delta}_i \leftarrow \frac{\Delta f_c}{\|\Delta \boldsymbol{g}_c\|_2^2} \Delta \boldsymbol{g}_c$; `// Update` $\boldsymbol{\delta}$ `to cross decision boundary`
9 $\tilde{\boldsymbol{x}}^{(i+1)} \leftarrow \tilde{\boldsymbol{x}}^{(i)} + \boldsymbol{\delta}^{(i)}$
10 $i \leftarrow i + 1$
11 **end while**
12 **return** $\tilde{\boldsymbol{x}}^{(i)}$

In (Moosavi-Dezfooli, Fawzi & Frossard, 2016), Moosavi-Dezfooli *et al*. consider a linear approximation of the model under attack to find the closest untargeted adversarial example (Equation 0.4). More specifically, they iteratively refine the perturbation using the gradient of the logits w.r.t. to the input to obtain the point that would cross the decision boundary under the linear approximation. The exact procedure for the ℓ_2-norm is described in Algorithm 1.2, where steps 7 and 8 use the ℓ_1-norm (not squared) when generating adversarial examples for the

ℓ_∞-norm. The attack is originally formulated for any ℓ_p norm but is only used for ℓ_2 and ℓ_∞ norms in practice.

While this algorithm allows obtaining adversarial examples in a few steps ($\sim 10^2$), it does not ensure the box-constraint to produce valid images (*i.e.* in the original range of possible values). Modifying the algorithm to add the box constraint strongly degrades the final performance of the attack (Rauber, Brendel & Bethge, 2017). Also, the DeepFool attack has been reported to be much less effective against adversarially trained models (Rony *et al.*, 2019).

1.2.1.5 Carlini and Wagner

While the DeepFool attack uses an approximation to find the closest adversarial example, Carlini and Wagner (C&W) propose to solve Equation 0.4 by transforming the constrained optimization problem into an unconstrained one using a penalty (Carlini & Wagner, 2017). They also include the box-constraint by performing a variable change, forcing the perturbations to be within bounds. The optimization for an untargeted attack is formulated as follows:

$$\begin{aligned} \underset{\boldsymbol{\delta}}{\text{minimize}} \quad & \|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_2^2 + c \max\{\max_{k \neq y}\{f_k(\tilde{\boldsymbol{x}})\} - f_y(\tilde{\boldsymbol{x}}), -\kappa\} \\ \text{where} \quad & \tilde{\boldsymbol{x}} = \frac{1}{2}(\tanh(\operatorname{arctanh}(\boldsymbol{x}) + \boldsymbol{\delta}) + 1) \end{aligned} \tag{1.10}$$

Increasing the confidence parameter κ will produce an adversarial example that is misclassified with higher probability. In this joint optimization, the trade-off between minimizing the norm of the perturbation and the misclassification is controlled by c, which is chosen in an ad-hoc way.

To produce adversarial examples minimizing the ℓ_2-norm, gradient descent is used and c is chosen using a line search starting at a low value (*e.g.* 10^{-3}) and multiplying it by 2 as long as an adversarial example is not found (c is further refined using a binary search). For the ℓ_0 variant of the algorithm, the ℓ_2 attack is used to find which pixels are *not* modified to obtain an adversarial example. Then these pixels are removed from the set of possible pixels to modify, and the ℓ_2 attack is run again until it fails to find an adversarial example. For the ℓ_∞ variant, the

norm of the perturbation $\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_2$ is replaced by $\sum \max(\mathbf{0}, |\tilde{\boldsymbol{x}} - \boldsymbol{x}| - \tau)$ where τ is heuristically decreased during the optimization.

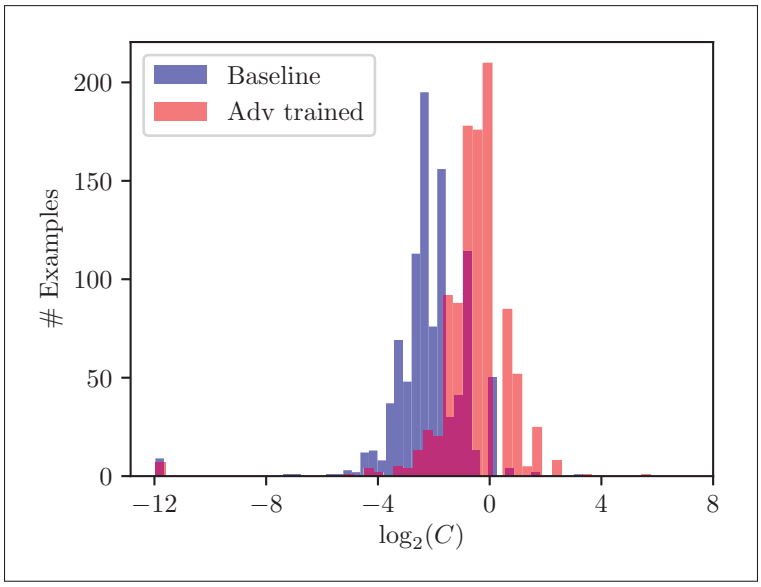


Figure 1.1 Histogram of the best c found by the C&W algorithm with 9 search steps on the MNIST dataset. From (Rony *et al.*, 2019)

In practice, the ℓ_2 variant of this attack is the most widely used, since it is able to find adversarial examples with small ℓ_2-norm while ensuring the box-constraint. However, this attack is particularly inefficient since it requires a line search on c (see Table 1.1). Indeed, if c is too small, the example will not be adversarial; but if it is too large, the perturbation will end up having a large norm. Penalty methods are well known to have slow convergence because of the choice of the weight for the penalty (Jensen & Bard, 2002; Boyd, Boyd & Vandenberghe, 2004). Moreover, this weight has to be chosen on a per model and sample basis. Adversarially trained models tend to be more robust and, therefore, need a larger weight to ensure the misclassification. Figure 1.1 shows the optimal c obtained for the MNIST digits test set on two identical models, where one was trained adversarially. This means that this attack has a prohibitive cost for adversarial training that cannot be reduced.

1.2.1.6 Elastic-Net Attacks

While several attacks have been proposed to find the closest adversarial example (Equation 0.4) for the ℓ_∞ and ℓ_2 norms, little has been done for the ℓ_1-norm. In (Chen, Sharma, Zhang, Yi & Hsieh, 2018b), Chen *et al*. propose the Elastic-Net Attack for DNNs (EAD) to generate adversarial examples with minimal ℓ_1-norm. They use a similar approach as Carlini and Wagner, but adding a ℓ_1 penalty in the optimization as such:

$$\underset{\boldsymbol{\delta}}{\text{minimize}} \quad \|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_2^2 + c \cdot g(\boldsymbol{x}) + \beta \|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_1 \tag{1.11}$$

where g is the second term of Equation 1.10 emphasizing the misclassification. To solve this optimization problem, they use the iterative shrinkage-thresholding algorithm (ISTA) (Beck & Teboulle, 2009). This attack is mostly a direct extension of the Carlini and Wagner ℓ_2 attack to the ℓ_1-norm and has a comparable computational cost.

1.2.1.7 Trust Region Attack

Trust region methods are well known in optimization for solving non-convex problems (Boyd *et al.*, 2004). Yao *et al*. propose to use a trust region based method to produce adversarial example with minimal ℓ_2 or ℓ_∞ norms (Yao, Gholami, Xu, Keutzer & Mahoney, 2019). The principle of trust region methods is to estimate the size of the region where an approximation using first and second order gradients can be trusted. However, computing the Hessian for DNNs is computationally too expensive to be used in practice. But for DNNs using piece-wise linear activation functions, such as Rectified Linear Units (ReLU), the Hessian is zero almost everywhere, meaning that we do not need to compute it. The resulting attack (see Algorithm 1.3) uses only first-order gradients. For the ℓ_∞ variant of the attack, the ℓ_2-norms in steps 2 and 6 are replaced by the ℓ_1 (which is the dual of the ℓ_∞-norm) and the $\boldsymbol{g}/\|\boldsymbol{g}\|_2$ term in step 7 is replaced by $\text{sign}(\boldsymbol{g})$.

Algorithm 1.3 Trust Region ℓ_2 Attack

Input: Classifier f, original image $\boldsymbol{x}$ and label y
Input: Radii $\epsilon^{(0)}$, $\epsilon_{\min}$ and $\epsilon_{\max}$, thresholds σ_{low} and σ_{high}, radius adjustment rate η
Initialize $\boldsymbol{\delta}^{(0)} \leftarrow \mathbf{0}$, $\tilde{\boldsymbol{x}}^{(0)} \leftarrow x$, $i \leftarrow 0$
$t \leftarrow \arg\min_{k \neq y} \frac{f_y(\boldsymbol{x}) - f_k(\boldsymbol{x})}{\left\| \nabla_{\boldsymbol{x}} f_y(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} f_k(\boldsymbol{x}) \right\|_2}$; `// Select the closest class as target`
$\Delta f^{(0)} \leftarrow f_y(\tilde{\boldsymbol{x}}^{(0)}) - f_t(\tilde{\boldsymbol{x}}^{(0)})$
while $\arg\max_k f_k(\tilde{\boldsymbol{x}}^{(i)}) = y$ **do**
  $\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{x}} \Delta f^{(i)}$
  $\tau \leftarrow \frac{f_y(\tilde{\boldsymbol{x}}^{(i)}) - f_t(\tilde{\boldsymbol{x}}^{(i)})}{\|\boldsymbol{g}\|_2}$
  $\tilde{\boldsymbol{x}}^{(i+1)} \leftarrow \mathcal{P}_\Lambda \left(\tilde{\boldsymbol{x}}^{(i)} + \min\{\epsilon^{(i)}, \tau\} \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|_2} \right)$; `// Project on feasible set`
  $\Delta f^{(i+1)} \leftarrow f_y(\tilde{\boldsymbol{x}}^{(i+1)}) - f_t(\tilde{\boldsymbol{x}}^{(i+1)})$
  $\rho \leftarrow \frac{\Delta f^{(i)} - \Delta f^{(i+1)}}{\epsilon^{(i)}}$; `// Estimate ratio for trust region`
  if $\rho > \sigma_{high}$ **then**
    $\epsilon^{(i+1)} \leftarrow \min\{\eta \epsilon^{(i)}, \epsilon_{\max}\}$
  else if $\rho < \sigma_{low}$ **then**
    $\epsilon^{(i+1)} \leftarrow \max\{\epsilon^{(i)} / \eta, \epsilon_{\max}\}$
  else
    $\epsilon^{(i+1)} \leftarrow \epsilon^{(i)}$
  $i \leftarrow i + 1$
end while
return $\tilde{\boldsymbol{x}}^{(i)}$

The method is designed to be an untargeted attack, but can be easily extended to produce targeted adversarial examples. In practice, it requires an order of magnitude less iterations than the C&W attack with similar performance in terms of ℓ_2-norm of the produced adversarial examples.

1.2.1.8 Wasserstein Attack

Moving away from traditional ℓ_p-norms attacks, Wong *et al.* proposed an attack to find an adversarial perturbation in the Wasserstein ball (Wong, Schmidt & Kolter, 2019). This corresponds to solving Equation 0.3 for the 1-Wasserstein distance. To solve this problem, they use a PGD-like algorithm, for which they need the projection onto the feasible set. Therefore,

the main contribution of (Wong *et al.*, 2019) is to propose an efficient projection onto the Wasserstein ball.

1.2.1.9 Perceptual Color Distance Attack

Algorithm 1.4 Perceptual Color Distance Alternating Loss (PerC-AL) Attack

Input: Classifier f, original image $\boldsymbol{x}$ and true label or target label y
Input: Number of iterations N, classification loss step size α_l, color difference step size α_c
Initialize $\boldsymbol{\delta}^{(0)} \leftarrow \mathbf{0}$, $\tilde{\boldsymbol{x}}^{(0)} \leftarrow x$, $i \leftarrow 0$
If targeted attack: $m \leftarrow -1$ else $m \leftarrow +1$; `// Gradient descent or ascent`
for $i \leftarrow 1$ *to* N **do**
  if $\tilde{\boldsymbol{x}}^{(i-1)}$ *is not adversarial* **then**
    $L \leftarrow J(\tilde{\boldsymbol{x}}^{(i-1)}, y)$; `// Classification loss`
    $\boldsymbol{g} \leftarrow m\nabla_{\boldsymbol{x}} L$
    $\hat{\boldsymbol{\delta}} \leftarrow \boldsymbol{\delta}^{(i-1)} + \alpha_l \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|}$
  else
    $\Delta E_{00} \leftarrow \text{CIEDE2000}(\tilde{\boldsymbol{x}}^{(i-1)}, \boldsymbol{x})$; `// CIEDE2000 color difference`
    $C_2 \leftarrow \|\Delta E_{00}\|$
    $\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{x}} C_2$
    $\hat{\boldsymbol{\delta}} \leftarrow \boldsymbol{\delta}^{(i-1)} - \alpha_c \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|}$
  $\tilde{\boldsymbol{x}}^{(i)} \leftarrow \mathcal{P}_\Lambda(\boldsymbol{x} + \hat{\boldsymbol{\delta}})$
  $\boldsymbol{\delta}^{(i)} \leftarrow \tilde{\boldsymbol{x}}^{(i)} - \boldsymbol{x}$
end for
return $\tilde{\boldsymbol{x}}^{(i)}$ *that has smallest* C_2 *and is adversarial*

Following the trend of proposing attacks for distances other than the traditional ℓ_p-norms, Zhao *et al*. focused on the CIEDE2000 color difference in their attack (Zhao, Liu & Larson, 2020). They propose two methods to solve Equation 0.4 for the CIEDE2000 color difference.[3] The first is based on a penalty approach similar to C&W (Carlini & Wagner, 2017) while the second is based on an alternating direction method, which is much more efficient in practice. Algorithm 1.4 describes the alternating direction attack from (Zhao *et al.*, 2020) for both targeted

[3] The CIEDE2000 color difference formula measures the perceptual difference between two colors, and is therefore defined for one pixel. In (Zhao *et al.*, 2020), the ℓ_2-norm of the CIEDE2000 for all pixels is minimzed.

and untargeted attacks. Although this attack presents an improvement over traditional penalty methods, the design of this attack is flawed. Indeed, the authors worked with an incorrect implementation of the CIEDE2000 color difference formula. This results in incorrect values for the CIEDE2000 color difference and incorrect gradients. We contacted the authors of this work, who confirmed the error.

1.2.1.10 Fast Adaptive Boundary Attacks

Algorithm 1.5 Fast Adaptive Boundary (FAB) Attack

Input: Classifier f, original image $\boldsymbol{x}$ and true label y
Input: Norm $p \in \{1, 2, \infty\}$ to minimize, number of iterations N, overshooting parameter η, maximum biased step size $\alpha_{\max}$, backward step size β

1 $q \leftarrow$ dual of the p-norm
2 Initialize $\boldsymbol{\delta}^{(0)} \leftarrow \mathbf{0}$, $\tilde{\boldsymbol{x}}^{(0)} \leftarrow x$, $i \leftarrow 0$
3 **for** $i \leftarrow 1$ *to* N **do**
4 $\quad z \leftarrow f(\tilde{\boldsymbol{x}}^{(i-1)})$; // Logits
5 $\quad s \leftarrow \arg\min_{k \neq y} \frac{z_k - z_y}{\left\| \nabla_x z_k - \nabla_x z_y \right\|_q}$; // Closest class under linear approximation
6 $\quad \boldsymbol{\delta}^{(i)} \leftarrow \text{proj}_p(\tilde{\boldsymbol{x}}^{(i-1)}, \pi_s)$; // Project $\tilde{x}$ on class s decision hyperplane
7 $\quad \boldsymbol{\delta}^{(i)}_{\text{orig}} \leftarrow \text{proj}_p(\boldsymbol{x}, \pi_s)$; // Project $\boldsymbol{x}$ on class s decision hyperplane
8 $\quad \alpha \leftarrow \min \left\{ \frac{\left\| \boldsymbol{\delta}^{(i)} \right\|_p}{\left\| \boldsymbol{\delta}^{(i)} \right\|_p + \left\| \boldsymbol{\delta}^{(i)}_{\text{orig}} \right\|_p}, \alpha_{\max} \right\}$; // Estimate biased step size
9 $\quad \tilde{\boldsymbol{x}}^* \leftarrow \mathcal{P}_\Lambda[(1-\alpha)(\tilde{\boldsymbol{x}}^{(i-1)} + \eta\boldsymbol{\delta}^{(i)}) + \alpha(\boldsymbol{x} + \eta\boldsymbol{\delta}^{(i)}_{\text{orig}})]$; // Biased step
10 $\quad \tilde{\boldsymbol{x}}^{(i)} \leftarrow (1-\beta)\boldsymbol{x} + \beta\tilde{\boldsymbol{x}}^*$; // Backward step
11 **end for**
12 **return** $\tilde{\boldsymbol{x}}^{(i)}$ *that has smallest* p*-norm and is adversarial*

Recently, Croce *et al*. noticed the inefficiency of most attacks solving Equation 0.4 for ℓ_p-norms with $p \in \{1, 2, \infty\}$. Indeed, for these three norms, there exists efficient projections (in $O(d \log d)$ (Hein & Andriushchenko, 2017)) onto a hyperplane. Using these projections and a linear approximation of the model to attack, they propose an attack that is more efficient than previously proposed methods for the ℓ_1 and ℓ_∞ norms. Algorithm 1.5 presents the Fast Adaptive Boundary (FAB) attack. Under the linear approximation of the classifier f, the algorithm finds the class s with the closest decision hyperplane (step 5) and then projects the current

perturbed input $\tilde{\boldsymbol{x}}$ onto it (step 6) while ensuring the box-constraint. Once the projection is obtained, it performs a biased step towards $\boldsymbol{x}$ to try to find a smaller perturbation. This algorithm proved to be well suited to find minimally perturbed adversarial examples for the three ℓ_p-norms mentioned above. However, it suffers from a large computational cost, especially on datasets with large number of classes. Indeed, step 5 requires computing the Jacobian of f w.r.t. the input, representing K (*i.e.* number of classes) backward propagation. This can slightly be reduced to $K-1$ since $\nabla_{\boldsymbol{x}} f_k(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} f_y(\boldsymbol{x}) = \nabla_{\boldsymbol{x}}[f_k(\boldsymbol{x}) - f_y(\boldsymbol{x})]$ but this still represents a high computational cost. Therefore, for datasets with large number of classes such as ImageNet, the gradient of the $f_k(\tilde{\boldsymbol{x}})$ is computed only for the K' most likely classes, with K' typically equal to 10 (Croce & Hein, 2020a).

1.2.1.11 Auto-PGD

The most widely used attack to evaluate the robustness of a deep learning model for a given perturbation budget is by far the PGD attack (Madry *et al.*, 2018), which solves Equation 0.3. However, it is often used in a suboptimal manner, yielding unreliable results and leading to overestimated robustness scores (Croce & Hein, 2020b). Mostly, the performance (success rate in this case) of the attack is highly dependent on the hyperparameters used, namely the step size and loss function. Therefore, Croce & Hein proposed a drop-in replacement for PGD: Auto-PGD (APGD) (Croce & Hein, 2020b). This variant of PGD uses a loss function robust to scaling and better optimization heuristics (*e.g.* momentum, step size scheduling) which greatly improve the success rate, especially for model which obfuscate the gradients. As such, this attack is becoming the standard for robustness evaluation under the ℓ_2 and ℓ_∞-norm threat models.

1.2.1.12 Fast Minimum Norm

The DDN attack (see chapter 3) has proven to be reliable to generate adversarial examples with minimum ℓ_2-norms, *i.e.* solving Equation 0.4. Therefore, Pintor, Roli, Brendel & Biggio proposed to improve it and extend it to the ℓ_0, ℓ_1 and ℓ_∞ norms, with the FMN attack (Pintor *et al.*, 2021). The main differences lie in the projection used, the use of a difference of logits as

in C&W for the loss instead of the cross-entropy, and a better scheduling of the norm for the projection. The resulting attack is presented in Algorithm 1.6.

Algorithm 1.6 Fast Minimum Norm (FMN) Attack

Input: Classifier f, original image $\boldsymbol{x}$ and true (or target) label y
Input: Norm $p \in \{0, 1, 2, \infty\}$ to minimize, number of iterations N
Input: Norm increase rates $(\gamma^{(i)})_{1 \leq i \leq N}$ and step-sizes $(\alpha^{(i)})_{1 \leq i \leq N}$.
Initialize $\tilde{\boldsymbol{x}}^{(0)} \leftarrow \boldsymbol{x}$, $\boldsymbol{\delta}^{(0)} \leftarrow \mathbf{0}$, $\boldsymbol{\delta}^{\star} \leftarrow \infty$
$q \leftarrow$ dual of p, $m \leftarrow -1$ if attack is targeted, else $m \leftarrow 1$
for i=*1,. . . ,N* **do**
  $\boldsymbol{z} = f(\boldsymbol{x} + \boldsymbol{\delta}^{(i-1)})$
  $\mathcal{L} = m(\boldsymbol{z}_y - \max_{k \neq y} \boldsymbol{z}_k)$
  $\boldsymbol{g} = \nabla_{\boldsymbol{\delta}} \mathcal{L}$
  if $\mathcal{L} > 0$ **then**
    if *no adversarial sample has been found yet* **then**
      $\epsilon^{(i)} \leftarrow \left\|\boldsymbol{\delta}^{(i-1)}\right\|_p + \frac{\mathcal{L}}{\|\boldsymbol{g}\|_q}$; // Estimate distance to boundary
    else
      $\epsilon^{(i)} = \epsilon^{(i-1)}(1 + \gamma^{(i)})$
  else
    if $\left\|\boldsymbol{\delta}^{(i-1)}\right\|_p \leq \left\|\boldsymbol{\delta}^{\star}\right\|_p$ **then**
      $\boldsymbol{\delta}^{\star} \leftarrow \boldsymbol{\delta}^{(i-1)}$; // Track best perturbation
    $\epsilon^{(i)} = \min\{\epsilon^{(i-1)}(1 - \gamma^{(i)}), \left\|\boldsymbol{\delta}^{\star}\right\|_p\}$
  $\tilde{\boldsymbol{\delta}} \leftarrow \boldsymbol{\delta}^{(i-1)} - \alpha^{(i)} \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|_2}$
  $\tilde{\boldsymbol{\delta}} \leftarrow \mathcal{P}_{\mathcal{B}_p(\boldsymbol{x}, \epsilon^{(i)})}(\boldsymbol{x} + \tilde{\boldsymbol{\delta}}) - \boldsymbol{x}$; // Projection on the ℓ_p-norm ball
  $\boldsymbol{\delta}^{(i)} \leftarrow \mathcal{P}_{\Lambda}(\boldsymbol{x} + \tilde{\boldsymbol{\delta}}) - \boldsymbol{x}$
end for
return $\boldsymbol{x} + \boldsymbol{\delta}^{\star}$

1.2.2 Defenses for deep neural networks

1.2.2.1 Adversarial training

Although many defense algorithms have been proposed in the literature, only one of them still holds as a working defense mechanism against all known adversarial attacks for a chosen ℓ_p-norm (Athalye, Carlini & Wagner, 2018; Tramer, Carlini, Brendel & Madry, 2020). The

principle of the adversarial training (Kurakin, Goodfellow & Bengio, 2017b; Madry *et al.*, 2018) is to train a classifier using adversarial examples crafted at train time instead of *clean* examples. The resulting optimization as formulated in (Madry *et al.*, 2018) is a saddle point problem, optimizing for the worst case:

$$\underset{\theta}{\text{minimize}} \quad \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\Big[\max_{\boldsymbol{\delta}\in\Lambda} \mathcal{L}(\boldsymbol{x}+\boldsymbol{\delta}, y, \theta)\Big] \tag{1.12}$$

where $\mathcal{D}$ is the training set and $\mathcal{S}$ indicates the feasible region for the attacker. Usually, $\mathcal{S}$ is an ℓ_p-ball such as $\mathcal{S} = \{\boldsymbol{\delta} : \|\boldsymbol{\delta}\|_p < \epsilon\}$ where ϵ is chosen beforehand. Empirically, increasing ϵ beyond a certain point reduces the final performance on clean images (Tsipras, Santurkar, Engstrom, Turner & Madry, 2019).

In the literature, this defense has been studied mostly for the ℓ_∞ and ℓ_2 norms. This is due to the fact that generating adversarial at each training iteration (*i.e.* the inner maximization in (1.12)) is computationally expensive. For any training iteration (*i.e.* backward propagation through the model), we need to find corresponding adversarial examples. This process also back propagates through the whole model under training several times. For instance, if an attack needs 20 iterations to find an adversarial example, the resulting training time will be ∼20 times longer (not accounting for any other hyperparameter change that may help adversarial training). The attack must also be robust to varying properties of the model; most attacks have been developed on models trained in a conventional fashion (*i.e.* cross-entropy on clean examples) but training them for defense modify their gradients, making some attacks fail on adversarially trained models (Rony *et al.*, 2019). Currently, most works performing adversarial training use PGD as the attack which is known to be both fast and robust for the ℓ_∞-norm and, to some extent, for the ℓ_2-norm as well.

1.2.2.2 Provable defenses and certified adversarial robustness

Adversarial training has empirically proven to be a working defense mechanism. However, for safety-critical applications, we need to obtain a guarantee that the model is robust to some perturbations, and quantify that robustness.

A first approach has been to extend the simplex method to ReLU activations which are widely spread in DNNs to provably find the smallest adversarial example (Katz, Barrett, Dill, Julian & Kochenderfer, 2017). The two main conclusions of Katz *et al*. are that they were able to certify only small models and that these combinatorial methods are not scalable to larger models, which are common in computer vision.

Another line of work is to consider convex relaxation of the ReLU activation over the set of reachable activations through a norm-bounded perturbation (Wong & Kolter, 2018; Raghunathan, Steinhardt & Liang, 2018). Once again, these methods lack scalability, which prevents them from being used on larger models typically used for datasets like ImageNet (and real-world applications).

More recently, Cohen *et al*. have proposed the randomized smoothing to certify adversarial robustness (Cohen, Rosenfeld & Kolter, 2019). The core idea of randomized smoothing is to construct a *smoothed* classifier f^* from a *base* classifier f. The smoothed classifier will return the average prediction for a sample $\boldsymbol{x}$ by the base classifier f when adding isotropic Gaussian noise:

$$f^*(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{\delta}\sim\mathcal{N}(0,\sigma^2 I)}[f(\boldsymbol{x}+\boldsymbol{\delta})] \tag{1.13}$$

Using this formulation, the authors are able to establish robustness guarantees for the ℓ_2-norm, which are naturally induced by the Gaussian smoothing. Extending this work by combining this approach with adversarial training, Salman *et al*. obtain state-of-the-art robustness guarantees for the ℓ_2-norm (Salman *et al.*, 2019).

1.3 Calibration for deep learning based classification

In this section, we present the main approaches used to improve the calibration of deep neural networks. These approaches are designed for the classification context in the general sense, meaning that it can easily be extended to segmentation, which is typically approached as a per-pixel classification problem.

In a deep learning based classification context, a model $f_\theta : \mathbb{R}^d \to \mathbb{R}^K$ parametrized by weights θ is trained by minimizing a loss $\mathcal{L}$ on a dataset $\mathcal{D} = \{(\boldsymbol{x}, y)\}_{i=1}^n$. This can be formulated as the following optimization problem:

$$\underset{\theta}{\text{minimize}} \quad \sum_{(\boldsymbol{x},y)\in\mathcal{D}} \mathcal{L}(f_\theta(\boldsymbol{x}), y) \tag{1.14}$$

The main loss used in this context is the Cross-Entropy (CE). We denote $\boldsymbol{s} = (s_k)_{1\leq k\leq K} \in \Delta^{K-1}$ the probability vector predicted by the model for a sample, usually obtained from the logits using the softmax function: $\boldsymbol{s} = \text{softmax}(\boldsymbol{z}) = \text{softmax}(f_\theta(\boldsymbol{x}))$. The cross-entropy is defined as:

$$\mathcal{L}_{\text{CE}}(\boldsymbol{z}, y) = -\log s_y = -\log\left(\text{softmax}_y(\boldsymbol{z})\right). \tag{1.15}$$

This loss can also be written considering a binary vector label $\mathbf{y} \in \{0, 1\}^K$, with $\mathbf{y}_y = 1$ and $\mathbf{y}_{k\neq y} = 0$, as:

$$\mathcal{L}_{\text{CE}}(\boldsymbol{z}, \mathbf{y}) = -\sum_{k=1}^{K} \mathbf{y}_k \log s_k. \tag{1.16}$$

Over the years, this has remained, by far, the main training loss for classification problems. However, as shown in (Guo *et al.*, 2017), training deep neural networks with the cross-entropy results in poorly calibrated predictions.

1.3.1 Temperature scaling

As a post-hoc solution, temperature scaling was proposed in the seminal work (Guo *et al.*, 2017). This method does not require modifying any part of the training procedure. It simply applies a

scaling $\tau \in \mathbb{R}_+$ to the logits, after the training is completed. The scaling is chosen to minimize the expected calibration error on the validation dataset, yielding new probabilities $\tilde{\boldsymbol{s}} \in \Delta^{K-1}$:

$$\tilde{\boldsymbol{s}} = \text{softmax}(\boldsymbol{z}/\tau) = \frac{\exp(\boldsymbol{z}/\tau)}{\sum_{k=1}^{K} \exp(z_k/\tau)}. \tag{1.17}$$

This scaling does not change the predicted class $\hat{y} = \arg\max_k \tilde{s}_k = \arg\max_k s_k$. A $\tau > 1$ results in logits closer to 0, and in turn, probabilities closer to $1/K$. Conversely, a $\tau < 1$ pushes the predicted probabilities to the vertices of the probability simplex. Note that the scaling could also be multiplicative, but the formulation in Equation 1.17 is more commonly found in the literature.

This method has the advantage of being computationally cheap, since it is done after training, and only requires computing (and storing) the logits on the validation set once. Additionally, it can be combined with any training-time method, and can only offer improvements in terms of calibration, since the worst-case would be to find an optimal temperature $\tau = 1$, while not keeping the same performance in terms of accuracy.

1.3.2 Maximum mean calibration error

The main difficulty in improving calibration is the impossibility to directly optimize the objective that would result in satisfying Equation 0.5. As such, the goal of Kumar, Sarawagi & Jain is to find a surrogate function for the calibration error, that is compatible with first-order optimization. Since the training of deep neural networks is usually done with mini-batches, they design a measure called maximum mean calibration error (MMCE) over the predictions for mini-batch, that is differentiable. Denoting $c^{(i)}$ the top predicted probability (*i.e.* often referred to as confidence) for sample i in a mini-batch of m samples, the MMCE_m^2 loss is defined as:

$$\text{MMCE}_\text{m}^2 = \sum_{i,j}^{m} \frac{\left([\hat{y}^{(i)} = y] - c^{(i)}\right)\left([\hat{y}^{(j)} = y] - c^{(j)}\right)\phi\left(c^{(i)}, c^{(j)}\right)}{m^2}, \tag{1.18}$$

where $[\cdot]$ is the Iverson bracket and ϕ is a kernel function. This loss is combined with the cross-entropy during training to improve the calibration, while still achieving good classification performance. In practice, a Laplacian kernel $\phi(c_1, c_2) = \exp\left(-\frac{-|c_1-c_2|}{0.4}\right)$ is used.

1.3.3 Label smoothing

Initially proposed as a training method to improve classification performance (Szegedy, Vanhoucke, Ioffe, Shlens & Wojna, 2016), label smoothing turned out to also improve calibration performance. It consists in replacing the one-hot label in the cross-entropy loss (1.16) by a "smoothed" label $\tilde{\mathbf{y}} = (\tilde{\mathbf{y}}_k)_{1 \leq k \leq K}$:

$$\tilde{\mathbf{y}}_k = \begin{cases} 1 - \xi & \text{if} \quad k = y, \\ \frac{\xi}{K-1} & \text{otherwise.} \end{cases} \tag{1.19}$$

When training with the cross-entropy with one-hot labels, the optimal solution for the logit of the sample's class is $z_y^\star = +\infty$. With the smoothed labels, the optimal solution for the logits becomes:

$$z_k^\star = \begin{cases} \log((K-1)(1-\xi)/\xi) + c & \text{if} \quad k = y, \\ c & \text{otherwise,} \end{cases} \tag{1.20}$$

where $c \in \mathbb{R}$ is an arbitrary constant (He *et al.*, 2019). In other words, the probabilities are not pushed indefinitely towards the vertices of the probability simplex during the training, which obviously results in better calibration.

1.3.4 Explicit confidence penalty

Another method to avoid over-confident predictions is to directly penalize the negative entropy of the model's predictions (Pereyra, Tucker, Chorowski, Kaiser & Hinton, 2017). This biases the model to produce probabilities that are closer to the uniform distribution. The resulting loss becomes:

$$\mathcal{L}(\boldsymbol{z}, y) = -\log s_y - \lambda \mathcal{H}(\boldsymbol{s}) = -\log s_y - \lambda \boldsymbol{s}^\top \log(\boldsymbol{s}), \tag{1.21}$$

where $\mathcal{H}$ is the entropy and $\lambda \in \mathbb{R}_+$ is the weight of the entropic regularization. This method is closely related to label smoothing as noted in (Pereyra *et al.*, 2017); the difference being the direction of the Kullback-Leibler (KL) divergence.

1.3.5 Focal loss

Initially proposed to improve recall in binary classification contexts, the focal loss (Lin, Goyal, Girshick, He & Dollár, 2017) has been observed to improve calibration (Mukhoti *et al.*, 2020). The focal loss is defined as:

$$\mathcal{L}_{\text{focal}}(z, y) = -(1 - s_y)^{\gamma} \log s_y, \tag{1.22}$$

where γ is a hyperparameter of the loss. For $\gamma = 0$, we recover the cross-entropy, while higher values correspond to a loss with a steeper slope when s_y is close to 0, as shown in Figure 1.2. This results in a reduction of the magnitude of the logits because the gradient of the loss w.r.t. s_y is smaller when s_y is close to 1. Mukhoti *et al.* further improve the calibration performance of the focal loss by using a varying value of γ as a function of the predicted probability s_y:

$$\gamma(s_y) = \begin{cases} 5 & \text{if} \quad s_y < 0.2, \\ 3 & \text{if} \quad s_y \geq 0.2. \end{cases} \tag{1.23}$$

1.3.6 Margin based label smoothing

Recently, Liu *et al.* (2022a) showed that the label smoothing and focal losses can be viewed as an approximation of a linear penalty for an equality constraint on the difference of logits. We define the non-negative *logit distance* function $\boldsymbol{d} : \mathbb{R}^K \to \mathbb{R}_+^K$ as:

$$\boldsymbol{d}(z) = \max_k z_k - z. \tag{1.24}$$

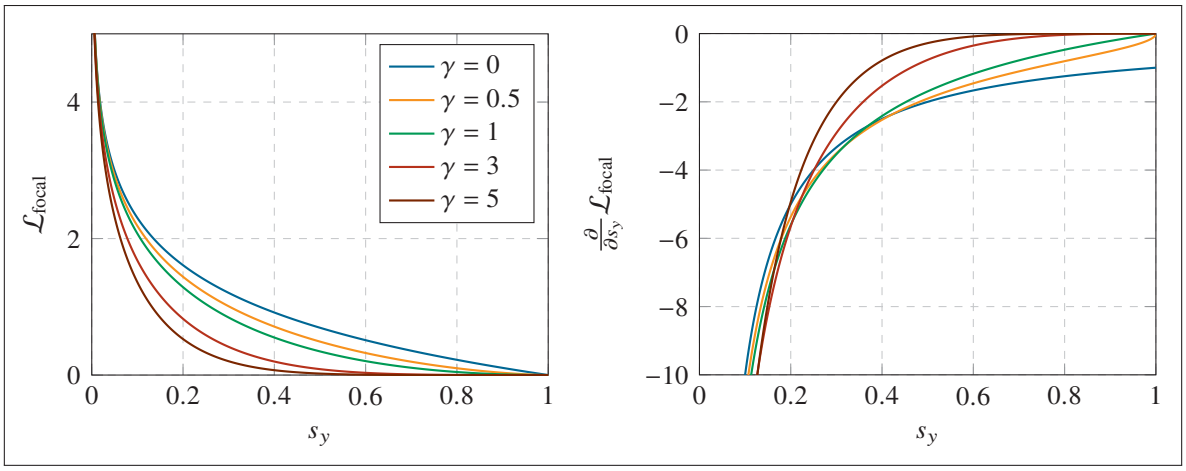


Figure 1.2 The focal loss for different values of γ, and the derivative w.r.t. s_y. When $\gamma = 0$, this corresponds to the cross-entropy loss

The label smoothing and focal loss objectives both correspond to a linear penalty for the constraint $\boldsymbol{d}(\boldsymbol{z}) = \mathbf{0}$ (Liu *et al.*, 2022a). However, imposing $\boldsymbol{d}(\boldsymbol{z}) = \mathbf{0}$ obviously leads to poor performance, since it amounts to have the model be a constant function. Therefore, Liu *et al.* propose to introduce a margin $m \in \mathbb{R}_+$ in the constraint on the logit distance, resulting in the following training objective:

$$\underset{\theta}{\text{minimize}} \quad \mathcal{L}_{\text{CE}}(f_\theta(\boldsymbol{x}), y) \quad \text{subject to} \quad \boldsymbol{d}(f_\theta(\boldsymbol{x})) \preceq m\mathbf{1}_K. \tag{1.25}$$

Solving this problem in the context of deep neural training is not an easy task, since it involves hard constraints. Therefore, Liu *et al.* approximately solve it with a linear penalty with the following margin based label smoothing (MbLS) loss:

$$\mathcal{L}_{\text{MbLS}}(\boldsymbol{z}, y) = \mathcal{L}_{\text{CE}}(\boldsymbol{z}, y) + \lambda \sum_{k=1}^{K} \max\{0, \boldsymbol{d}_k(\boldsymbol{z}) - m)\}, \tag{1.26}$$

where $\lambda \in \mathbb{R}_+$ is the weight of the penalty.

CHAPTER 2

A UNIFYING MUTUAL INFORMATION VIEW OF METRIC LEARNING: CROSS-ENTROPY VS. PAIRWISE LOSSES

Malik Boudiaf* [1] , Jérôme Rony* [1] , Imtiaz Masud Ziko* [1] , Eric Granger[1] , Marco Pedersoli[1] , Pablo Piantanida[2] , Ismail Ben Ayed[1]

* Equal contribution

[1] Systems Engineering Department, École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada

[2] Laboratoire des Signaux et Systèmes (L2S),
CentraleSupelec-CNRS-Université Paris-Saclay, France

Paper published at the
European Conference on Computer Vision (ECCV), August 2020

Abstract

Recently, substantial research efforts in Deep Metric Learning (DML) focused on designing complex pairwise-distance losses, which require convoluted schemes to ease optimization, such as sample mining or pair weighting. The standard cross-entropy loss for classification has been largely overlooked in DML. On the surface, the cross-entropy may seem unrelated and irrelevant to metric learning as it does not explicitly involve pairwise distances. However, we provide a theoretical analysis that links the cross-entropy to several well-known and recent pairwise losses. Our connections are drawn from two different perspectives: one based on an explicit optimization insight; the other on discriminative and generative views of the mutual information between the labels and the learned features. First, we explicitly demonstrate that the cross-entropy is an upper bound on a new pairwise loss, which has a structure similar to various pairwise losses: it minimizes intra-class distances while maximizing inter-class distances. As a result, minimizing the cross-entropy can be seen as an approximate *bound-optimization* (or *Majorize-Minimize*) algorithm for minimizing this pairwise loss. Second, we show that, more generally, minimizing the cross-entropy is actually equivalent to maximizing the mutual information, to which we connect several well-known pairwise losses. Furthermore, we show that various standard

pairwise losses can be explicitly related to one another via bound relationships. Our findings indicate that the cross-entropy represents a proxy for maximizing the mutual information – as pairwise losses do – without the need for convoluted sample-mining heuristics. Our experiments[4] over four standard DML benchmarks strongly support our findings. We obtain state-of-the-art results, outperforming recent and complex DML methods.

2.1 Introduction

The core task of metric learning consists in learning a metric from high-dimensional data, such that the distance between two points, as measured by this metric, reflects their semantic similarity. Applications of metric learning include image retrieval, zero-shot learning or person re-identification, among others. Initial attempts to tackle this problem tried to learn metrics directly on the input space (Lowe, 1995). Later, the idea of learning suitable embedding was introduced, with the goal of learning Mahalanobis distances (Xing, Jordan, Russell & Ng, 2003; Schultz & Joachims, 2004; Goldberger *et al.*, 2005; Weinberger & Saul, 2009; Davis, Kulis, Jain, Sra & Dhillon, 2007), which corresponds to learning the best linear projection of the input space onto a lower-dimensional manifold, and using the Euclidean distance as a metric. Building on the embedding-learning ideas, several papers proposed to learn more complex mappings, either by kernelization of already existing linear algorithms (Davis *et al.*, 2007), or by using a more complex hypothesis such as linear combinations of gradient boosted regressions trees (Kedem, Tyree, Sha, Lanckriet & Weinberger, 2012).

The recent success of deep neural networks at learning complex, nonlinear mappings of high-dimensional data aligns with the problem of learning a suitable embedding. Following works on Mahalanobis distance learning, most Deep Metric Learning (DML) approaches are based on pairwise distances. Specifically, the current paradigm is to learn a deep encoder that maps points with high semantic similarity close to each other in the embedded space (w.r.t. pairwise Euclidean or cosine distances). This paradigm concretely translates into *pairwise losses* that encourage small distances for pairs of samples from the same class and large distances for

[4] Code available at: https://github.com/jeromerony/dml_cross_entropy

pairs of samples from different classes. While such formulations seem intuitive, the practical implementations and optimization schemes for pairwise losses may become cumbersome, and randomly assembling pairs of samples typically results in slow convergence or degenerate solutions (Hermans *et al.*, 2017). Hence, research in DML focused on finding efficient ways to reformulate, generalize and/or improve sample mining and/or sample weighting strategies over the existing pairwise losses. Popular pairwise losses include triplet loss and its derivatives (Hermans *et al.*, 2017; Sohn, 2016; Song *et al.*, 2016; Zheng *et al.*, 2019; Ge, 2018), contrastive loss and its derivatives (Hadsell *et al.*, 2006; Wang *et al.*, 2019b), Neighborhood Component Analysis and its derivatives (Goldberger *et al.*, 2005; Movshovitz-Attias *et al.*, 2017; Wu *et al.*, 2018), among others. However, such modifications are often heuristic-based, and come at the price of increased complexity and additional hyper-parameters, reducing the potential of these methods in real-world applications. Furthermore, the recent experimental study in (Musgrave *et al.*, 2020) showed that the improvement brought by an abundant metric learning literature in the last 15 years is at best marginal when the methods are compared fairly.

Admittedly, the objective of learning a useful embedding of data points intuitively aligns with the idea of directly acting on the distances between pairs of points in the embedded space. Therefore, the standard cross-entropy loss, widely used in classification tasks, has been largely overlooked by the DML community, most likely due to its apparent irrelevance for Metric Learning (Wen *et al.*, 2016). As a matter of fact, why would anyone use a point-wise prediction loss to enforce pairwise-distance properties on the embedding space? Even though the cross-entropy was shown to be competitive for face recognition applications (Liu *et al.*, 2017; Wang *et al.*, 2018b; Wang, Cheng, Liu & Liu, 2018a), to the best of our knowledge, only one paper empirically observed competitive results of a normalized, temperature-weighted version of the cross-entropy in the context of deep metric learning (Zhai & Wu, 2019). However, the authors did not provide any theoretical insights for these results.

On the surface, the standard cross-entropy loss may seem unrelated to the pairwise losses used in DML. Here, we provide theoretical justifications that connect directly the cross-entropy to several well-known and recent pairwise losses. Our connections are drawn from two different

perspectives; one based on an explicit optimization insight and the other on mutual-information arguments. We show that four of the most prominent pairwise metric-learning losses, as well as the standard cross-entropy, are maximizing a common underlying objective: the Mutual Information (MI) between the learned embeddings and the corresponding samples' labels. As sketched in section 2.2, this connection can be intuitively understood by writing this MI in two different, but equivalent ways. Specifically, we establish tight links between pairwise losses and the *generative* view of this MI. We study the particular case of contrastive loss (Hadsell *et al.*, 2006), explicitly showing its relation to this MI. We further generalize this reasoning to other DML losses by uncovering tight relations with contrastive loss. As for the cross-entropy, we demonstrate that the cross-entropy is an upper bound on an underlying pairwise loss – on which the previous reasoning can be applied – which has a structure similar to various existing pairwise losses. As a result, minimizing the cross-entropy can be seen as an approximate *bound-optimization* (or *Majorize-Minimize*) algorithm for minimizing this pairwise loss, implicitly minimizing intra-class distances and maximizing inter-class distances. We also show that, more generally, minimizing the cross-entropy is equivalent to maximizing the *discriminative* view of the mutual information. Our findings indicate that the cross-entropy represents a proxy for maximizing the mutual information, as pairwise losses do, without the need for complex sample-mining and optimization schemes. Our comprehensive experiments over four standard DML benchmarks (CUB200, Cars-196, Stanford Online Product and In-Shop) strongly support our findings. We consistently obtained state-of-the-art results, outperforming many recent and complex DML methods.

In summary, our contributions are:

1. Establishing relations between several pairwise DML losses and a generative view of the mutual information between the learned features and labels;
2. Proving explicitly that optimizing the standard cross-entropy corresponds to an approximate bound-optimizer of an underlying pairwise loss;
3. More generally, showing that minimizing the standard cross-entropy loss is equivalent to maximizing a discriminative view of the mutual information between the features and labels.

4. Demonstrating state-of-the-art results with cross-entropy on several DML benchmark datasets.

2.2 On the two views of the mutual information

Table 2.1 Definition of the random variables and information measures used in this paper

General	Labeled dataset	$\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$
	Input feature space	$\mathcal{X}$
	Embedded feature space	$\mathcal{Z} \subset \mathbb{R}^d$
	Label/Prediction space	$\mathcal{Y} \subset \mathbb{R}^K$
	Euclidean distance	$D_{ij} = \left\|\boldsymbol{z}_i - \boldsymbol{z}_j\right\|_2$
	Cosine distance	$D_{ij}^{\cos} = \frac{\boldsymbol{z}_i^\top \boldsymbol{z}_j}{\|\boldsymbol{z}_i\|\|\boldsymbol{z}_j\|}$
Model	Encoder	$\phi_{\mathcal{W}} : \mathcal{X} \to \mathcal{Z}$
	Soft-classifier	$f_\theta : \mathcal{Z} \to [0,1]^K$
Random variables (RVs)	Data	X, Y
	Embedding	$\widehat{Z}\|X \sim \phi_{\mathcal{W}}(X)$
	Prediction	$\widehat{Y}\|\widehat{Z} \sim f_\theta(\widehat{Z})$
Information measures	Entropy of Y	$\mathcal{H}(Y) \coloneqq \mathbb{E}_{p_Y}\left[-\log p_Y(Y)\right]$
	Conditional entropy of Y given Z	$\mathcal{H}(Y\|\widehat{Z}) \coloneqq \mathbb{E}_{p_{Y\widehat{Z}}}\left[-\log p_{Y\|\widehat{Z}}(Y\|\widehat{Z})\right]$
	Cross entropy (CE) between Y and $\widehat{Y}$	$\mathcal{H}(Y;\widehat{Y}) \coloneqq \mathbb{E}_{p_Y}\left[-\log p_{\widehat{Y}}(Y)\right]$
	Conditional CE given $\widehat{Z}$	$\mathcal{H}(Y;\widehat{Y}\|\widehat{Z}) \coloneqq \mathbb{E}_{p_{\widehat{Z}Y}}\left[-\log p_{\widehat{Y}\|\widehat{Z}}(Y\|\widehat{Z})\right]$
	Mutual information between $\widehat{Z}$ and Y	$\mathcal{I}(\widehat{Z};Y) \coloneqq \mathcal{H}(Y) - \mathcal{H}(Y\|\widehat{Z})$

The Mutual Information (MI) is a well known-measure designed to quantify the amount of information shared by two random variables. Its formal definition is presented in Table 2.1. Throughout this work, we will be particularly interested in $\mathcal{I}(\widehat{Z};Y)$ which represents the MI between learned features $\widehat{Z}$ and labels Y. Due to its symmetry property, the MI can be written in two ways, which we will refer to as the *discriminative view* and *generative view* of MI:

$$\mathcal{I}(\widehat{Z};Y) = \underbrace{\mathcal{H}(Y) - \mathcal{H}(Y|\widehat{Z})}_{\text{discriminative view}} = \underbrace{\mathcal{H}(\widehat{Z}) - \mathcal{H}(\widehat{Z}|Y)}_{\text{generative view}} \tag{2.1}$$

While being analytically equivalent, these two views present two different, complementary interpretations. In order to maximize $\mathcal{I}(\widehat{Z};Y)$, the discriminative view conveys that the labels should be balanced (out of our control) and easily identified from the features. On the other hand, the generative view conveys that the features learned should spread as much as possible in the feature space, while keeping samples sharing the same class close to each other. Hence, the discriminative view is more focused on label identification, while the generative view focuses on more explicitly shaping the distribution of the features learned by the model. Therefore, the MI enables us to draw links between classification losses (*e.g.* cross-entropy) and feature-shaping losses (including all the well-known pairwise metric learning losses).

2.3 Pairwise losses and the generative view of the MI

In this section, we study four pairwise losses used in the DML community: center loss (Wen *et al.*, 2016), contrastive loss (Hadsell *et al.*, 2006), Scalable Neighbor Component Analysis (SNCA) loss (Wu *et al.*, 2018) and Multi-Similarity (MS) loss (Wang *et al.*, 2019b). We show that these losses can be interpreted as proxies for maximizing the generative view of mutual information $\mathcal{I}(\widehat{Z};Y)$. We begin by analyzing the specific example of contrastive loss, establishing its tight link to the MI, and further generalize our analysis to the other pairwise losses (see Table 2.2). Furthermore, we show that these pairwise metric-learning losses can be explicitly linked to one another via bound relationships.

2.3.1 The example of contrastive loss

We start by analyzing the representative example of contrastive loss (Hadsell *et al.*, 2006). For a given margin $m \in \mathbb{R}^+$, this loss is formulated as:

$$\mathcal{L}_{\text{contrast}} = \underbrace{\frac{1}{n}\sum_{i=1}^{n}\sum_{j:y_j=y_i} D_{ij}^2}_{T_{\text{contrast}}} + \underbrace{\frac{1}{n}\sum_{i=1}^{n}\sum_{j:y_j\neq y_i}[m - D_{ij}]_+^2}_{C_{\text{contrast}}} \tag{2.2}$$

where $[x]_+ = \max(0, x)$. This loss naturally breaks down into two terms: a *tightness* part T_{contrast} and a *contrastive* part C_{contrast}. The tightness part encourages samples from the same class to be close to each other and form *tight* clusters. As for the *contrastive* part, it forces samples from different classes to stand far apart from one another in the embedded feature space. Let us analyze these two terms from a mutual-information perspective.

As shown in the next subsection, the tightness part of contrastive loss is equivalent to the tightness part of the center loss (Wen *et al.*, 2016): $T_{\text{contrast}} \overset{\text{c}}{=} T_{\text{center}} = \frac{1}{2}\sum_{i=1}^{n} \left\|\boldsymbol{z}_i - \boldsymbol{c}_{y_i}\right\|^2$, where $\boldsymbol{c}_k = \frac{1}{|\mathcal{Z}_k|}\sum_{z\in\mathcal{Z}_k} \boldsymbol{z}$ denotes the mean of feature points from class k in embedding space $\mathcal{Z}$ and symbol $\overset{\text{c}}{=}$ denotes equality up to a multiplicative and/or additive constant. Written in this way, we can interpret T_{contrast} as a conditional cross entropy between $\widehat{Z}$ and another random variable $\bar{Z}$, whose conditional distribution given Y is a standard Gaussian centered around $\boldsymbol{c}_Y$: $\bar{Z}|Y \sim \mathcal{N}(c_Y, I)$:

$$T_{\text{contrast}} \overset{\text{c}}{=} \mathcal{H}(\widehat{Z}; \bar{Z}|Y) = \mathcal{H}(\widehat{Z}|Y) + \mathcal{D}_{KL}(\widehat{Z}||\bar{Z}|Y) \tag{2.3}$$

As such, T_{contrast} is an upper bound on the conditional entropy that appears in the mutual information:

$$T_{\text{contrast}} \geq \mathcal{H}(\widehat{Z}|Y) \tag{2.4}$$

This bound is tight when $\widehat{Z}|Y \sim \mathcal{N}(\boldsymbol{c}_Y, I)$. Hence, minimizing T_{contrast} can be seen as minimizing $\mathcal{H}(\widehat{Z}|Y)$, which exactly encourages the encoder $\phi_{\mathcal{W}}$ to produce low-entropy (=compact) clusters in the feature space for each given class. Notice that using this term only will inevitably lead to a trivial encoder that maps all data points in $\mathcal{X}$ to a single point in the embedded space $\mathcal{Z}$, hence achieving a global optimum.

To prevent such a trivial solution, a second term needs to be added. This second term – that we refer to as the *contrastive* term – is designed to push each point away from points that have a different label. In this term, only pairs such that $D_{ij} \leq m$ produce a cost. Given a pair (i, j), let us define $x = D_{ij}/m$. Given that $x \in [0, 1]$, one can show the following: $1 - 2x \leq (1 - x)^2 \leq 1 - x$.

Using linear approximation $(1-x)^2 \approx 1-2x$ (with error at most x), we obtain:

$$C_{\text{contrast}} \overset{\text{c}}{\approx} -\frac{2m}{n}\sum_{i=1}^{n}\sum_{j:y_j\neq y_i} D_{ij} = -\frac{2m}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} D_{ij} + \frac{2m}{n}\sum_{i=1}^{n}\sum_{j:y_j=y_i} D_{ij} \tag{2.5}$$

While the second term in Equation 2.5 is redundant with the tightness objective, the first term is close to the differential entropy estimator proposed in (Wang & Sha, 2011):

$$\widehat{\mathcal{H}}(\widehat{Z}) = \frac{d}{n(n-1)}\sum_{i=1}^{n}\sum_{j=1}^{n}\log D_{ij}^2 \overset{\text{c}}{=} \sum_{i=1}^{n}\sum_{j=1}^{n}\log D_{ij} \tag{2.6}$$

Both terms measure the spread of $\widehat{Z}$, even though they present different gradient dynamics. All in all, minimizing the whole contrastive loss can be seen as a proxy for maximizing the MI between the labels Y and the embedded features $\widehat{Z}$:

$$\mathcal{L}_{\text{contrast}} = \underbrace{\frac{1}{n}\sum_{i=1}^{n}\sum_{j:y_j=y_i}(D_{ij}^2 + 2mD_{ij})}_{\propto\mathcal{H}(\widehat{Z}|Y)} \underbrace{-\frac{2m}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} D_{ij}}_{\propto\mathcal{H}(\widehat{Z})} \quad \propto \; -\mathcal{I}(\widehat{Z};Y) \tag{2.7}$$

2.3.2 Generalizing to other pairwise losses

A similar analysis can be carried out on other, more recent metric learning losses. More specifically, they can also be broken down into two parts: a *tightness* part that minimizes intra-class distances to form compact clusters, which is related to the *conditional entropy* $\mathcal{H}(\widehat{Z}|Y)$, and a second *contrastive* part that prevents trivial solutions by maximizing inter-class distances, which is related to the *entropy* of features $\mathcal{H}(\widehat{Z})$. Note that, in some pairwise losses, there might be some redundancy between the two terms, *i.e.*, the tightness term also contains some contrastive subterm, and vice-versa. For instance, the cross-entropy loss is used as the contrastive part of the center-loss but, as we show in subsection 2.4.2, the cross-entropy, used alone, already contains both tightness (conditional entropy) and contrastive (entropy) parts. Table 2.2 presents the split for four DML losses. The rest of the section is devoted to exhibiting

Table 2.2 Several well-known and/or recent DML losses broken into a *tightness* term and a *contrastive* term. Minimizing the cross-entropy corresponds to an approximate bound optimization of PCE

Loss	**Tightness part** $\propto \mathcal{H}(\widehat{Z}\|Y)$	**Contrastive part** $\propto \mathcal{H}(\widehat{Z})$
Center (Wen *et al.*, 2016)	$\frac{1}{2}\sum_{i=1}^{n}\left\|z_i - \boldsymbol{c}_{y_i}\right\|^2$	$-\frac{1}{n}\sum_{i=1}^{n}\log p_{iy_i}$
Contrastive (Hadsell *et al.*, 2006)	$\frac{1}{n}\sum_{i=1}^{n}\sum_{j:y_j=y_i} D_{ij}^2$	$\frac{1}{n}\sum_{i=1}^{n}\sum_{j:y_j\neq y_i}[m - D_{ij}]_+^2$
SNCA (Wu *et al.*, 2018)	$-\frac{1}{n}\sum_{i=1}^{n}\log\left[\sum_{j:y_j=y_i}\exp\frac{D_{ij}^{\cos}}{\sigma}\right]$	$\frac{1}{n}\sum_{i=1}^{n}\log\left[\sum_{k\neq i}\exp\frac{D_{ik}^{\cos}}{\sigma}\right]$
Multi-Similarity (Wang *et al.*, 2019b)	$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\alpha}\log\left[1+\sum_{j:y_j=y_i} e^{-\alpha(D_{ij}^{\cos}-m)}\right]$	$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\beta}\log\left[1+\sum_{j:y_j\neq y_i} e^{\beta(D_{ij}^{\cos}-m)}\right]$
PCE Proposition 2.1	$-\frac{1}{2\lambda n^2}\sum_{i=1}^{n}\sum_{j:y_j=y_i} z_i^\top z_j$	$\frac{1}{n}\sum_{i=1}^{n}\log\left[\sum_{k=1}^{K}\exp\left[\frac{1}{\lambda n}\sum_{j=1}^{n} p_{jk} z_i^\top z_j\right]\right] - \frac{1}{2K^2\lambda^2}\sum_{k=1}^{K}\left\|\boldsymbol{c}_k^s\right\|^2$

the close relationships between several pairwise losses and the tightness and contrastive terms (*i.e.* T and C).

Links between losses: In this section, we show that the tightness and contrastive parts of the pairwise losses in Table 2.2, even though different at first sight, can actually be related to one another.

Lemma 2.1. *Let T_A denote the tightness part of the loss from method A. Assuming that features are ℓ_2-normalized, and that classes are balanced, the following relations between Center (Wen* et al.*, 2016), Contrastive (Hadsell* et al.*, 2006), SNCA (Wu* et al.*, 2018) and MS (Wang* et al.*, 2019b) losses hold:*

$$T_{SNCA} \overset{\text{c}}{\leq} T_{Center} \overset{\text{c}}{=} T_{Contrastive} \overset{\text{c}}{\leq} T_{MS} \tag{2.8}$$

Where $\overset{\text{c}}{\leq}$ stands for lower than, up to a multiplicative and an additive constant, and $\overset{\text{c}}{=}$ stands for equal to, up to a multiplicative and an additive constant.

The detailed proof of Lemma 2.1 is deferred to the supplemental material. As for the contrastive parts, we show in the supplemental material that both C_{SNCA} and C_{MS} are lower bounded by a common contrastive term that is directly related to $\mathcal{H}(\hat{Z})$. We do not mention the *contrastive* term of center-loss, as it represents the cross-entropy loss, which is exhaustively studied in section 2.4.

2.4 Cross-entropy does it all

We now completely change gear to focus on the widely used *unary* classification loss: cross-entropy. On the surface, the cross-entropy may seem unrelated to metric-learning losses as it does not involve pairwise distances. We show that a close relationship exists between these pairwise losses widely used in deep metric learning and the cross-entropy classification loss. This link can be drawn from two different perspectives, one is based on an explicit optimization insight and the other is based on a discriminative view of the mutual information. First, we explicitly demonstrate that the cross-entropy is an upper bound on a new pairwise loss, which has a structure similar to all the metric-learning losses listed in Table 2.2, *i.e.*, it contains a tightness term and a contrastive term. Hence, minimizing the cross-entropy can be seen as an approximate *bound-optimization (or Majorize-Minimize)* algorithm for minimizing this pairwise loss. Second, we show that, more generally, minimization of the cross-entropy is actually equivalent to maximization of the mutual information, to which we connected various DML losses. These findings indicate that the cross-entropy represents a proxy for maximizing $\mathcal{I}(\widehat{Z}, Y)$, just like pairwise losses, without the need for dealing with the complex sample mining and optimization schemes associated to the latter.

2.4.1 The pairwise loss behind unary cross-entropy

Bound optimization: Given a function $f(\mathcal{W})$ that is either intractable or hard to optimize, bound optimizers are iterative algorithms that instead optimize auxiliary functions (upper bounds on f). These auxiliary functions are usually more tractable than the original function f. Let t

be the current iteration index, then a_t is an auxiliary function if:

$$\begin{aligned} f(\mathcal{W}) &\leq a_t(\mathcal{W}) \quad ,\forall\, \mathcal{W} \\ f(\mathcal{W}_t) &= a_t(\mathcal{W}_t) \end{aligned} \tag{2.9}$$

A bound optimizer follows a two-step procedure: first an auxiliary function a_t is computed, then a_t is minimized, such that:

$$\mathcal{W}_{t+1} = \arg\min_{\mathcal{W}} a_t(\mathcal{W}) \tag{2.10}$$

This iterative procedure is guaranteed to decrease the original function f:

$$f(\mathcal{W}_{t+1}) \leq a_t(\mathcal{W}_{t+1}) \leq a_t(\mathcal{W}_t) = f(\mathcal{W}_t) \tag{2.11}$$

Note that bound optimizers are widely used in machine learning. Examples of well-known bound optimizers include the concave-convex procedure (CCCP) (Yuille & Rangarajan, 2001), expectation maximization (EM) algorithms or submodular-supermodular procedures (SSP) (Narasimhan & Bilmes, 2005). Such optimizers are particularly used in clustering (Tang, Marin, Ben Ayed & Boykov, 2019) and, more generally, in problems involving latent-variable optimization.

Pairwise Cross-Entropy: We now prove that minimizing cross-entropy can be viewed as an approximate bound optimization of a more complex pairwise loss.

Proposition 2.1. *Alternately minimizing the cross-entropy loss $\mathcal{L}_{CE}$ with respect to the encoder's parameters $\mathcal{W}$ and the classifier's weights θ can be viewed as an approximate bound-optimization of a Pairwise Cross-Entropy (PCE) loss, which we define as follows:*

$$\mathcal{L}_{PCE} = \underbrace{-\frac{1}{2\lambda n^2}\sum_{i=1}^{n}\sum_{j:y_j=y_i} \boldsymbol{z}_i^\top \boldsymbol{z}_j}_{\textit{TIGHTNESS PART}} + \underbrace{\frac{1}{n}\sum_{i=1}^{n}\log\sum_{k=1}^{K} e^{\frac{1}{\lambda n}\sum_{j=1}^{n} p_{jk}\boldsymbol{z}_i^\top \boldsymbol{z}_j} - \frac{1}{2\lambda}\sum_{k=1}^{K}\left\|\boldsymbol{c}_k^s\right\|^2}_{\textit{CONTRASTIVE PART}} \tag{2.12}$$

Where $\boldsymbol{c}_k^s = \frac{1}{n}\sum_{i=1}^{n} p_{ik}\boldsymbol{z}_i$ represents the soft-mean of class k, p_{ik} represents the softmax probability of point $\boldsymbol{z}_i$ belonging to class k, and $\lambda \in \mathbb{R}, \lambda > 0$ depends on the encoder $\phi_{\mathcal{W}}$.

The full proof of Proposition 2.1 is provided in the supplemental material. We hereby provide a quick sketch.

Considering the usual softmax parametrization for our model's predictions $\widehat{Y}$, the idea is to break the cross-entropy loss in two terms, and artificially add and remove the regularization term $\frac{\lambda}{2}\sum_{k=1}^{K}\theta_k^\top\theta_k$:

$$\mathcal{L}_{\text{CE}} = \underbrace{-\frac{1}{n}\sum_{i=1}^{n}\theta_{y_i}^\top \boldsymbol{z}_i + \frac{\lambda}{2}\sum_{k}\theta_k^\top\theta_k}_{f_1(\theta)} + \underbrace{\frac{1}{n}\sum_{i=1}^{n}\log\sum_{k=1}^{K}e^{\theta_k^\top \boldsymbol{z}_i} - \frac{\lambda}{2}\sum_{k=1}^{K}\theta_k^\top\theta_k}_{f_2(\theta)} \tag{2.13}$$

By properly choosing $\lambda \in \mathbb{R}$ in Equation 2.13, both f_1 and f_2 become convex functions of θ. For any class k, we then show that the optimal values of θ_k for f_1 and f_2 are proportional to, respectively, the hard mean $\boldsymbol{c}_k = \frac{1}{|\mathcal{Z}_k|}\sum_{i:y_i=k}\boldsymbol{z}_i$ and the soft mean $\boldsymbol{c}_k^s = \frac{1}{n}\sum_{i=1}^{n}p_{ik}\boldsymbol{z}_i$ of class k. By plugging-in those optimal values, we can lower bound f_1 and f_2 individually in Equation 2.13 and get the result.

Proposition 2.1 casts a new light on the cross-entropy loss by explicitly relating it to a new pairwise loss (PCE), following the intuition that the optimal weights θ^* of the final layer, *i.e.*, the linear classifier, are related to the centroids of each class in the embedded feature space $\mathcal{Z}$. Specifically, finding the optimal classifier's weight θ^* for cross-entropy can be interpreted as building an auxiliary function $a_t(\mathcal{W}) = \mathcal{L}_{\text{CE}}(\mathcal{W}, \theta^*)$ on $\mathcal{L}_{\text{PCE}}(\mathcal{W})$. Subsequently minimizing cross-entropy w.r.t. the encoder's weights $\mathcal{W}$ can be interpreted as the second step of bound optimization on $\mathcal{L}_{\text{PCE}}(\mathcal{W})$. Similarly to other metric learning losses, PCE contains a *tightness* part that encourages samples from the same classes to align with one another. In echo to Lemma 2.1, this tightness term, noted T_{PCE}, is equivalent, up to multiplicative and additive constants, to T_{center} and T_{contrast}, when the features are assumed to be normalized:

$$T_{\text{PCE}} \overset{\text{c}}{=} T_{\text{center}} \overset{\text{c}}{=} T_{\text{contrast}} \tag{2.14}$$

PCE also contains a *contrastive* part, divided into two terms. The first pushes all samples away from one another, while the second term forces soft means c_k^s far from the origin. Hence, minimizing the cross-entropy can be interpreted as implicitly minimizing a pairwise loss whose structure appears similar to the well-established metric-learning losses in Table 2.2.

Simplified Pairwise Cross-Entropy: While PCE brings interesting theoretical insights, the computation of the parameter λ at every iteration requires computating the eigenvalues of a $d \times d$ matrix at every iteration (cf. full proof in supplemental material), which makes the implementation of PCE difficult in practice. In order to remove the dependence upon λ, one can plug in the same θ for both f_1 and f_2 in Equation 2.13. We choose to use $\theta_1^* = \arg\min_\theta f_1(\theta) \propto [\boldsymbol{c}_1, ..., \boldsymbol{c}_K]^\top$. This yields a simplified version of PCE, that we call SPCE:

$$\mathcal{L}_{\text{SPCE}} = \underbrace{-\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j:y_j=y_i} \boldsymbol{z}_i^\top \boldsymbol{z}_j}_{\text{TIGHTNESS}} + \underbrace{\frac{1}{n}\sum_{i=1}^{n}\log\sum_{k=1}^{K}\exp\left(\frac{1}{n}\sum_{j:y_j=k}\boldsymbol{z}_i^\top \boldsymbol{z}_j\right)}_{\text{CONTRASTIVE}} \tag{2.15}$$

SPCE and PCE are similar (the difference is that PCE was derived after plugging in the soft means instead of hard means in f_2). Contrary to PCE, however, SPCE is easily computable, and the preliminary experiments we provide in the supplementary material indicate that CE and SPCE exhibit similar behaviors at training time. Interestingly, our derived SPCE loss has a form similar to contrastive learning losses in unsupervised representation learning (Oord, Li & Vinyals, 2018; Tschannen, Djolonga, Rubenstein, Gelly & Lucic, 2020; Chen, Kornblith, Norouzi & Hinton, 2020).

2.4.2 A discriminative view of mutual information

Lemma 2.2. *Minimizing the conditional cross-entropy loss, denoted by* $\mathcal{H}(Y;\widehat{Y}|\widehat{Z})$*, is equivalent to maximizing the mutual information* $\mathcal{I}(\widehat{Z};Y)$.

The proof of Lemma 2.2 is provided in the supplementary material. Such result is compelling. Using the discriminative view of mutual information allows to show that minimizing cross-

entropy loss is equivalent to maximizing the mutual information $\mathcal{I}(\widehat{Z};Y)$. This information theoretic argument reinforces our conclusion from Proposition 2.1 that cross-entropy and the previously described metric learning losses are essentially doing the same job.

2.4.3 Then why would cross-entropy work better?

We showed that cross-entropy essentially optimizes the same underlying mutual information $\mathcal{I}(\widehat{Z};Y)$ as other DML losses. This fact alone is not enough to explain why the cross-entropy is able to consistently achieve better results than DML losses as shown in section 2.5. We argue that the difference is in the optimization process. On the one hand, pairwise losses require careful sample mining and weighting strategies to obtain the most informative pairs, especially when considering mini-batches, in order to achieve convergence in a reasonable amount of time, using a reasonable amount of memory. On the other hand, optimizing cross-entropy is substantially easier as it only implies minimization of unary terms. Essentially, cross-entropy does it all without dealing with the difficulties of pairwise terms. Not only it makes optimization easier, but also it simplifies the implementation, thus increasing its potential applicability in real-world problems.

2.5 Experiments

2.5.1 Metric

Most methods, especially recent ones, use the cosine distance to compute the recall for the evaluation. They include ℓ_2 normalization of the features in the model (Oh Song, Jegelka, Rathod & Murphy, 2017; Movshovitz-Attias *et al.*, 2017; Wang, Zhou, Wen, Liu & Lin, 2017; Opitz, Waltner, Possegger & Bischof, 2017; Ge, 2018; Yuan, Yang & Zhang, 2017; Xuan, Stylianou & Pless, 2020; Zhai & Wu, 2019; Wang *et al.*, 2019b; Sanakoyeu, Tschernezki, Buchler & Ommer, 2019; Xuan, Souvenir & Pless, 2018), which makes cosine and Euclidean distances equivalent. Computing cosine similarity is also more memory efficient and typically leads to better results (Schroff, Kalenichenko & Philbin, 2015). For these reasons, the Euclidean

distance on non normalized features has rarely been used for both training and evaluation. In our experiments, ℓ_2-normalization of the features during training actually hindered the final performance, which might be explained by the fact that we add a classification layer on top of the feature extractor. Thus, we did not ℓ_2-normalize the features during training and reported the recall with both Euclidean and cosine distances.

2.5.2 Datasets

Table 2.3 Summary of the datasets used for evaluation in metric learning

Name	Objects	Categories	Images
Caltech-UCSD Birds-200-2011 (CUB)	Birds	200	11 788
Cars Dataset	Cars	196	16 185
Stanford Online Products (SOP)	House furniture	22 634	120 053
In-shop Clothes Retrieval	Clothes	7 982	52 712

Four datasets are commonly used in metric learning to evaluate the performances. These datasets are summarized in Table 2.3. CUB (Wah, Branson, Welinder, Perona & Belongie, 2011), Cars (Krause, Stark, Deng & Fei-Fei, 2013) and SOP (Song *et al.*, 2016) datasets are divided into train and evaluation splits. For the evaluation, the recall is computed between each sample of the evaluation set and the rest of the set. In-Shop (Liu, Luo, Qiu, Wang & Tang, 2016) is divided into a query and a gallery set. The recall is computed between each sample of the query set and the whole gallery set.

2.5.3 Training specifics

2.5.3.1 Model architecture and pre-training

In the metric learning literature, several architectures have been used, which historically correspond to the state-of-the-art image classification architectures on ImageNet (Deng *et al.*, 2009), with an additional constraint on model size (*i.e.*, the ability to train on one or two GPUs in a reasonable time). These include GoogLeNet (Szegedy *et al.*, 2015) as in (Kim, Goyal,

Chawla, Lee & Kwon, 2018), BatchNorm-Inception (Szegedy *et al.*, 2016) as in (Wang *et al.*, 2019b) and ResNet-50 (He, Zhang, Ren & Sun, 2016b) as in (Xuan *et al.*, 2020). They have large differences in classification performances on ImageNet, but the impact on performances over DML benchmarks has rarely been studied in controlled experiments. As this is not the focus of our paper, we use ResNet-50 for our experiments. We concede that one may obtain better performances by modifying the architecture (*e.g.*, reducing model stride and performing multi-level fusion of features). Here, we limit our comparison to standard architectures. Our implementation uses the PyTorch (Paszke *et al.*, 2019) library, and initializes the ResNet-50 model with weights pre-trained on ImageNet.

2.5.3.2 Sampling

To the best of our knowledge, all DML papers – including (Zhai & Wu, 2019) – use a form of pairwise sampling to ensure that, during training, each mini-batch contains a fixed number of classes and samples per class (*e.g.* mini-batch size of 75 with 3 classes and 25 samples per class in (Zhai & Wu, 2019)). Deviating from that, we use the common random sampling among all samples (as in most classification training schemes) and set the mini-batch size to 128 in all experiments (contrary to (Wang *et al.*, 2019b) in which the authors use a mini-batch size of 80 for CUB, 1 000 for SOP and did not report for Cars and In-Shop).

2.5.3.3 Data Augmentation

As is common in training deep learning models, data augmentation improves the final performances of the methods. For CUB, the images are first resized so that their smallest side has a length of 256 (*i.e.*, keeping the aspect ratio) while for Cars, SOP and In-Shop, the images are resized to 256×256. Then a patch is extracted at a random location and size, and resized to 224×224. For CUB and Cars, we found that random jittering of the brightness, contrast and saturation slightly improves the results. All of the implementation details can be found in the publicly available code.

2.5.3.4 Cross-entropy

The focus of our experiments is to show that, with careful tuning, it is possible to obtain similar or better performance than most recent DML methods, while using only the cross-entropy loss. To train with the cross-entropy loss, we add a linear classification layer (with bias) on top of the feature extraction – similar to many classification models – which produces logits for all the classes present in the training set. Both the weights and biases of this classification layer are initialized to $\mathbf{0}$. We also add dropout with a probability of 0.5 before this classification layer. To further reduce overfitting, we use label smoothing for the target probabilities of the cross-entropy. We set the probability of the true class to $1 - \epsilon$ and the probabilities of the other classes to $\frac{\epsilon}{K-1}$ with $\epsilon = 0.1$ in all our experiments.

2.5.3.5 Optimizer

In most DML papers, the hyper-parameters of the optimizer are the same for Cars, SOP and In-Shop whereas, for CUB, the methods typically use a smaller learning rate. In our experiments, we found that the best results were obtained by tuning the learning rate on a per dataset basis. In all experiments, the models are trained with SGD with Nesterov acceleration and a weight decay of 0.0005, which is applied to convolution and fully-connected layers' weights (but not to biases) as in (Jia *et al.*, 2018). For CUB and Cars, the learning rate is set to 0.02 and 0.05 respectively, with 0 momentum. For both SOP and In-Shop, the learning rate is set to 0.003 with a momentum of 0.99.

2.5.3.6 Batch normalization

Following (Wang *et al.*, 2019b), we freeze all the batch normalization layers in the feature extractor. For Cars, SOP and In-Shop, we found that adding batch normalization – without scaling and bias – on top of the feature extractor improves our final performance and reduces the gap between ℓ_2 and cosine distances when computing the recall. On CUB, however, we obtained the best recall without this batch normalization.

2.5.4 Results

Results for the experiments are reported in Table 2.4. We also report the architecture used in the experiments as well as the distance used in the evaluation to compute the recall. ℓ_2 refers to the Euclidean distance on non normalized features while *cos* refers to either the cosine distance or the Euclidean distance on ℓ_2-normalized features, both of which are equivalent.

On all datasets, we report state-of-the-art results except on Cars, where the only method achieving similar recall uses cross-entropy for training. We also notice that, contrary to common beliefs, using Euclidean distance can actually be competitive as it also achieves near state-of-the-art results on all four datasets. These results clearly highlight the potential of cross-entropy for metric learning, and confirm that this loss can achieve the same objective as pairwise losses.

2.6 Conclusion

Throughout this paper, we revealed non-obvious relations between the cross-entropy loss, widely adopted in classification tasks, and pairwise losses commonly used in DML. These relations were drawn under two different perspectives. First, cross-entropy minimization was shown equivalent to an approximate bound-optimization of a pairwise loss, introduced as Pairwise Cross-Entropy (PCE), which appears similar in structure to already existing DML losses. Second, adopting a more general information theoretic view of DML, we showed that both pairwise losses and cross-entropy were, in essence, maximizing a common mutual information $\mathcal{I}(\hat{Z}, Y)$ between the embedded features and the labels. This connection becomes particularly apparent when writing mutual information in both its *generative* and *discriminative* views. Hence, we argue that most of the differences in performance observed in previous works come from the optimization process during training. Cross-entropy contains only unary terms, while traditional DML losses are based on pairwise-term optimization, which requires substantially more tuning (*e.g.* mini-batch size, sampling strategy, pair weighting). While we acknowledge that some losses have better properties than others regarding optimization, we empirically showed that the cross-entropy loss was also able to achieve state-of-the-art results when fairly tuned, highlighting the fact that most

Table 2.4 Performance on CUB200, Cars-196, SOP and In-Shop datasets. d refers to the distance used to compute the recall when evaluating

	Method	d	Architecture	Recall at					
Caltech-UCSD Birds-200-2011				1	2	4	8	16	32
	Lifted Structure (Song *et al.*, 2016)	ℓ_2	GoogLeNet	47.2	58.9	70.2	80.2	89.3	93.2
	Proxy-NCA (Movshovitz-Attias *et al.*, 2017)	cos	BN-Inception	49.2	61.9	67.9	81.9	-	-
	HTL (Ge, 2018)	cos	GoogLeNet	57.1	68.8	78.7	86.5	92.5	95.5
	ABE (Kim *et al.*, 2018)	cos	GoogLeNet	60.6	71.5	79.8	87.4	–	–
	HDC (Yuan *et al.*, 2017)	cos	GoogLeNet	60.7	72.4	81.9	89.2	93.7	96.8
	DREML (Xuan *et al.*, 2018)	cos	ResNet-18	63.9	75.0	83.1	89.7	–	–
	EPSHN (Xuan *et al.*, 2020)	cos	ResNet-50	64.9	75.3	83.5	–	–	–
	NormSoftmax (Zhai & Wu, 2019)	cos	ResNet-50	65.3	76.7	85.4	91.8	–	–
	Multi-Similarity (Wang *et al.*, 2019b)	cos	BN-Inception	65.7	77.0	86.6	91.2	95.0	97.3
	D&C (Sanakoyeu *et al.*, 2019)	cos	ResNet-50	65.9	76.6	84.4	90.6	–	–
	Cross-Entropy	ℓ_2	ResNet-50	67.6	78.1	85.6	91.1	94.7	97.2
		cos		69.2	79.2	86.9	91.6	95.0	97.3
Stanford Cars				1	2	4	8	16	32
	Lifted Structure (Song *et al.*, 2016)	ℓ_2	GoogLeNet	49.0	60.3	72.1	81.5	89.2	92.8
	Proxy-NCA (Movshovitz-Attias *et al.*, 2017)	cos	BN-Inception	73.2	82.4	86.4	88.7	–	–
	HTL (Yuan *et al.*, 2017)	cos	GoogLeNet	81.4	88.0	92.7	95.7	97.4	99.0
	EPSHN (Xuan *et al.*, 2020)	cos	ResNet-50	82.7	89.3	93.0	–	–	–
	HDC (Yuan *et al.*, 2017)	cos	GoogLeNet	83.8	89.8	93.6	96.2	97.8	98.9
	Multi-Similarity (Wang *et al.*, 2019b)	cos	BN-Inception	84.1	90.4	94.0	96.5	98.0	98.9
	D&C (Sanakoyeu *et al.*, 2019)	cos	ResNet-50	84.6	90.7	94.1	96.5	–	–
	ABE (Kim *et al.*, 2018)	cos	GoogLeNet	85.2	90.5	94.0	96.1	–	–
	DREML (Xuan *et al.*, 2018)	cos	ResNet-18	86.0	91.7	95.0	97.2	–	–
	NormSoftmax (Zhai & Wu, 2019)	cos	ResNet-50	89.3	94.1	96.4	98.0	–	–
	Cross-Entropy	ℓ_2	ResNet-50	89.1	93.7	96.5	98.1	99.0	99.4
		cos		89.3	93.9	96.6	98.4	99.3	99.7
Stanford Online Product				1		10	100	1000	
	Lifted Structure (Song *et al.*, 2016)	ℓ_2	GoogLeNet	62.1		79.8	91.3	97.4	
	HDC (Yuan *et al.*, 2017)	cos	GoogLeNet	70.1		84.9	93.2	97.8	
	HTL (Ge, 2018)	cos	GoogLeNet	74.8		88.3	94.8	98.4	
	D&C (Sanakoyeu *et al.*, 2019)	cos	ResNet-50	75.9		88.4	94.9	98.1	
	ABE (Kim *et al.*, 2018)	cos	GoogLeNet	76.3		88.4	94.8	98.2	
	Multi-Similarity (Wang *et al.*, 2019b)	cos	BN-Inception	78.2		90.5	96.0	98.7	
	EPSHN (Xuan *et al.*, 2020)	cos	ResNet-50	78.3		90.7	96.3	–	
	NormSoftmax (Zhai & Wu, 2019)	cos	ResNet-50	79.5		91.5	96.7	–	
	Cross-Entropy	ℓ_2	ResNet-50	80.8		91.2	95.7	98.1	
		cos		81.1		91.7	96.3	98.8	
In-Shop Clothes Retrieval				1	10	20	30	40	50
	HDC (Yuan *et al.*, 2017)	cos	GoogLeNet	62.1	84.9	89.0	91.2	92.3	93.1
	DREML (Xuan *et al.*, 2018)	cos	ResNet-18	78.4	93.7	95.8	96.7	–	–
	HTL (Ge, 2018)	cos	GoogLeNet	80.9	94.3	95.8	97.2	97.4	97.8
	D&C (Sanakoyeu *et al.*, 2019)	cos	ResNet-50	85.7	95.5	96.9	97.5	–	98.0
	ABE (Kim *et al.*, 2018)	cos	GoogLeNet	87.3	96.7	97.9	98.2	98.5	98.7
	EPSHN (Xuan *et al.*, 2020)	cos	ResNet-50	87.8	95.7	96.8	–	–	–
	NormSoftmax (Zhai & Wu, 2019)	cos	ResNet-50	89.4	97.8	98.7	99.0	–	–
	Multi-Similarity (Wang *et al.*, 2019b)	cos	BN-Inception	89.7	97.9	98.5	98.8	99.1	99.2
	Cross-Entropy	ℓ_2	ResNet-50	90.6	97.8	98.5	98.8	98.9	99.0
		cos		90.6	98.0	98.6	98.9	99.1	99.2

improvements have come from enhanced training schemes (*e.g*. data augmentation, learning rate policies, batch normalization freeze) rather than the intrinsic properties of pairwise losses. We strongly advocate that cross-entropy should be carefully tuned to be compared against as a baseline in future works.

CHAPTER 3

DECOUPLING DIRECTION AND NORM FOR EFFICIENT GRADIENT-BASED ℓ_2 ADVERSARIAL ATTACKS AND DEFENSES

Jérôme Rony*[1] , Luiz G. Hafemann*[1] , Luiz S. Oliveira[2] , Ismail Ben Ayed[1] , Robert Sabourin[1] , Eric Granger[1]

* Equal contribution
[1] Systems Engineering Department, École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada

[2] Department of Informatics, Federal University of Parana,
Curitiba, PR, Brazil

Paper published at the
IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019

Abstract

Research on adversarial examples in computer vision tasks has shown that small, often imperceptible changes to an image can induce misclassification, which has security implications for a wide range of image processing systems. Considering ℓ_2 norm distortions, the Carlini and Wagner attack is presently the most effective white-box attack in the literature. However, this method is slow since it performs a line-search for one of the optimization terms, and often requires thousands of iterations. In this paper, an efficient approach is proposed to generate gradient-based attacks that induce misclassifications with low ℓ_2 norm, by decoupling the direction and the norm of the adversarial perturbation that is added to the image. Experiments conducted on the MNIST, CIFAR-10 and ImageNet datasets indicate that our attack achieves comparable results to the state-of-the-art (in terms of ℓ_2 norm) with considerably fewer iterations (as few as 100 iterations), which opens the possibility of using these attacks for adversarial training. Models trained with our attack achieve state-of-the-art robustness against white-box gradient-based ℓ_2 attacks on the MNIST and CIFAR-10 datasets, outperforming the Madry defense when the attacks are limited to a maximum norm.

3.1 Introduction

Deep neural networks have achieved state-of-the-art performances on a wide variety of computer vision applications, such as image classification, object detection, tracking, and activity recognition (Gu *et al.* (2018)). In spite of their success in addressing these challenging tasks, they are vulnerable to active *adversaries*. Most notably, they are susceptible to *adversarial examples*[5], in which adding small perturbations to an image, often imperceptible to a human observer, causes a misclassification (Biggio & Roli (2018); Szegedy *et al.* (2014)).

Recent research on adversarial examples developed *attacks* that allow for evaluating the robustness of models, as well as *defenses* against these attacks. Attacks have been proposed to achieve different objectives, such as minimizing the amount of noise that induces misclassification (Carlini & Wagner (2017); Szegedy *et al.* (2014)), or being fast enough to be incorporated into the training procedure (Goodfellow *et al.* (2015); Tramèr *et al.* (2018)). In particular, considering the case of obtaining adversarial examples with lowest perturbation (measured by its ℓ_2 norm), the state-of-the-art attack has been proposed by Carlini and Wagner (C&W) (Carlini & Wagner (2017)). While this attack generates adversarial examples with low ℓ_2 noise, it also requires a high number of iterations, which makes it impractical for training a robust model to defend against such attacks. In contrast, one-step attacks are fast to generate, but using them for training does not increase model robustness on white-box scenarios, with full knowledge of the model under attack (Tramèr *et al.* (2018)). Developing an attack that finds adversarial examples with low noise in few iterations would enable adversarial training with such examples, which could potentially increase model robustness against white-box attacks.

Developing attacks that minimize the norm of the adversarial perturbations requires optimizing two objectives: 1) obtaining a low ℓ_2 norm, while 2) inducing a misclassification. With the current state-of-the-art method, C&W (Carlini & Wagner (2017)), this is addressed by using a two-term loss function, with the weight balancing the two competing objectives found via an

[5] This also affects other machine learning classifiers, but we restrict our analysis to CNNs, that are most commonly used in computer vision tasks.

expensive line search, requiring a large number of iterations. This makes the evaluation of a system's robustness very slow and it is unpractical for adversarial training.

In this paper, we propose an efficient gradient-based attack called *Decoupled Direction and Norm*[6] (DDN) that induces misclassification with a low ℓ_2 norm. This attack optimizes the cross-entropy loss, and instead of penalizing the norm in each iteration, projects the perturbation onto a ℓ_2-sphere centered at the original image. The change in norm is then based on whether the sample is adversarial or not. Using this approach to decouple the direction and norm of the adversarial noise leads to an attack that needs significantly fewer iterations, achieving a level of performance comparable to state-of-the-art, while being amenable to be used for adversarial training.

A comprehensive set of experiments was conducted using the MNIST, CIFAR-10 and ImageNet datasets. Our attack obtains comparable results to the state-of-the-art while requiring much fewer iterations (~100 times less than C&W). For untargeted attacks on the ImageNet dataset, our attack achieves better performance than the C&W attack, taking less than 10 minutes to attack 1 000 images, versus over 35 hours to run the C&W attack.

Results for adversarial training on the MNIST and CIFAR-10 datasets indicate that DDN can achieve state-of-the-art robustness compared to the Madry defense (Madry *et al.* (2018)). These models require that attacks use a higher average ℓ_2 norm to induce misclassifications. They also obtain a higher accuracy when the ℓ_2 norm of the attacks is bounded. On MNIST, if the attack norm is restricted to 1.5, the model trained with the Madry defense achieves 67.3% accuracy, while our model achieves 87.2% accuracy. On CIFAR-10, for attacks restricted to a norm of 0.5, the Madry model achieves 56.1% accuracy, compared to 67.6% in our model.

3.2 Related Work

In this section, we formalize the problem of adversarial examples, the threat model, and review the main attack and defense methods proposed in the literature.

[6] Code available at https://github.com/jeromerony/fast_adversarial.

3.2.1 Problem Formulation

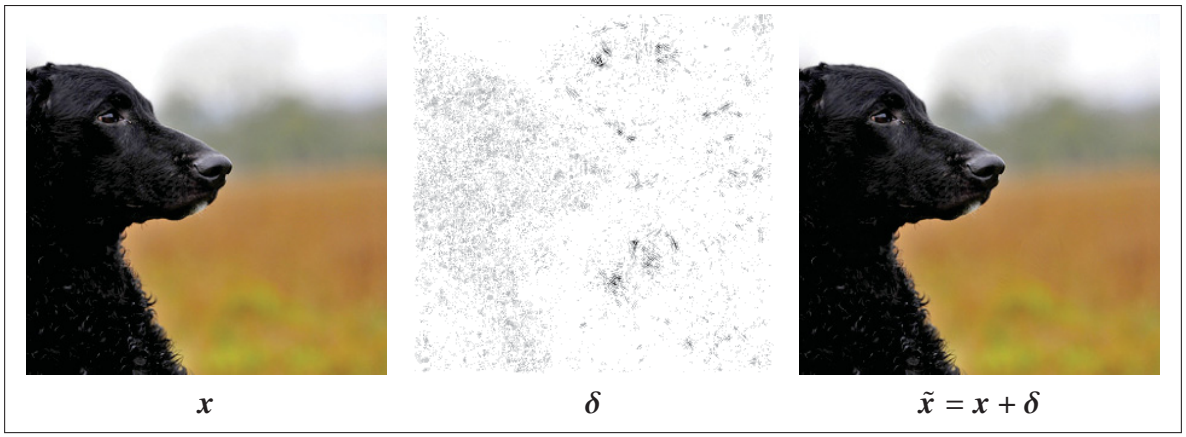


Figure 3.1 Example of an adversarial image on the ImageNet dataset. The sample $\boldsymbol{x}$ is recognized as a Curly-coated retriever. Adding a perturbation $\boldsymbol{\delta}$ we obtain an adversarial image $\tilde{\boldsymbol{x}}$ that is classified as a microwave (with $\|\boldsymbol{\delta}\|_2 = 0.7$)

Let $\boldsymbol{x}$ be an sample from the input space $\mathcal{X}$, with label y_{true} from a set of possible labels $\mathcal{Y}$. Let $D(\boldsymbol{x}_1, \boldsymbol{x}_2)$ be a distance measure that compares two input samples (ideally capturing their perceptual similarity). $P(y|\boldsymbol{x}, \theta) = f(\boldsymbol{x}; \theta)_y$ is a model (classifier) parameterized by θ. An example $\tilde{\boldsymbol{x}} \in \mathcal{X}$ is called *adversarial* (for non-targeted attacks) against the classifier if $\arg\max_j P(y_j|\tilde{\boldsymbol{x}}, \theta) \neq y_{\text{true}}$ and $D(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \leq \epsilon$, for a given maximum perturbation ϵ. A *targeted attack* with a given desired class y_{target} further requires that $\arg\max_j P(y_j|\tilde{\boldsymbol{x}}, \theta) = y_{\text{target}}$. We denote as $J(\boldsymbol{x}, y, \theta)$, the cross-entropy between the prediction of the model for an input $\boldsymbol{x}$ and a label y. Figure 3.1 illustrates a targeted attack on the ImageNet dataset, against an Inception v3 model (Szegedy *et al.*, 2016).

In this paper, attacks are considered to be generated by a gradient-based optimization procedure, restricting our analysis to differentiable classifiers. These attacks can be formulated either to obtain a minimum distortion $D(\boldsymbol{x}, \tilde{\boldsymbol{x}})$, or to obtain the worst possible loss in a region $D(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \leq \epsilon$. As an example, consider that the distance function is a norm (*e.g.* ℓ_0, ℓ_2 or L_∞), and the inputs are images (where each pixel's value is constrained between 0 and M). In a white-box scenario, the optimization procedure to obtain an non-targeted attack with minimum distortion $\boldsymbol{\delta}$ can be

formulated as:

$$\begin{aligned} \underset{\boldsymbol{\delta}}{\text{minimize}} \quad \|\boldsymbol{\delta}\| \quad \text{subject to} \quad & \arg\max_{j} P(y_j|\boldsymbol{x}+\boldsymbol{\delta},\theta) \neq y_{\text{true}} \\ & 0 \leq \boldsymbol{x}+\boldsymbol{\delta} \leq M \end{aligned} \tag{3.1}$$

With a similar formulation for *targeted attacks*, by changing the constraint to be equal to the target class.

If the objective is to obtain the worst possible loss for a given maximum noise of norm ϵ, the problem can be formulated as:

$$\begin{aligned} \underset{\boldsymbol{\delta}}{\text{minimize}} \quad P(y_{\text{true}}|\boldsymbol{x}+\boldsymbol{\delta},\theta) \quad \text{subject to} \quad & \|\boldsymbol{\delta}\| \leq \epsilon \\ & 0 \leq \boldsymbol{x}+\boldsymbol{\delta} \leq M \end{aligned} \tag{3.2}$$

With a similar formulation for *targeted attacks*, by maximizing $P(y_{\text{target}}|\boldsymbol{x}+\boldsymbol{\delta},\theta)$.

We focus on gradient-based attacks that optimize the ℓ_2 norm of the distortion. While this distance does not perfectly capture perceptual similarity, it is widely used in computer vision to measure similarity between images (*e.g.* comparing image compression algorithms, where Peak Signal-to-Noise Ratio is used, which is directly related to the ℓ_2 measure). A differentiable distance measure that captures perceptual similarity is still an open research problem.

3.2.2 Threat Model

In this paper, a *white-box* scenario is considered, also known as a Perfect Knowledge scenario (Biggio & Roli (2018)). In this scenario, we consider that an attacker has perfect knowledge of the system, including the neural network architecture and the learned weights θ. This threat model serves to evaluate system security under the *worst case* scenario. Other scenarios can be conceived to evaluate attacks under different assumptions on the attacker's knowledge, for instance, no access to the trained model, no access to the same training set, among others. These scenarios are referred as *black-box* or Limited-Knowledge (Biggio & Roli (2018)).

3.2.3 Attacks

Several attacks were proposed in the literature, either focusing on obtaining adversarial examples with a small $\boldsymbol{\delta}$ (Equation 3.1) (Carlini & Wagner (2017); Moosavi-Dezfooli *et al.* (2016); Szegedy *et al.* (2014)), or on obtaining adversarial examples in one (or few) steps for adversarial training (Goodfellow *et al.* (2015); Kurakin *et al.* (2017a)).

L-BFGS. Szegedy *et al.* (2014) proposed an attack for minimally distorted examples (Equation 3.1), by considering the following approximation:

$$\underset{\boldsymbol{\delta}}{\text{minimize}} \quad C\,\|\boldsymbol{\delta}\|_2 + \log P(y_{\text{true}}|\boldsymbol{x}+\boldsymbol{\delta},\theta) \quad \text{subject to} \quad 0 \leq \boldsymbol{x}+\boldsymbol{\delta} \leq M \tag{3.3}$$

where the constraint $\boldsymbol{x}+\boldsymbol{\delta} \in [0,M]^n$ was addressed by using a box-constrained optimizer (L-BFGS: Limited memory Broyden–Fletcher–Goldfarb–Shanno), and a line-search to find an appropriate value of C.

FGSM. Goodfellow *et al.* (2015) proposed the Fast Gradient Sign Method, a one-step method that could generate adversarial examples. The original formulation was developed considering the L_∞ norm, but it has also been used to generate attacks that focus on the ℓ_2 norm as follows:

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} + \epsilon \frac{\nabla_{\boldsymbol{x}} J(\boldsymbol{x}, y, \theta)}{\|\nabla_{\boldsymbol{x}} J(\boldsymbol{x}, y, \theta)\|} \tag{3.4}$$

where the constraint $\tilde{\boldsymbol{x}} \in [0,M]^n$ was addressed by simply clipping the resulting adversarial example.

DeepFool. This method considers a linear approximation of the model, and iteratively refines an adversary example by choosing the point that would cross the decision boundary under this approximation. This method was developed for untargeted attacks, and for any L_p norm (Moosavi-Dezfooli *et al.* (2016)).

C&W. Similarly to the L-BFGS method, the C&W ℓ_2 attack (Carlini & Wagner (2017)) minimizes two criteria at the same time – the perturbation that makes the sample adversarial

(*e.g.* misclassified by the model), and the ℓ_2 norm of the perturbation. Instead of using a box-constrained optimization method, they propose changing variables using the tanh function, and instead of optimizing the cross-entropy of the adversarial example, they use a difference between logits. For a targeted attack aiming to obtain class t, with Z denoting the model output before the softmax activation (logits), it optimizes:

$$\begin{aligned} &\underset{\boldsymbol{\delta}}{\text{minimize}} \quad \|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_2^2 + Cg(\tilde{\boldsymbol{x}}) \\ &\text{where} \quad g(\tilde{\boldsymbol{x}}) = \max(\max_{i \neq t}\{f(\tilde{\boldsymbol{x}})_i\} - f(\tilde{\boldsymbol{x}})_t, -\kappa) \\ &\text{and} \quad \tilde{\boldsymbol{x}} = \frac{1}{2}(\tanh(\operatorname{arctanh}(\boldsymbol{x}) + \boldsymbol{\delta}) + 1) \end{aligned} \tag{3.5}$$

where $f(\tilde{\boldsymbol{x}})_i$ denotes the logit corresponding to the i-th class. By increasing the confidence parameter κ, the adversarial sample will be misclassified with higher confidence. To use this attack in the untargeted setting, the definition of g is modified to $g(\tilde{\boldsymbol{x}}) = \max(f(\tilde{\boldsymbol{x}})_y - \max_{i \neq y}\{f(\tilde{\boldsymbol{x}})_i\}, -\kappa)$ where y is the original label.

3.2.4 Defenses

Developing defenses against adversarial examples is an active area of research. To some extent, there is an *arms race* on developing defenses and attacks that break them. Goodfellow *et al*. proposed a method called *adversarial training* (Goodfellow *et al.* (2015)), in which the training data is augmented with FGSM samples. This was later shown not to be robust against iterative white-box attacks, nor black-box single-step attacks (Tramèr *et al.* (2018)). Papernot, McDaniel, Wu, Jha & Swami (2016b) proposed a *distillation* procedure to train robust networks, which was shown to be easily broken by iterative white-box attacks (Carlini & Wagner (2017)). Other defenses involve *obfuscated gradients* (Athalye *et al.* (2018)), where models either incorporate non-differentiable steps (such that the gradient cannot be computed) (Buckman, Roy, Raffel & Goodfellow (2018); Guo, Rana, Cisse & van der Maaten (2018)), or randomized elements (to induce incorrect estimations of the gradient) (Dhillon *et al.* (2018); Xie, Wang, Zhang, Ren & Yuille (2018)). These defenses were later shown to be ineffective when attacked

with Backward Pass Differentiable Approximation (BPDA) (Athalye *et al.* (2018)), where the actual model is used for forward propagation, and the gradient in the backward-pass is approximated. The Madry defense (Madry *et al.* (2018)), which considers a worst-case optimization, is the only defense that has been shown to be somewhat robust (on the MNIST and CIFAR-10 datasets). Below we provide more detail on the general approach of adversarial training, and the Madry defense.

Adversarial Training. This defense considers augmenting the training objective with adversarial examples (Goodfellow *et al.* (2015)), with the intention of improving robustness. Given a model with loss function $J(\boldsymbol{x}, y, \theta)$, training is augmented as follows:

$$\tilde{J}(\boldsymbol{x}, y, \theta) = \alpha J(\boldsymbol{x}, y, \theta) + (1 - \alpha) J(\tilde{\boldsymbol{x}}, y, \theta) \tag{3.6}$$

where $\tilde{\boldsymbol{x}}$ is an adversarial sample. In (Goodfellow *et al.* (2015)), the FGSM is used to generate the adversarial example in a single step. Tramèr *et al.* (2018) extended this method, showing that generating one-step attacks using the model under training introduced an issue. The model can converge to a degenerate solution where its gradients produce "easy" adversarial samples, causing the adversarial loss to have a limited influence on the training objective. They proposed a method in which an ensemble of models is also used to generate the adversarial examples $\tilde{\boldsymbol{x}}$. This method displays some robustness against black-box attacks using surrogate models, but does not increase robustness in white-box scenarios.

Madry Defense. Madry *et al.* (2018) proposed a saddle point optimization problem, in which we optimize for the worst case:

$$\begin{aligned} &\underset{\theta}{\text{minimize}} \quad p(\theta) \\ &\text{where} \quad p(\theta) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\Big[\max_{\boldsymbol{\delta}\in\mathcal{S}} J(\boldsymbol{x}+\boldsymbol{\delta}, y, \theta)\Big] \end{aligned} \tag{3.7}$$

where $\mathcal{D}$ is the training set, and $\mathcal{S}$ indicates the feasible region for the attacker (*e.g.* $\mathcal{S} = \{\boldsymbol{\delta} : \|\boldsymbol{\delta}\| < \epsilon\}$). They show that Equation 3.7 can be optimized by stochastic gradient descent – during each training iteration, it first finds the adversarial example that maximizes the loss

around the current training sample x (*i.e.* maximizing the loss over $\boldsymbol{\delta}$, which is equivalent to minimizing the probability of the correct class as in Equation 3.2), and then, it minimizes the loss over θ. Experiments by Athalye *et al.* (2018) show that it was the only defense not broken under white-box attacks.

3.3 Decoupled Direction and Norm Attack

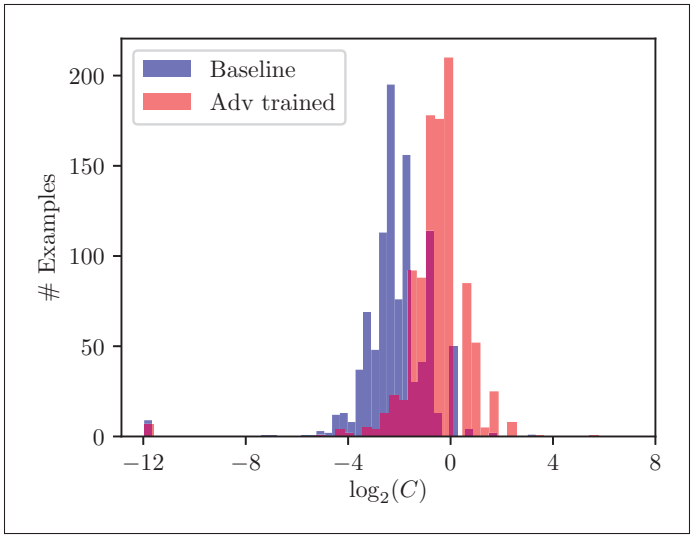


Figure 3.2 Histogram of the best C found by the C&W algorithm with 9 search steps on the MNIST dataset

From the problem definition, we see that finding the worst adversary in a fixed region is an easier task. In Equation 3.2, both constraints can be expressed in terms of $\boldsymbol{\delta}$, and the resulting equation can be optimized using projected gradient descent. Finding the closest adversarial example is harder: Equation 3.1 has a constraint on the prediction of the model, which cannot be addressed by a simple projection. A common approach, which is used by Szegedy *et al.* (2014) and Carlini & Wagner (2017) is to approximate the constrained problem in Equation 3.1 by an unconstrained one, replacing the constraint with a *penalty*. This amounts to jointly optimizing both terms, the norm of $\boldsymbol{\delta}$ and a classification term (see Eq. 3.3 and 3.5), with a sufficiently high parameter C. In the general context of constrained optimization, such a penalty-based approach is a well known general principle (Jensen & Bard (2002)). While tackling an unconstrained

problem is convenient, penalty methods have well-known difficulties in practice. The main difficulty is that one has to choose parameter C in an *ad hoc* way. For instance, if C is too small in Equation 3.5, the example will not be adversarial; if it is too large, this term will dominate, and result in an adversarial example with more noise. This can be particularly problematic when optimizing with a low number of steps (*e.g.* to enable its use in adversarial training). Figure 3.2 plots a histogram of the values of C that were obtained by running the C&W attack on the MNIST dataset. We can see that the optimum C varies significantly among different examples, ranging from 2^{-11} to 2^5. We also see that the distribution of the best constant C changes whether we attack a model with or without adversarial training (adversarially trained models often require higher C). Furthermore, penalty methods typically result in slow convergence (Jensen & Bard (2002)).

Algorithm 3.1 Decoupled Direction and Norm Attack

Input: x: original image to be attacked
Input: y: true label (untargeted) or target label (targeted)
Input: K: number of iterations
Input: α: step size
Input: γ: factor to modify the norm in each iteration
Initialize $\boldsymbol{\delta}_0 \leftarrow \mathbf{0}$, $\tilde{\boldsymbol{x}}_0 \leftarrow \boldsymbol{x}$, $\epsilon_0 \leftarrow 1$
If targeted attack: $m \leftarrow -1$ else $m \leftarrow +1$
for $k \leftarrow 1$ *to* K **do**
  $\boldsymbol{g} \leftarrow m\nabla_{\tilde{\boldsymbol{x}}_{k-1}} J(\tilde{\boldsymbol{x}}_{k-1}, y, \theta)$
  $\boldsymbol{g} \leftarrow \alpha \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|_2}$ // Step of size α in the direction of g
  $\boldsymbol{\delta}_k \leftarrow \boldsymbol{\delta}_{k-1} + \boldsymbol{g}$
  if $\tilde{\boldsymbol{x}}_{k-1}$ *is adversarial* **then**
    $\epsilon_k \leftarrow (1-\gamma)\epsilon_{k-1}$ // Decrease norm
  else
    $\epsilon_k \leftarrow (1+\gamma)\epsilon_{k-1}$ // Increase norm
  $\tilde{\boldsymbol{x}}_k \leftarrow \boldsymbol{x} + \epsilon_k \frac{\boldsymbol{\delta}_k}{\|\boldsymbol{\delta}_k\|_2}$ // Project $\boldsymbol{\delta}_k$ onto an ϵ_k-sphere around x
  $\tilde{\boldsymbol{x}}_k \leftarrow \text{clip}(\tilde{\boldsymbol{x}}_k, 0, 1)$ // Ensure $\tilde{\boldsymbol{x}}_k \in \mathcal{X}$
end for
return $\tilde{\boldsymbol{x}}_k$ *that has lowest norm* $\|\tilde{\boldsymbol{x}}_k - \boldsymbol{x}\|_2$ *and is adversarial*

Given the difficulty of finding the appropriate constant C for this optimization, we propose an algorithm that does not impose a penalty on the ℓ_2 norm during the optimization. Instead,

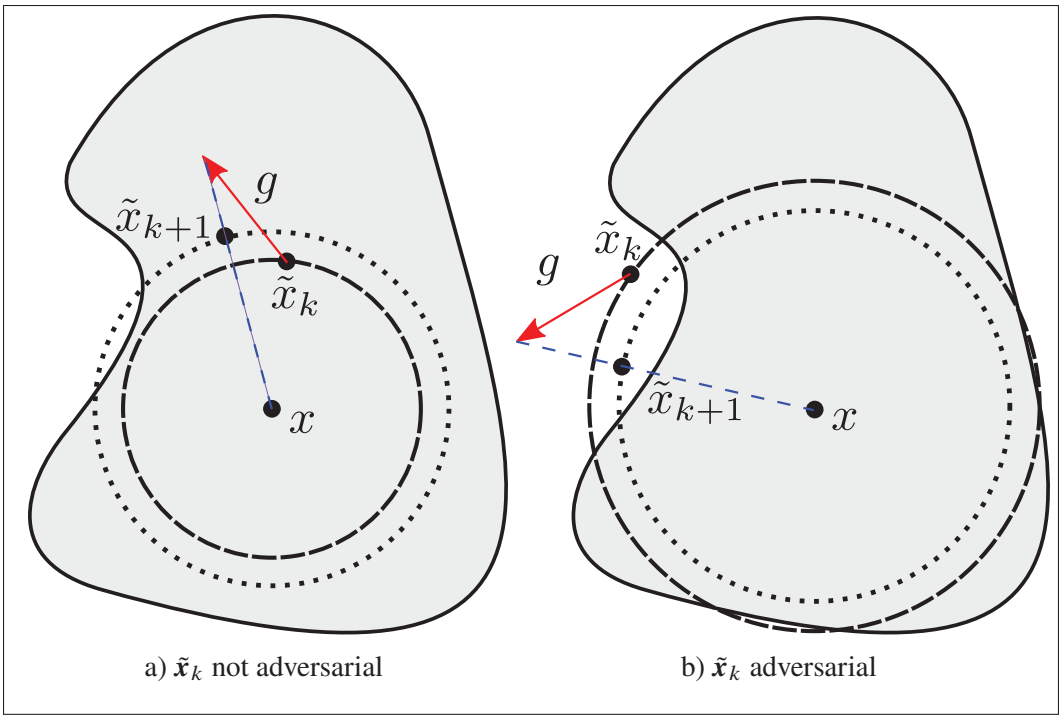


Figure 3.3 Illustration of an untargeted attack. The shaded area denotes the region of the input space classified as y_{true}. In (a), $\tilde{\boldsymbol{x}}_k$ is still not adversarial, and we increase the norm ϵ_{k+1} for the next iteration, otherwise it is reduced in (b). In both cases, we take a step g starting from the current point $\tilde{\boldsymbol{x}}$, and project back to an ϵ_{k+1}-sphere centered at x

the norm is constrained by projecting the adversarial perturbation $\boldsymbol{\delta}$ on an ϵ-sphere around the original image x. Then, the ℓ_2 norm is modified through a binary decision. If the sample x_k is not adversarial at step k, the norm is increased for step $k+1$, otherwise it is decreased.

We also note that optimizing the cross-entropy may present two other difficulties. First, the function is not bounded, which can make it dominate in the optimization of Equation 3.3. Second, when attacking trained models, often the predicted probability of the correct class for the original image is very close to 1, which causes the cross entropy to start very low and increase by several orders of magnitude during the search for an adversarial example. This affects the norm of the gradient, making it hard to find an appropriate learning rate. C&W address these issues by optimizing the difference between logits instead of the cross-entropy. In this work, the issue of it being unbounded does not affect the attack procedure, since the decision to update the

norm is done on the model's prediction (not on the cross-entropy). In order to handle the issue of large changes in gradient norm, we normalize the gradient to have unit norm before taking a step in its direction.

The full procedure is described in Algorithm 3.1 and illustrated in Figure 3.3. We start from the original image x, and iteratively refine the noise $\boldsymbol{\delta}_k$. In iteration k, if the current sample $\tilde{\boldsymbol{x}}_k = \boldsymbol{x} + \boldsymbol{\delta}_k$ is still not adversarial, we consider a larger norm $\epsilon_{k+1} = (1 + \gamma)\epsilon_k$. Otherwise, if the sample is adversarial, we consider a smaller $\epsilon_{k+1} = (1 - \gamma)\epsilon_k$. In both cases, we take a step g (step 5 of Algorithm 3.1) from the point $\tilde{\boldsymbol{x}}_k$ (red arrow in Figure 3.3), and project it back onto an ϵ_{k+1}-sphere centered at x (the direction given by the dashed blue line in Figure 3.3), obtaining $\tilde{\boldsymbol{x}}_{k+1}$. Lastly, $\tilde{\boldsymbol{x}}_{k+1}$ is projected onto the feasible region of the input space $\mathcal{X}$. In the case of images normalized to $[0, 1]$, we simply clip the value of each pixel to be inside this range (step 13 of Algorithm 3.1). Besides this step, we can also consider quantizing the image in each iteration, to ensure the attack is a valid image.

It's worth noting that, when reaching a point where the decision boundary is tangent to the ϵ_k-sphere, g will have the same direction as $\boldsymbol{\delta}_{k+1}$. This means that $\boldsymbol{\delta}_{k+1}$ will be projected on the direction of $\boldsymbol{\delta}_k$. Therefore, the norm will oscillate between the two sides of the decision boundary in this direction. Multiplying ϵ by $1 + \gamma$ and $1 - \gamma$ will result in a global decrease (on two steps) of the norm by $1 - \gamma^2$, leading to a finer search of the best norm.

3.4 Attack Evaluation

Experiments were conducted on the MNIST, CIFAR-10 and ImageNet datasets, comparing the proposed attack to the state-of-the-art ℓ_2 attacks proposed in the literature: DeepFool (Moosavi-Dezfooli *et al.* (2016)) and C&W ℓ_2 attack (Carlini & Wagner (2017)). We use the same model architectures with identical hyperparameters for training as in (Carlini & Wagner (2017)) for MNIST and CIFAR-10 (see the supplementary material for details). Our base classifiers obtain 99.44% and 85.51% accuracy on the test sets of MNIST and CIFAR-10, respectively. For the ImageNet experiments, we use a pre-trained Inception V3 (Szegedy *et al.*

Table 3.1 Performance of our DDN attack compared to C&W and DeepFool attacks on MNIST, CIFAR-10 and ImageNet in the untargeted scenario

	Attack	Budget	Success	Mean ℓ_2	Median ℓ_2	#Grads	Run-time (s)
MNIST	C&W	4×25	100.0	1.7382	1.7400	100	1.7
		1×100	99.4	1.5917	1.6405	100	1.7
		9×10 000	100.0	**1.3961**	1.4121	54 007	856.8
	DeepFool	100	75.4	1.9685	2.2909	98	-
	DDN	100	100.0	1.4563	1.4506	100	1.5
		300	100.0	1.4357	1.4386	300	4.5
		1 000	100.0	1.4240	1.4342	1 000	14.9
CIFAR-10	C&W	4×25	100.0	0.1924	0.1541	60	3.0
		1×100	99.8	0.1728	0.1620	91	4.6
		9×10 000	100.0	0.1543	0.1453	36 009	1 793.2
	DeepFool	100	99.7	0.1796	0.1497	25	-
	DDN	100	100.0	0.1503	0.1333	100	4.7
		300	100.0	0.1487	0.1322	300	14.2
		1 000	100.0	**0.1480**	0.1317	1 000	47.6
ImageNet	C&W	4×25	100.0	1.5812	1.3382	63	379.3
		1×100	100.0	0.9858	0.9587	48	287.1
		9×10 000	100.0	0.4692	0.3980	21 309	127 755.6
	DeepFool	100	98.5	0.3800	0.2655	41	-
	DDN	100	99.6	0.3831	0.3227	100	593.6
		300	100.0	0.3749	0.3210	300	1 779.4
		1 000	100.0	**0.3617**	0.3188	1 000	5 933.6

Table 3.2 Comparison of the DDN attack to the C&W ℓ_2 attack on MNIST

	Average case		Least Likely	
Attack	Success	Mean ℓ_2	Success	Mean ℓ_2
C&W 4×25	96.11	2.8254	69.9	5.0090
C&W 1×100	86.89	2.0940	31.7	2.6062
C&W 9×10 000	100.00	1.9481	100.0	2.5370
DDN 100	100.00	1.9763	100.0	2.6008
DDN 300	100.00	1.9577	100.0	2.5503
DDN 1 000	100.00	1.9511	100.0	2.5348

(2016)), that achieves 22.51% top-1 error on the validation set. Inception V3 takes images of size 299×299 as input, which are cropped from images of size 342×342.

Table 3.3 Comparison of the DDN attack to the C&W ℓ_2 attack on CIFAR-10

	Average case		Least Likely	
Attack	**Success**	**Mean ℓ_2**	**Success**	**Mean ℓ_2**
C&W 4×25	99.78	0.3247	98.7	0.5060
C&W 1×100	99.32	0.3104	95.8	0.4159
C&W 9×10 000	100.00	0.2798	100.0	0.3905
DDN 100	100.00	0.2925	100.0	0.4170
DDN 300	100.00	0.2887	100.0	0.4090
DDN 1 000	100.00	0.2867	100.0	0.4050

Table 3.4 Comparison of the DDN attack to the C&W ℓ_2 attack on ImageNet. For C&W 9×10 000, we report the results from Carlini & Wagner (2017)

	Average case		Least Likely	
Attack	**Success**	**Mean ℓ_2**	**Success**	**Mean ℓ_2**
C&W 4×25	99.13	4.2826	80.6	8.7336
C&W 1×100	96.74	1.7718	66.2	2.2997
C&W 9×10 000	100.00	0.96	100.0	2.22
DDN 100	99.98	1.0260	99.5	1.7074
DDN 300	100.00	0.9021	100.0	1.3634
DDN 1 000	100.00	0.8444	100.0	1.2240

For experiments with DeepFool (Moosavi-Dezfooli *et al.* (2016)), we used the implementation from Foolbox (Rauber *et al.* (2017)), with a budget of 100 iterations. For the experiments with C&W, we ported the attack (originally implemented on TensorFlow) on PyTorch to evaluate the models in the frameworks in which they were trained. We use the same hyperparameters from (Carlini & Wagner (2017)): 9 search steps on C with an initial constant of 0.01, with 10 000 iterations for each search step (with early stopping) - we refer to this scenario as C&W 9×10 000 in the tables. As we are interested in obtaining attacks that require few iterations, we also report experiments in a scenario where the number of iterations is limited to 100. We consider a scenario of running 100 steps with a fixed C (1×100), and a scenario of running 4 search steps on C, of 25 iterations each (4×25). Since the hyperparameters proposed in (Carlini & Wagner (2017)) were tuned for a larger number of iterations and search steps, we performed a grid search

for each dataset, using learning rates in the range [0.01, 0.05, 0.1, 0.5, 1], and C in the range [0.001, 0.01, 0.1, 1, 10, 100, 1 000]. We report the results for C&W with the hyperparameters that achieve best Median ℓ_2. Selected parameters are listed in the supplementary material.

For the experiments using DDN, we ran attacks with budgets of 100, 300 and 1 000 iterations, in all cases, using $\epsilon_0 = 1$ and $\gamma = 0.05$. The initial step size $\alpha = 1$, was reduced with cosine annealing to 0.01 in the last iteration. The choice of γ is based on the encoding of images. For any correctly classified image, the smallest possible perturbation consists in changing one pixel by $1/255$ (for images encoded in 8 bit values), corresponding to a norm of $1/255$. Since we perform quantization, the values are rounded, meaning that the algorithm must be able to achieve a norm lower than $1.5/255 = 3/510$. When using K steps, this imposes:

$$\epsilon_0(1-\gamma)^K < \frac{3}{510} \Rightarrow \gamma > 1 - \left(\frac{3}{510\,\epsilon_0}\right)^{\frac{1}{K}} \tag{3.8}$$

Using $\epsilon_0 = 1$ and $K = 100$ yields $\gamma \simeq 0.05$. Therefore, if there exists an adversarial example with smallest perturbation, the algorithm may find it in a fixed number of steps.

For the results with DDN, we consider quantized images (to 256 levels). The quantization step is included in each iteration (see step 13 of Algorithm 3.1). All results reported in the paper consider images in the $[0, 1]$ range.

Two sets of experiments were conducted: untargeted attacks and targeted attacks. As in (Carlini & Wagner (2017)), we generated attacks on the first 1 000 images of the test set for MNIST and CIFAR-10, while for ImageNet we randomly chose 1 000 images from the validation set that are correctly classified. For the untargeted attacks, we report the success rate of the attack (percentage of samples for which an attack was found), the mean ℓ_2 norm of the adversarial noise (for successful attacks), and the median ℓ_2 norm over all attacks while considering unsuccessful attacks as worst-case adversarial (distance to a uniform gray image, as introduced by (Brendel *et al.* (2020))). We also report the average number (for batch execution) of gradient computations and the total run-times (in seconds) on a NVIDIA GTX 1080 Ti with 11GB of memory. We did not report run-times for the DeepFool attack, since the implementation from foolbox generates

adversarial examples one-by-one and is executed on CPU, leading to unrepresentative run-times. Attacks on MNIST and CIFAR-10 have been executed in a single batch of 1 000 samples, whereas attacks on ImageNet have been executed in 20 batches of 50 samples.

For the targeted attacks, following the protocol from (Carlini & Wagner (2017)), we generate attacks against all possible classes on MNIST and CIFAR-10 (9 attacks per image), and against 100 randomly chosen classes for ImageNet (10% of the number of classes). Therefore, in each targeted attack experiment, we run 9 000 attacks on MNIST and CIFAR-10, and 100 000 attacks on ImageNet. Results are reported for two scenarios: 1) average over all attacks; 2) average performance when choosing the least likely class (*i.e*. choosing the worst attack performance over all target classes, for each image). The reported ℓ_2 norms are, as in the untargeted scenario, the means over successful attacks.

Table 3.1 reports the results of DDN compared to the C&W ℓ_2 and DeepFool attacks on the MNIST, CIFAR-10 and ImageNet datasets. For the MNIST and CIFAR-10 datasets, results with DDN are comparable to the state-of-the-art. DDN obtains slightly worse ℓ_2 norms on the MNIST dataset (when compared to the C&W 9×10 000), however, our attack is able to get within 5% of the norm found by C&W in only 100 iterations compared to the 54 007 iterations required for the C&W ℓ_2 attack. When the C&W attack is restricted to use a maximum of 100 iterations, it always performed worse than DDN with 100 iterations. On the ImageNet dataset, our attack obtains better Mean ℓ_2 norms than both other attacks. The DDN attack needs 300 iterations to reach 100% success rate. DeepFool obtains close results but fails to reach 100% success rate. It is also worth noting that DeepFool seems to performs worse against adversarially trained models (discussed in section 3.6). Supplementary material reports curves of the perturbation size against accuracy of the models for the three attacks.

Tables 3.2, 3.3 and 3.4 present the results on targeted attacks on the MNIST, CIFAR-10 and ImageNet datasets, respectively. For the MNIST and CIFAR-10 datasets, DDN yields similar performance compared to the C&W attack with 9×10 000 iterations, and always perform better than the C&W attack when it is restricted to 100 iterations (we re-iterate that the hyperparameters

for the C&W attack were tuned for each dataset, while the hyperparameters for DDN are fixed for all experiments). On the ImageNet dataset, DDN run with 100 iterations obtains superior performance than C&W. For all datasets, with the scenario restricted to 100 iterations, the C&W algorithm has a noticeable drop in success rate for finding adversarial examples to the least likely class.

3.5 Adversarial Training with DDN

Since the DDN attack can produce adversarial examples in relatively few iterations, it can be used for adversarial training. For this, we consider the following loss function:

$$\tilde{J}(\boldsymbol{x}, y, \theta) = J(\tilde{\boldsymbol{x}}, y, \theta) \tag{3.9}$$

where $\tilde{\boldsymbol{x}}$ is an adversarial example produced by the DDN algorithm, that is projected to an ϵ-ball around x, such that the classifier is trained with adversarial examples with a maximum norm of ϵ. It is worth making a parallel of this approach with the Madry defense (Madry *et al.* (2018)) where, in each iteration, the loss of the worst-case adversarial (see Equation 3.2) in an ϵ-ball around the original sample x is used for optimization. In our proposed adversarial training procedure, we optimize the loss of the closest adversarial example (see Equation 3.1). The intuition of this defense is to push the decision boundary away from x in each iteration. We do note that this method does not have the theoretical guarantees of the Madry defense. However, since in practice the Madry defense uses approximations (when searching for the global maximum of the loss around x), we argue that both methods deserve empirical comparison.

3.6 Defense Evaluation

We trained models using the same architectures as (Carlini & Wagner (2017)) for MNIST, and a Wide ResNet (WRN) 28-10 (Zagoruyko & Komodakis (2016)) for CIFAR-10 (similar to (Madry *et al.* (2018)) where they use a WRN 34-10). As described in section 3.5, we augment the

training images with adversarial perturbations. For each training step, we run the DDN attack with a budget of 100 iterations, and limit the norm of the perturbation to a maximum $\epsilon = 2.4$ on the MNIST experiments, and $\epsilon = 1$ for the CIFAR-10 experiments. For MNIST, we train the model for 30 epochs with a learning rate of 0.01 and then for 20 epochs with a learning rate of 0.001. To reduce the training time with CIFAR-10, we first train the model on original images for 200 epochs using the hyperparameters from (Zagoruyko & Komodakis (2016)). Then, we continue training for 30 more epochs using Equation 3.9, keeping the same final learning rate of 0.0008. Our robust MNIST model has a test accuracy of 99.01% on the clean samples, while the Madry model has an accuracy of 98.53%. On CIFAR-10, our model reaches a test accuracy of 89.0% while the model by Madry *et al*. obtains 87.3%.

Table 3.5 Evaluation of the robustness of our adversarial training on MNIST against the Madry defense

Defense	**Attack**	**Attack Success**	**Mean ℓ_2**	**Median ℓ_2**	**Model Accuracy at $\epsilon \leq 1.5$**
Baseline	C&W 9×10 000	100.0	1.3961	1.4121	42.1
	DeepFool 100	75.4	1.9685	2.2909	81.8
	DDN 1 000	100.0	1.4240	1.4342	45.2
	All	100.0	1.3778	1.3946	40.8
Madry *et al*.	C&W 9×10 000	100.0	2.0813	2.1071	73.0
	DeepFool 100	91.6	4.9585	5.2946	93.1
	DDN 1 000	99.6	1.8436	1.8994	69.9
	All	100.0	1.6917	1.8307	67.3
Ours	C&W 9×10 000	100.0	2.5181	2.6146	88.0
	DeepFool 100	94.3	3.9449	4.1754	92.7
	DDN 1 000	100.0	2.4874	2.5781	87.6
	All	100.0	**2.4497**	2.5538	**87.2**

We evaluate the adversarial robustness of the models using three untargeted attacks: Carlini 9×10 000, DeepFool 100 and DDN 1 000. For each sample, we consider the smallest adversarial perturbation produced by the three attacks and report it in the "**All**" row. Tables 3.5 and 3.6 report the results of this evaluation with a comparison to the defense of Madry *et al.* (2018)[7] and

[7] Models taken from https://github.com/MadryLab

Table 3.6 Evaluation of the robustness of our adversarial training on CIFAR-10 against the Madry defense

Defense	**Attack**	**Attack Success**	**Mean ℓ_2**	**Median ℓ_2**	**Model Accuracy at $\epsilon \leq 0.5$**
Baseline WRN 28-10	C&W 9×10 000	100.0	0.1343	0.1273	0.2
	DeepFool 100	99.3	0.5085	0.4241	38.3
	DDN 1 000	100.0	0.1430	0.1370	0.1
	All	100.0	0.1282	0.1222	0.1
Madry *et al.* WRN 34-10	C&W 9×10 000	100.0	0.6912	0.6050	57.1
	DeepFool 100	95.6	1.4856	0.9576	64.7
	DDN 1 000	100.0	0.6732	0.5876	56.9
	All	100.0	0.6601	0.5804	56.1
Ours WRN 28-10	C&W 9×10 000	100.0	0.8860	0.8254	67.9
	DeepFool 100	99.7	1.5298	1.1163	69.9
	DDN 1 000	100.0	0.8688	0.8177	68.0
	All	100.0	**0.8597**	0.8151	**67.6**

the baseline (without adversarial training) for CIFAR-10. For MNIST, the baseline corresponds to the model used in section 3.4. We observe that for attacks with unbounded norm, the attacks can successfully generate adversarial examples almost 100% of the time. However, an increased ℓ_2 norm is required to generate attacks against the model trained with DDN.

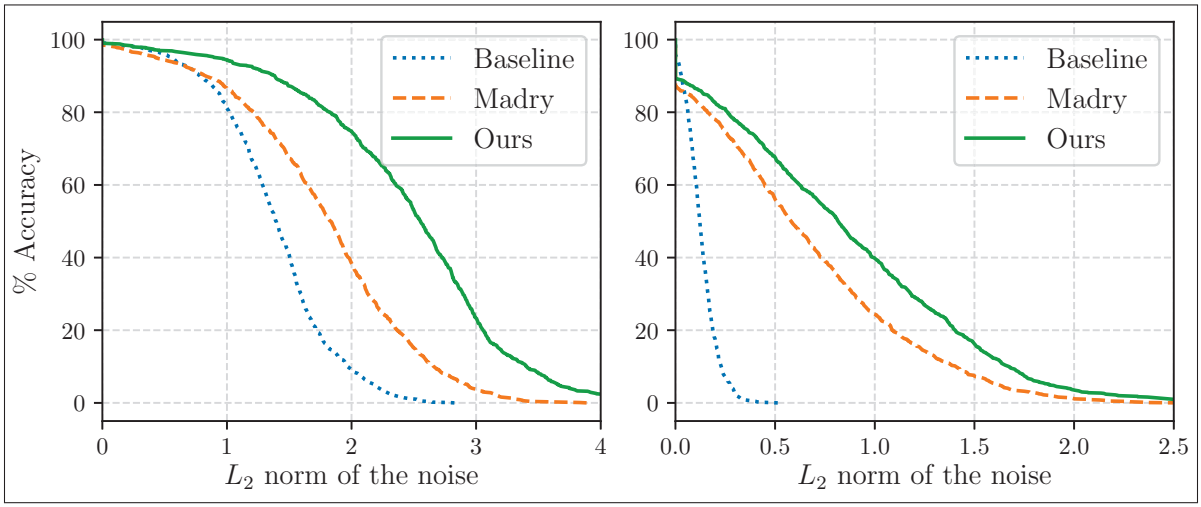


Figure 3.4 Models robustness on MNIST (left) and CIFAR-10 (right): impact on accuracy as we increase the maximum perturbation ϵ

Figure 3.4 shows the robustness of the MNIST and CIFAR-10 models respectively for different attacks with increasing maximum ℓ_2 norm. These figures can be interpreted as the expected accuracy of the systems in a scenario where the adversary is constrained to make changes with norm $\ell_2 \leq \epsilon$. For instance on MNIST, if the attacker is limited to a maximum norm of $\epsilon = 1.5$, the baseline performance decreases to 40.8%; Madry to 67.3% and our defense to 87.2%. At $\epsilon = 2.0$, baseline performance decreases to 9.2%, Madry to 38.6% and our defense to 74.8%. On CIFAR-10, if the attacker is limited to a maximum norm of $\epsilon = 0.5$, the baseline performance decreases to 0.1%; Madry to 56.1% and our defense to 67.6%. At $\epsilon = 1.0$, baseline performance decreases to 0%, Madry to 24.4% and our defense to 39.9%. For both datasets, the model trained with DDN outperforms the model trained with the Madry defense for all values of ϵ.

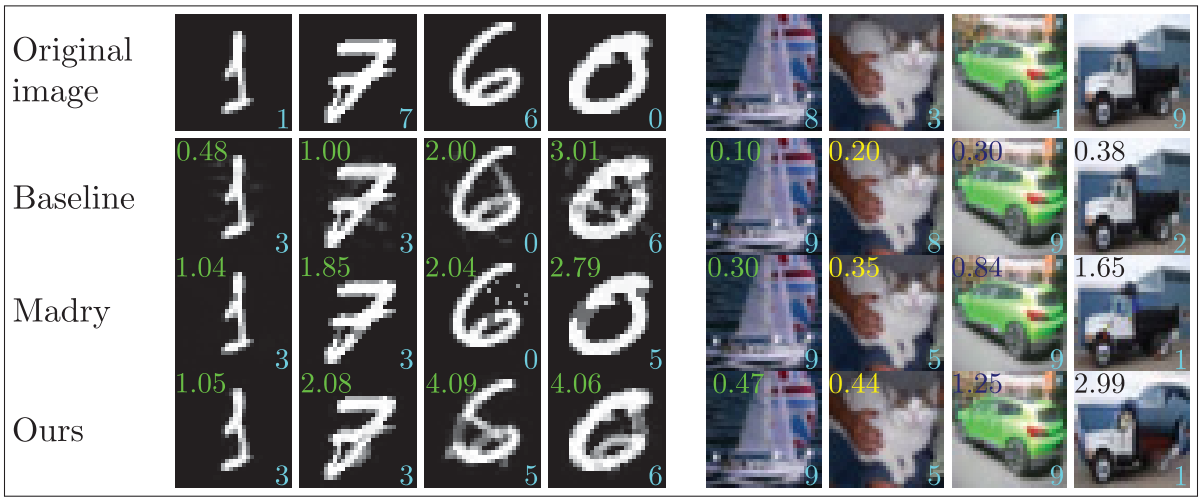


Figure 3.5 Adversarial examples with varied levels of noise $\boldsymbol{\delta}$ against three models: baseline, Madry defense and our defense. Text on top-left of each image indicate $\|\boldsymbol{\delta}\|_2$; text on bottom-right indicates the predicted class. For CIFAR-10: 1: automobile, 2: bird, 3: cat, 5: dog, 8: ship, 9: truck

Figure 3.5 shows adversarial examples produced by the DDN 1 000 attack for different models on MNIST and CIFAR-10. On MNIST, adversarial examples for the baseline are not meaningful (the still visually belong to the original class), whereas some adversarial examples obtained for the adversarially trained model (DDN) actually change classes (bottom right: 0 changes to 6). For all models, there are still some adversarial examples that are very close to the original images (first column). On CIFAR-10, while the adversarially trained models require higher

norms for the attacks, most adversarial examples still perceptually resemble the original images. In few cases (bottom-right example for CIFAR-10), it could cause a confusion: it can appear as changing to class 1 - a (cropped) automobile facing right.

3.7 Conclusion

We presented the *Decoupled Direction and Norm* attack, which obtains comparable results with the state-of-the-art for ℓ_2 norm adversarial perturbations, but in much fewer iterations. Our attack allows for faster evaluation of the robustness of differentiable models, and enables a novel adversarial training, where, at each iteration, we train with examples close to the decision boundary. Our experiments with MNIST and CIFAR-10 show state-of-the-art robustness against ℓ_2-based attacks in a white-box scenario. Future work includes the evaluation of the transferability of attacks in black-box scenarios.

The methods presented in this paper were used in NIPS 2018 Adversarial Vision Challenge Brendel *et al.* (2020), ranking first in untargeted attacks, and third in targeted attacks and robust models (both attacks and defense in a black-box scenario). These results highlight the effectiveness of the defense mechanism, and suggest that attacks using adversarially-trained surrogate models can be effective in black-box scenarios, which is a promising future direction.

CHAPTER 4

AUGMENTED LAGRANGIAN ADVERSARIAL ATTACKS

Jérôme Rony , Eric Granger , Marco Pedersoli , Ismail Ben Ayed

Systems Engineering Department, École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada

Paper published at the
IEEE/CVF International Conference on Computer Vision (ICCV), October 2021

Abstract

Adversarial attack algorithms are dominated by penalty methods, which are slow in practice, or more efficient distance-customized methods, which are heavily tailored to the properties of the distance considered. We propose a white-box attack algorithm to generate minimally perturbed adversarial examples based on Augmented Lagrangian principles. We bring several algorithmic modifications, which have a crucial effect on performance. Our attack enjoys the generality of penalty methods and the computational efficiency of distance-customized algorithms, and can be readily used for a wide set of distances. We compare our attack to state-of-the-art methods on three datasets and several models, and consistently obtain competitive performances with similar or lower computational complexity.[8]

4.1 Introduction

The last few years have seen an arms race in adversarial attacks, where several methods have been proposed to find minimally perturbed adversarial examples. In most cases, adversarial example generation is stated as a constrained optimization, which seeks the smallest additive perturbation, according to some distance, to misclassify an input. Existing methods fall within two categories: attacks using *penalty* methods for constrained optimization, and distance-customized attacks leveraging the properties of the distance considered (typically ℓ_p-norms).

[8] Code: https://github.com/jeromerony/augmented_lagrangian_adversarial_attacks

Penalties are a natural choice, as they transform a constrained-optimization problem into an unconstrained one. Within this category, the most notorious attack is the one from Carlini & Wagner (2017) known for its ℓ_2 variant. Building on the work of Carlini & Wagner, several other attacks have been proposed, *e.g.* EAD for ℓ_1 (Chen *et al.*, 2018b) and StrAttack (Xu *et al.*, 2019), which generalizes EAD and introduces sparsity in the perturbations. More recently, other works have tackled other distances than the standard ℓ_p-norms, such as SSIM (Gragnaniello, Marra, Verdoliva & Poggi, 2021), CIEDE2000 (Zhao *et al.*, 2020) or LPIPS (Laidlaw, Singla & Feizi, 2021), and followed similar penalty-based strategies. Although convenient and applicable to a wide class of distances, penalty methods are known to result in slow convergence in the general field of optimization (Jensen & Bard, 2002). Furthermore, choosing the weight of the penalty is not a trivial task (Kervadec *et al.*, 2022). In fact, in adversarial attacks, it has recently been shown that the optimal penalty weight actually varies by orders of magnitude across samples and models (Rony *et al.*, 2019). Although penalty methods can achieve competitive performance, they typically require an expensive line-search to find optimal penalty weights (Carlini & Wagner, 2017), thereby requiring large numbers of iterations. This may impede their practical deployment for training robust models and efficiently evaluating robustness.

To accelerate the generation of adversarial examples, and improve performance over penalty methods, there has been an intensive focus on developing efficient algorithms customized for specific ℓ_p-norms (Brendel, Rauber, Kümmerer, Ustyuzhaninov & Bethge, 2019; Croce & Hein, 2020a; Moosavi-Dezfooli *et al.*, 2016; Pintor *et al.*, 2021; Rony *et al.*, 2019). However, such distance-customized methods are not generally applicable because they rely heavily on the geometry/properties of the distance considered (*e.g.* using projections and dual norms) to find minimally perturbed adversarial examples. The most notable attacks within this second category are: DeepFool for the ℓ_2 and ℓ_∞ norms, which uses a linear approximation of the model at each iteration (Moosavi-Dezfooli *et al.*, 2016); DDN for the ℓ_2-norm, which uses projections on the ℓ_2-ball to decouple the direction and the norm of the perturbation (Rony *et al.*, 2019); and FAB for the ℓ_1, ℓ_2 and ℓ_∞, which combines a linear approximation of the model and projections w.r.t.

the norm considered (Croce & Hein, 2020a). The recently proposed FMN attack (Pintor *et al.*, 2021) extends the DDN attack to other norms. Beyond ℓ_p-norms, Wong *et al*. proposed an attack (Wong *et al.*, 2019) to produce adversarial perturbations with minimal Wasserstein distance using projected Sinkhorn iterations. Finally, one attack that does not strictly fall in either of the two categories was proposed by Brendel *et al*., and designed for ℓ_p-norms with $p \in \{0, 1, 2, \infty\}$ (Brendel *et al.*, 2019). In this attack, the optimization is formulated such that the perturbation follows the decision boundary of a classifier, while minimizing the considered distance. This is not limited to ℓ_p-norms but, in practice, the implementation leverages a trust-region solver designed for each ℓ_p-norm specifically, which limits applicability to other distances.

Penalty methods are generally applicable, and can be used for distances other than the standard ℓ_p-norms; for instance, CIEDE2000 (Zhao *et al.*, 2020) or LPIPS (Laidlaw *et al.*, 2021; Zhang, Isola, Efros, Shechtman & Wang, 2018). They replace constrained problems with unconstrained ones by adding a penalty, which increases when the constraint is violated. A weight of the penalty is chosen and increased heuristically, while the unconstrained optimization is repeated several times. Powerful Augmented Lagrangian principles have well-established advantages over penalties in the general context of optimization (Bertsekas, 2014, 2016; Fletcher, 2013; Nocedal & Wright, 2006), and completely avoid penalty-weight heuristics by automatically estimating the multipliers. Furthermore, the multiplier estimates tend to the Lagrange multipliers, which avoids the ill-conditioning problems often encountered in penalty methods (Bertsekas, 2014; Conn, Gould & Toint, 1997). Finally, Augmented Lagrangian methods avoid the need for explicitly solving the dual problem, unlike basic Lagrangian-dual optimization, which might be intractable/unstable for non-convex problems.

Despite their well-established advantages and popularity in the optimization community for solving non-convex problems, Augmented Lagrangian methods have not been investigated previously for adversarial attacks. While this seems rather surprising, we found in this work that the vanilla Augmented Lagrangian methods are not competitive in the context of adversarial attacks (*e.g.* in terms of computational efficiency), which might explain why they have been avoided so far. However, we introduce several algorithmic modifications to design a customized

Augmented Lagrangian algorithm for adversarial example generation. Our modifications are crucial to achieve highly competitive performances in comparisons to state-of-the-art methods. The modifications include integrating the Augmented Lagrangian inner and outer iterations with joint updates of the perturbation and the multipliers, relaxing the need for an inner-convergence criterion, introducing an exponential moving average of the multipliers, and adapting the learning rates to the distance function. All-in-all, we propose a white-box Augmented Lagrangian Method for Adversarial (ALMA) attacks, which enjoys both the general applicability of penalty approaches and computational efficiency of distance-customized methods. Our attack can be readily used to generate adversarial examples for a large set of distances, including ℓ_1-norm, ℓ_2-norm, CIEDE2000, LPIPS and SSIM, and we advocate its use for other distances that might be investigated in future research in adversarial attacks. We evaluate our attack on three datasets (MNIST, CIFAR10 and ImageNet) and several models (regularly and adversarially-trained). For each distance, we compare our method against state-of-the-art attacks proposed specifically for that distance, and consistently observe competitive performance.

4.2 Preliminaries

Let $\boldsymbol{x}$ be a sample from the input space $\mathcal{X} \subset \mathbb{R}^d$, and $y \in \mathcal{Y}$ its associated label, where $\mathcal{Y}$ is a set of discrete labels of size K. Let $f : \mathbb{R}^d \to \mathbb{R}^K$ be a model that outputs logits (*i.e.* pre-softmax scores) $\boldsymbol{z} \in \mathbb{R}^K$ given an input $\boldsymbol{x}$; $f_k(\boldsymbol{x})$ denotes the k-th component of the vector $f(\boldsymbol{x})$. In a classification scenario, the probability $p_y = P(y|x)$ is obtained using the softmax function: $p_y = \text{softmax}_y(\boldsymbol{z})$. In this work, we assume that $\mathcal{X}$ is the hypercube $\mathcal{X} = [0, 1]^d$, which is general enough for computer vision applications.

4.2.1 Problem formulation

The problem of adversarial example generation has been mainly formulated in two ways. One way is to find adversarial examples satisfying a distance constraint:

$$\begin{aligned} \text{find} \quad \boldsymbol{\delta} \quad \text{subject to} \quad & \arg\max_k f_k(\boldsymbol{x}+\boldsymbol{\delta}) \neq y \\ & D(\boldsymbol{x}+\boldsymbol{\delta}, \boldsymbol{x}) \leq \epsilon \\ & \boldsymbol{x}+\boldsymbol{\delta} \in \mathcal{X} \end{aligned} \tag{4.1}$$

Alternatively, the objective is to find adversarial examples that are minimally distorted w.r.t. a distance function D:

$$\begin{aligned} \underset{\boldsymbol{\delta}}{\text{minimize}} \quad D(\boldsymbol{x}+\boldsymbol{\delta}, \boldsymbol{x}) \quad \text{subject to} \quad & \arg\max_k f_k(\boldsymbol{x}+\boldsymbol{\delta}) \neq y \\ & \boldsymbol{x}+\boldsymbol{\delta} \in \mathcal{X} \end{aligned} \tag{4.2}$$

In this work, we are interested in solving Equation 4.2, which is equivalent to solving Equation 4.1 for every ϵ. Thus, it is a more general but more difficult problem.

4.2.2 Equivalent problem

In the general case, constraint $\boldsymbol{x}+\boldsymbol{\delta} \in \mathcal{X}$ is not necessarily trivial. However, in our context, this corresponds to a box constraint: $\mathbf{0} \leq \boldsymbol{x}+\boldsymbol{\delta} \leq \mathbf{1}$. We therefore handle it with a simple projection $\mathcal{P}_{[0,1]}$. For brevity, we will omit this constraint in the rest of the paper. In the above formulations, arg max is not differentiable, and therefore not readily amenable to gradient-based optimization. We replace the arg max constraint with an inequality constraint on the logits, as done in several works, most notably Carlini & Wagner (2017):

$$\underset{\boldsymbol{\delta}}{\text{minimize}} \quad D(\boldsymbol{x}+\boldsymbol{\delta}, \boldsymbol{x}) \quad \text{subject to} \quad f_y(\boldsymbol{x}+\boldsymbol{\delta}) - \max_{k \neq y} f_k(\boldsymbol{x}+\boldsymbol{\delta}) < 0 \tag{4.3}$$

While more suited to gradients, this constraint is not scale invariant, as noted in (Croce & Hein, 2020b): extreme scaling of the logits may result in gradient masking. We use a slightly modified Difference of Logits Ratio (DLR) for this constraint (Croce & Hein, 2020b):

$$\mathrm{DLR}^{+}(z, y) = \frac{z_y - \max_{i \neq y} z_i}{z_{\pi_1} - z_{\pi_3}} \tag{4.4}$$

where $z = f(\boldsymbol{x})$ and π is the ordering of the elements of z in decreasing order. This loss is negative if and only if $\boldsymbol{x}$ is not classified as y, and its maximum is 1. Therefore, we solve the following optimization problem:

$$\underset{\boldsymbol{\delta}}{\text{minimize}} \quad D(\boldsymbol{x} + \boldsymbol{\delta}, \boldsymbol{x}) \quad \text{subject to} \quad \mathrm{DLR}^{+}(f(\boldsymbol{x} + \boldsymbol{\delta}), y) < 0 \tag{4.5}$$

4.2.3 Distances

Most attacks in the literature measure the size of the perturbations in terms of ℓ_p-norms. In this work, we propose an attack that can find minimally perturbed adversarial examples w.r.t. several distances. We limit this work to four common measures: the ℓ_1 and ℓ_2 norms, the CIEDE2000 (Sharma, Wu & Dalal, 2005) and the LPIPS distance (Zhang *et al.*, 2018). The CIEDE2000 color difference (Sharma *et al.*, 2005) is a metric designed to assess the perceptual difference between two colors. It is widely used to evaluate color accuracy of displays and printed materials. This metric was designed to be aligned with the perception of the human eye. Generally, a value smaller than 1 means that the color difference is imperceptible, between 1 and 2 is perceptible through close examination, between 2 and 10 is perceptible at a glance, and above 10 means that the colors are different. This metric is calculated in the CIELAB color space, so a conversion is needed (see section 2). The CIEDE2000 is defined between two color pixels, so we use the image level accumulated version as in (Zhao *et al.*, 2020). The LPIPS distance (Zhang *et al.*, 2018) is a recently proposed perceptual metric based on the distance between deep features of two images for a chosen model. Zhang *et al.* showed that this distance aligns with human

perception better than other perceptual similarity metrics such as SSIM. We use the LPIPS with AlexNet as in (Laidlaw *et al.*, 2021).

4.3 Methodology

4.3.1 General Augmented Lagrangian algorithm

To describe a minimization problem with one inequality constraint, we use the following notation:

$$\text{minimize} \quad g(\boldsymbol{x}) \quad \text{subject to} \quad h(\boldsymbol{x}) < 0 \tag{4.6}$$

with $g : \mathbb{R}^d \to \mathbb{R}$ the *objective function* and $h : \mathbb{R}^d \to \mathbb{R}$ the *inequality-constraint function*.

Penalty methods trade a constrained problem (4.6) with an unconstrained one by adding a term (penalty), which increases when the constraint is violated. A weight of the penalty, λ, is chosen heuristically, and the unconstrained optimization is repeated several times with increasing values of λ, until the constraint is satisfied. Augmented Lagrangian methods have well-established advantages over penalty methods in the general context of optimization (Bertsekas, 2014, 2016; Fletcher, 2013; Nocedal & Wright, 2006), and avoid completely such heuristics. They estimate automatically the multipliers, which yields adaptive and optimal weights for the constraints. Such estimates tend to the Lagrange multiplier, which avoids the ill-conditioning problems often encountered in penalty methods (Bertsekas, 2014; Conn *et al.*, 1997). Augmented Lagrangian methods also avoid the need to explicitly solve the dual problem, unlike basic Lagrangian optimization, which may be intractable/unstable for non-convex problems. From a more practical standpoint, the efficiency of Augmented Lagrangian methods depends solely on the ability to solve the inner minimization (which we detail below), making the implementation simpler and more robust. Despite these advantages and their popularity in the optimization community, Augmented Lagrangian methods have not (to our knowledge) been investigated for adversarial attacks. In the following, we customize Augmented Lagrangian principles to solve adversarial-attack problems of the form (4.2).

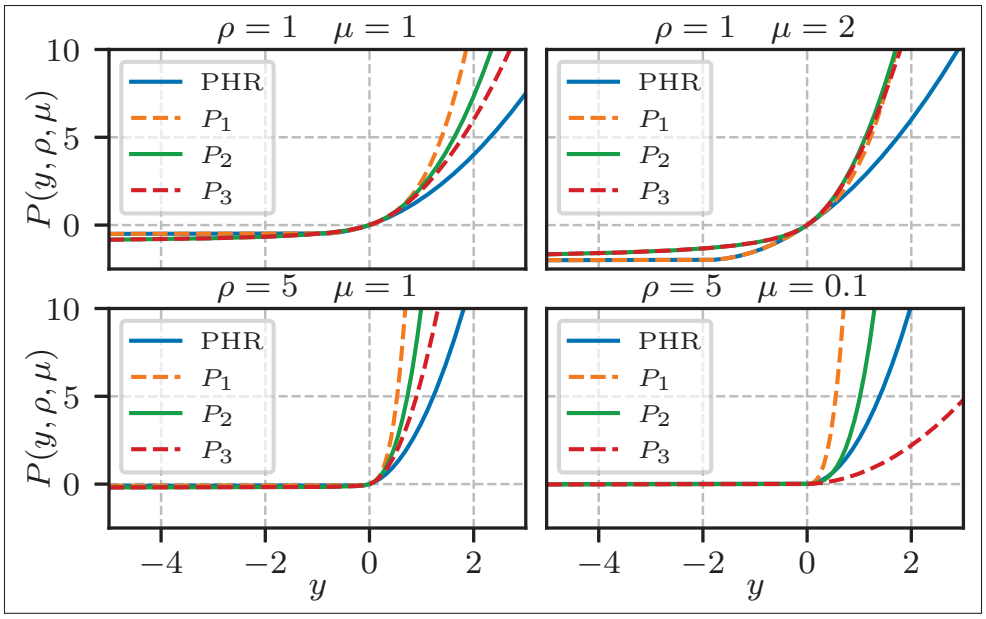


Figure 4.1 Examples of penalty-Lagrangian functions for different values of ρ and μ. The plotted functions are defined in (Birgin *et al.*, 2005) and given in section 4

In general, an Augmented Lagrangian algorithm uses a succession of unconstrained optimization problems, each solved approximately. The algorithm can be broken down in two types of iterations: *outer* iterations indexed by i and *inner* iterations. During the inner iterations, the following Augmented Lagrangian function:

$$G(\boldsymbol{x}) = g(\boldsymbol{x}) + P(h(\boldsymbol{x}), \rho^{(i)}, \mu^{(i)}) \tag{4.7}$$

is approximately minimized w.r.t. to $\boldsymbol{x}$, using the previous solution as initialization, up to an inner convergence criterion. $P : \mathbb{R} \times \mathbb{R}_+^* \times \mathbb{R}_+^* \to \mathbb{R}$ is a *penalty-Lagrangian function* such that $P'(y, \rho, \mu) = \frac{\partial}{\partial y} P(y, \rho, \mu)$ exists and is continuous for all $y \in \mathbb{R}$ and $(\rho, \mu) \in (\mathbb{R}_+^*)^2$. Any candidate function P should satisfy these four axioms Birgin *et al.* (2005):

Axiom 1. $\forall y \in \mathbb{R}, \forall (\rho, \mu) \in (\mathbb{R}_+^*)^2, \ \frac{\partial}{\partial y} P(y, \rho, \mu) \geq 0$

Axiom 2. $\forall (\rho, \mu) \in (\mathbb{R}_+^*)^2, \ \frac{\partial}{\partial y} P(0, \rho, \mu) = \mu$

Axiom 3. If, for all $j \in \mathbb{N}$, $0 < \mu_{\min} \leq \mu^{(j)} \leq \mu_{\max} < \infty$, then: $\lim_{j \to \infty} \rho^{(j)} = \infty$ and $\lim_{j \to \infty} y^{(j)} = y > 0$ imply that $\lim_{j \to \infty} \frac{\partial}{\partial y} P(y^{(j)}, \rho^{(j)}, \mu^{(j)}) = \infty$

Axiom 4. If, for all $j \in \mathbb{N}$, $0 < \mu_{\min} \leq \mu^{(j)} \leq \mu_{\max} < \infty$, then: $\lim_{j\to\infty} \rho^{(j)} = \infty$ and $\lim_{j\to\infty} y^{(j)} = y < 0$ imply that $\lim_{j\to\infty} \frac{\partial}{\partial y} P(y^{(j)}, \rho^{(j)}, \mu^{(j)}) = 0$

The first and second axioms guarantee that the derivative of the penalty-Lagrangian function P is positive and equal to μ when $y = 0$. The third and fourth axioms guarantee that the derivative of P w.r.t. y tends to infinity if the constraint is not satisfied (*i.e.* $h(\boldsymbol{x}) \geq 0$), and tends to 0 if the constraint is satisfied (*i.e.* $h(\boldsymbol{x}) < 0$). Figure 4.1 contains examples of widely used penalty-Lagrangian functions. Once the inner convergence criterion is satisfied, an outer iteration is performed, during which the *penalty multiplier* μ and the *penalty parameter* ρ are modified. The penalty multiplier μ is updated to the derivative of P at the current value w.r.t. the constraint function:

$$\mu^{(i+1)} = P'(h(\boldsymbol{x}), \rho^{(i)}, \mu^{(i)}) \tag{4.8}$$

This means that the penalty multiplier increases when the constraint is not satisfied and, otherwise, is reduced. This can be seen as an *adaptive* way of choosing the penalty weight in a penalty method. The penalty parameter ρ is increased based on the value of the constraint function at the current outer iteration, compared to the previous one. If the constraint function has not improved (*i.e.* reduced) significantly, then ρ is multiplied by a fixed factor, typically between 2 and 100 (Birgin *et al.*, 2005). With higher values of the penalty parameter ρ, the penalty-Lagrangian function tends to an ideal penalty, as can be seen in Figure 4.1. Algorithm 4.1 describes a generic Augmented Lagrangian method.

4.3.2 Augmented Lagrangian Attack

We propose to use Augmented Lagrangian methods to solve (4.5). However, we found that a vanilla Augmented Lagrangian algorithm is not well-suited for adversarial attacks. Indeed, alternating between inner and outer iterations is rather slow compared to the few hundreds iterations in typical adversarial attacks. Moreover, we wish to obtain an algorithm with a fixed number of iterations for practical purposes. This is clearly incompatible with the use of an inner convergence criterion required to stop the approximate minimization of G (step 3

Algorithm 4.1 Generic Augmented Lagrangian method

Input: Function to minimize g, constraint function h
Input: Penalty function P, initial multiplier $\mu^{(0)}$, $\rho^{(0)}$
Input: Initial value $\boldsymbol{x}^{(0)}$
for $i \leftarrow 0$ *to* $N-1$ **do**
  Using $\boldsymbol{x}^{(i)}$ as initialization, minimize (approximately):
    $G(\boldsymbol{x}) = g(\boldsymbol{x}) + P(h(\boldsymbol{x}), \rho^{(i)}, \mu^{(i)})$
  $\boldsymbol{x}^{(i+1)} \leftarrow$ approximate minimizer of G
  $\mu^{(i+1)} \leftarrow P'(h(\boldsymbol{x}^{(i+1)}), \rho^{(i)}, \mu^{(i)})$
  if *the constraint does not improve* **then**
    Set $\rho^{(i+1)} > \rho^{(i)}$
  else
    $\rho^{(i+1)} \leftarrow \rho^{(i)}$
end for

of Algorithm 4.1). Designing a good inner convergence criterion is not a trivial task either. Therefore, we propose a modification of the traditional Augmented Lagrangian algorithm. We combine the inner and outer iterations, resulting in a joint update of the perturbation $\boldsymbol{\delta} = \tilde{\boldsymbol{x}} - \boldsymbol{x}$ and the penalty multiplier μ. This means that we do not need an inner convergence criterion as we are always adapting μ. This also requires to adapt the increase of ρ, which depends on the value of the inequality-constraint function. Algorithm 4.2 presents our approach and, in the following sections, we detail several important design choices for the algorithm.

4.3.2.1 Penalty parameters adaptation

The most critical design choices for our attack are the adaptation of the penalty parameters μ and ρ.

Penalty multiplier. μ is usually modified in each outer iteration (after the inner problem is approximately solved), and taken as the derivative of the penalty function w.r.t. the constraint function (see Equation 4.8). In our algorithm, we modify μ at each iteration. This approach aims at reducing the budget needed for the optimization by combining inner and outer iterations. However, this leads to spiking values of μ, which tend to make the optimization unstable.

Algorithm 4.2 ALMA attack

Input: Classifier f, original image $\boldsymbol{x}$, true or target label y
Input: Number of iterations N, initial step size $\eta^{(0)}$, penalty parameter increase rate $\gamma > 1$, constraint improvement rate $\tau \in [0, 1]$, M number of steps between ρ increase.
Input: D distance function
Input: Penalty function P, initial multiplier $\mu^{(0)}$, initial penalty parameter $\rho^{(0)}$
1 Initialize $\tilde{\boldsymbol{x}}^{(0)} \leftarrow \boldsymbol{x}$
2 **for** $i \leftarrow 0$ *to* $N - 1$ **do**
3 $\quad z \leftarrow f(\tilde{\boldsymbol{x}}^{(i)})$
4 $\quad d^{(i)} \leftarrow \mathrm{DLR}^{+}(z, y)$; // tDLR$^+$ for targeted attack
5 $\quad \hat{\mu} \leftarrow \nabla_d P(d^{(i)}, \rho^{(i)}, \mu^{(i)})$; // New penalty multiplier
6 $\quad \mu^{(i+1)} \leftarrow \mathcal{P}_{[\mu_{\min}, \mu_{\max}]}(\alpha\mu^{(i)} + (1-\alpha)\hat{\mu})$; // EMA
7 $\quad L \leftarrow D(\tilde{\boldsymbol{x}}^{(i)}, \boldsymbol{x}) + P(d^{(i)}, \rho^{(i)}, \mu^{(i+1)})$; // Loss
8 $\quad \boldsymbol{g} \leftarrow \nabla_{\tilde{\boldsymbol{x}}} L$; // Gradient of loss w.r.t. $\tilde{\boldsymbol{x}}$
9 $\quad \tilde{\boldsymbol{x}}^{(i+1)} \leftarrow \mathcal{P}_{[0,1]}(\tilde{\boldsymbol{x}}^{(i)} - \eta^{(i)}\boldsymbol{g})$; // Step and box-constraint
10 $\quad$ **if** $(i+1) \bmod M = 0$ ***and*** $d^{(j)} > 0, \forall j \in \{0, \ldots, i\}$ ***and*** $d^{(i)} > \tau d^{(i-M)}$ **then**
11 $\quad\quad \rho^{(i+1)} \leftarrow \gamma\rho^{(i)}$ // If no adversarial has been found and d does not decrease significantly, increase ρ by a factor of γ
12 $\quad$ **else**
13 $\quad\quad \rho^{(i+1)} \leftarrow \rho^{(i)}$
14 **end for**
15 **return** $\tilde{\boldsymbol{x}}^{(i)}$ *that is adversarial and has smallest* $D(\tilde{\boldsymbol{x}}^{(i)}, \boldsymbol{x})$

Figure 4.2 shows the evolution of μ during the attack, as well as the ℓ_2-norm of the perturbation, when attacking (with ALMA ℓ_2) a single MNIST sample with the SmallCNN model. We can see that μ regularly spike to high values when not using EMA, which in turns increases the ℓ_2-norm of the perturbation because the penalty term dominates. To solve this issue, we smooth the values of μ using an Exponential Moving Average (EMA) of μ:

$$\begin{aligned} \hat{\mu} &= P'(g(\boldsymbol{x}), \rho^{(i)}, \mu^{(i)}) \\ \mu^{(i+1)} &= \alpha\mu^{(i)} + (1-\alpha)\hat{\mu} \end{aligned} \tag{4.9}$$

where $\alpha \in \mathbb{R}$ is a hyper-parameter. This corresponds to steps 5 and 6 in Algorithm 4.2. When using an EMA with $\alpha = 0.9$, the evolution of μ is smoother, with a much smaller spike at

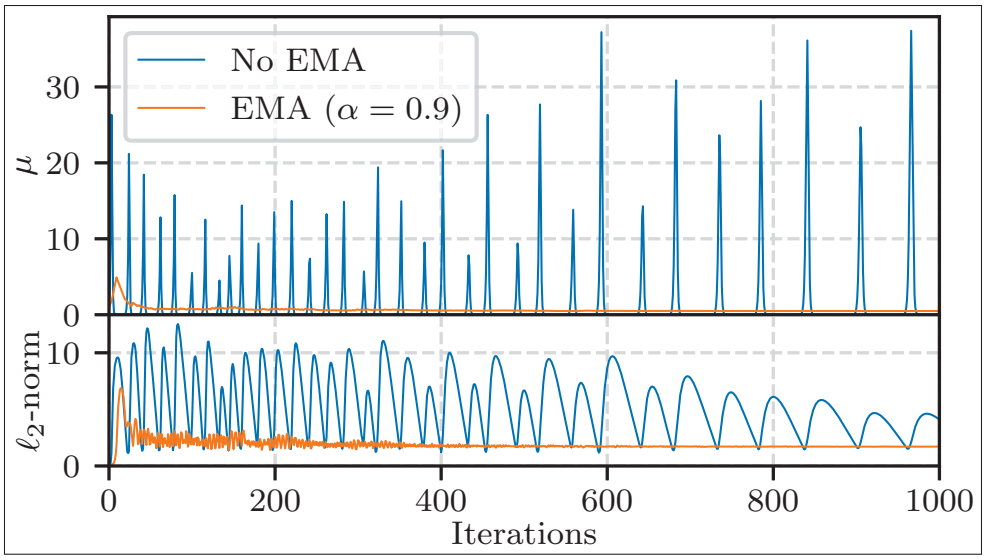


Figure 4.2 Example of the evolution of the penalty multiplier μ and the ℓ_2-norm of the perturbation during the optimization when attacking a single MNIST sample for the SmallCNN model. This leads to a ℓ_2-norm of the final adversarial perturbation of 2.13 without EMA and 1.71 with EMA ($\alpha = 0.9$). The norm of the perturbation can be lower without EMA in some iterations, but it is associated with an increase in μ, meaning that the perturbed sample $\tilde{\boldsymbol{x}}^{(i)}$ is not adversarial

the beginning and then stabilizing. As a consequence, the ℓ_2-norm of the final perturbation is smaller when using an EMA at 1.71 for $\alpha = 0.9$, compared to 2.13 when $\alpha = 0$.

For numerical stability, we also need to constrain the value of the penalty multiplier such that it never becomes too small or too large. Therefore, we also project μ on the safeguarding interval $[\mu_{\min}, \mu_{\max}] \subset \mathbb{R}_+^*$.

Penalty parameter. ρ is typically increased in each outer iteration if the constraint has not improved during the inner minimization. Since inner and outer iterations are combined in our attack, we adopt a simple strategy to adapt ρ. Every M iterations, we verify if an adversarial example was found and, if not, if the constraint has improved. If no adversarial example was found, and the constraint does not improve (*e.g.* reduced by 5% in the last M iterations), we set $\rho^{(i+1)} = \gamma\rho^{(i)}$. This corresponds to steps 11 and 13 of Algorithm 4.2.

4.3.2.2 Choice of penalty function

Experiments have shown that the choice of the penalty function is of great importance, especially when considering non-convex problems (Birgin *et al.*, 2005). Many functions have been proposed in the literature. In our experiment, we use the P_2 penalty function proposed in (Kort & Bertsekas, 1976), which is defined as followed:

$$P_2(y, \rho, \mu) = \begin{cases} \mu y + \mu \rho y^2 + \frac{1}{6}\rho^2 y^3 & \text{if} \quad y \geq 0 \\ \frac{\mu y}{1-\rho y} & \text{if} \quad y < 0 \end{cases} \tag{4.10}$$

where y represents the value of the constraint: $\text{DLR}^+(\boldsymbol{z}, y)$ in our case (or $\text{tDLR}^+(\boldsymbol{z}, y)$ in a targeted scenario). In the numerical comparison of several penalty-Lagrangian functions done by Birgin *et al.*, P_2 is second to PHRQuad in terms of robustness. However, we found that P_2 is more suited to our problem. Experimentally, PHRQuad's derivative is smaller than P_2, resulting in smaller increase of μ in each iteration. This, in turn, results in an increase of ρ because no adversarial example is found (see step 10 of Algorithm 4.2). Generally, increasing ρ helps in finding a feasible point (*i.e.* adversarial examples), at the cost of a larger final distance for the perturbation. The choice of P_2 also depends on the algorithm design. In our attack, we chose to increase ρ every M steps if no adversarial example has been found, regardless of the usual convergence criterion used in more traditional Augmented Lagrangian methods. Therefore, we need a penalty function with an "aggressive" derivative such that μ is modified quickly.

4.3.2.3 Learning rate scheduling

Initial learning rate. Different distance functions can have widely different scales (see Tables 4.1 and 4.2, median results column). Therefore, the initial learning rate is chosen adaptively for each sample $\boldsymbol{x}$, such that the first gradient step (step 9 in Algorithm 4.2) increases D by ϵ, or formally, such that: $D(\tilde{\boldsymbol{x}}^{(1)}, \boldsymbol{x}) = \epsilon$. To obtain that value, we compute the gradient $\boldsymbol{g}$ of DLR^+ w.r.t. $\boldsymbol{x}$, and find the scalar $\eta^{(0)}$ such that $D(\mathcal{P}_{[0,1]}[\boldsymbol{x} - \eta^{(0)}\boldsymbol{g}], \boldsymbol{x}) = \epsilon$, using a line search followed by a binary search.

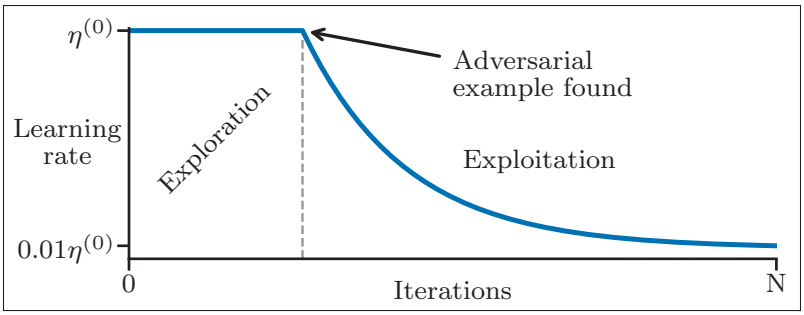


Figure 4.3 Exponential learning rate decay for the attack

Learning rate decay. The optimization process can be partitioned in two phases: exploration and exploitation. The exploration phase corresponds to finding a feasible point (*i.e.* an adversarial example) for Equation 4.2. Then, the exploitation phase consists in refining the feasible point, *i.e.* finding a minimally distorted adversarial example. In our algorithm, the learning rate remains constant during the exploration phase, and decays exponentially during the exploitation phase to reach a final learning rate of $\eta^{(0)}/100$. Figure 4.3 illustrates the scheduling of the learning rate decay.

4.3.2.4 Optimization algorithm

It is well known that gradient descent has a slow convergence rate, especially for ill-conditioned problem. Several optimization algorithms have been proposed to accelerate first-order methods, such as the momentum method or Nesterov's acceleration (Nesterov, 1983). One such algorithm is RMSProp (Tieleman & Hinton, 2012), in which the learning rate is divided by the square root of the EMA of the squared gradient. We use RMSProp combined with the momentum method in our attack, with a slight modification. Originally, the EMA of the squared gradient is initialized at 0. To avoid dividing by a small value in the first iteration (which would result in a large increase of the distance during the first iteration), we initialize the EMA of the squared gradient at 1. For simplicity, Algorithm 4.2 only describes a regular gradient descent update at step 9.

4.4 Experiments

Datasets. To evaluate our attack, we conduct experiments on three datasets: MNIST, CIFAR10 and ImageNet with two different budgets: 100 and 1 000 iterations. On MNIST, we test the attacks on the whole test set. On CIFAR10 and ImageNet, we test the attacks on 1 000 samples. These 1 000 samples correspond to the first 1 000 samples of the test set on CIFAR10 and 1 000 randomly chosen samples from the validation set for ImageNet.

Models. It has been observed that some attacks can work well for regularly trained models, and fail against defended (*i.e*. adversarially trained) models (Carlini *et al.*, 2019; Croce & Hein, 2020b; Rony *et al.*, 2019). Therefore, for each dataset, we evaluate the attack against a collection of models. Specifically, for MNIST, we use four models. The first three are the same model as in (Carlini & Wagner, 2017; Pintor *et al.*, 2021; Rony *et al.*, 2019) with three training schemes: a regularly trained model we call SmallCNN, a ℓ_2-adversarially trained model from (Rony *et al.*, 2019) we call SmallCNN-DDN and a ℓ_∞-adversarially trained model from (Zhang *et al.*, 2019a) we call SmallCNN-TRADES. To vary architectures, we also test on the ℓ_∞-adversarially trained model from (Zhang *et al.*, 2019b) we call CROWN-IBP. This is the large variant trained with $\epsilon = 0.4$. For CIFAR10, we use three models: a regularly trained Wide ResNet 28-10 (Zagoruyko & Komodakis, 2016), a ℓ_∞-adversarially trained Wide ResNet 28-10 from (Carmon, Raghunathan, Schmidt, Duchi & Liang, 2019) and a ℓ_2-adversarially trained ResNet-50 (He, Zhang, Ren & Sun, 2016a) from (Augustin, Meinke & Hein, 2020). These models are fetched from the RobustBench library (Croce *et al.*, 2021). On ImageNet, we test the attacks on three ResNet-50 models: one regularly trained, and two adversarially trained (ℓ_2 and ℓ_∞), available through the robustness library (Engstrom, Ilyas, Salman, Santurkar & Tsipras, 2019). All the models have been selected because they have obtained high robust accuracies in third-party evaluations (Croce & Hein, 2020b).

Metrics. For each attack, we report the Attack Success Rate (ASR) which is the fraction of examples for which an adversarial perturbation was found, and the median perturbation size. To compare the complexities of the attacks, we use the average number of forward and backward

model propagations per sample needed to perform each attack. In the deep learning context, the number of model propagations (forward and backward) is a good proxy for the complexity of an attack, as these operations generally require orders of magnitude more computations than the other operations of an attack. We also prefer this measure to the attack run-time since this makes for comparisons that are independent of both software optimizations and hardware differences.

Hyperparameters. For our attack, we consider two budgets: 100 and 1 000 iterations. The goal of this work is to propose a framework to generate minimally perturbed adversarial example w.r.t. any distance. Therefore, we chose one α (*i.e.* the coefficient of the EMA) per budget and one ϵ (*i.e.* the increase of D in the first iteration, see subsubsection 4.3.2.3) per distance. α and ϵ are shared across all datasets and models. For the 100 iterations budget, we set $\alpha = 0.5$ and for the 1 000 iterations budget, we set $\alpha = 0.9$. For ϵ, we chose an initial value in relation to the usual scale of each distance: typically a tenth of the expected distances between the adversarial examples and the original inputs works well. Although it requires prior knowledge on the expected distances, a good ϵ can quickly be found when experimenting with a new distance function. section 6 gives the initial values of ϵ used in our experiments for each distance. The other hyperparameters are fixed. We use $\mu^{(0)} = \rho^{(0)} = 1$, $\mu_{\min} = 10^{-6}$ and $\mu_{\max} = 10^{12}$, $\gamma = 1.2$, $\tau = 0.95$ and $M = 10$ in all our experiments. These values are not usual for Augmented Lagrangian algorithms; they reflect our design choice of combining inner and outer iterations. Since we update our penalty parameters (*i.e.* μ and ρ) more frequently, γ can have a smaller value.

Attacks. We compare our attack with various state-of-the-art attacks from the literature. Specifically, we evaluate the ℓ_1 variant of our attack against the EAD attack (Chen *et al.*, 2018b) with budgets of 9×100 and 9×1 000, and FAB ℓ_1 (Croce & Hein, 2020a) and FMN ℓ_1 (Pintor *et al.*, 2021) attacks with budgets of 100 and 1 000 iterations. For the ℓ_2 variant of our attack, we compare it to the C&W ℓ_2 (Carlini & Wagner, 2017) attack with budgets of 9×1 000 and 9×10 000, DDN (Rony *et al.*, 2019), FAB ℓ_2 (Croce & Hein, 2020a) and FMN ℓ_2 (Pintor *et al.*,

2021) all with budgets of 100 and 1 000 iterations.[9] For FAB ℓ_1 and ℓ_2 on ImageNet, we use the targeted variant of the attack, as was done in the original work.[10] We compare the CIEDE2000 variant of our attack with, to the best of our knowledge, the only attack using this metric: the PerC-AL attack (Zhao *et al.*, 2020) with budgets of 100 and 1 000 iterations. Lastly, for the LPIPS variant, we compare our attack to the LPA attack (Laidlaw *et al.*, 2021), which is designed to solve Equation 4.1 (*i.e.* find a perturbation within a distance budget). We perform a binary search on the distance budget to find the minimum budget for which an adversarial example can be found. We also add the $\text{APGD}^{\text{T}}_{\text{DLR}}$ ℓ_2 (Croce & Hein, 2020b) to the ℓ_2 comparison with a binary search. For the binary search, we start with a large enough budget (depending on the dataset) such that the attacks can reach 100% ASR and perform enough binary search steps to reach a precision of 0.01 for the ℓ_2-norm and 0.001 for the LPIPS.

Finally, we perform a C&W type attack as a baseline for the CIEDE2000 and LPIPS distances. We replace the ℓ_2-norm with the corresponding target distance and use a budget of 9×1 000 which is enough to get a good performance while keeping an acceptable budget (compared to 9×10 000).

4.5 Results

To summarize the results, we report the geometric mean of each metric over the models considered for each dataset, in Table 4.1 for MNIST and Table 4.2 for CIFAR10 and ImageNet. Tables containing the detailed results for each model can be found in section 8. We also present robust accuracy curves, which reflect directly the performance in terms of minimum distance of the adversarial examples, by showing an expected robust accuracy as a function of a perturbation budget. For all curves, the dotted lines denote the reduced budget attacks of the corresponding colors. With these curves, we can also find the median distance (see section 8) for each attack by

[9] We tried to include the B&B ℓ_1 and ℓ_2 attacks (Brendel *et al.*, 2019) in our comparison, but their official implementations kept crashing. In the cases where it did not, we observed median distances worse than FAB. As such, we omitted the method completely from the experiments.

[10] The FAB attack needs to compute the Jacobian of the output of the model w.r.t. to the input. This increases the complexity by a factor K (*i.e.* the number of classes), making it impractical for datasets with large number of classes, such as ImageNet. See Section 4 of (Croce & Hein, 2020a).

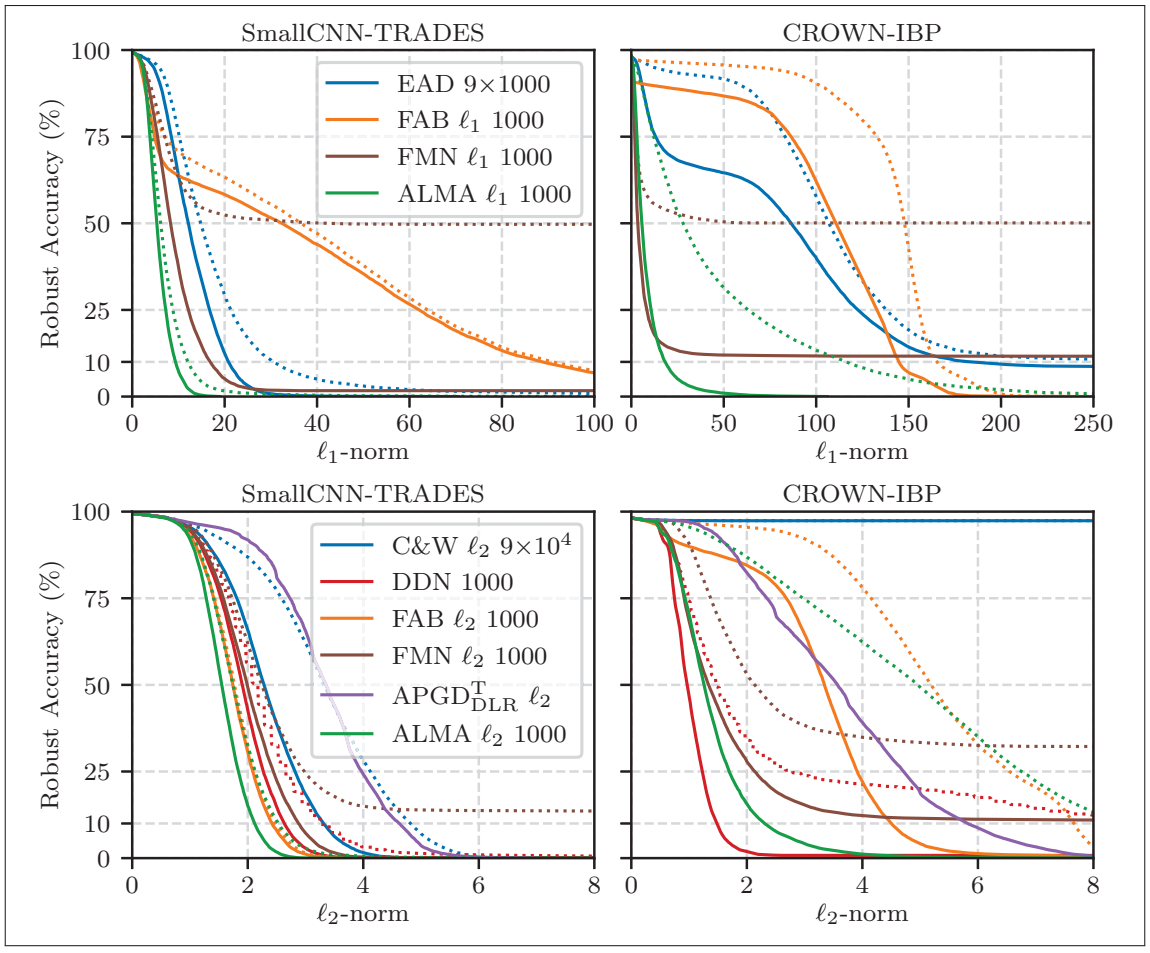


Figure 4.4 Robust accuracy curves for the SmallCNN-TRADES and CROWN-IBP adversarially trained models on MNIST against ℓ_1 (top row) and ℓ_2 (bottom row) attacks

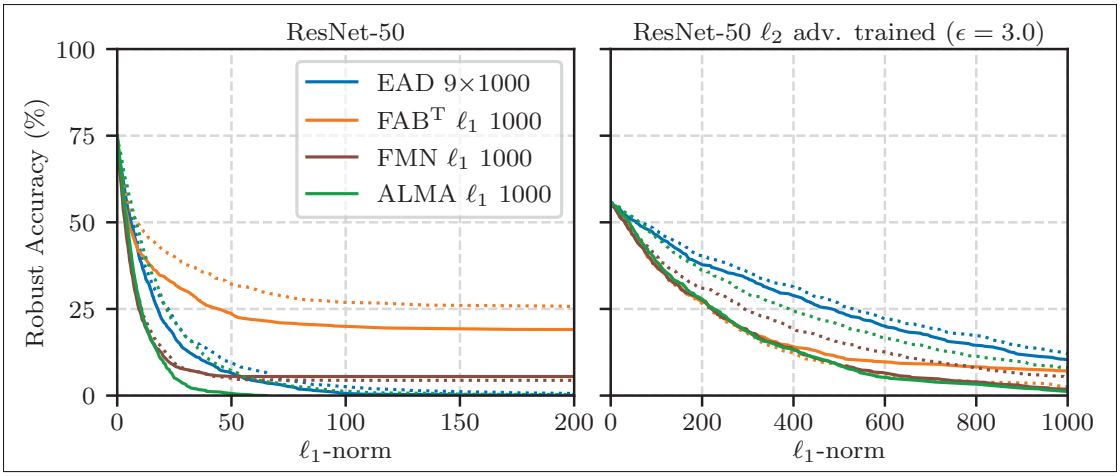


Figure 4.5 Robust accuracy curves for regular and ℓ_2-adversarial ResNet-50 on ImageNet against ℓ_1 attacks

Table 4.1 Performance for attacks on the MNIST dataset. Geometric mean over the four models. FMN ℓ_1 100 and C&W ℓ_2 do not reach 50% ASR on the IBP model, so the median distance is undefined. ‡A binary search is performed on each sample to get a minimal perturbation attack (Equation 4.2)

Distance	Attack	ASR (%)	Median Distance	Forwards / Backwards
ℓ_1-norm	EAD 9×100 (Chen *et al.*, 2018b)	97.18	21.94	807 / 407
	EAD 9×1 000 (Chen *et al.*, 2018b)	97.76	19.19	5 025 / 2 516
	FAB ℓ_1 100 (Croce & Hein, 2020a)	99.80	27.41	201 / 1 000
	FAB ℓ_1 1 000 (Croce & Hein, 2020a)	99.83	24.21	2 001 / 10 000
	FMN ℓ_1 100 (Pintor *et al.*, 2021)	69.87	–	100 / 100
	FMN ℓ_1 1 000 (Pintor *et al.*, 2021)	95.35	7.34	1 000 / 1 000
	ALMA ℓ_1 100 (Ours)	99.90	11.45	100 / 100
	ALMA ℓ_1 1 000 (Ours)	100	7.16	1 000 / 1 000
ℓ_2-norm	C&W ℓ_2 9×1 000 (Carlini & Wagner, 2017)	40.19	–	8 643 / 8 643
	C&W ℓ_2 9×10 000 (Carlini & Wagner, 2017)	40.20	–	85 907 / 85 907
	DDN 100 (Rony *et al.*, 2019)	98.48	1.86	100 / 100
	DDN 1 000 (Rony *et al.*, 2019)	99.83	1.61	1 000 / 1 000
	FAB ℓ_2 100 (Croce & Hein, 2020a)	83.25	2.10	201 / 1 000
	FAB ℓ_2 1 000 (Croce & Hein, 2020a)	96.28	1.77	2 001 / 10 000
	FMN ℓ_2 100 (Pintor *et al.*, 2021)	70.99	3.18	100 / 100
	FMN ℓ_2 1 000 (Pintor *et al.*, 2021)	96.02	1.77	1 000 / 1 000
	$\text{APGD}^{\text{T}}_{\text{DLR}}$ ℓ_2‡ (Croce & Hein, 2020b)	99.98	2.52	12 271 / 12 253
	ALMA ℓ_2 100 (Ours)	99.72	2.38	100 / 100
	ALMA ℓ_2 1 000 (Ours)	100	1.61	1 000 / 1 000

looking at the distance for which the accuracy is 50%. All the robust accuracy curves can be found in section 9. Here we display a few for which there are significant differences between the attacks. For MNIST, we show the curves in Figure 4.4 for the SmallCNN-TRADES and CROWN-IBP models against ℓ_1 and ℓ_2 attacks. For ImageNet, we show the curves for ResNet-50 regularly and ℓ_2-adversarially trained from (Engstrom *et al.*, 2019) against ℓ_1 attacks.

Across all datasets and models, our attack consistently obtains results competitive with attacks tailored to each distance. On MNIST, ALMA ℓ_1 is the only attack to reach an ASR of 100% with only FAB ℓ_1 outperforming it on the SmallCNN model. FMN ℓ_1 reaches a lower median ℓ_1 at the cost of a reduced ASR on the CROWN IBP model. ALMA ℓ_2 is only marginally outperformed by $\text{APGD}^{\text{T}}_{\text{DLR}}$ ℓ_2 on the SmallCNN and SmallCNN-DDN models, but again, ALMA is the only attack to consistently reach 100% ASR. The results on MNIST get confirmed by the experiments

Table 4.2 Performance for attacks on the CIFAR10 and ImageNet datasets. Geometric mean over the three models for each dataset. †For ImageNet, we use the targeted variant of the attack as in (Croce & Hein, 2020a) (see section 4.4). ‡A binary search is performed on each sample to get a minimal perturbation attack (Equation 4.2)

Distance	Attack	CIFAR10			ImageNet		
		ASR (%)	Median Distance	Forwards / Backwards	ASR (%)	Median Distance	Forwards / Backwards
ℓ_1-norm	EAD 9×100	100	6.11	572 / 290	100	13.87	488 / 248
	EAD 9×1 000	100	5.44	4 284 / 2 146	100	12.83	3 758 / 1 883
	FAB† ℓ_1 100	96.58	4.26	201 / 1 000	88.82	10.72	1 810 / 900
	FAB† ℓ_1 1 000	99.00	3.78	2 001 / 10 000	89.07	8.88	18 010 / 9 000
	FMN ℓ_1 100	99.90	3.64	100 / 100	94.33	8.43	100 / 100
	FMN ℓ_1 1 000	99.83	3.54	1 000 / 1 000	93.93	7.58	1 000 / 1 000
	ALMA ℓ_1 100 (Ours)	100	4.31	100 / 100	100	19.79	100 / 100
	ALMA ℓ_1 1 000 (Ours)	100	3.69	1 000 / 1 000	100	12.10	1 000 / 1 000
ℓ_2-norm	C&W ℓ_2 9×1 000	100	0.40	7 976 / 7 974	99.83	0.57	7 248 / 7 246
	C&W ℓ_2 9×10 000	100	0.40	78 081 / 78 079	99.83	0.57	67 479 / 67 476
	DDN 100	100	0.43	100 / 100	99.70	0.51	100 / 100
	DDN 1 000	100	0.42	1 000 / 1 000	99.87	0.50	1 000 / 1 000
	FAB† ℓ_2 100	100	0.41	201 / 1 000	99.70	0.35	1 810 / 900
	FAB† ℓ_2 1 000	100	0.41	2 001 / 10 000	98.90	0.35	18 010 / 9 000
	FMN ℓ_2 100	99.90	0.43	100 / 100	99.43	0.38	100 / 100
	FMN ℓ_2 1 000	99.83	0.40	1 000 / 1 000	99.63	0.36	1 000 / 1 000
	$\text{APGD}^{\text{T}}_{\text{DLR}}$ ℓ_2‡	100	0.38	5 345 / 5 321	100	0.34	6 096 / 6 068
	ALMA ℓ_2 100 (Ours)	100	0.40	100 / 100	100	0.38	100 / 100
	ALMA ℓ_2 1 000 (Ours)	100	0.38	1 000 / 1 000	100	0.35	1 000 / 1 000
CIEDE 2000	C&W CIEDE2000 9×1 000	100	0.93	6 729 / 6 726	100	1.39	5 635 / 5 632
	PerC-AL 100	100	2.87	201 / 100	99.90	3.55	201 / 100
	PerC-AL 1 000	100	2.72	2 001 / 1 000	99.93	3.42	2 001 / 1 000
	ALMA CIEDE2000 100 (Ours)	100	1.09	100 / 100	100	0.75	100 / 100
	ALMA CIEDE2000 1 000 (Ours)	100	0.78	1 000 / 1 000	100	0.63	1 000 / 1 000
LPIPS $\times 10^{-2}$	C&W LPIPS 9×1 000	100	0.47	6 658 / 6 655	100	2.07	4 950 / 4 944
	LPA‡	100	5.39	1 118 / 1 108	100	5.79	1 211 / 1 201
	ALMA LPIPS 100 (Ours)	99.97	2.47	100 / 100	100	1.59	100 / 100
	ALMA LPIPS 1 000 (Ours)	100	0.60	1 000 / 1 000	100	1.13	1 000 / 1 000

on CIFAR10 and ImageNet. All the variants of ALMA consistently obtain 100% success rate (except for ALMA LPIPS 100). ALMA ℓ_1 obtains worse median norms than FAB ℓ_1 and FMN ℓ_1, but again, at the cost of a reduced ASR for both attacks. Surprisingly, FMN ℓ_1 obtains lower ASR for the higher budget variant, but with a lower median. For the ℓ_2 attacks, several perform similarly, with only $\text{APGD}^{\text{T}}_{\text{DLR}}$ ℓ_2 reaching a lower ℓ_2 median with a 100% ASR. This is expected given that this is a distance budget attack, with a much higher overall complexity when combining it with a binary search. For the CIEDE2000 distance, the PerC-AL attack (Zhao *et al.*, 2020) obtains significantly larger median distance on both CIFAR10 and ImageNet.

Investigating the original code, we found errors in the implementation of the CIEDE2000, resulting in both wrong values and wrong gradients.[11]. The LPIPS variant of ALMA performs much better than LPA combined with a binary search for both datasets. Being a penalty base approach, our C&W baseline gets performance that is on par or better than ALMA LPIPS on both datasets, but at a much higher computational cost. However, it does not perform as well for the CIEDE2000 distance.

Overall, our attack offers a reliable trade-off between speed and performance in terms of ASR and perturbation size, given that the hyper-parameters are not tuned beyond ϵ, which is distance specific, and α which is directly related to the number of steps.

4.6 Conclusion

In this paper, an adversarial attack based on Augmented Lagrangian methods is proposed, which acts as a general framework for generating minimally perturbed adversarial examples w.r.t. several distances. In most of our experiments, it offers a good trade-off between performance and computational complexity in comparison to state-of-the-art methods. We believe that our general method could serve as a starting point for designing efficient attacks minimizing new distances.

[11] To verify this, we tested both implementations of the CIEDE2000 against the test values provided for this purpose in (Sharma *et al.*, 2005).

CHAPTER 5

PROXIMAL SPLITTING ADVERSARIAL ATTACK FOR SEMANTIC SEGMENTATION

Jérôme Rony[1] , Jean-Christophe Pesquet[2] , Ismail Ben Ayed[1]

[1] Systems Engineering Department, École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada

[2] Centre de Vision Numérique
Université Paris-Saclay, CentraleSupélec, Inria

Paper accepted for publication at the
IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2023

Abstract

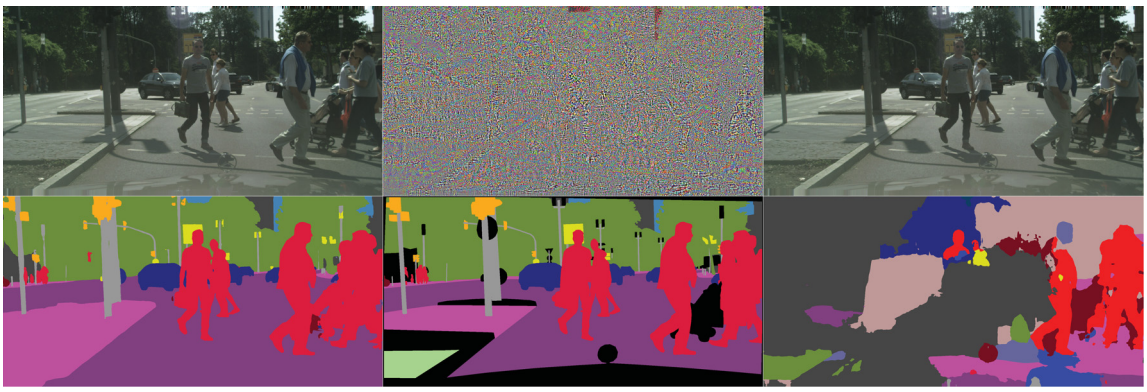
Classification has been the focal point of research on adversarial attacks, but only a few works investigate methods suited to denser prediction tasks, such as semantic segmentation. The methods proposed in these works do not accurately solve the adversarial segmentation problem and, therefore, overestimate the size of the perturbations required to fool models. Here, we propose a white-box attack for these models based on a proximal splitting to produce adversarial perturbations with much smaller ℓ_∞ norms. Our attack can handle large numbers of constraints within a nonconvex minimization framework via an Augmented Lagrangian approach, coupled with adaptive constraint scaling and masking strategies. We demonstrate that our attack significantly outperforms previously proposed ones, as well as classification attacks that we adapted for segmentation, providing a first comprehensive benchmark for this dense task.[12]

5.1 Introduction

Research on white-box adversarial attacks has mostly focused on classification tasks, with several methods proposed over the years for ℓ_p-norms (Kurakin *et al.*, 2017b; Moosavi-Dezfooli *et al.*, 2016; Dong *et al.*, 2018; Carlini & Wagner, 2017; Rony *et al.*, 2019; Yao *et al.*, 2019;

[12] Code: https://github.com/jeromerony/alma_prox_segmentation

a) Pascal VOC 2012: $\|\boldsymbol{\delta}\|_\infty = {}^{0.67}/_{255}$



b) Cityscapes: $\|\boldsymbol{\delta}\|_\infty = {}^{0.53}/_{255}$

Figure 5.1 Untargeted adversarial examples for FCN HRNetV2 W48 on Pascal VOC 2012 and Cityscapes. In both cases, more than 99% of pixels are incorrectly classified. For each dataset, left is the original image and its predicted segmentation, middle is the amplified perturbation and the ground truth segmentation and right is the adversarial image with its predicted segmentation. For Pascal VOC 2012, the predicted classes are *TV monitor* (blue), *person* (beige) and *chair* (bright red).

Brendel *et al.*, 2019; Croce & Hein, 2020a,b; Rony *et al.*, 2021; Matyasko & Chau, 2021). In contrast, the literature on adversarial attacks for segmentation tasks has been much scarcer, with few works proposing attacks (Xie *et al.*, 2017; Cisse, Adi, Neverova & Keshet, 2017; Ozbulak, Van Messem & De Neve, 2019). The lack of studies on adversarial attacks in segmentation may

appear surprising because of the prominence of this computer vision task in many applications where a semantic understanding of image contents is needed. In many safety-critical application areas, it is thus crucial to assess the robustness of the employed segmentation models.

Although segmentation is treated as a per-pixel classification problem, designing adversarial attacks for semantic segmentation is much more challenging for several reasons. First, from an optimization perspective, the problem of adversarial example generation is more difficult. In a classification task, producing minimal adversarial examples is a nonconvex constrained problem with a single constraint. In a segmentation task, this optimization problem now has multiple constraints, since at least one constraint must be addressed for each pixel in the image. For a dataset such as Cityscapes (Cordts *et al.*, 2016), the images have a size of 2 048×1 024, resulting in more than 2 million constraints. Consequently, most attacks originally designed for classification cannot be directly extended to segmentation. For instance, penalty methods such as C&W (Carlini & Wagner, 2017) cannot tackle multiple constraints since they rely on a binary search of the penalty weight.

Second, the computational and memory cost of generating adversarial examples can be prohibitive. White-box adversarial attacks usually rely on computing the gradient of a loss w.r.t. the input. In segmentation tasks, the dense outputs result in high memory usage to perform a backward propagation of the gradient. For reference, computing the gradients of the logits w.r.t. the input requires ~22 GiB of memory for FCN HRNetV2 W48 (Wang *et al.*, 2020) on Cityscapes with a 2 048×1 024 image. Additionally, most recent classification adversarial attacks require between 100 and 1 000 iterations, resulting in a run-time of up to a few seconds per image (Rony *et al.*, 2019; Pintor *et al.*, 2021) (on GPU) depending on the dataset and model. For segmentation, this increases to tens or even hundreds of seconds per image with larger models.

In this article, we propose an adversarial attack to produce minimal adversarial perturbations w.r.t. the ℓ_∞-norm, for deep semantic segmentation models. Building on Augmented Lagrangian principles, we introduce adaptive strategies to handle a large number of constraints (*i.e.* $>10^6$). Furthermore, we tackle the nonsmooth ℓ_∞-norm minimization with a proximal splitting instead

of gradient descent. In particular, we show that we can efficiently compute the proximity operator of the sum of the ℓ_∞-norm and the indicator function of the space of possible perturbations. This results in an adversarial attack that significantly outperforms the DAG attacks (Xie *et al.*, 2017), in addition to several classification attacks that we were able to adapt for segmentation. We propose a methodology to evaluate adversarial attacks in segmentation, and compare the different approaches on the Cityscapes (Cordts *et al.*, 2016) and Pascal VOC 2012 (Everingham, Van Gool, Williams, Winn & Zisserman, 2012) datasets with a diverse set of architectures, including well-known DeepLabV3+ models Chen, Zhu, Papandreou, Schroff & Adam (2018a), the recently proposed transformer-based SegFormer (Xie *et al.*, 2021), and the robust DeepLabV3 DDC-AT model from (Xu *et al.*, 2021). The proposed approach yields outstanding performances for all models and datasets considered in our experiments. For instance, our attack finds untargeted adversarial perturbations with ℓ_∞-norms lower than $1/255$ on average for all models on Cityscapes. With this attacker, we provide better means to assess the robustness of deep segmentation models, which has often been overestimated until now, as the existing attacks could only find adversarial examples with larger norms.

5.2 Related Works

While the literature on minimal adversarial attacks for classification is vast, the research on attacks for segmentation is much less developed. The main work on adversarial attacks for segmentation is done by Xie *et al.*. It proposes a simple algorithm to generate adversarial perturbations for dense prediction tasks, including object detection and segmentation, called the Dense Adversary Generation (DAG) attack. In this attack, the rescaled gradient of the loss w.r.t. the input is added to the current perturbation, until the stopping criterion is reached, *i.e*. a given percentage of pixels is adversarial. In each iteration, the total loss is the sum of the losses over pixels that are not adversarial. This can be seen as a form of greedy algorithm. See section 1 for the complete algorithm of the DAG attack and a discussion on the stopping criterion used. In practice, this attack is quite efficient, however, it simply accumulates gradients until the stopping criterion is reached. Therefore, it does not minimize the norm considered. Cisse *et al*. propose

the Houdini attack for several tasks (Cisse *et al.*, 2017), including segmentation. The goal of this approach is to maximize a surrogate loss for a given perturbation budget (*i.e.* constraint on the ℓ_∞-norm), hence not producing minimal perturbations. More recently, Ozbulak *et al.* studied adversarial examples on a medical image segmentation task (Ozbulak *et al.*, 2019). They propose a targeted attack to minimize the ℓ_2-norm, which is a regular penalty method. The weight of the penalty terms is fixed to 1, however, leading to large perturbations.

Other works study the robustness of segmentation models against adversarial attacks (Fischer, Kumar, Metzen & Brox, 2017; Arnab, Miksik & Torr, 2018; Kang, Song, Du & Guizani, 2020). In these works, the authors use FGSM (Kurakin *et al.*, 2017b) or an iterative version of FGSM. However, FGSM is not a minimization attack and is known to provide rough robustness evaluations. This leads to largely overestimated robustness results on both Pascal VOC 2012 and Cityscapes in (Arnab *et al.*, 2018).

Even though most adversarial attacks were designed for classification, some may be adapted for segmentation tasks. In particular, ℓ_∞ attacks that do not rely on projections onto an estimated decision boundary can be used for segmentation (as opposed to DeepFool (Moosavi-Dezfooli *et al.*, 2016) or FAB (Croce & Hein, 2020a)). These attacks are PGD (Madry *et al.*, 2018), FMN (Pintor *et al.*, 2021) and PDPGD (Matyasko & Chau, 2021). Note that PDPGD (Matyasko & Chau, 2021) relies on a proximal splitting method, but uses the AdaProx algorithm (Melchior, Joseph & Moolekamp, 2019); the latter is appealing but, unlike the prox-Newton algorithm it is inspired from (Becker & Fadili, 2012), introduces a mismatch between the scaling in the computation of the proximity operator and the step-size of the gradient step. The convergence study of such an algorithm would be quite challenging in the non-convex case. It is also known that, even in the convex case, when such a mismatched algorithm converges, the asymptotic point differs from the solution to the original optimization problem (Savanier, Chouzenoux, Pesquet & Riddell, 2022).

5.3 Preliminaries

Let $\boldsymbol{x} \in \mathcal{X}$ be an input image with its corresponding label map $\boldsymbol{y} \in \mathcal{Y}$. Usually, in computer vision problems, $\mathcal{X} = [0,1]^{C \times H \times W}$ and $\mathcal{Y} = \{1, \dots, K\}^{H \times W}$, where H and W are the height and width of the image, C is the number of channels (*e.g.* 3 for RGB images), and K is the number of labeling classes. Our objective is to fool the model $f \colon \mathcal{X} \to \mathbb{R}^{K \times H \times W}$ producing *logits* $\boldsymbol{z} = f(\boldsymbol{x}) = \big(f(\boldsymbol{x})_{k,i,j}\big)_{k,i,j} \in \mathbb{R}^{K \times H \times W}$. This means that we are looking for a perturbation vector $\boldsymbol{\delta}$ such that, for every pixel $(i,j) \in \{1, \dots, H\} \times \{1, \dots, W\}$, the maximum value of $(f(\boldsymbol{x}+\boldsymbol{\delta})_{k,i,j})_{1 \leq k \leq K}$ is reached for a label different from the true one $\boldsymbol{y}_{i,j}$. In addition, the perturbation should satisfy the value range constraints: $\boldsymbol{x} + \boldsymbol{\delta} \in \mathcal{X}$. We denote Λ the set of admissible perturbations, so here $\Lambda = [0,1]^{C \times H \times W} - \boldsymbol{x}$. For simplicity, with a slight abuse of notation, we index the pixels with $i \in \{1, \dots, d\}$ where $d = HW$ is the total number of pixels.

5.3.1 Problem formulation

The minimal adversarial perturbation problem for the ℓ_∞-norm can be formulated as follows:

$$\begin{aligned} \underset{\boldsymbol{\delta}}{\text{minimize}} \quad \|\boldsymbol{\delta}\|_\infty \quad \text{subject to} \quad & \underset{k \in \{1,\dots,K\}}{\arg\max}\, f(\boldsymbol{x}+\boldsymbol{\delta})_{k,i} \neq \boldsymbol{y}_i, \quad i = 1, \dots, d, \\ & \boldsymbol{x} + \boldsymbol{\delta} \in \mathcal{X}. \end{aligned} \tag{5.1}$$

The d arg max constraints correspond to the misclassification of each pixel. The $\boldsymbol{x} + \boldsymbol{\delta} \in \mathcal{X}$ constraint corresponds to producing a perturbation that results in a valid image, and can be re-written as $\boldsymbol{\delta} \in \Lambda$. Note that this constraint can also be encoded in the objective using the indicator function:

$$\iota_\Lambda(\boldsymbol{\delta}) = \begin{cases} 0 & \text{if} \quad \boldsymbol{\delta} \in \Lambda; \\ +\infty & \text{else,} \end{cases} \tag{5.2}$$

resulting in the minimization of $\|\boldsymbol{\delta}\|_\infty + \iota_\Lambda(\boldsymbol{\delta})$.

5.3.2 Equivalent problem

As is common in several attacks (Carlini & Wagner, 2017; Croce & Hein, 2020b; Rony *et al.*, 2021), we replace the non-differentiable misclassification constraints by differentiable ones, to make it compatible with first-order optimization methods. Here, the arg max constraints are replaced by constraints on the Difference of Logits Ratio (DLR) (Croce & Hein, 2020a), or rather the DLR^+ (Rony *et al.*, 2021):

$$\begin{aligned} \underset{\boldsymbol{\delta}}{\text{minimize}} \quad \|\boldsymbol{\delta}\|_\infty \quad \text{subject to} \quad & \text{DLR}^+(f(\boldsymbol{x}+\boldsymbol{\delta})_i, \boldsymbol{y}_i) + \varepsilon \leq 0,\ i = 1, \dots, d, \\ & \boldsymbol{\delta} \in \Lambda, \end{aligned} \tag{5.3}$$

where $\text{DLR}^+(\boldsymbol{z}, y) = \frac{z_y - \max_{i \neq y} z_i}{z_{\pi_1} - z_{\pi_3}}$, with π_i the index of the i-th largest logit and ε a small positive constant.

5.3.3 Augmented Lagrangian method

One way to handle the misclassification constraints is to use an Augmented Lagrangian approach (Rony *et al.*, 2021), which in the classification setting, performs a gradient descent on the following quantity:

$$D(\boldsymbol{x}+\boldsymbol{\delta}, \boldsymbol{x}) + P\Big(\text{DLR}^+(g(\boldsymbol{x}+\boldsymbol{\delta})), \rho, \mu\Big), \tag{5.4}$$

where D is a discrepancy measure (*e.g.* ℓ_2-norm, LPIPS (Zhang *et al.*, 2018)), P is a penalty-Lagrangian function parametrized by a parameter ρ and a multiplier μ, and $g : \mathcal{X} \to \mathbb{R}$ is a classification model. In (Rony *et al.*, 2021), the penalty multiplier μ is updated after every gradient descent iteration to increase or decrease the weight of penalty, and eventually satisfy the misclassification constraint.

5.4 Proposed Method

Our segmentation attack is built on the general Augmented Lagrangian principle, which has led to competitive performances in the ALMA classification attack (Rony *et al.*, 2021). It provides

an efficient solution to challenging nonconvex problems arising when designing an attacker, and we will see that it can be extended to cope with multiple constraints. This can be achieved by introducing one penalty per constraint, with its associated parameter ρ and multiplier μ. Denoting $\boldsymbol{d} = \mathrm{DLR}^+(f(\boldsymbol{x}+\boldsymbol{\delta}), \boldsymbol{y}) \in \mathbb{R}^d$, we tackle problem (5.1) by minimizing the following objective:

$$\underbrace{\|\boldsymbol{\delta}\|_\infty + \iota_\Lambda(\boldsymbol{\delta})}_{h_1} \quad + \quad \underbrace{\boldsymbol{m}^\top P(\boldsymbol{d}, \boldsymbol{\rho}, \boldsymbol{\mu})}_{h_2}, \tag{5.5}$$

where $\boldsymbol{m} \in \{0,1\}^d$ is a binary mask (detailed in the following section), $(\boldsymbol{\rho}, \boldsymbol{\mu}) \in \mathbb{R}^d_{++} \times \mathbb{R}^d_{++}$ are the penalty parameters and multipliers associated with each constraint, and P is the penalty function applied componentwise. This formulation raises many technical challenges in the context of segmentation, when dealing with millions of constraints. Additionally, gradient based optimization does not accommodate nonsmooth functions such as the ℓ_∞-norm. In this work, we bring several modifications to make it (*i*) suitable for segmentation and (*ii*) applicable to the ℓ_∞-norm.

Our attack, called ALMA prox, consists in minimizing (5.5) using a proximal splitting (Combettes & Pesquet, 2011) to handle the nonsmooth term h_1 and an Augmented Lagrangian method to satisfy the constraints by minimizing h_2 using a gradient descent. In subsection 5.4.1, we introduce the adaptive constraint strategies to handle large numbers of constraints in the Augmented Lagrangian framework, and in subsection 5.4.2, we detail the proximal splitting iteration. The complete algorithm of our attack is provided in section 3.

5.4.1 Adaptive constraints strategies

5.4.1.1 Constraint masking

In segmentation tasks, some regions are unlabeled (*e.g.* object boundaries in Pascal VOC, *void* class in Cityscapes), and should be ignored during an attack. We use a binary mask $\boldsymbol{m} \in \{0,1\}^d$ to encode this, effectively reducing the number of constraints from d to $\|\boldsymbol{m}\|_1$. Given the number of constraints considered in this problem (*e.g.* $\sim 10^6$ for Cityscapes), we consider an attack as

successful if it satisfies at least a fraction ν of the constraints. In this paper, we use $\nu = 99\%$. To consolidate this in our attack, we compute a mask $\tilde{\boldsymbol{m}}^{(t)} \in \{0,1\}^d$ at each iteration $t \in \{1,\dots,N\}$ such that $\nu \|\boldsymbol{m}\|_1 \leq \|\tilde{\boldsymbol{m}}^{(t)}\|_1 \leq \|\boldsymbol{m}\|_1$. This mask discards the largest constraints, aligning the optimization objective with the criterion of successful attack. The discarded constraints are less likely to be satisfied, so this avoids the continuous increase of their associated multipliers, which may result in larger perturbations. In practice, at each iteration t, we use a threshold $\xi^{(t)}$ as a percentile of the constraints $\boldsymbol{d}^{(t)}$ to obtain $\tilde{\boldsymbol{m}}^{(t)}$, and linearly decay $\xi^{(t)}$ from 100% at the first iteration to ν at the last iteration:

$$\begin{aligned} \xi^{(t)} &= \Big(1 - (1-\nu)\frac{t-1}{N-1}\Big)\text{-percentile of } \boldsymbol{d}^{(t)}, \\ \tilde{\boldsymbol{m}}^{(t)} &= [\boldsymbol{d}^{(t)} \leq \xi^{(t)}], \end{aligned} \tag{5.6}$$

where $[\cdot]$ is the Iverson bracket applied component-wise.

5.4.1.2 Constraint scaling

When dealing with large numbers of constraints in Augmented Lagrangian methods, it is standard practice to scale them (see section 12.5 of (Birgin & Martínez, 2014)). However, there is no principled way of choosing this scale, as it is problem dependent. At iteration t, we multiply all the constraints by a scaling factor $w^{(t)} \in]0,1]$ by computing

$$h_2(\boldsymbol{\delta}) = (\tilde{\boldsymbol{m}}^{(t)})^\top P\big(w^{(t)}\boldsymbol{d}^{(t)}, \boldsymbol{\rho}^{(t)}, \boldsymbol{\mu}^{(t)}\big). \tag{5.7}$$

This scaling factor is adjusted during the attack with a binary decision: if the pixel success rate at the current iteration is less than the target pixel success rate ν, the scaling factor is increased, otherwise it is decreased:

$$w^{(t)} = w^{(t-1)} \times \begin{cases} \frac{1}{1-\gamma_w} & \text{if} \quad \frac{\boldsymbol{m}^\top[\boldsymbol{d}^{(t)} \leq 0]}{\|\boldsymbol{m}\|_1} < \nu; \\ \frac{1}{1+\gamma_w} & \text{otherwise.} \end{cases} \tag{5.8}$$

Multiplying by $\frac{1}{1\pm\gamma_w}$ instead of $1 \pm \gamma_w$ biases $w^{(t)}$ to increase in case of oscillations, with two iterations resulting in a multiplication of $\frac{1}{1-\gamma_w^2}$ instead of $1 - \gamma_w^2$. $w^{(t)}$ is further projected on a safeguarding interval $[w_{\min}, 1]$. We use $w_{\min} = 0.1$ and set the initial scale to $w^{(0)} = 1$.

5.4.1.3 Penalty

For the penalty function, we use a slightly modified version of the P_2 penalty from (Birgin & Martínez, 2014). It corresponds to the following mapping applied component-wise defined $(\forall y \in \mathbb{R})(\forall(\rho, \mu) \in [0, +\infty[^2)$:

$$P(y, \rho, \mu) = \begin{cases} \mu y + \mu\rho y^2 + \frac{1}{6}\rho^2 y^3 & \text{if} \quad y \geq 0; \\ \frac{\mu y}{1-\max(1,\rho)y} & \text{otherwise.} \end{cases} \tag{5.9}$$

This penalty also satisfies the requirements in (Birgin & Martínez, 2014), while allowing the penalty multipliers to decrease faster for negative values of the constraint with $\rho < 1$.

5.4.2 Proximal splitting

Proximal methods have been the workhorse of nonsmooth large scale optimization in the last two decades (Combettes & Pesquet, 2011; Bach, Jenatton, Mairal, Obozinski *et al.*, 2012). Originally designed for solving convex optimization problems, they have been also successfully employed in the nonconvex case (Attouch, Bolte & Svaiter, 2013). One of their main advantages is that they offer the ability to split the cost function in a sum of terms which can be addressed either by computing their proximity operator, or their gradient when they are smooth. One important point is that, generally, the limitations on the step-size arising in proximal algorithms are related to the Lipschitz-regularity of the gradient of the smooth part, whereas the proximity operators of the other functions do not introduce any restriction.

As a common practice in adversarial attacks, we are interested in minimizing the ℓ_∞-norm. To handle this term, at each iteration of the proposed algorithm, our problem is expressed as the minimization of a sum of two functions as in (5.5), where h_1 is convex and h_2 is smooth. With

this formulation, we can solve the problem using a *forward-backward* splitting algorithm where the update reads

$$\boldsymbol{\delta}^{(t+1)} = \underbrace{\operatorname{prox}_{\lambda h_1}}_{\text{backward step}} \big(\underbrace{\boldsymbol{\delta}^{(t)} - \lambda \nabla_{\boldsymbol{\delta}} h_2\big(\boldsymbol{\delta}^{(t)}\big)}_{\text{forward step}} \big), \tag{5.10}$$

λ is a positive step-size, and $\operatorname{prox}_{\lambda h_1}$ is the proximity operator of the function λh_1. Global convergence guarantees of the forward-backward algorithm exist in the convex case (Combettes & Wajs, 2005) and local ones in the nonconvex case (Attouch *et al.*, 2013). To carry out our attack, we thus need to find the proximity operator of h_1.

5.4.2.1 Proximity operator of h_1

The function h_1 is a sum of two functions: $\|\cdot\|_\infty$ and ι_Λ. In general, the proximity operator of a sum of functions *is neither* the sum of the proximity operators of the functions *nor* their composition. One way to solve the sub-problem of finding the prox of $h_1 = \|\cdot\|_\infty + \iota_\Lambda$ would be to resort to an iterative proximal splitting algorithm, such as the dual forward-backward splitting (Combettes & Pesquet, 2011). However, we will propose a more efficient numerical approach by going back to the definition of the proximity operator (Bauschke, Combettes *et al.*, 2011):

$$\begin{aligned} \operatorname{prox}_{\lambda h_1}(\boldsymbol{\delta}) &= \operatorname{prox}_{\lambda\|\cdot\|_\infty + \iota_\Lambda}(\boldsymbol{\delta}) \\ &= \operatorname*{arg\,min}_{\boldsymbol{p}\in\mathbb{R}^{Cd}} \frac{1}{2}\|\boldsymbol{p}-\boldsymbol{\delta}\|_2^2 + \lambda\|\boldsymbol{p}\|_\infty + \iota_\Lambda(\boldsymbol{p}). \end{aligned} \tag{5.11}$$

As $\mathcal{X} = [0,1]^{Cd}$, we can reformulate this problem as:

$$\begin{aligned} \underset{\boldsymbol{p},\beta}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{p}-\boldsymbol{\delta}\|_2^2 + \lambda\beta \quad \text{subject to} \quad -\beta\mathbf{1}_{Cd} \le \boldsymbol{p} \le \beta\mathbf{1}_{Cd}, \\ & \qquad\qquad\qquad\qquad\qquad\qquad -\boldsymbol{x} \le \boldsymbol{p} \le \mathbf{1}_{Cd} - \boldsymbol{x}, \end{aligned} \tag{5.12}$$

where $\mathbf{1}_{Cd} = [1,\dots,1]^\top \in \mathbb{R}^{Cd}$. Since $\Lambda = [0,1]^{Cd} - \boldsymbol{x} \subset [-1,1]^{Cd}$, the maximum possible ℓ_∞-norm of the solution $\boldsymbol{p}^\star$ is 1, so we need to solve (5.12) for $\beta \in [0,1]$. For $\beta = 0$, the

solution is trivially $\boldsymbol{p}^\star = 0$. For $\beta = 1$, the solution is $\boldsymbol{p}^\star = \mathcal{P}_\Lambda(\boldsymbol{\delta})$ since the problem reduces to

$$\underset{\boldsymbol{p}}{\text{minimize}} \quad \frac{1}{2}\left\|\boldsymbol{p} - \boldsymbol{\delta}\right\|_2^2 \quad \text{subject to} \quad -\boldsymbol{x} \leq \boldsymbol{p} \leq \mathbf{1}_{Cd} - \boldsymbol{x}. \tag{5.13}$$

For a given $\beta \in]0, 1[$, (5.12) becomes a projection problem, so we can express the solution for $\boldsymbol{p}$ as $\boldsymbol{p}_\beta = \mathcal{P}_{[-\beta,\beta]^{Cd} \cap \Lambda}(\boldsymbol{\delta})$. The optimal value of β has then to be found by solving

$$\underset{\beta \in [0,1]}{\text{minimize}} \quad \frac{1}{2}\left\|\boldsymbol{p}_\beta - \boldsymbol{\delta}\right\|_2^2 + \lambda\beta. \tag{5.14}$$

Proposition 5.1. *Problem* (5.14) *is convex and admits a unique optimal solution* $\beta^\star \in [0, 1]$*, for which the following inequality holds:*

$$0 \leq \beta^\star \leq \left\|\mathcal{P}_\Lambda(\boldsymbol{\delta})\right\|_\infty. \tag{5.15}$$

The proof of this result is provided in section 2. Several methods can be used to solve (5.14). We employ a ternary search since it offers a linear convergence rate, avoids the computation of the gradient of the objective w.r.t. β, and is numerically stable. It also yields a number of iterations depending on the required absolute precision $\Delta\beta$. Based on inequality (5.15), the number of iterations for the ternary search is $\log(\Delta\beta / \left\|\mathcal{P}_\Lambda(\boldsymbol{\delta})\right\|_\infty) / \log(2/3)$. Since we are performing adversarial attacks on images, the precision is usually 8-bit, or $1/255 \approx 3.9 \times 10^{-3}$, we solve the problem with $\Delta\beta = 10^{-5}$, yielding 29 iterations at worse, when $\left\|\mathcal{P}_\Lambda(\boldsymbol{\delta})\right\|_\infty = 1$. The procedure to compute $\boldsymbol{p}^\star = \operatorname{prox}_{\lambda\|\cdot\|_\infty + \iota_\Lambda}(\boldsymbol{\delta})$ is described in Algorithm 5.1.

5.4.2.2 Variable Metric Forward-Backward

Forward-Backward algorithms can suffer from slow convergence. Therefore, we propose to use a Variable Metric Forward Backward (VMFB) (Chouzenoux, Pesquet & Repetti, 2014; Combettes & Vũ, 2014) algorithm as an acceleration.

Algorithm 5.1 Ternary search for $\boldsymbol{p}^\star = \mathrm{prox}_{\lambda\|\cdot\|_\infty + \iota_\Lambda}(\boldsymbol{\delta})$

Input: Input vector $\boldsymbol{\delta}$ and feasible set Λ.
Input: Absolute precision $\Delta\beta$ on the solution.
$\boldsymbol{\delta}_\Lambda \leftarrow \mathcal{P}_\Lambda(\boldsymbol{\delta})$
$l \leftarrow 0$; // Lower bound for $\beta^\star$
$u \leftarrow \|\boldsymbol{\delta}_\Lambda\|_\infty$; // Upper bound for $\beta^\star$
$n \leftarrow \lceil \log(\Delta\beta/u)/\log(2/3) \rceil$; // Number of steps
for $t \leftarrow 1, \ldots, n$ **do**
  $\beta_l \leftarrow l + (u-l)/3$; // Points to evaluate the objective
  $\beta_u \leftarrow u - (u-l)/3$
  $\boldsymbol{p}_l \leftarrow \mathcal{P}_{[-\beta_l, \beta_l]}(\boldsymbol{\delta}_\Lambda)$; // Projection on interval
  $\boldsymbol{p}_u \leftarrow \mathcal{P}_{[-\beta_u, \beta_u]}(\boldsymbol{\delta}_\Lambda)$
  $f_l \leftarrow \frac{1}{2}\|\boldsymbol{p}_l - \boldsymbol{\delta}\|_2^2 + \lambda\beta_l$; // Value of objective
  $f_u \leftarrow \frac{1}{2}\|\boldsymbol{p}_u - \boldsymbol{\delta}\|_2^2 + \lambda\beta_u$
  if $f_l \geq f_u$ **then**
    $l \leftarrow \beta_l$; // Update bounds for search
  else
    $u \leftarrow \beta_u$
end for
$\beta^\star = (l+u)/2$; // Minimum is in $[l, u]$
return $\boldsymbol{p}^\star = \mathcal{P}_{[-\beta^\star, \beta^\star]}(\boldsymbol{\delta}_\Lambda)$; // Return prox using $\boldsymbol{\delta}_\Lambda$

Definition 5.1. *Let* H *be a positive definite matrix in* $\mathbb{R}^{Cd \times Cd}$. *For all* $\lambda > 0$, *the proximity operator of* λh_1 *in the metric* H *is defined as*

$$\mathrm{prox}_{\lambda h_1}^{\mathsf{H}}(\boldsymbol{\delta}) = \underset{\boldsymbol{p} \in \mathbb{R}^{Cd}}{\arg\min} \frac{1}{2}\|\boldsymbol{p} - \boldsymbol{\delta}\|_{\mathsf{H}}^2 + \lambda h_1(\boldsymbol{\delta}), \tag{5.16}$$

where $\|\boldsymbol{p}\|_{\mathsf{H}} = \sqrt{\boldsymbol{p}^\top \mathsf{H} \boldsymbol{p}}$ *is a weighted norm.*

Following the definition, the update in VMFB reads

$$\boldsymbol{\delta}^{(t+1)} = \mathrm{prox}_{\lambda h_1}^{\mathsf{H}}\left(\boldsymbol{\delta}^{(t)} - \lambda \mathsf{H}^{-1} \nabla_{\boldsymbol{\delta}} h_2\big(\boldsymbol{\delta}^{(t)}\big)\right). \tag{5.17}$$

This method requires inverting a $Cd \times Cd$ square matrix, which may be impractical. Therefore, we use a diagonal metric $\mathsf{H} = \mathrm{Diag}(\boldsymbol{s})$ with $\boldsymbol{s} \in]0, +\infty[^{Cd}$. At iteration t, the diagonal vector

$\boldsymbol{s}^{(t)}$ is estimated from the gradient as

$$\begin{aligned} \boldsymbol{v}^{(t)} &= \alpha \boldsymbol{v}^{(t-1)} + (1-\alpha)\big(\nabla_{\boldsymbol{\delta}} h_2(\boldsymbol{\delta}^{(t)})\big)^2 \\ \boldsymbol{s}^{(t)} &= \sqrt{\frac{\boldsymbol{v}^{(t)}}{1-\alpha^t}} + \varepsilon \end{aligned} \tag{5.18}$$

where the square and square root operations are performed componentwise, $\alpha \in [0, 1[$ is a smoothing parameter, and $\boldsymbol{v}^{(0)} = \mathbf{0}_{Cd}$.

Remark. With this choice for the metric, the forward (*i.e.* gradient) step becomes equivalent to the one in the Adam algorithm (Kingma & Ba, 2015) with $(\beta_1, \beta_2) = (0, \alpha)$.

Note that Proposition 5.1 still holds for a diagonal metric. Indeed, the projection on Λ w.r.t. a diagonal metric $\mathcal{P}_\Lambda^{\mathsf{H}} = \arg\min_{\boldsymbol{y}\in\Lambda} \|\cdot - \boldsymbol{y}\|_{\mathsf{H}}^2$ is equal to the Euclidean projection $\mathcal{P}_\Lambda = \arg\min_{\boldsymbol{y}\in\Lambda} \|\cdot - \boldsymbol{y}\|_2^2$. This is readily deduced from the fact that both projection problems are separable in each component, since H is diagonal and the constraint $\boldsymbol{y} \in \Lambda$ is separable. Therefore, since $\mathcal{P}_\Lambda^{\mathsf{H}} = \mathcal{P}_\Lambda$, the proof is unchanged. Additionally, the ternary search approach to compute $\boldsymbol{p}^\star$ can still be used: the norms in steps 10 and 11 are replaced by the weighted norm.

5.5 Experiments

5.5.1 Ternary Search

To evaluate the efficiency of our method to compute $\mathrm{prox}_{\lambda h_1}$ using a ternary search, we provide a comparison with several traditional iterative splitting algorithms: Dual Forward-Backward (DFB) splitting (Combettes, Dũng & Vũ, 2010), an Accelerated variant (ADFB) using Nesterov acceleration on the dual problem (Chambolle & Dossal, 2015), and a Douglas-Rachford (DR) splitting algorithm (Lions & Mercier, 1979). We generate pseudorandom samples $\boldsymbol{x} \sim \mathcal{U}(\mathbf{0}_d, \mathbf{1}_d)$, perturbation vectors $\boldsymbol{\delta} \sim \mathcal{N}(0, \sigma^2 I_d)$, and scale $\lambda \sim 10^{\mathcal{U}(-1,3)}$, with $d = 2^{18} = 512{\times}512$. For all methods, we use an absolute stopping criterion on the solution $\left\|\boldsymbol{p}^{(t+1)} - \boldsymbol{p}^{(t)}\right\|_\infty \leq 10^{-5}$. For

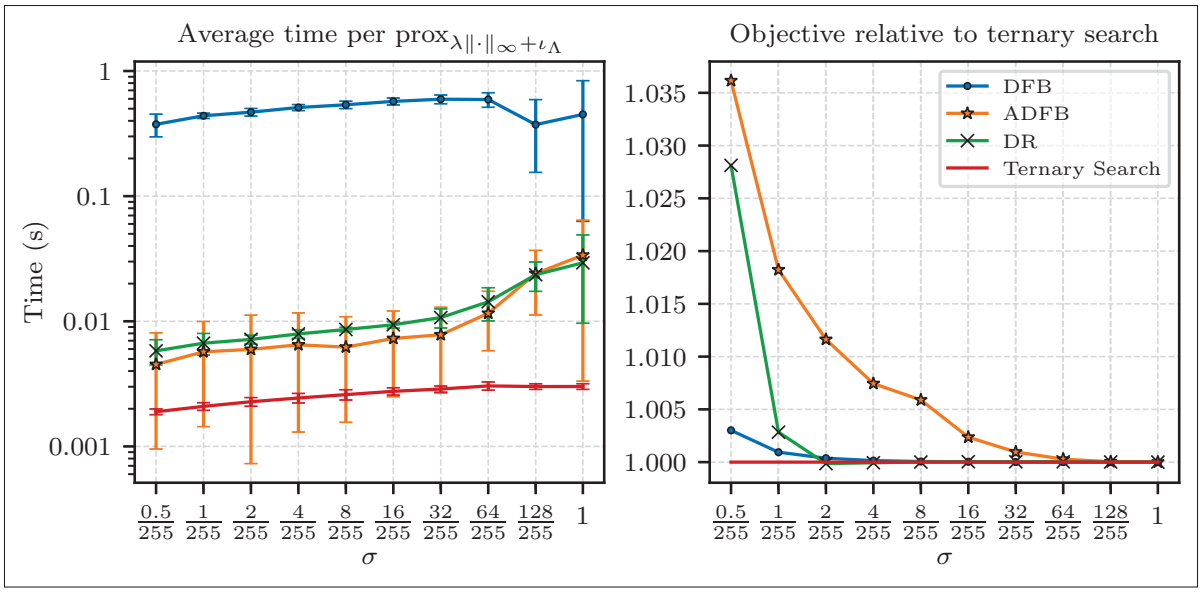


Figure 5.2 Comparison of average run-time and quality of solution in terms of optimization objective of DFB, ADFB, DR and Ternary Search for pseudorandom vectors $\boldsymbol{\delta} \sim \mathcal{N}(0, \sigma^2 I_d)$

each σ, we repeat the evaluation 100 times, and report the average compute time[13] and the value of the objective from (5.11) relative to the one obtained using the ternary search.

The results are shown in Figure 5.2. Overall, the Ternary Search approach is faster and provides more accurate solutions to the computation of the proximity operator of interest than DFB, ADFB and DR. In some occasions, it may happen that ADFB converges faster, but the trade-off is a worse solution in terms of objective on average.

5.5.2 Datasets and Models

We perform the experiments on the validation set of two well-known segmentation datasets: Pascal VOC 2012 (Everingham *et al.*, 2012) and Cityscapes (Cordts *et al.*, 2016).

We evaluate the attacks on a collection of models chosen based on several criteria: widespread usage, high performance, and architectures diversity. With these criteria, we selected DeepLabV3+

[13] Experiments were run on NVIDIA A100 SXM4 40 GB GPUs.

ResNet-50, ResNet-101 (Chen *et al.*, 2018a), and FCN HRNetV2 W48 (Wang *et al.*, 2020) which are CNN based models, and SegFormer MiT-B0 and MiT-B3 (Xie *et al.*, 2021) which are transformer based models. We also consider the robust model DeepLabV3 ResNet-50 DDC-AT from (Xu *et al.*, 2021).

For white-box attacks, most attack algorithms compute the gradient of a loss w.r.t. the input. In the context of adversarial attacks, this quantity is computed for validation images, which can be larger than the image crops used in training. For Cityscapes specifically, models are evaluated on 2 048×1 024 images. This leads to high memory usage, and, in particular, DeepLabV3+ ResNet-101 and larger variants of the SegFormer family (*i.e*. MiT-B4 and MiT-B5) require more than 40 GB of memory per gradient computation on 2 048×1 024 images. Therefore, we refrain from doing experiments on models requiring more than 40 GB of GPU memory[13] to ease future comparisons.

For most segmentation models, the evaluation protocol involves resizing the image to a specified size (*e.g*. such that the smallest side has a specified length, while keeping the aspect ratio), using the model to produce a segmentation, and then resizing this segmentation to the original image size and compute the performance metrics. In our context, we do not perform this resizing before and after. This slight modification of the evaluation protocol leads to marginal differences in the typical performance metrics, which we report in section 4. All the models weights are fetched from the MMSegmentation library (MMSegmentation Contributors, 2020), except for the robust model DeepLabV3 DDC-AT from (Xu *et al.*, 2021), which was obtained from the repository associated with the paper.

5.5.3 Metrics

In segmentation, the concept of a *successful attack* is more ambiguous than in classification, where success is a binary criterion. From a security perspective in a segmentation task, *mostly* making wrong predictions, except for a few pixels, can be considered as a successful attack (or a model failure), even though all the constraints are not satisfied.

Previous works on adversarial examples in a segmentation context (Arnab *et al.*, 2018; Xie *et al.*, 2017; Xu *et al.*, 2021) measure the model robustness (or equivalently, attack performance) using the mean Intersection over Union (mIoU) over all classes. However, this metric is biased towards small regions, and does not indicate how well the adversarial optimization problem (5.1) is solved. Therefore, to measure the success of an attack, we simply measure the constraint satisfaction rate over all pixels in the mask $\boldsymbol{m}$, irrespective of the original class. For a given image, we call this constraint satisfaction rate the Attack Pixel Success Rate (APSR). For untargeted attacks, the APSR is defined as:

$$\text{APSR} = \frac{\boldsymbol{m}^\top [\arg\max_k f(\boldsymbol{x}+\boldsymbol{\delta})_{k,i} \neq \boldsymbol{y}_i]}{\|\boldsymbol{m}\|_1} \in [0,1] \tag{5.19}$$

Although our approach has been presented in the context of untargeted attacks, it can also be straightforwardly extended to targeted attacks. In such a case, a target label $\boldsymbol{t} = (\boldsymbol{t}_i)_{1\leq i\leq d}$ is provided and the statement becomes $\arg\max_k f(\boldsymbol{x}+\boldsymbol{\delta})_{k,i} = \boldsymbol{t}_i$.

From there, choosing a specific threshold is arbitrary. In our experiments, we use $\nu = 99\%$ as a threshold for the APSR, meaning that an attack is considered successful if APSR $\geq 99\%$. From an optimization perspective, this indicates that we satisfy at least 99% of the constraints in problem (5.1). A lower threshold could also result in low quality segmentations. However, given the small ℓ_∞-norms of the perturbations observed to satisfy such a high threshold, we argue that it highlights the effectiveness of our attack.

5.5.4 Attack objectives

The most common scenario for adversarial attacks is the untargeted setting. While it is natural for classification, this can produce unrealistic segmentations that could easily be filtered or rejected (see Figure 5.1). Therefore, we need to find a more natural objective for the adversarial attack depending on the context. For Pascal VOC, there is a *background* class, which can be chosen as the target class for the whole image. This means that a successful targeted attack would produce a segmentation with no object of interest.

In contrast, Cityscapes does not consider a *background* class, so we need a plausible target. A strategy explored in (Fischer *et al.*, 2017) is to *erase* a class and select the nearest neighbor that is not from the same class as the target label. This produces natural segmentation labels given the context, but it is not clear which class should be erased in general. For our experiments, we compute a target label based on the majority label for each pixel over the whole training set. This produces a more natural looking segmentation, but with high spatial frequencies, which differs from the dataset segmentation labels. Therefore, we draw a smoothed version of this target, avoiding the high frequencies, while keeping the structure. This target is provided in section 5.

5.5.5 Attacks

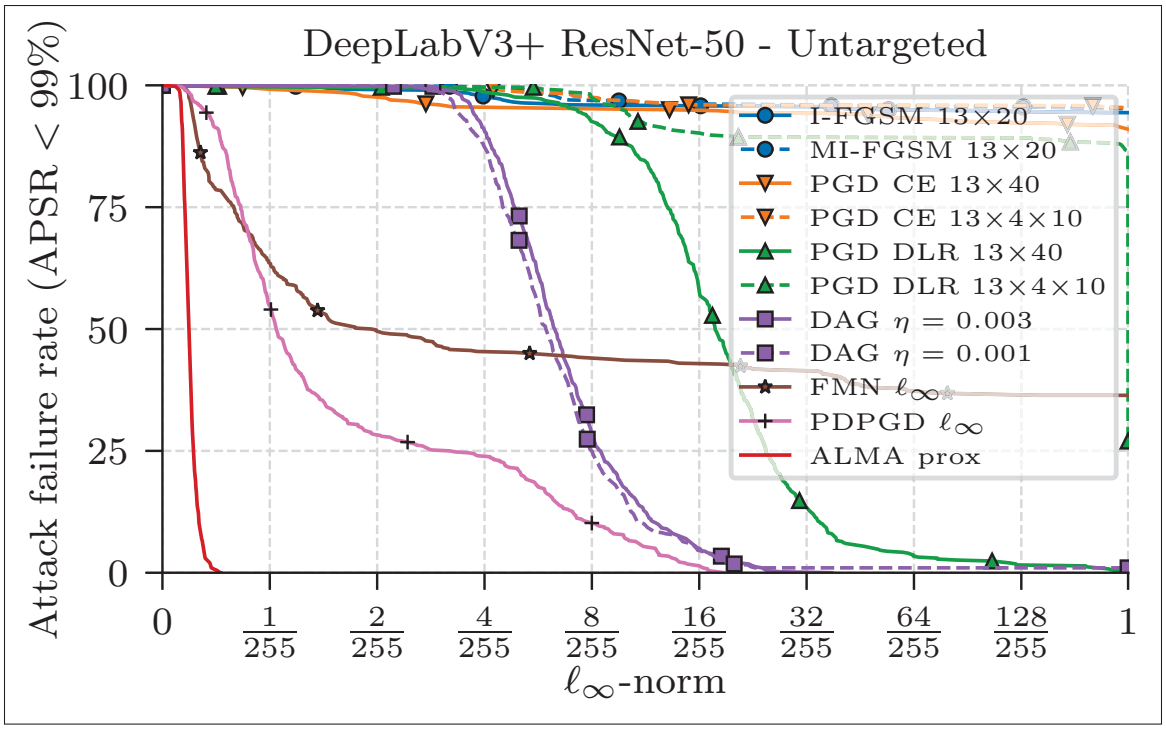


Figure 5.3 Percentage of unsuccessful untargeted attacks for DeepLabV3+ ResNet-50 on Cityscapes. Horizontal axis is linear on [0, 2/255] and logarithmic on [2/255, 1]

For the minimization attacks, we set a 500 iterations budget, corresponding to a ~24 hours run-time to attack the entire validation set of Cityscapes with the largest model.

5.5.5.1 DAG

The main baseline in our experiments is DAG (Xie *et al.*, 2017). We report the results with two step-sizes: 0.003 and 0.001;, such that the larger does not fail to find adversarial perturbations, while the smaller finds smaller perturbations for some samples, but fails on others.

5.5.5.2 Other baselines

As mentioned in section 5.2, we can adapt some attacks originally designed for classification. We consider I-FGSM (Kurakin *et al.*, 2017b) and MI-FGSM (Dong *et al.*, 2018) with 20 iterations and use a 13 steps binary search on the ℓ_∞-norm, yielding an error of $2^{-13} \approx 10^{-4}$, to find the smallest norm for which the attack succeeds. Similarly, we include four variants of the PGD attack (Madry *et al.*, 2018) with a binary search. These variants use the Cross-Entropy (CE) or the DLR loss from (Croce & Hein, 2020b), in combination with 40 steps or 10 steps with 3 random restarts (totaling 40 steps).[14] We also adapt the ℓ_∞ variant of FMN (Pintor *et al.*, 2021) by taking the loss as the average over the mask $\boldsymbol{m}$ and testing if APSR $\geq \nu$ as the binary decision for the projection step. We use $\alpha = 10$ in FMN (see section 3.1 of (Pintor *et al.*, 2021)). Finally, we modify the PDPGD attack (Matyasko & Chau, 2021) to take into account the constraint masking strategies. We provide the details of the modifications, as well as an ablation study on the impact of these strategies in section 6. We could not include the Houdini attack (Cisse *et al.*, 2017) as no public implementation is available.

5.5.5.3 ALMA prox

For our attack, we use an initial step-size $\lambda^{(0)} = 10^{-3}$ and decay it to $\lambda^{(N)} = 10^{-4}$. With a budget of 500 iterations, we set $\alpha = 0.8$, compared to 0.9 for 1 000 iterations in (Rony *et al.*, 2021). We set the $\mu^{(0)} = \mathbf{1}_d$ and $\rho^{(0)} = 0.01 \cdot \mathbf{1}_d$, the penalty parameter increase rate $\gamma = 2$ and the constraint scale adjustment rate to $\gamma_w = 0.02$.

[14] Given the poor performance of these attacks on the regularly trained models, we do not evaluate them on the robust model from (Xu *et al.*, 2021).

5.6 Results

5.6.1 Perturbation size

Table 5.1 Median and average norms $\|\boldsymbol{\delta}\|_\infty \times 255$ for each adversarial attack on Pascal VOC 2012 and Cityscapes

	Attack	Pascal VOC 2012				Cityscapes			
		DeepLabV3+ ResNet-50		FCN HRNetV2 W48		DeepLabV3+ ResNet-50		SegFormer MiT-B3	
Untargeted	I-FGSM 13×20	146.32	136.55	227.11	145.69	255.00	242.15	255.00	208.97
	MI-FGSM 13×20	195.35	145.96	255.00	157.63	255.00	244.43	255.00	228.43
	PGD CE 13×40	80.10	120.93	152.08	136.05	255.00	236.77	255.00	188.15
	PGD CE $13 \times 4 \times 10$	24.90	74.15	66.93	118.76	255.00	244.92	255.00	231.51
	PGD DLR 13×40	7.22	26.42	11.11	23.85	17.75	24.23	84.22	102.80
	PGD DLR $13 \times 4 \times 10$	4.42	24.99	10.46	29.75	255.00	227.89	110.82	134.41
	DAG $\eta = 0.003$	5.69	6.63	8.80	10.61	6.30	7.51	8.83	9.02
	DAG $\eta = 0.001$	5.23	8.22	8.49	14.61	5.95	9.39	8.59	53.65
	FMN ℓ_∞	0.46	38.34	0.91	46.39	1.97	96.57	1.08	6.42
	PDPGD ℓ_∞	0.73	1.77	1.52	2.43	1.06	2.82	1.39	3.05
	ALMA prox	**0.32**	**0.34**	**0.51**	**0.56**	**0.24**	**0.26**	**0.33**	**0.33**
Targeted	I-FGSM 13×20	0.47	0.50	0.59	0.64	255.00	255.00	51.73	88.24
	MI-FGSM 13×20	0.59	0.66	0.77	0.85	4.26	55.35	2.54	2.60
	PGD CE 13×40	0.37	0.43	0.50	0.54	3.49	3.71	1.87	1.92
	PGD CE $13 \times 4 \times 10$	0.50	0.51	0.62	0.65	255.00	255.00	255.00	255.00
	PGD DLR 13×40	0.62	0.68	0.81	0.94	8.37	67.86	3.98	4.09
	PGD DLR $13 \times 4 \times 10$	0.87	1.07	1.12	1.28	255.00	255.00	255.00	255.00
	DAG $\eta = 0.003$	4.21	4.50	5.32	5.66	11.34	12.96	9.82	10.06
	DAG $\eta = 0.001$	3.92	4.21	5.07	5.36	10.96	40.28	11.53	119.47
	FMN ℓ_∞	0.42	0.45	0.47	0.49	255.00	254.36	255.00	255.00
	PDPGD ℓ_∞	0.28	0.36	0.35	0.46	14.51	14.41	19.26	19.20
	ALMA prox	**0.25**	**0.26**	**0.32**	**0.34**	**1.15**	**1.17**	**0.65**	**0.66**

For each dataset, model, and scenario (untargeted and targeted), we plot the attack failure rate as a function of the perturbation size. This failure rate corresponds to the fraction of samples for which an attack has *not* found an adversarial example with APSR $\geq 99\%$. This kind of plot can be interpreted in two ways: model-centric and attack-centric. In a model-centric analysis, a more robust model will require larger perturbations to be fooled, and therefore, correspond to a curve towards the upper right. In an attack-centric analysis, a stronger attack will find smaller perturbations, and have a curve towards the lower left.

Table 5.2 Median and average norms $\|\delta\|_\infty \times 255$ for each adversarial attack on the robust model DeepLabV3 DDC-AT Xu *et al.* (2021)

	Attack	Pascal VOC 2012		Cityscapes	
Untargeted	DAG $\eta = 0.003$	12.05	25.96	16.99	25.81
	DAG $\eta = 0.001$	11.71	57.93	17.02	81.30
	FMN ℓ_∞	1.28	75.66	37.54	128.67
	PDPGD ℓ_∞	3.53	11.86	43.61	58.10
	ALMA prox	**0.78**	**2.17**	**0.81**	**1.01**
Targeted	DAG $\eta = 0.003$	7.01	7.68	31.60	69.84
	DAG $\eta = 0.001$	6.78	8.78	255.00	230.29
	FMN ℓ_∞	1.31	36.10	255.00	255.00
	PDPGD ℓ_∞	0.84	1.23	255.00	255.00
	ALMA prox	**0.52**	**0.54**	**3.64**	**3.92**

Figure 5.3 shows this plot for untargeted attacks on Cityscapes with DeepLabV3+ ResNet-50. To improve readability, we use a linear scale on $[0, 2/255]$ and a logarithmic scale on $[2/255, 1]$. Here, DAG needs a norm of $8/255$ to successfully fool ~75% of the samples (with APSR $\geq$ 99%), and $24/255$ to fool 99% of the samples. In contrast, ALMA prox finds adversarial perturbations with ℓ_∞-norms smaller than $0.55/255$ for *all* samples. This contradicts the robustness results in Arnab *et al.* (2018), where models have mIoUs of ~50% at $1/255$, as opposed to 2.1% at $0.55/255$ here.

Table 5.1 reports the median and average ℓ_∞-norm (multiplied by 255 for readability) of the perturbations produced by the attacks for a subset of the regular models: DeepLabV3+ ResNet-50 and FCN HRNetV2 W46 on Pascal VOC 2012, and DeepLabV3+ ResNet-50 and SegFormer MiT-B3 on Cityscapes. For unsuccessful attacks (*i.e.* APSR < 99%), the perturbation size is considered to be 1 (*i.e.* $255/255$). Overall, the ALMA prox attack outperforms all the attacks considered in our experiments. On Pascal VOC, most attacks can find small (*e.g.* $\leq 1/255$) perturbations in the targeted scenario. This is expected because the most frequent class in Pascal VOC is the background. However, the differences are much larger in the untargeted scenario, which corresponds to predicting mostly non-background classes. On Cityscapes, the scale of the problem ($\sim 10^6$ constraints) highlights the difficulty of generating adversarial perturbations for segmentation models. PDPGD handles the untargeted scenario well, with median ℓ_∞-norms

of $\sim 1/255$, but produces much larger perturbations in the targeted case. Contrarily, PDG CE 13×40 finds small targeted perturbations, but fails in the untargeted scenario. Finally, FMN has inconsistent performance across samples: on Cityscapes in the untargeted scenario, the low median and high average indicates that it fails to find for a large portion of the samples. This can also be seen in Figure 5.3, where FMN finds perturbation with ℓ_∞-norms smaller than $2/255$ for $\sim$50% of the samples, but fails for $\sim$35% of them. The results for all models can be found in section 8 with similar trends on the other models.

Finally, the median and average perturbation norms for the robust model DeepLabV3 DDC-AT are reported in Table 5.2. On Pascal VOC, several attacks succeed in finding small perturbations. However, Cityscapes is, again, more challenging, especially in the targeted scenario.

5.6.2 Attack complexity

The observed average complexities in terms of number of forward and backward propagations of the model (see Section 4 of Rony *et al.* (2021)) and run-times are reported in section 7. The number of propagations is equal to the iteration budget for all attacks except DAG, which uses an early stopping criterion. Therefore, DAG has lower complexity on average. For smaller step-sizes, it can reach the limit of 500 iterations for some samples (*e.g.* for SegFormer models on Cityscapes, see Figure IV-4) and fails to find adversarial perturbations. The second observation is that the proximity operator used in ALMA prox does not significantly increase the run-time compared to the other attacks with similar number of forward and backward propagations.

5.7 Conclusion

We proposed an adversarial attack for deep semantic segmentation models to produce minimal perturbations w.r.t. the ℓ_∞-norm. Our attack is based on an Augmented Lagrangian method, which allows us to tackle large numbers of misclassification constraints using gradient-based optimization, coupled with a proximal splitting of the objective to minimize the ℓ_∞-norm and satisfy the input space constraints with a proximity operator. Additionally, we devised an efficient

method to compute this proximity operator, which is compatible with a VMFB acceleration based on a diagonal metric. Our attack offers significant improvements in terms of ℓ_∞-norm minimization for segmentation tasks, even for a robust model. One limitation of our method is that it does not produce valid digitized images, *i.e*. encoded with a reduced number of bits, such as 8. For a discussion on this issue, see (Bonnet, Furon & Bas, 2020). Note that all the attacks considered in our experiments do not produce valid images as well.

CHAPTER 6

CLASS ADAPTIVE NETWORK CALIBRATION

Bingyuan Liu*[1] , Jérôme Rony*[1] , Adrian Galdran [2] , Jose Dolz [1] , Ismail Ben Ayed[1]

* Equal contribution
[1] Systems Engineering Department, École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada

[2] Universitat Pompeu Fabra, Barcelona, Spain

Paper accepted for publication at the
IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2023

Abstract

Recent studies have revealed that, beyond conventional accuracy, calibration should also be considered for training modern deep neural networks. To address miscalibration during learning, some methods have explored different penalty functions as part of the learning objective, alongside a standard classification loss, with a hyper-parameter controlling the relative contribution of each term. Nevertheless, these methods share two major drawbacks: 1) the scalar balancing weight is the same for all classes, hindering the ability to address different intrinsic difficulties or imbalance among classes; and 2) the balancing weight is usually fixed without an adaptive strategy, which may prevent from reaching the best compromise between accuracy and calibration, and requires hyper-parameter search for each application. We propose Class Adaptive Label Smoothing (CALS) for calibrating deep networks, which allows to learn class-wise multipliers during training, yielding a powerful alternative to common label smoothing penalties. Our method builds on a general Augmented Lagrangian approach, a well-established technique in constrained optimization, but we introduce several modifications to tailor it for large-scale, class-adaptive training. Comprehensive evaluation and multiple comparisons on a variety of benchmarks, including standard and long-tailed image classification, semantic segmentation, and text classification, demonstrate the superiority of the proposed method. The code is available at https://github.com/by-liu/CALS.

6.1 Introduction

Deep Neural Networks (DNNs) have become the prevailing model in machine learning, particularly for computer vision(He *et al.*, 2016a) and natural language processing applications(Vaswani *et al.*, 2017). Increasingly powerful architectures(He *et al.*, 2016a; Chen, Papandreou, Schroff & Adam, 2017; Liu *et al.*, 2021), learning methods (Chen *et al.*, 2020; He *et al.*, 2022) and a large body of other techniques (Ioffe & Szegedy, 2015; Loshchilov & Hutter, 2019) are constantly introduced. Nonetheless, recent studies(Guo *et al.*, 2017; Mukhoti *et al.*, 2020) have shown that regardless of their superior discriminative performance, high-capacity modern DNNs are poorly calibrated, *i.e*. failing to produce reliable predictive confidences. Specifically, they tend to yield over-confident predictions, where the probability associated with the predicted class overestimates the actual likelihood. Since this is a critical issue in safety-sensitive applications like autonomous driving or computational medical diagnosis, the problem of DNN calibration has been attracting increasing attention in recent years (Guo *et al.*, 2017; Mukhoti *et al.*, 2020; Pereyra *et al.*, 2017).

Current calibration methods can be categorized into two main families. The first family involves techniques that perform an additional post-processing parameterized operation on the output logits (or pre-softmax activations) (Guo *et al.*, 2017), with the calibration parameters of that operation obtained from a validation set by either learning or grid-search. Despite the simplicity and low computational cost, these methods have empirically proven to be highly effective (Guo *et al.*, 2017; Ding, Han, Liu & Niethammer, 2021). However, their main drawback is that the choice of the optimal calibration parameters is highly sensitive to the trained model instance and validation set (Mukhoti *et al.*, 2020; Liu *et al.*, 2022a).

The second family of methods attempts to simultaneously optimize for accuracy and calibration during network training. This is achieved by introducing, explicitly or implicitly, a secondary optimization goal involving the model's predictive uncertainty, alongside the main training objective. As a result, a scalar balancing hyper-parameter is required to tune the relative contribution of each term in the overall loss function. Some examples of this type of approaches

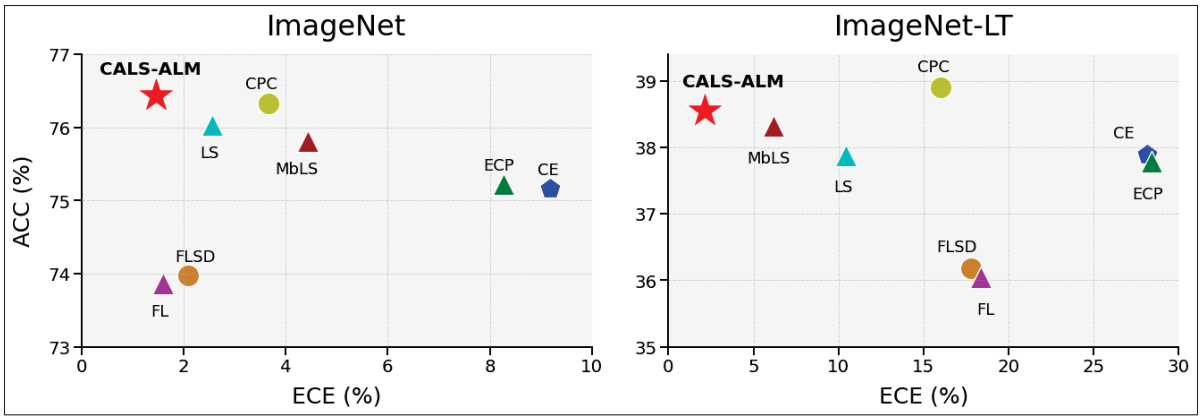


Figure 6.1 Many techniques have been proposed for jointly improving accuracy and calibration during training (Guo *et al.*, 2017; Mukhoti *et al.*, 2020), but they fail to consider uneven learning scenarios like high class imbalance or long-tail distributions. We show a comparison of the proposed **CALS-ALM** method and different learning approaches in terms of Calibration Error (ECE) vs Accuracy on the (a) ImageNet and (b) ImageNet-LT (long-tailed ImageNet) datasets. A lower ECE indicates better calibration: a better model should attain **high ACC and low ECE**. Among all the considered methods, **CALS-ALM** shows superior performance when considering both discriminative power and well-balanced probabilistic predictions, achieving best accuracy and calibration on ImageNet, and best calibration and second best accuracy on ImageNet-LT.

include: Explicit Confidence Penalty (ECP)(Pereyra *et al.*, 2017), Label Smoothing (LS)(Müller, Kornblith & Hinton, 2019), Focal Loss (FL) (Lin *et al.*, 2017) and its variant, Sample-Dependent Focal Loss (FLSD) (Mukhoti *et al.*, 2020). It has been recently demonstrated in(Liu *et al.*, 2022a) that all these methods can be formulated as different penalty terms that enforce the same equality constraint on the logits of the DNN: driving the *logit distances* towards zero. Here, logit distances refers to the vector of L1 distances between the highest logit value and the rest. Observing the non-informative nature of this equality constraint, (Liu *et al.*, 2022a) proposed to use a generalized inequality constraint, only penalizing those logits for which the distance is larger than a pre-defined margin, achieving state-of-the-art calibration performance on many different benchmarks.

Although learning based methods achieve greater calibration performance (Mukhoti *et al.*, 2020; Liu *et al.*, 2022a), they have two major limitations: 1) The scalar balancing weight is equal for all classes. This hinders the network performance when some classes are harder to learn or less

represented than others, such as in datasets with a large number of categories (ImageNet) or considerable class imbalance (ImageNet-LT). 2) The balancing weight is usually fixed before network optimization, with no learning or adaptive strategy throughout training. This can prevent the model from reaching the best compromise between accuracy and calibration. To address the above issues, we introduce Class Adaptive Label Smoothing method based on an Augmented Lagrangian Multiplier algorithm, which we refer to as CALS-ALM. Our **Contributions** can be summarized as follows:

- We propose Class Adaptive Label Smoothing (CALS) for network calibration. Adaptive class-wise multipliers are introduced instead of the widely used single balancing weight, which addresses the above two issues: 1) CALS can handle a high number of classes with different intrinsic difficulties, *e.g*. ImageNet; 2) CALS can effectively learn from data suffering from class imbalance or a long-tailed distribution, *e.g*. ImageNet-LT.
- Different from previous penalty based methods, we solve the resulting constrained optimization problem by implementing a modified Augmented Lagrangian Multiplier (ALM) algorithm, which yields adaptive and optimal weights for the constraints. We make some critical design decisions in order to adapt ALM to the nature of modern learning techniques: 1) The inner convergence criterion in ALM is relaxed to a fixed number of iterations in each inner stage, which is amenable to mini-batch stochastic gradient optimization in deep learning. 2) Popular techniques, such as data augmentation, batch normalization (Ioffe & Szegedy, 2015) and dropout (Gal & Ghahramani, 2016), rule out the possibility of tracking original samples and applying sample-wise multipliers. To overcome this complication, we introduce class-wise multipliers, instead of sample-wise multipliers in the standard ALM. 3) The outer-step update for estimating optimal ALM multipliers is performed on the validation set, which is meaningful for training on large-scale training set and avoids potential overfitting.
- Comprehensive experiments over a variety of applications and benchmarks, including standard image classification (Tiny-ImageNet and ImageNet), long-tailed image classification (ImageNet-LT), semantic segmentation (PASCAL VOC 2012), and text classification (20 Newsgroups), demonstrate the effectiveness of our CALS-ALM method. As shown in Figure 6.1, CALS-ALM yields superior performance over baselines and state-of-the-art

calibration losses when considering both accuracy and calibration, especially for more realistic large-scaled datasets with large number of classes or class imbalance.

6.2 Related Work

6.2.1 Problem Formulation

Given a dataset $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$ with $N \in \mathbb{N}$ pairs of samples $\boldsymbol{x} \in \mathcal{X}$ and corresponding labels $y \in \mathcal{Y}$, with $\mathcal{Y} = \{1, \dots, K\}$, a deep neural network (DNN) $f_\theta : \mathcal{X} \to \mathbb{R}^K$ parameterized by θ yields *logits* $\boldsymbol{z} = f_\theta(\boldsymbol{x}) = (f_\theta(\boldsymbol{x})_k)_{1 \le k \le K} \in \mathbb{R}^K$. In a classification scenario, the output probability $\boldsymbol{s} = (s_k)_{1 \le k \le K} \in \Delta^{K-1}$, where $\Delta^{K-1} \subset [0, 1]^K$ denotes the probability simplex, is obtained by applying the softmax function on the output *logits*, *i.e.* $\boldsymbol{s} = \mathrm{softmax}(\boldsymbol{z}) = \frac{\exp \boldsymbol{z}}{\sum \exp \boldsymbol{z}}$. Therefore, the predicted class $\hat{y}$ is computed as $\hat{y} = \arg\max_k s_k$, and the predicted confidence is $\hat{p} = \boldsymbol{s}_{\hat{y}} = \max_k s_k$. A perfectly calibrated model should satisfy that the predicted confidence of any input is equal to the accuracy of the model: $\hat{p} = P(\hat{y} = y | \hat{p})$. Hence, an over-confident model yields on average larger confidences than the associated accuracy, whereas an under-confident model yields lower confidence than its accuracy.

A number of recent studies (Guo *et al.*, 2017; Mukhoti *et al.*, 2020; Minderer *et al.*, 2021; Liu *et al.*, 2022a) have shown that DNNs tend to become over-confident during training as a result of minimizing the popular cross-entropy (CE) training loss:

$$\mathcal{L}_{\mathrm{CE}}(\boldsymbol{x}, y) = -\sum_{k=1}^{K} y_k \log s_k \tag{6.1}$$

where $\boldsymbol{y} \in \{0, 1\}^K$ is the one-hot encoding of y. This objective function is minimized when the predictions for all the training samples fully match the ground-truth labels y, *i.e.* $s_y = 1$ and $\forall k \neq y, s_k = 0$. The negative logarithmic term on the logit of the correct category renders the global minimization of the CE loss unreachable, as it keeps pushing the predicted probabilities *vs*. towards the vertices of the $(K-1)$-simplex even after the classification error is zero (Mukhoti *et al.*, 2020), resulting in over-confident models.

6.2.2 Post-processing methods

To address mis-calibration, different post-processing techniques applied after model training have been proposed recently (Guo *et al.*, 2017; Ding *et al.*, 2021; Tomani, Gruber, Erdem, Cremers & Buettner, 2021). The most popular of these strategies is Temperature Scaling (TS) (Guo *et al.*, 2017), which applies a single scalar temperature parameter to manipulate logit outputs monotonely, resulting in softened prediction confidences without affecting predicted labels. Note that here the temperature parameter needs to be tuned on a separate validation set. Despite its simplicity, TS has been shown effective in fixing over-confidence predictions (Guo *et al.*, 2017). As a local alternative to TS, (Ding *et al.*, 2021) proposed to train a regression model for learning position-specific temperature for semantic segmentation problems. Unfortunately, TS and its variants can be sensitive to both the model and the validation set, and do not work well under data distribution shifts (Ovadia *et al.*, 2019). Thus, some subsequent works (Tomani *et al.*, 2021; Ma & Blaschko, 2021) have attempted to provide solutions for improving performance under domain shift.

6.2.3 Learning-based methods

Another popular direction is to directly address mis-calibration during training by introducing an additional penalty or supervision regarding model calibration with the standard training loss. In (Kumar *et al.*, 2018), the authors introduced a trainable calibration measure based on RKHS kernels, while (Karandikar *et al.*, 2021) proposed a differential calibration loss based on a soft version of the binning operation in the ECE metric. In (Cheng & Vasconcelos, 2022), two types of binary pairwise calibration constraints were proposed as additional penalty terms during training. Other methods try to decrease over-fitting on the cross-entropy loss, which has been demonstrated to be the main reason of over-confidence (Guo *et al.*, 2017; Mukhoti *et al.*, 2020). In (Pereyra *et al.*, 2017) an explicit confidence penalty (ECP) is proposed to maximize the entropy and reduce over-fitting, while Label Smoothing (Szegedy *et al.*, 2016) has also been shown to implicitly improve the calibration (Müller *et al.*, 2019) by softening the hard one-hot targets in the cross-entropy. The Focal Loss (Lin *et al.*, 2017), originally proposed

to tackle class imbalance, can also be effective for calibration (Mukhoti *et al.*, 2020), as it implicitly minimizes the Kullback-Leibler (KL) divergence between the uniform distribution and the network softmax probabilities, thereby increasing the entropy of the predictions. As an extension the Sample-Dependent Focal Loss (FLSD) was also proposed in (Mukhoti *et al.*, 2020) to further boost calibration performance.

Margin-based Label Smoothing (MbLS) A unifying constrained-optimization formulation of loss functions promoting calibration has been recently presented in (Liu *et al.*, 2022a). Specifically, the additional penalties integrated in these methods, including ECP (Pereyra *et al.*, 2017), LS (Müller *et al.*, 2019) and FL (Mukhoti *et al.*, 2020), can be viewed as different forms of approximations to the same constraint, *i.e*. enforcing the logit distances to be zero. Noticing that this constraint is non-informative (its solution being uniformly distributed probabilities), (Liu *et al.*, 2022a) further proposed a generalized formulation by relaxing the constraint to allow the logit distances being lower than a strictly positive margin.

The specific formulation of MbLS (Liu *et al.*, 2022a) is as follows. Given a margin $m \in \mathbb{R}_+$, the constrained optimization problem for network training is:

$$\underset{\theta}{\text{minimize}} \quad \sum_{i=1}^{N} \mathcal{L}_{\text{CE}}(\boldsymbol{x}^{(i)}, y^{(i)}) \quad \text{subject to} \quad \max_k \{z_k^{(i)}\} - \boldsymbol{z}^{(i)} \preceq m\mathbf{1}_K, \quad i = 1, \ldots, N, \tag{6.2}$$

where $\boldsymbol{z}^{(i)} = f_\theta(\boldsymbol{x}^{(i)})$. The minimum can be approximated by penalty-based optimization methods, transforming the above formulation into an unconstrained problem by means of simple ReLU functions:

$$\underset{\theta}{\text{minimize}} \quad \sum_{i=1}^{N} \mathcal{L}_{\text{CE}}(\boldsymbol{x}^{(i)}, y^{(i)}) + \lambda \sum_{i=1}^{N} \sum_{j=1}^{K} \max\{0, \max_k \{z_k^{(i)}\} - z_j^{(i)} - m\}, \tag{6.3}$$

where $\lambda \in \mathbb{R}_+$ is a scalar weight balancing the contributions of the CE loss and the corresponding penalty.

6.3 Sample-wise Constrained DNN Optimization

Although MbLS can significantly improve calibration, the associated constrained problem (6.2) is not solved accurately. It is approximated by an unconstrained problem with a single uniform penalty, regardless of the data sample or category. However, the samples and classes considered in a classification problem have different intrinsic learning difficulties. Therefore, an improved training scheme would involve considering distinct penalty weights λ for each sample and class. This would result in having to chose $N \times K$ penalty weights $\Lambda \in \mathbb{R}_+^{N \times K}$, with the resulting optimization problem being:

$$\underset{\theta}{\text{minimize}} \quad \sum_{i=1}^{N} \mathcal{L}_{\text{CE}}(\boldsymbol{x}^{(i)}, y^{(i)}) + \sum_{i=1}^{N} \sum_{j=1}^{K} \Lambda_{ij} \max\{0, \max_{k} \{z_k^{(i)}\} - z_j^{(i)} - m\}. \tag{6.4}$$

From an optimization perspective, supposing that optimal weights $\theta^\star$ exists for problem (6.2), there exists $\Lambda^\star \in \mathbb{R}_+^{N \times K}$ such that $(\theta^\star, \Lambda^\star)$ is a saddle point of the Lagrangian associated to (6.2). These $\Lambda^\star$ are the Lagrange multipliers of the problem. Therefore, using $\Lambda = \Lambda^\star$ would be the best choice to solve (6.4).

In practice, using the Lagrange multipliers of problem (6.2) as the weights for the penalties may not be computationally feasible, and it could even result in degraded performance. Indeed, in the context of machine learning, we optimize a model's weights θ to solve (6.2) on a training set $\mathcal{D}_{\text{train}}$, and expect to generalize on a test set $\mathcal{D}_{\text{test}}$, which we do not have access to during training. Because of the bias-variance trade-off, solving (6.2) optimally would likely result in overfitting, *i.e.* we may solve problem (6.2) accurately on the train set, but not generalize properly on the test set, resulting in poor calibration and classification performance overall. This suggests that it could be preferable to evaluate during training the quality of multipliers on a separate validation set $\mathcal{D}_{\text{valid}}$. Additionally, several mechanisms for training DNNs are not compatible with a straightforward minimization. First, the use of batch normalization yields predictions that are not independent between samples in a minibatch. Second, the use of regularization techniques such as dropout may lead to virtually inaccurate predictions on certain training samples, impacting the correct estimation of multipliers. Third, data augmentation, which is

standard in DNN training, would result in additional penalty weights for the augmented samples: they can be easier or harder to classify than the original ones.

In addition to the above obstacles, applying a penalty weight per sample and per class (resulting in $N \times K$ weights) would not scale well for large datasets and dense predictive tasks, such as semantic segmentation, which is typically formulated as a per-pixel classification task. Assuming that images in the dataset have a size of $H \times W$, this would result in $N \times H \times W \times K$ penalty weights. This rapidly becomes a limiting factor for moderately sized segmentation datasets. For instance, Pascal VOC 2012 (Everingham *et al.*, 2012) contains 21 classes and 1464 training images, amounting to $2.62{\times}10^8$ pixels, or $5.5{\times}10^9$ penalty weights, which, stored as `float32`, would use ~20 GiB. For Cityscapes (Cordts *et al.*, 2016), containing approximately 3000 training images of size 2048×1024 in 19 classes, this amounts to ~445 GiB.

Following the above observations, we introduce a relaxation of sample-wise penalties, and propose to solve the following problem:

$$\underset{\theta}{\text{minimize}} \quad \sum_{i=1}^{N} \mathcal{L}_{\text{CE}}(\boldsymbol{x}^{(i)}, y^{(i)}) + \sum_{i=1}^{N} \sum_{j=1}^{K} \lambda_j \max\{0, \max_k\{z_k^{(i)}\} - z_j^{(i)} - m\}, \tag{6.5}$$

where $(\lambda_j)_{1 \leq j \leq K} \in \mathbb{R}_+^K$. Since penalties are now class-wise, we need K penalty weights. This has the advantage to scale well to denser classification tasks such as segmentation. However, we still face a challenging optimization problem (6.5), since we still need to chose K weights, which can be extremely complicated for large-scale datasets with many classes such as ImageNet, which contains 1000 classes. In the next section, we introduce a numerical technique to deal with this challenge.

6.4 Class Adaptive Network Calibration

The challenge of the previous formulation stems from correctly choosing the weights $\boldsymbol{\lambda} \in \mathbb{R}_+^K$, which can be cumbersome when K is large. Therefore, we propose to use an Augmented Lagrangian Multiplier (ALM) method to adaptively learn the weights of the penalties.

6.4.1 General ALM

ALM methods combine penalties and primal-dual updates to solve a constrained problem. They have well-established advantages and enjoy widespread popularity in the general context of optimization (Bertsekas, 2014; Nocedal & Wright, 2006; Sangalli, Erdil, Hötker, Donati & Konukoglu, 2021). Specifically, we have the following generic constrained optimization problem:

$$\underset{x}{\text{minimize}} \quad g(x) \quad \text{subject to} \quad h_i(x) \leq 0, \quad i = 1, \dots, n \tag{6.6}$$

where $g : \mathbb{R}^d \to \mathbb{R}$ is the objective function and $h_i : \mathbb{R}^d \to \mathbb{R}, i = 1, \dots, n$ are the constraint functions. We tackle it by approximately solving a sequence $j \in \mathbb{N}$ of unconstrained problems:

$$\underset{x}{\text{minimize}} \quad \mathcal{L}^{(j)}(x) = g(x) + \sum_{i=1}^{n} P(h_i(x), \rho_i^{(j)}, \lambda_i^{(j)}) \tag{6.7}$$

with $P : \mathbb{R} \times \mathbb{R}^n_{++} \times \mathbb{R}^n_{++} \to \mathbb{R}$ a penalty-Lagrangian function, and $\boldsymbol{\rho}^{(j)} = (\rho_i)_{1 \leq i \leq n} \in \mathbb{R}^n_{++}$, $\boldsymbol{\lambda}^{(j)} = (\lambda_i^{(j)})_{1 \leq i \leq n} \in \mathbb{R}^n_{++}$ the penalty parameters and multipliers associated to P at the j-th iteration. This sequence of unconstrained problems is called *outer* iterations, while the steps in the minimization of $\mathcal{L}^{(j)}$ are called *inner* iterations.

The main components of ALM methods are (*i*) the *penalty-Lagrangian function* P, (*ii*) the update of the *penalty multipliers* $\boldsymbol{\lambda}^{(j)}$ and (*iii*) the increase of the *penalty parameters* $\boldsymbol{\rho}^{(j)}$. First, the penalty function P needs to satisfy a set of axioms (Birgin *et al.*, 2005) (see section 1): these axioms constrain the function to be continuously differentiable w.r.t. its first variable and to have a non-negative derivative: $\forall z \in \mathbb{R}, P'(z, \rho, \lambda) = \frac{\partial}{\partial z} P(z, \rho, \lambda) \geq 0$, with $P'(0, \rho, \lambda) = \lambda$. Figure 6.2 gives an example of a penalty, and how ρ and λ affect it. The choice of penalty function is critical to the performance of ALM methods, especially for nonconvex problems (Birgin *et al.*, 2005). Typical functions include PHR (Hestenes, 1969; Powell, 1969), P_2 (Kort & Bertsekas, 1976) and P_3 (Nakayama, Sayama & Sawaragi, 1975) (see section 3.2 of (Birgin *et al.*, 2005)). Second, the penalty multipliers $\boldsymbol{\lambda}^{(j)}$ are updated to the derivative of P w.r.t. the solution obtained during the last inner minimization. Formally, let $x^{(j)}$ be the approximate minimizer of $\mathcal{L}^{(j)}$,

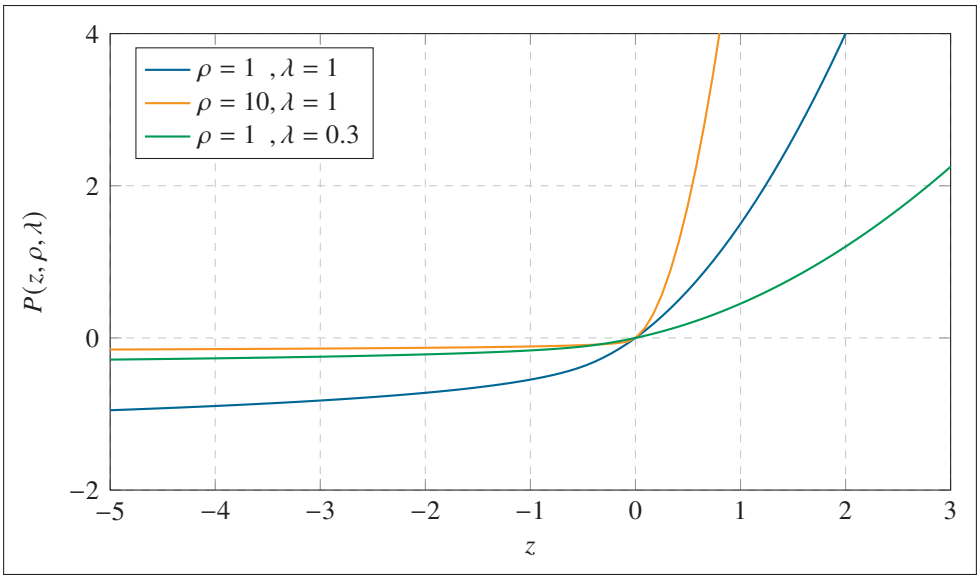


Figure 6.2 A penalty-Lagrangian function P with varying values of ρ and μ. Higher values of ρ bring P closer to an ideal penalty. The multiplier λ is the derivative of P w.r.t. the constraint at $z = 0$

then $\forall i \in \{1, \ldots, n\}$:

$$\lambda_i^{(j+1)} = P'(h_i(x^{(j)}), \rho_i^{(j)}, \lambda_i^{(j)}) \tag{6.8}$$

This update rule corresponds to a first-order multiplier estimate for the constrained problem. Third, the penalty parameters $\boldsymbol{\rho}^{(j)}$ are increased during the outer iterations if the constraints do not improve (*i.e.* is closer to being satisfied) compared to the previous outer iteration. Typically, $\rho_i^{(j+1)} = \gamma \rho_i^{(j)}$ if h_i does not improve, with $\gamma > 1$.

When the problem is convex, alternating between the approximate minimization of (6.7) and the update of the multipliers (6.8) leads to a solution for the constrained problem. The inner minimization corresponds to minimizing the primal while the outer iterations correspond to solving the dual problem. The complete procedure is presented in Algorithm 6.1. Although guarantees exist only in the convex case, it is well-known that ALM methods can efficiently solve nonconvex problems as well (Birgin *et al.*, 2005). In the context of deep learning, their use has been surprisingly under-explored (Rony *et al.*, 2021; Sangalli *et al.*, 2021).

Algorithm 6.1 Augmented Lagrangian Multiplier algorithm

Input: Objective function f
Input: Constraint functions $h_i, i = 1, \dots, n$
Input: Penalty function P, initial $\boldsymbol{\lambda}^{(0)} \in \mathbb{R}^n_{++}$, $\boldsymbol{\rho}^{(0)} \in \mathbb{R}^n_{++}$
Input: Initial variable $x^{(0)}$, iterations $j = 1$
while *not converged* **do**
  Initialize with $x^{(j-1)}$ and minimize (approximately):
  $\mathcal{L}^{(j)}(x) = f(x) + \sum_{i=1}^{n} P(h_i(x), \rho_i^{(j)}, \lambda_i^{(j)})$
  $x^{(j)} \leftarrow$ (approximate) minimizer of $\mathcal{L}^{(j)}$
  for $i = 1, \dots, n$ **do**
    $\lambda_i^{(j+1)} \leftarrow P'(h_i(x^{(j)}), \rho_i^{(j)}, \lambda_i^{(j)})$
    if *the i-th constraint does not improve* **then**
      $\rho_i^{(j+1)} \leftarrow \gamma \rho_i^{(j)}$
    else
      $\rho_i^{(j+1)} \leftarrow \rho_i^{(j)}$
  end for
  $j \leftarrow j + 1$
end while

6.4.2 ALM for calibration

Our goal now is to build an ALM method effective for calibration purposes. We can achieve this by reformulating problem (6.5) using a penalty function P parametrized by $(\boldsymbol{\rho}, \boldsymbol{\lambda}) \in \mathbb{R}^K_{++} \times \mathbb{R}^K_{++}$ as follows:

$$\underset{\theta}{\text{minimize}} \quad \sum_{i=1}^{N} \mathcal{L}_{\text{CE}}(\boldsymbol{x}^{(i)}, y^{(i)}) + \sum_{k=1}^{K} P(d_k^{(i)} - m, \rho_k, \lambda_k) \tag{6.9}$$

with $d_k^{(i)} = \max\{\boldsymbol{z}^{(i)}\} - z_k^{(i)} \in \mathbb{R}_+$. With this formulation, it is natural to use a penalty-Lagrangian function for P. To avoid numerical issues typically associated with non-linear penalties, we normalize the constraints by the margin $m > 0$:

$$d_k^{(i)} - m \leq 0 \Leftrightarrow \frac{d_k^{(i)}}{m} - 1 \leq 0 \tag{6.10}$$

This leads to improved numerical stability for the ALM multiplier update as well. Additionally, we average the constraints instead of summing them. This makes the method independent of the

number of classes, and eases the choice of initial penalty parameters $\boldsymbol{\rho}^{(0)}$. The resulting loss is:

$$\sum_{i=1}^{N} \mathcal{L}_{\text{CE}}(\boldsymbol{x}^{(i)}, y^{(i)}) + \frac{1}{K} \sum_{k=1}^{K} P\left(\frac{d_k^{(i)}}{m} - 1, \rho_k, \lambda_k\right) \tag{6.11}$$

Algorithm 6.2 CALS-ALM training

Input: DNN initial $\theta^{(0)}$, margin m
Input: Dataset: $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{valid}}$, batch size B
Input: Penalty function P, $\gamma > 1$, $\tau \in (0, 1)$
Input: Initial $\boldsymbol{\lambda}^{(0)} \in \mathbb{R}^n_{++}$, $\boldsymbol{\rho}^{(0)} \in \mathbb{R}^n_{++}$
for $j = 0, \ldots, T$ **do**
  for *each mini-batch* $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{B}$ *in* $\mathcal{D}_{train}$ **do**
    $\mathcal{L}_{\text{c}} = \sum_{i=1}^{B} \mathcal{L}_{\text{CE}}(\boldsymbol{x}^{(i)}, y^{(i)})$; // Cross-entropy
    $\mathcal{L}_{\text{p}} = \sum_{i=1}^{B} \frac{1}{K} \sum_{k=1}^{K} P\left(\frac{d_k^{(i)}}{m} - 1, \rho_k^{(j)}, \lambda_k^{(j)}\right)$; // Penalties
    $\mathcal{L} = \frac{1}{B}(\mathcal{L}_{\text{c}} + \mathcal{L}_{\text{p}})$
    $\theta^{(t+1)} \leftarrow \theta^{(t)} - \alpha \cdot \nabla_\theta \mathcal{L}$; // Gradient descent
  end for
  for $k = 1, \ldots, K$ **do**
    $\lambda_k^{(j+1)} = \frac{1}{|\mathcal{D}_{\text{valid}}|} \sum_{(\boldsymbol{x},y) \in \mathcal{D}_{\text{valid}}} P'\left(\frac{d_k}{m} - 1, \rho_k^{(j)}, \lambda_k^{(j)}\right)$
    $\overline{d_k}^{(j)} = \frac{1}{|\mathcal{D}_{\text{valid}}|} \sum_{(\boldsymbol{x},y) \in \mathcal{D}_{\text{valid}}} \frac{d_k}{m} - 1$; // Average constraint
    if $j \geq 1$ ***and*** $\overline{d_k}^{(j)} > \tau \max\{0, \overline{d_k}^{(j-1)}\}$ **then**
      $\rho_k^{(j+1)} \leftarrow \gamma \rho_k^{(j)}$; // Constraint has not improved
    else
      $\rho_k^{(j+1)} \leftarrow \rho_k^{(j)}$
  end for
end for

As noted in section 6.3, one of the main downsides of estimating Lagrange multipliers from the training set is that we could quickly overfit the data. Therefore, we propose to use the validation set to obtain a reliable estimate of the penalty multipliers at each epoch. We consider that an epoch of training corresponds to the approximate minimization of the loss function, and then compute the average penalty multiplier on the validation set. Formally after a training epoch j,

the penalty multipliers for epoch j+1 will be, for all $k = 1, \ldots, K$:

$$\lambda_k^{(j+1)} = \frac{1}{|\mathcal{D}_{\text{valid}}|} \sum_{(\boldsymbol{x},y)\in\mathcal{D}_{\text{valid}}} P'\left(\frac{d_k}{m} - 1, \rho_k^{(j)}, \lambda_k^{(j)}\right) \tag{6.12}$$

Finally, the penalty multiplier is projected on a safeguarding interval $[\lambda_{\min}, \lambda_{\max}] = [10^{-6}, 10^{6}]$ in our case. To update the penalty parameters $\boldsymbol{\rho}$, we compute the average constraint per class on the validation set. Then, for each class, if the average constraint is positive and has not decreased compared to the previous epoch, we multiply the corresponding penalty parameter by γ.

Finally, as suggested by (Birgin *et al.*, 2005) and confirmed by our empirical results, we utilize the PHR function in our implementation, defined as follows:

$$\text{PHR}(z, \rho, \lambda) = \begin{cases} \lambda z + \frac{1}{2}\rho z^2 & \text{if} \quad \lambda + \rho z \geq 0; \\ -\frac{\lambda^2}{2\rho} & \text{otherwise.} \end{cases} \tag{6.13}$$

Overall, the proposed method, consolidated in Algorithm 6.2, corresponds to approximately solving the constrained problem (6.2), by learning class-wise penalty multipliers on the validation set, to avoid overfitting and training specificities (*i.e.* batch normalization, dropout, augmentations) which would result in unreliable penalty multipliers estimates.

6.5 Experiments

6.5.1 Experimental Setup

Datasets. We perform experiments on a variety of popular benchmarks. First, we include three widely used image classification benchmarks, including *Tiny-ImageNet* (Deng *et al.*, 2009), *ImageNet* (Deng *et al.*, 2009) and one long-tailed image classification, *ImageNet-LT* (Liu *et al.*, 2019). Tiny-ImageNet is widely used in the calibration literature (Mukhoti *et al.*, 2020; Liu *et al.*, 2022a), with relatively small 64×64 resolution, while ImageNet (Deng *et al.*, 2009) is a large-scale benchmark consisting of 1000 categories and over 1M images. The

main characteristic of ImageNet-LT is that the number of samples is extremely imbalanced across classes, ranging from 5 to 1280. To evaluate performance in dense prediction tasks, we include one semantic segmentation benchmark, *PASCAL VOC 2012* (Everingham *et al.*, 2012). Furthermore, one benchmark from the NLP domain, *20 Newsgroups* (Lang, 1995), is included to show the general applicability. For a detailed description of each dataset and the pre-processing settings, please refer to section 2.

Evaluation Metrics. For calibration, we report the most widely used Expected Calibration Error (ECE) (Naeini, Cooper & Hauskrecht, 2015). Samples are grouped into M equi-spaced bins according to prediction confidence, and a weighted average of the absolute difference between accuracy and confidence in each bin is calculated:

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{N} |A_m - C_m|, \tag{6.14}$$

where M is the number of bins, N the amount of test samples, B_m the samples with prediction confidence in the m^{th} bin, A_m the accuracy and C_m the mean confidence of samples in the m^{th} bin. Note we fix N to 15 according to (Mukhoti *et al.*, 2020; Liu *et al.*, 2022a). In accordance with (Liu *et al.*, 2022a), we also report Adaptive ECE (AECE), a variant of ECE where the bins are configured to evenly distribute the test samples across them. Additionally, Classwise Calibration Error (CWCE) (Maier-Hein *et al.*, 2022), a classwise extension of ECE, is included in section 3. For discriminative performance, we use standard measures: accuracy (Acc) for classification, and intersection over union (mIoU) for segmentation.

Compared methods. We compare our method to other learning based calibration losses, including *(i)* methods that impose constraints on predictions (either logits or softmax probabilities), *i.e*. Explicit Confidence Penalty (ECP) (Pereyra *et al.*, 2017), Label Smoothing (LS) (Szegedy *et al.*, 2016), Focal Loss (FL) (Lin *et al.*, 2017) and its sample-dependent version (FLSD) (Mukhoti *et al.*, 2020), Margin-based Label Smoothing (MbLS) (Liu *et al.*, 2022a) and CPC (Cheng & Vasconcelos, 2022), and *(ii)* techniques that directly optimize calibration measures, *i.e*. MMCE (Kumar *et al.*, 2018). We refer to the related literature (Mukhoti *et al.*, 2020; Liu

et al., 2022a) to set the hyper-parameters for various methods. For instance, the smoothing factor in LS and FL is set to 0.05 and 3 respectively, and we set margin to 10 in MbLS. A detailed description of hyper-parameter values can be found in section 6.

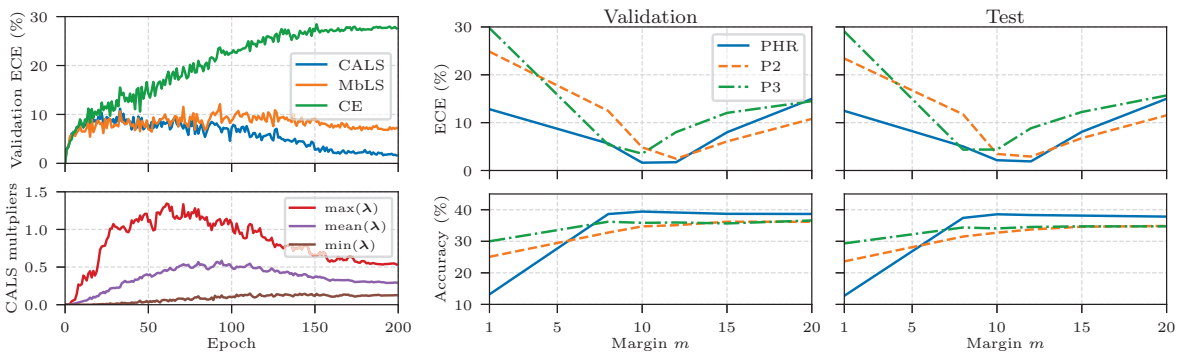
Our methods. A simple alternative to the algorithm presented in subsection 6.4.2 would be to heuristically tune multipliers by scaling them according to penalty values: if $P_k^{(j+1)}$ increases we also increase λ_k^{j+1} and vice versa. This strategy, akin to learning rate scheduling, can be formulated as:

$$\lambda_k^{(j+1)} = \begin{cases} \mu\lambda_k^{(j)} & \text{if} \quad P_k^{(j+1)} > \tau P_k^{(j)} \\ \lambda_k^{(j)}/\mu & \text{if} \quad P_k^{(j)} > \tau P_k^{(j+1)} \\ \lambda_k^{(j)} & \text{otherwise} \end{cases} \tag{6.15}$$

where $\mu > 1$ and $\tau > 1$ are hyper-parameters that we fix to 1.1. We refer to our main algorithm as **CALS-ALM** and to this heuristic rule as **CALS-HR** in what follows.

We fix the margin to $m = 10$ on vision tasks and $m = 6$ on the NLP benchmark, as in (Liu *et al.*, 2022a), for a fair comparison. We also perform an ablation study to investigate the impact of the margin value. For other hyper-parameters, we set $\boldsymbol{\lambda}^{(0)} = 10^{-6} \cdot \mathbf{1}_K$, $\boldsymbol{\rho}^{(0)} = \mathbf{1}_K$, $\gamma = 1.2$, and we update the penalty parameters $\boldsymbol{\rho}$ every 10 epochs. Please refer to section 6 for a detailed description of all hyper-parameters.

Implementation Details. For image classification, we experiment with ResNet (He *et al.*, 2016a) and a vision Transformer model, *i.e*. Swin Transformer V2 (SwinV2-T) (Liu *et al.*, 2022b). DeepLabV3 (Chen *et al.*, 2017) is employed for semantic segmentation on PASCAL VOC 2012. Following (Mukhoti *et al.*, 2020; Liu *et al.*, 2022a), we use the Global Pooling CNN (GPool-CNN) architecture (Lin, Chen & Yan, 2014) on the NLP recognition task. Further training details on each dataset can be found in section 2.



a) Validation ECE and multipliers during training.

b) Effect of penalty function and margin on accuracy and ECE for CALS.

Figure 6.3 **Ablation study on ImageNet-LT.** (a) **Evolution of ECE on validation and multipliers for CALS**: ECE on the validation set for our method (CALS), CE and MbLS (Liu *et al.*, 2022a) and values of multipliers $\boldsymbol{\lambda}$ for CALS after each training epoch. (b) **Effect of penalty functions and margin**: ECE and accuracy on validation and test set are shown across different choices of penalty functions and margin values.

6.5.2 Results

Results on image classification. Table 6.1 presents the discriminative and calibration performance of our methods across three widely used classification benchmarks, compared to baselines and related works. We can observe that our CALS-ALM approach consistently outperforms existing techniques in terms of calibration. Specifically, the results indicate that the standard CE loss and other approaches often lead to miscalibrated models, with the severity of miscalibration substantially increasing in correlation with dataset difficulty. This is particularly evident in large-scaled datasets with numerous classes, such as ImageNet, or those with long-tail class distributions, like ImageNet-LT. Although other learning-based methods could provide better calibrated networks, their performance is not stable across different settings. For example, while FL achieves a relatively low ECE of 1.60 with a Resnet50 trained on ImageNet, it only yields an ECE of 25.50 when training a SwinV2-T network on ImageNet-LT, revealing a limitation in adapting to different learning scenarios. In contrast, CALS-ALM attains the best calibration performance in all cases, frequently outperforming exsiting approaches by a significant margin, with minimal variations across datasets and architectures. This trends persists when compared

to the most closely related technique, MbLS. It is noteworthy that the improvements on the long-tailed ImageNet-LT dataset are substantial; for example, we decrese ECE from 28.12 to 2.15 for ResNet-50, and from 31.82 to 2.32 for SwinV2-T, validating the effectiveness of class adaptive learning. Another interesting finding is that CALS-HR, employing a naive update strategy for class-wise penalty weights, achieves nearly the second-best performance in the ImageNet-LT dataset. This further demonstrates the effectiveness of CALS for learning under class-imbalance scenarios. For the reliability diagrams of various models, please refer to section 5.

In terms of model accuracy, our method delivers competitive performances, surpassing existing methods in certain cases. It is important to emphasize that, while the proposed method achieves discriminative results comparable to the best-performing approach for each dataset, the differences in calibration are considerable. This highlights the superiority of the proposed formulation for training highly discriminative and well-calibrated networks. Figure 6.1 provides a more visual comparison considering both accuracy and ECE. It is demonstrated that CALS-ALM provides the optimal compromise between accuracy and calibration performance.

Ablation Analysis. Figure 6.3 illustrates the evolution of ECE in (a) and penalty multipliers $\boldsymbol{\lambda}$ in (b) during training. It is interesting to observe that the evolution of $\boldsymbol{\lambda}$ is consistent with the ECE. Specifically, the average penalty weight gradually increases while the ECE initially deteriorates because the model is focused on increasing accuracy. However, the value of the penalty weight begins to decline alongside the ECE, as the network starts to become better calibrated. For a visualization of classwise multipliers, please refer to section 4. Figure 6.3(c) highlights the impact of the choice of penalty functions and margin values. This demonstrates that the PHR penalty function is preferable over the other two options, P2 and P3, for both calibration and accuracy. Regarding the margin, the best performance is achieved with $m \approx 10$, which is consistent with the findings in (Liu *et al.*, 2022a).

Semantic Segmentation. Table 6.2 presents the performance on Pascal VOC dataset. Note here CALS refers to our best method, *i.e*. CALS-ALM. It can be observed that the trend is consistent

Table 6.1 Calibration performance for different approaches on three image classification benchmarks. We report two lower-is-better calibration metrics, *i.e*. ECE and AECE. Best method is highlighted in bold, while the second-best one is underlined

	TinyImageNet			ImageNet					
	ResNet-50			ResNet-50			SwinV2-T		
Method	Acc	ECE	AECE	Acc	ECE	AECE	Acc	ECE	AECE
CE	65.02	3.73	3.69	75.16	9.19	9.18	75.60	9.95	9.94
MMCE	<u>65.34</u>	2.81	2.61	74.85	8.57	8.56	76.68	9.07	9.08
ECP	64.90	4.00	3.92	75.22	8.27	8.26	75.82	9.88	9.86
LS	**65.78**	3.17	3.16	76.04	2.57	2.88	75.42	7.32	7.33
FL	63.09	2.96	3.12	73.87	<u>1.60</u>	<u>1.65</u>	75.60	3.19	3.18
FLSD	64.09	2.91	2.95	73.97	2.08	2.06	74.70	2.44	2.37
CPC	64.49	4.88	4.91	76.33	3.66	3.59	76.34	5.50	5.33
MbLS	64.74	<u>1.64</u>	<u>1.73</u>	75.82	4.44	4.26	<u>77.18</u>	<u>1.95</u>	<u>1.73</u>
CALS-HR	65.09	2.50	2.42	<u>76.34</u>	5.63	5.69	**77.58**	3.06	2.95
CALS-ALM	65.03	**1.54**	**1.38**	**76.44**	**1.46**	**1.32**	77.10	**1.61**	**1.69**

	ImageNet-LT					
	ResNet-50			SwinV2-T		
Method	Acc	ECE	AECE	Acc	ECE	AECE
CE	37.90	28.12	28.12	31.82	31.82	36.68
MMCE	37.79	28.41	28.40	33.14	26.41	26.41
ECP	37.69	28.14	28.13	31.22	33.70	33.70
LS	37.88	10.46	10.38	31.70	11.42	11.40
FL	36.04	18.37	18.36	30.73	25.50	25.50
FLSD	36.18	17.77	17.78	32.56	25.16	25.17
CPC	**38.90**	16.00	15.99	32.54	13.21	13.19
MbLS	38.32	6.16	6.16	32.05	7.65	7.64
CALS-HR	38.50	<u>2.83</u>	<u>2.78</u>	**34.31**	<u>2.37</u>	**2.45**
CALS-ALM	<u>38.56</u>	**2.15**	**2.30**	<u>33.94</u>	**2.32**	**2.45**

with image classification experiments: CALS outperforms counterparts in terms of ECE, and yields competitive results on discriminative performance, *i.e*. mIoU in segmentation. It is worth noting that some methods like MMCE, FLSD and CPC, are not included here because their computation demands were too heavy for pixel-wise segmentation tasks. In contrast, our method is unlimited in dense prediction tasks as the computation cost it adds is moderate.

Table 6.2 Segmentation results on PASCAL VOC 2012

Loss	CE	ECP	LS	FL	MbLS	CALS
ECE	14.75	5.15	6.90	10.87	5.22	**4.66**
mIoU	66.46	65.57	**67.73**	64.25	65.29	66.77

Text Classification. Last, we demonstrate the general applicability of the proposed method by analyzing its performance on a non-vision task, *i.e*. text classification on 20 Newsgroups dataset. The results are reported in Table 6.3. Remarkably, CALS again brings substantial improvement in terms of calibration, with ECE decreasing to 2.04%, while yielding the best accuracy 68.32%. This reveals that the proposed class adaptive learning method is also able to handle class differences in NLP applications and provide promising performance in terms of both accuracy and calibration.

Table 6.3 Results on the text classification task, 20 Newsgroups

Loss	CE	MMCE	ECP	LS	FL	FLSD	CPC	MbLS	CALS
ECE	22.75	23.02	22.97	8.07	10.80	10.87	9.46	5.40	**2.04**
ACC	67.01	66.23	66.48	67.14	66.08	65.85	68.27	67.89	**68.32**

6.6 Limitations and Future Work

We have proposed Class Adaptive Label Smoothing for network calibration based on a modified Augmented Lagrangian Multiplier algorithm. Despite its superior performance over previous methods, there are potential limitations in this work. For instance, our method requires the validation set to have the same distribution as the training set. Although this is satisfied in nearly every benchmark, it will be interesting to investigate the impact of using non-independent and identically distributed (*i.i.d*) validation sets.

CONCLUSION AND RECOMMENDATIONS

In this thesis, we revisited several optimization problems related to the training and verification of deep learning models.

First, we established a link between the cross-entropy and several pairwise losses that are used in the deep metric learning literature. With this link, we prove that cross-entropy is relevant for feature learning, and experimentally show that it often outperforms pairwise methods thanks to its simplicity and training stability.

Second, we devised several adversarial attack algorithms. The first one, DDN, which is an efficient and robust solution to generate adversarial examples with small ℓ_2-norms. However, like many attack in the literature, it lacks the generality of penalty-based approaches to minimize other distances. Therefore, we propose to leverage a well-known augmented Lagrangian multiplier method to tackle the general constrained optimization problem of generating adversarial attacks. The resulting ALMA attack can be used for a wide family of smooth discrepancy measures such as ℓ_2, ℓ_1, SSIM, LPIPS, CIEDE2000, *etc*. This attack framework allows tackling a more general – but also more difficult – problem: generating adversarial attacks for semantic segmentation. For this task, we need to satisfy potentially millions of constraints. To be able to minimize the non-smooth ℓ_∞-norm, we combine it with a proximal splitting. We prove that the proximity operator of the sum of the ℓ_∞-norm and the hypercube indicator function can be efficiently computed. This allows us to better evaluate the robustness of deep semantic segmentation models, and show that it was largely overestimated, when done with attacks adapted from segmentation.

Finally, with the new insights provided by Liu *et al.*, we notice that improving the calibration of deep neural networks is closely related to the satisfaction of inequality constraints on the logit distance. However, manually tuning naive penalties to satisfy this constraint is infeasible in practice. Therefore, we resort to an augmented Lagrangian multiplier method to automatically

learn class-wise penalty weights. With this strategy, we are able to achieve state-of-the-art classification and calibration performances on several tasks, including semantic segmentation where we can take advantage of the scalability of our method.

Recommendations and future work

- **Simplicity and equivalence of methods**: as shown in chapter 2, many methods proposed to solve a particular task tend to make it more complex. This creates a distorted view of the effectiveness of such methods, where a finer analysis reveals that the performance improvements can be mainly attributed to training strategies or tricks. While several methods may look different on the surface, they can, in fact, be equivalent and provide little to no additional benefit while introducing more confounding variables. Therefore, we recommend to carefully evaluate robust baseline methods, which often perform as well, or better than more complex alternatives.
- **Benchmark of adversarial attacks**: an increasing number of adversarial attacks have been published over the years. Some tackle new problems, while many try to improve the efficiency and quality of the solutions for the adversarial perturbation generation problems. This has created a confusion around which attacks are best suited to accurately evaluate the robustness of deep learning models, especially the one trained with defense methods. This often leads to over-estimation of the actual robustness of models, and lures researchers to a false sense of security. Therefore, our future work includes establishing recommendations for which combination of attacks to use to best assess the robustness of deep models, based on a comprehensive benchmark of the proposed methods.
- **Learning with constraints**: many tasks in machine learning can be formulated as constrained optimization problems. In chapter 6, we have shown that using well-known optimization methods to improve constraints satisfactions results in better calibration. This can be applied to several other problems, which have been solely approached with naive regularization, or linear and quadratic penalties. Such problems include, for instance, incorporating domain

knowledge in medical imaging by constraining the size of organs, which allows training a model with limited amounts of data (Kervadec *et al.*, 2022).

APPENDIX I

SUPPLEMENTARY MATERIAL FOR THE PAPER TITLED *A UNIFYING MUTUAL INFORMATION VIEW OF METRIC LEARNING: CROSS-ENTROPY* VS. *PAIRWISE LOSSES*

1. Proof of Lemma 2.1

Proof. Throughout the following proofs, we will use the fact that classes are assumed to be balanced in order to consider $\mathcal{Z}_k$, for any class k, as a constant $|\mathcal{Z}_k| = \frac{n}{K}$. We will also use the feature normalization assumption to connect cosine and Euclidean distances. On the unit-hypersphere, we will use that: $D_{i,j}^{\cos} = 1 - \frac{\|z_i - z_j\|^2}{2}$.

Tightness terms: Let us start by linking center loss to contrastive loss. For any specific class k, let $\boldsymbol{c}_k = \frac{1}{|\mathcal{Z}_k|} \sum\limits_{z \in \mathcal{Z}_k} z$ denotes the hard mean. We can write:

$$
\begin{aligned}
\sum_{z_i \in \mathcal{Z}_k} \|z_i - \boldsymbol{c}_k\|^2 &= \sum_{z_i \in \mathcal{Z}_k} [\|z_i\|^2 - 2 z_i^\top \boldsymbol{c}_k] + |\mathcal{Z}_k| \, \|\boldsymbol{c}_k\|^2 \\
&= \sum_{z_i \in \mathcal{Z}_k} \|z_i\|^2 - 2 \frac{1}{|\mathcal{Z}_k|} \sum_{z_i \in \mathcal{Z}_k} \sum_{z_j \in \mathcal{Z}_k} z_i^\top z_j + \frac{1}{|\mathcal{Z}_k|} \sum_{z_i \in \mathcal{Z}_k} \sum_{z_j \in \mathcal{Z}_k} z_i^\top z_j \\
&= \sum_{z_i \in \mathcal{Z}_k} \|z_i\|^2 - \frac{1}{|\mathcal{Z}_k|} \sum_{z_i \in \mathcal{Z}_k} \sum_{z_j \in \mathcal{Z}_k} z_i^\top z_j \\
&= \frac{1}{2} [\sum_{z_i \in \mathcal{Z}_k} \|z_i\|^2 + \sum_{z_j \in \mathcal{Z}_k} \left\|z_j\right\|^2] - \frac{1}{|\mathcal{Z}_k|} \sum_{z_i \in \mathcal{Z}_k} \sum_{z_j \in \mathcal{Z}_k} z_i^\top z_j \\
&= \frac{1}{2|\mathcal{Z}_k|} [\sum_{z_i \in \mathcal{Z}_k} \sum_{z_j \in \mathcal{Z}_k} \|z_i\|^2 + \sum_{z_i \in \mathcal{Z}_k} \sum_{z_j \in \mathcal{Z}_k} \left\|z_j\right\|^2] - \frac{1}{2|\mathcal{Z}_k|} \sum_{z_i \in \mathcal{Z}_k} \sum_{z_j \in \mathcal{Z}_k} 2 z_i^\top z_j \\
&= \frac{1}{2|\mathcal{Z}_k|} \sum_{z_i, z_j \in \mathcal{Z}_k} \|z_i\|^2 - 2 z_i^\top z_j + \left\|z_j\right\|^2 \\
&= \frac{1}{2|\mathcal{Z}_k|} \sum_{z_i, z_j \in \mathcal{Z}_k} \left\|z_i - z_j\right\|^2 \\
&\overset{\text{c}}{=} \sum_{z_i, z_j \in \mathcal{Z}_k} \left\|z_i - z_j\right\|^2
\end{aligned}
\tag{A I-1}
$$

Summing over all classes k, we get the desired equivalence. Note that, in the context of K-means clustering, where the setting is different[15], a technically similar result could be established (Tang *et al.*, 2019), linking K-means to pairwise graph clusteirng objectives.

Now we link contrastive loss to SNCA loss. For any class k, we can write:

$$\begin{aligned}
-\sum_{z_i\in\mathcal{Z}_k}\log\sum_{z_j\in\mathcal{Z}_k\setminus\{i\}}e^{\frac{D^{\cos}_{i,j}}{\sigma}} &\overset{\text{c}}{=} -\sum_{z_i\in\mathcal{Z}_k}\log\left(\frac{1}{|\mathcal{Z}_k|-1}\sum_{z_j\in\mathcal{Z}_k\setminus\{i\}}e^{\frac{D^{\cos}_{i,j}}{\sigma}}\right)\\
&\leq -\sum_{z_i\in\mathcal{Z}_k}\sum_{z_j\in\mathcal{Z}_k\setminus\{i\}}\frac{D^{\cos}_{i,j}}{(|\mathcal{Z}_k|-1)\sigma}\\
&\overset{\text{c}}{=} \sum_{z_i\in\mathcal{Z}_k}\sum_{z_j\in\mathcal{Z}_k\setminus\{i\}}\frac{\left\|z_i-z_j\right\|^2}{2\sigma(|\mathcal{Z}_k|-1)}\\
&\overset{\text{c}}{=} \sum_{z_i\in\mathcal{Z}_k}\sum_{z_j\in\mathcal{Z}_k\setminus\{i\}}\left\|z_i-z_j\right\|^2
\end{aligned} \tag{A I-2}$$

where we used the convexity of $x\rightarrow-\log(x)$ and Jensen's inequality. The proof can be finished by summing over all classes k.

Finally, we link MS loss (Wang *et al.*, 2019b) to contrastive loss:

$$\begin{aligned}
\sum_{z_i\in\mathcal{Z}_k}\frac{1}{\alpha}\log\left(1+\sum_{z_j\in\mathcal{Z}_k\setminus\{i\}}e^{-\alpha(D^{\cos}_{i,j}-1)}\right) &= \sum_{z_i\in\mathcal{Z}_k}\frac{1}{\alpha}\log\sum_{z_j\in\mathcal{Z}_k}e^{-\alpha(D^{\cos}_{i,j}-1)}\\
&\overset{\text{c}}{=} \sum_{z_i\in\mathcal{Z}_k}\frac{1}{\alpha}\log\left(\frac{1}{|\mathcal{Z}_k|}\sum_{z_j\in\mathcal{Z}_k}e^{-\alpha(D^{\cos}_{i,j}-1)}\right)\\
&\geq \frac{1}{|\mathcal{Z}_k|}\sum_{z_i,z_j\in\mathcal{Z}_k}-(D^{\cos}_{i,j}-1)\\
&\overset{\text{c}}{=} \sum_{z_i,z_j\in\mathcal{Z}_k}\left\|z_i-z_j\right\|^2,
\end{aligned} \tag{A I-3}$$

where we used the concavity of $x\rightarrow\log(x)$ and Jensen's inequality.

[15] In clustering, the optimization is performed over assignment variables, as opposed to DML, where assignments are already known and optimization is carried out over the embedding.

Contrastive terms: In this part, we first show that the contrastive terms C_{SNCA} and C_{MS} represent upper bounds on $C = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j:y_j\neq y_i} D_{ij}^2$:

$$\begin{aligned} C_{MS} = \frac{1}{\beta n}\sum_{i=1}^{n}\log\left(1+\sum_{j:y_j\neq y_i} e^{\beta(D_{ij}^{\cos}-1)}\right) &\geq \frac{1}{\beta n}\sum_{i=1}^{n}\log\left(\sum_{j:y_j\neq y_i} e^{\beta(D_{ij}^{\cos}-1)}\right) \\ &\overset{c}{\geq} \frac{1}{\beta n}\sum_{i=1}^{n}\sum_{j:y_j\neq y_i}\beta(D_{ij}^{\cos}-1) \\ &\overset{c}{=} -\frac{1}{n}\sum_{i=1}^{n}\sum_{j:y_j\neq y_i} D_{ij}^2 \\ &= C \end{aligned} \tag{A I-4}$$

where, again, we used Jensen's inequality in the second line above. The link between SNCA and contrastive loss can be established quite similarly:

$$\begin{aligned} C_{SNCA} = \frac{1}{n}\sum_{i=1}^{n}\log\left(\sum_{j\neq i} e^{\frac{D_{ij}^{\cos}}{\sigma}}\right) &= \frac{1}{n}\sum_{i=1}^{n}\log\left(\sum_{j\neq i:y_i=y_j} e^{\frac{D_{ij}^{\cos}}{\sigma}} + \sum_{j:y_j\neq y_i} e^{\frac{D_{ij}^{\cos}}{\sigma}}\right) \\ &\geq \frac{1}{n}\sum_{i=1}^{n}\log\left(\sum_{j:y_j\neq y_i} e^{\frac{D_{ij}^{\cos}}{\sigma}}\right) \\ &\overset{c}{\geq} \frac{1}{n}\sum_{i=1}^{n}\sum_{j:y_j\neq y_i}\frac{D_{ij}^{\cos}}{\sigma} \\ &\overset{c}{=} -\frac{1}{n}\sum_{i=1}^{n}\sum_{j:y_j\neq y_i} D_{ij}^2 \\ &= C \end{aligned} \tag{A I-5}$$

Now, similarly to the reasoning carried out in subsection 2.3.1, we can write:

$$C = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j:y_j\neq y_i} D_{ij}^2 = \underbrace{-\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} D_{ij}^2}_{\text{contrast}\,\propto\,\mathcal{H}(\hat{Z})} + \underbrace{\frac{1}{n}\sum_{i=1}^{n}\sum_{j:y_j=y_i} D_{ij}^2}_{\text{tightness subterm}\,\propto\,\mathcal{H}(\hat{Z}|Y)} \tag{A I-6}$$

Where the redundant tightness term is very similar to the tightness term in contrastive loss $T_{contrast}$ treated in details in subsection 2.3.1. As for the truly contrastive part of C, it can also be related to the differential entropy estimator used in (Wang & Sha, 2011):

$$\widehat{\mathcal{H}}(\widehat{Z}) = \frac{d}{n(n-1)} \sum_{i=1}^{n} \sum_{j=1}^{n} \log D_{ij}^2 \overset{c}{=} \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} \log D_{ij}^2 \tag{A I-7}$$

In summary, we just proved that the contrastive parts of MS and SNCA losses are upper bounds on the contrastive term C. The latter term is composed of a proxy for the entropy of features $\mathcal{H}(\hat{Z})$, as well as a tightness sub-term.

□

2. Proof of Proposition 2.1

Proof. First, let us show that $\mathcal{L}_{CE} \geq \mathcal{L}_{PCE}$. Consider the usual softmax parametrization of point i belonging to class k: $p_{ik} = (f_\theta(z_i))_k = \frac{\exp \theta_k^\top z_i}{\sum_j \exp \theta_j^\top z_i}$, where $z = \phi_{\mathcal{W}}(x)$. We can explicitly write the cross-entropy loss:

$$\begin{aligned} \mathcal{L}_{CE} &= -\frac{1}{n} \sum_{i=1}^{n} \log f_\theta(z_i) \\ &= \underbrace{-\frac{1}{n} \sum_{i=1}^{n} \theta_{y_i}^\top z_i + \frac{\lambda}{2} \sum_{k=1}^{K} \theta_k^\top \theta_k}_{h_1(\theta)} + \underbrace{\frac{1}{n} \sum_{i=1}^{n} \log \sum_{j=1}^{K} e^{\theta_j^\top z_i} - \frac{\lambda}{2} \sum_{k=1}^{K} \theta_k^\top \theta_k}_{h_2(\theta)} . \end{aligned} \tag{A I-8}$$

Where we introduced $\lambda \in \mathbb{R}$. How to specifically set λ will soon become clear. Let us now write the gradients of h_1 and h_2 in Equation A I-8 with respect to θ_k:

$$\frac{\partial h_1}{\partial \theta_k} = -\frac{1}{n} \sum_{i : y_i = k} z_i + \lambda \theta_k \tag{A I-9}$$

$$\frac{\partial h_2}{\partial \theta_k} = \frac{1}{n} \sum_{i} \underbrace{\frac{\exp(\theta_k^\top z_i)}{\sum_{j=1}^{K} \exp(\theta_j^\top z_i)}}_{p_{ik}} z_i - \lambda \theta_k \tag{A I-10}$$

Notice that h_1 is a convex function of θ, regardless of λ. As for h_2, we set λ such that h_2 becomes a convex function of θ. Specifically, by setting:

$$\lambda = \min_{k,l} \sigma_l(A_k) \tag{A I-11}$$

where $A_k = \frac{1}{n}\sum_{i=1}^{n}(p_{ik} - p_{ik}^2)\boldsymbol{z}_i\boldsymbol{z}_i^\top$ and $\sigma_l(A)$ represents the l^{th} eigenvalue of A, we make sure that the hessian of h_2 is semi-definite positive. Therefore, we can look for the minima of h_1 and h_2.

Setting gradients in Equation A I-9 and Equation A I-10 to 0, we obtain that for all $k \in [1, K]$, the optimal θ_k for h_1 is, up to a multiplicative constant, the hard mean of features from class k: $\theta_k^{h_1*} = \frac{1}{\lambda n}\sum\limits_{i:y_i=k} \boldsymbol{z}_i \propto \boldsymbol{c}_k$, while the optimal θ_k for h_2 is, up to a multiplicative constant, the soft mean of features: $\theta_k^{h_2*} = \frac{1}{\lambda n}\sum_{i=1}^{n}\ p_{ik}\boldsymbol{z}_i = \boldsymbol{c}_k^s/\lambda$. Therefore, we can write:

$$h_1(\theta) \geq h_1(\theta^{h_1*}) = -\frac{1}{\lambda n^2}\sum_{i=1}^{n}\sum_{j:y_j=y_i}\boldsymbol{z}_i^\top\boldsymbol{z}_j + \frac{\lambda}{2\lambda^2}\sum_{i=1}^{n}\sum_{j:y_j=y_i}\boldsymbol{z}_i^\top\boldsymbol{z}_j \tag{A I-12}$$

$$= -\frac{1}{2\lambda n^2}\sum_{i=1}^{n}\sum_{j:y_j=y_i}\boldsymbol{z}_i^\top\boldsymbol{z}_j \tag{A I-13}$$

And

$$\begin{aligned} h_2(\theta) &\geq h_2(\theta^{h_2*}) \\ &= \frac{1}{n}\sum_{i=1}^{n}\log\sum_{k=1}^{K}\exp\left(\frac{1}{\lambda n}\sum_{j=1}^{n}p_{jk}\boldsymbol{z}_i^\top\boldsymbol{z}_j\right) - \frac{1}{2\lambda}\sum_{k=1}^{K}\left\|\boldsymbol{c}_k^s\right\|^2 \end{aligned} \tag{A I-14}$$

Putting it all together, we can obtain the desired result:

$$\begin{aligned} \mathcal{L}_{CE} &\geq -\frac{1}{2\lambda n^2}\sum_{i=1}^{n}\sum_{j:y_j=y_i}\boldsymbol{z}_i^\top\boldsymbol{z}_j + \frac{1}{n}\sum_{i=1}^{n}\log\sum_{k=1}^{K}e^{\frac{1}{\lambda n}\sum_j p_{jk}\boldsymbol{z}_i^\top\boldsymbol{z}_j} - \frac{1}{2\lambda}\sum_{k=1}^{K}\left\|\boldsymbol{c}_k^s\right\|^2 \\ &= \mathcal{L}_{PCE} \end{aligned} \tag{A I-15}$$

where $\boldsymbol{c}_k^s = \frac{1}{n}\sum_{i=1}^{n} p_{ik}\boldsymbol{z}_i$ represents the soft mean of class k.

Let us now justify that minimizing cross-entropy can be seen as an approximate bound optimization on $\mathcal{L}_{PCE}$. At every iteration t of the training, cross-entropy represents an upper bound on Pairwise Cross-entropy.

$$\mathcal{L}_{CE}(\mathcal{W}(t), \theta(t)) \geq \mathcal{L}_{PCE}(\mathcal{W}(t), \theta(t)) \tag{A I-16}$$

When optimizing w.r.t θ, the bound almost becomes tight. The approximation comes from the fact that $\theta_k^{h_1*}$ and $\theta_k^{h_2*}$ are quite dissimilar in early training, but become very similar as training progresses and the model's softmax probabilities align with the labels. Therefore, using the notation:

$$\theta(t+1) = \min_{\theta} \mathcal{L}_{CE}(\mathcal{W}(t), \theta) \tag{A I-17}$$

We can write:

$$\mathcal{L}_{CE}(\mathcal{W}(t), \theta(t+1)) \approx \mathcal{L}_{PCE}(\mathcal{W}(t), \theta(t+1)) \tag{A I-18}$$

Then, minimizing $\mathcal{L}_{CE}$ and $\mathcal{L}_{PCE}$ w.r.t $\mathcal{W}$ becomes approximately equivalent.

□

2.1 Proof of Lemma 2.2

Proof. Using the discriminative view of MI, we can write:

$$\mathcal{I}(\widehat{Z}; Y) = \mathcal{H}(Y) - \mathcal{H}(Y|\widehat{Z}) \tag{A I-19}$$

The entropy of labels $\mathcal{H}(Y)$ is a constant and, therefore, can be ignored. From this view of MI, maximization of $\mathcal{I}(\widehat{Z}; Y)$ can only be achieved through a minimization of $\mathcal{H}(Y|\widehat{Z})$, which depends on our embeddings $\widehat{Z} = \phi_{\mathcal{W}}(X)$. We can relate this term to our cross-entropy loss using the following relation:

$$\mathcal{H}(Y; \widehat{Y}|\widehat{Z}) = \mathcal{H}(Y|\widehat{Z}) + \mathcal{D}_{KL}(Y||\widehat{Y}|\widehat{Z}) \tag{A I-20}$$

Therefore, while minimizing cross-entropy, we are implicitly both minimizing $\mathcal{H}(Y|\widehat{Z})$ as well as $\mathcal{D}_{KL}(Y||\widehat{Y}|\widehat{Z})$. In fact, following Equation A I-20, optimization could naturally be decoupled in 2 steps, in a *Maximize-Minimize* fashion. One step would consist in fixing the encoder's weights $\mathcal{W}$ and only minimizing Equation A I-20 w.r.t to the classifier's weights θ. At this step, $\mathcal{H}(Y|\widehat{Z})$ would be fixed while $\widehat{Y}$ would be adjusted to minimize $\mathcal{D}_{KL}(Y||\widehat{Y}|\widehat{Z})$. Ideally, the KL term would vanish at the end of this step. In the following step, we would minimize Equation A I-20 w.r.t to the encoder's weights $\mathcal{W}$, while keeping the classifier fixed.

□

3. Preliminary results with SPCE

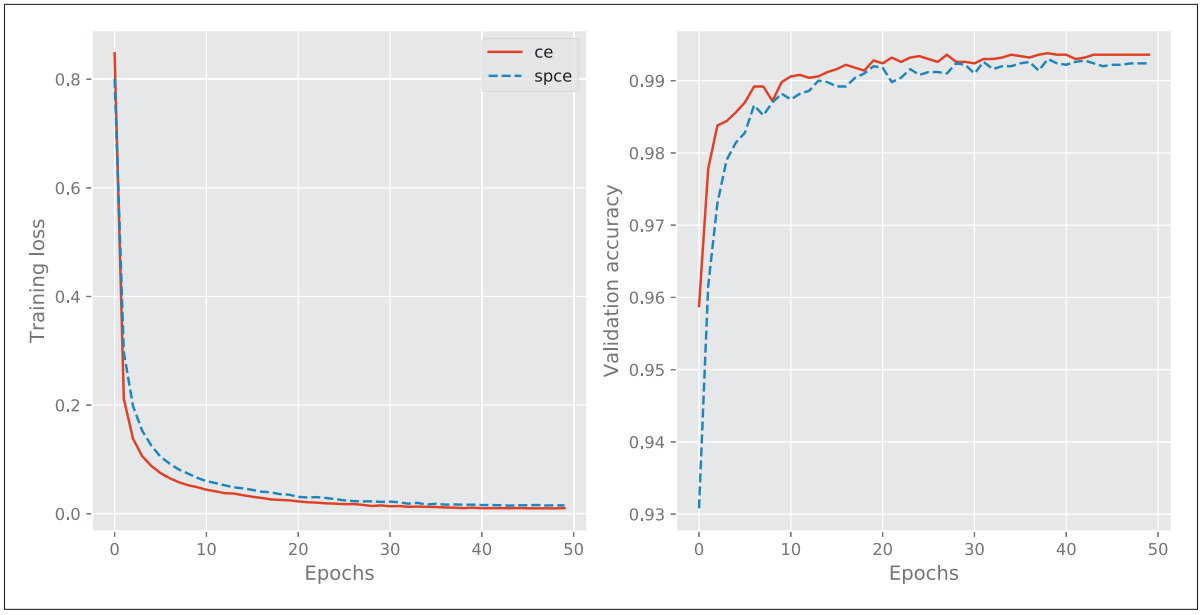


Figure-A I-1 Evolution of the cross-entropy loss (CE) and the simplified pairwise cross-entropy (SPCE) during training on MNIST, as well as the validation accuracy for both losses

In Figure I-1, we track the evolution of both loss functions and validation accuracy when training with $\mathcal{L}_{CE}$ and $\mathcal{L}_{SPCE}$ on MNIST dataset. We use a small CNN composed of four convolutional layers. The optimizer used is Adam. Batch size is set to 128, learning rate to $1e^{-4}$ with cosine annealing, weight decay to $1e^{-4}$ and feature dimension to $d = 100$. Figure I-1 supports the

theoretical links that were drawn between Cross-Entropy and its simplied pairwise version SPCE. Particularly, this preliminary result demonstrates that SPCE is indeed employable as a loss, and exhibits a very similar behavior to the original cross-entropy. Both losses remain very close to each other throughout the training, and so remain the validation accuracies.

4. Analysis of ranking losses for Deep Metric Learning

Some recent works (Cakir, He, Xia, Kulis & Sclaroff, 2019; Wang *et al.*, 2019a; Rolínek *et al.*, 2020) tackle the problem of deep metric learning using a rank-based approach. In other words, given a point in feature space z_i, the pairwise losses studied throughout this work try to impose manual margins m, so that the distance between z_i and any negative point z_j^- is at least m. Rank-based losses rather encourage that all points are well ranked, distance-wise, such that $d(z_i, z_j^+) \leq d(z_i, z_j^-)$ for any positive and negative points z_j^+ and z_j^-. We show that our tightness/contrastive analysis also holds for such ranking losses. In particular, we analyse the loss proposed in (Cakir *et al.*, 2019). For any given query embedded point z_i, let us call D the random variable associated to the distance between z_i and all other points in the embedded space, defined over all possible (discretized) distances $\mathcal{D}$. Furthermore, let us call R the binary random variable that describes the relation to the current query point (R^+ and R^- describe respectively a positive and negative relationship to z_i). The loss maximized in (Cakir *et al.*, 2019) reads:

$$\text{FastAP} = \sum_{d \in \mathcal{D}} \frac{P(D < d|R^+)P(R^+)}{P(D < d)} P(D = d|R^+) \tag{A I-21}$$

Taking the logarithm, and using Jensen's inequality, we can lower bound this loss:

$$\begin{aligned} \log(\text{FastAP}) &\geq \sum_{d \in \mathcal{D}} P(D = d, R^+) \log\left(\frac{P(D < d|R^+)}{P(D < d)}\right) \\ &= \underbrace{\mathop{\mathbb{E}}_{d \sim P(.,R^+)} \log P(D < d|R^+)}_{T_{AP}=\text{TIGHTNESS}} - \underbrace{\mathop{\mathbb{E}}_{d \sim P(.,R^+)} \log P(D < d)}_{C_{AP}=\text{CONTRASTIVE}} \end{aligned} \tag{A I-22}$$

To intuitively understand what those two terms are doing, let us imagine we approximate each of the expectations with a single point Monte-Carlo approximation. In other words, we sample a

positive point z_j^+, take its associated distance to z_i, which we call d^+, then we approximate the tightness term as:

$$T_{AP} \approx \log P(D < d^+ | R^+) \tag{A I-23}$$

Maximizing T_{AP} has a clear interpretation: it encourages all positive points to lie inside the hypersphere of radius d^+ around query point z_i. Similarly:

$$C_{AP} \approx -\log P(D < d^+) \tag{A I-24}$$

Maximizing C_{AP} also has a clear interpretation: it encourages all points (both positive and negative ones) to lie outside the hypersphere of radius d^+ around query point z_i. Now, Equation A I-22 is nothing more than an expectation over all positive distance d^+ one could sample. Therefore, such loss can be analyzed through the same lens as other DML losses, i.e., one tightness term that encourages all points from the same class as z_i to lie close to it in the embedded space, and one contrastive term that oppositely refrains all points from approaching z_i closer than its current positive points.

5. On the limitations of cross-entropy

While we demonstrated that the cross-entropy loss could be competitive in comparison to pairwise losses, while being easier to optimize, there still exist scenarios for which a straightforward use of the CE loss becomes prohibitive. Hereafter, we describe two such scenarios.

Case of relative labels: The current setting assumes that absolute labels are given for each sample, *i.e.*, each sample $\boldsymbol{x}_i$ belongs to a single absolute class y_i. However, DML can be applied to more general problems where the absolute class labels are not available. Instead, one has access to relative labels that only describe the relationships between points (*e.g.*, a pair is similar or dissimilar). From these relative labels, one could still define absolute classes as sets of samples inside which every pair has a positive relationship. Note that with this definition, each sample may belong to multiple classes simultaneously, which makes the use of standard cross-entropy difficult. However, with such re-formulation, our Simplified Pairwise Cross-Entropy (SPCE),

which we hereby remind:

$$\mathcal{L}_{\text{SPCE}} = \underbrace{-\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j:y_j=y_i} z_i^\top z_j}_{\text{TIGHTNESS}} + \underbrace{\frac{1}{n}\sum_{i=1}^{n}\log\sum_{k=1}^{K}\exp\left(\frac{1}{n}\sum_{j:y_j=k} z_i^\top z_j\right)}_{\text{CONTRASTIVE}} \tag{2.15}$$

can handle such problems, just like any other pairwise loss.

Case of large number of classes: In some problems, the total number of classes K can grow to several millions. In such cases, even simply storing the weight matrix $\theta \in \mathbb{R}^{K\times d}$ of the final classifier required by cross-entropy becomes prohibitive. Note that there exist heuristics to handle such problems with standard cross-entropy, such as sampling subsets of classes and solving those sub-problems instead, as was done in (Zhai & Wu, 2019). However, we would be introducing new training heuristics (e.g., class sampling), which defeats the initial objective of using the cross-entropy loss. Again, the SPCE loss underlying the unary cross-entropy could again handle such cases, similarly to other pairwise losses, given that it doesn't require storing such weight matrix.

APPENDIX II

SUPPLEMENTARY MATERIAL FOR THE PAPER TITLED *DECOUPLING DIRECTION AND NORM FOR EFFICIENT GRADIENT-BASED ℓ_2 ADVERSARIAL ATTACKS AND DEFENSES*

1. Model architectures

Table II-1 lists the architectures of the CNNs used in the Attack Evaluation - we used the same architecture as in (Carlini & Wagner (2017)) for a fair comparison against the C&W and DeepFool attacks. Table II-2 lists the architecture used in the robust model (defense) trained on CIFAR-10. We used a Wide ResNet with 28 layers and widening factor of 10 (WRN-28-10). The residual blocks used are the "basic block" (He *et al.* (2016a); Zagoruyko & Komodakis (2016)), with stride 1 for the first group and stride 2 for the second an third groups. This architecture is slightly different from the one used by Madry *et al.* (2018), where Madry *et al.* (2018) they use a modified version of Wide ResNet with 5 residual blocks instead of 4 in each group, and without convolutions in the residual connections (when the shape of the output changes, *e.g.* with stride=2).

2. Hyperparameters selected for the C&W attack

Table-A II-1 CNN architectures used for the Attack Evaluation

Layer Type	**MNIST Model**	**CIFAR-10 Model**
Convolution + ReLU	$3 \times 3 \times 32$	$3 \times 3 \times 64$
Convolution + ReLU	$3 \times 3 \times 32$	$3 \times 3 \times 64$
Max Pooling	2×2	2×2
Convolution + ReLU	$3 \times 3 \times 64$	$3 \times 3 \times 128$
Convolution + ReLU	$3 \times 3 \times 64$	$3 \times 3 \times 128$
Max Pooling	2×2	2×2
Fully Connected + ReLU	200	256
Fully Connected + ReLU	200	256
Fully Connected + Softmax	10	10

Table-A II-2 CIFAR-10 architecture used for the Defense evaluation

Layer Type	**Size**
Convolution	$3 \times 3 \times 16$
Residual Block	$\begin{bmatrix} 3 \times 3, 160 \\ 3 \times 3, 160 \end{bmatrix} \times 4$
Residual Block	$\begin{bmatrix} 3 \times 3, 320 \\ 3 \times 3, 320 \end{bmatrix} \times 4$
Residual Block	$\begin{bmatrix} 3 \times 3, 640 \\ 3 \times 3, 640 \end{bmatrix} \times 4$
Batch Normalization + ReLU	-
Average Pooling	8×8
Fully Connected + Softmax	10

We considered a scenario of running the C&W attack with 100 steps and a fixed C (1×100), and a scenario of running 4 search steps on C, of 25 iterations each (4×25). Since the hyperparameters proposed in (Carlini & Wagner (2017)) were tuned for a larger number of iterations and search steps, we performed a grid search for each dataset, using learning rates in the range [0.01, 0.05, 0.1, 0.5, 1], and C in the range [0.001, 0.01, 0.1, 1, 10, 100, 1 000]. We selected the hyperparameters that resulted in targeted attacks with lowest Median ℓ_2 for each dataset. Table II-3 lists the hyperparameters found through this search procedure.

Table-A II-3 Hyperparameters used for the C&W attack when restricted to 100 iterations

Dataset	**# Iterations**	**Parameters**
MNIST	1×100	$\alpha = 0.1$, $C = 1$
MNIST	4×25	$\alpha = 0.5$, $C = 1$
CIFAR-10	1×100	$\alpha = 0.01$, $C = 0.1$
CIFAR-10	4×25	$\alpha = 0.01$, $C = 0.1$
ImageNet	1×100	$\alpha = 0.01$, $C = 1$
ImageNet	4×25	$\alpha = 0.01$, $C = 10$

3. Examples of adversarial images

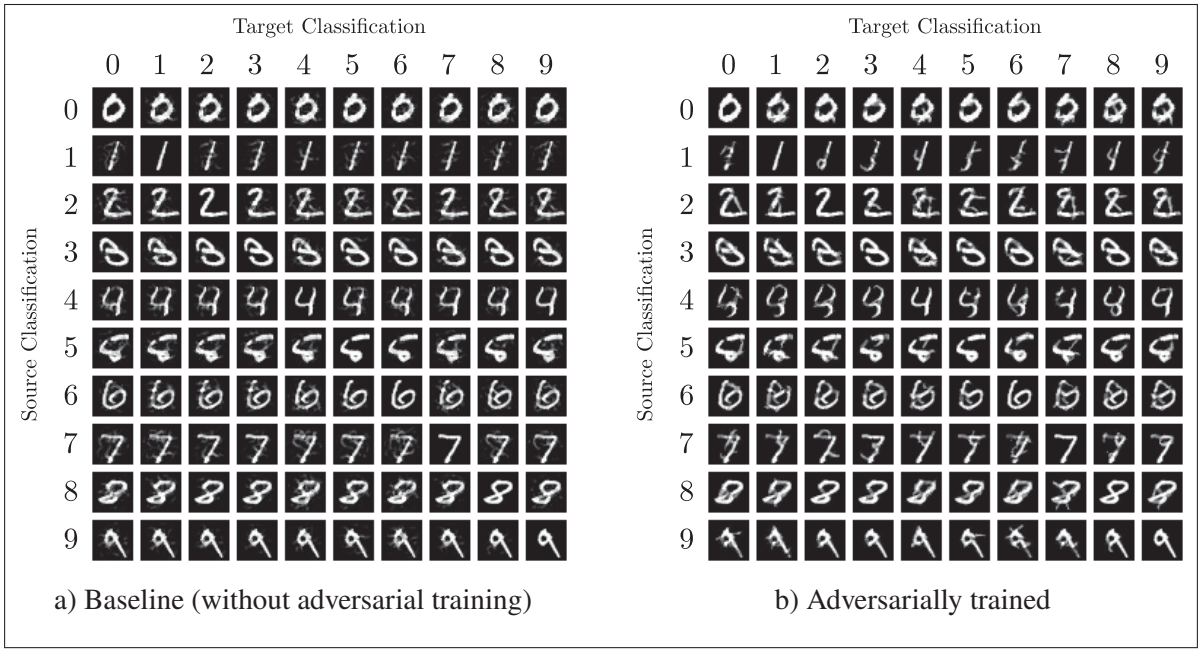


Figure-A II-1 Adversarial examples obtained using the C&W ℓ_2 attack on two models: (a) Baseline, (b) model adversarially trained with our attack

Figure II-1 plots a grid of attacks (obtained with the C&W attack) against the first 10 examples in the MNIST dataset. The rows indicate the source classification (label), and the columns indicate the target class used to generate the attack (images on the diagonal are the original samples). We can see that in the adversarially trained model, the attacks need to introduce much larger changes to the samples in order to make them adversarial, and some of the adversarial samples visually resemble another class.

Figure II-2 shows randomly-selected adversarial examples for the CIFAR-10 dataset, comparing the baseline model (WRN 28-10), the Madry defense and our proposed defense. For each image and model, we ran three attacks (DDN 1 000, C&W 9×10 000, DeepFool 100), and present the adversarial example with minimum ℓ_2 perturbation among them. Figure II-3 shows cherry-picked adversarial examples on CIFAR-10, that visually resemble another class, when attacking the proposed defense. We see that on the average case (randomly-selected), adversarial examples against the defenses still require low amounts of noise (perceptually) to induce misclassification.

On the other hand, we notice that on adversarially trained models, some examples do require a much larger change on the image, making it effectively resemble another class.

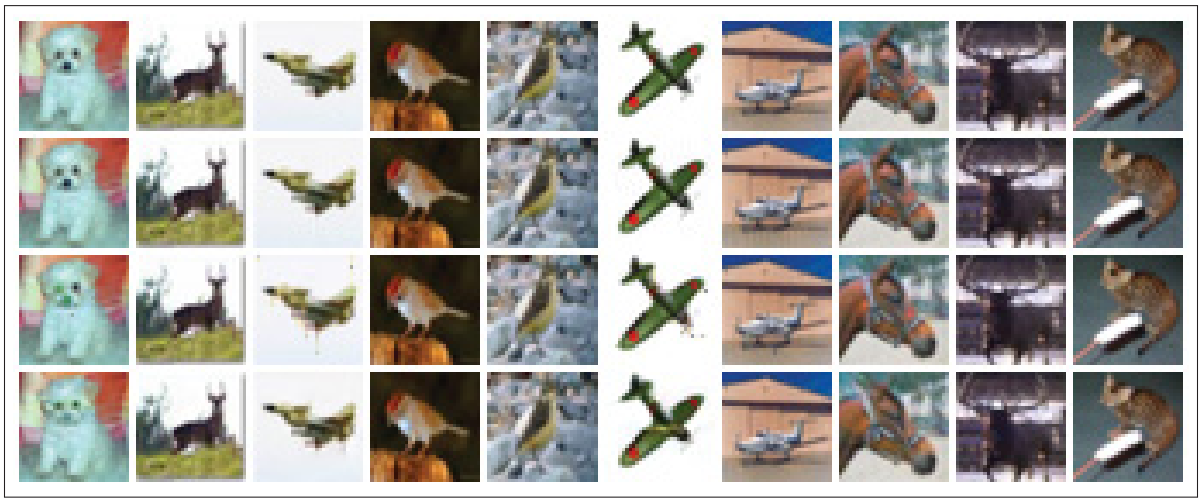


Figure-A II-2 Randomly chosen adversarial examples on CIFAR-10 for three models. **Top row**: original images; **second row**: attacks against the baseline; **third row**: attacks against the Madry defense

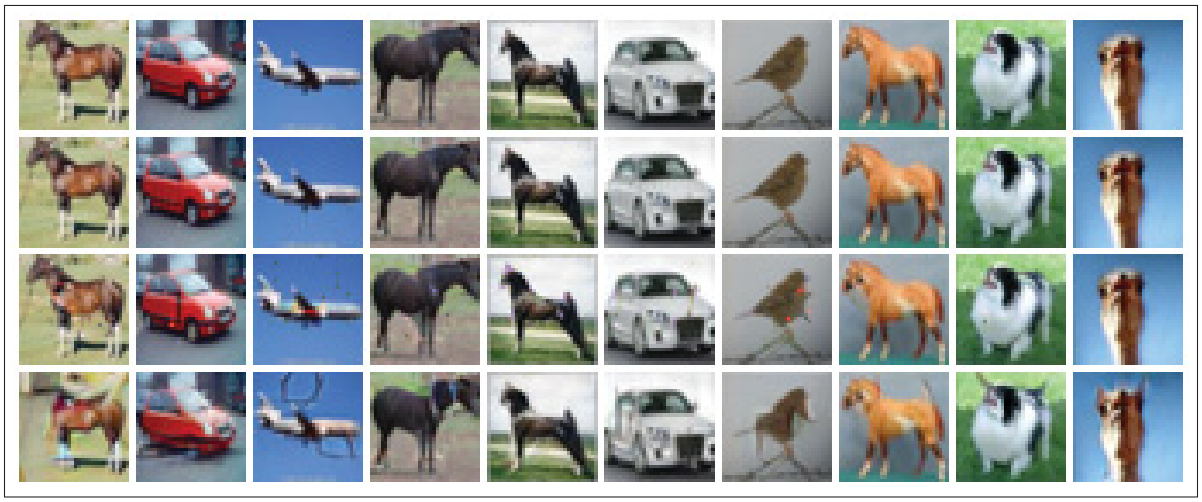
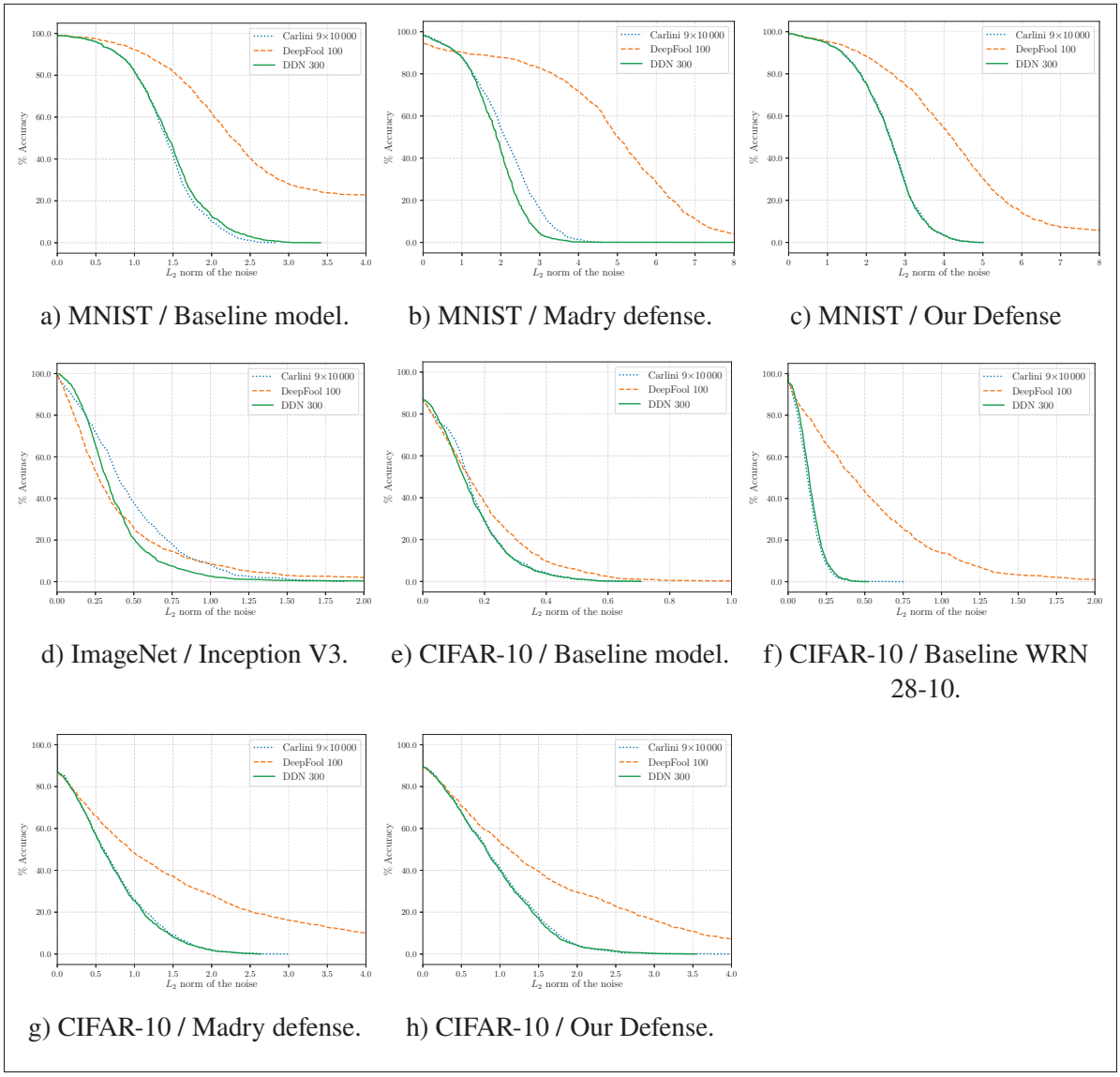


Figure-A II-3 Cherry-picked adversarial examples on CIFAR-10 for three models. **Top row**: original images; **second row**: attacks against the baseline; **third row**: attacks against the Madry defense; **bottom row**: attacks against the proposed defense. Predicted labels for the last row are, from left to right: dog, ship, deer, dog, dog, truck, horse, dog, cat, cat

4. Attack performance curves

Figure II-4 reports curves of the perturbation size against accuracy of the models for three attacks: Carlini 9×10 000, DeepFool 100 and DDN 300.



a) MNIST / Baseline model. b) MNIST / Madry defense. c) MNIST / Our Defense

d) ImageNet / Inception V3. e) CIFAR-10 / Baseline model. f) CIFAR-10 / Baseline WRN 28-10.

g) CIFAR-10 / Madry defense. h) CIFAR-10 / Our Defense.

Figure-A II-4 Attacks performances on different datasets and models

APPENDIX III

SUPPLEMENTARY MATERIAL FOR THE PAPER TITLED *AUGMENTED LAGRANGIAN ADVERSARIAL ATTACKS*

1. Differences with ADMM attack (Zhao, Xu, Liu, Wang & Lin, 2019) and StrAttack (Xu *et al.*, 2019)

There is a significant technical difference between our proposed attack and the two attacks based on ADMM approaches (Zhao *et al.*, 2019; Xu *et al.*, 2019) as the multipliers in the ADMM attacks are used for the problem-splitting constraints, but not for the attack constraints as in our ALM. For two terms, ADMM replaces a one-variable problem of the form $\underset{x}{\text{minimize}}\, g(x) + h(x)$ by a two-variable problem:

$$\underset{x,y}{\text{minimize}} \quad g(x) + h(y) \quad \text{subject to} \quad x = y \tag{A III-1}$$

a splitting that gives raise to variable-consistency constraints $x = y$. In fact, both ADMM attacks (Xu *et al.*, 2019; Zhao *et al.*, 2019) are based on a decomposition of the Carlini-Wagner penalty formulation (Equation 7 in (Zhao *et al.*, 2019) and Equation 4 in (Xu *et al.*, 2019)):

$$D(\boldsymbol{x} + \boldsymbol{\delta}, \boldsymbol{x}) + g_{\text{CW}}(\beta) \quad \text{subject to} \quad \boldsymbol{\delta} = \beta \tag{A III-2}$$

where D is the distance and g_{CW} is the standard CW penalty for attack constraint $f_y(\boldsymbol{x} + \boldsymbol{\delta}) - \max_{k \neq y} f_k(\boldsymbol{x} + \boldsymbol{\delta}) < 0$; see Equation 5 in (Zhao *et al.*, 2019) and Equation 3 in (Xu *et al.*, 2019). The Lagrange multipliers in these ADMM attacks are for the decomposition constraints $\boldsymbol{\delta} = \beta$, but the attack constraints are still handled with the standard CW penalty. In our case, we address the attack constraints with augmented Lagrangian principles, and there is no ADMM splitting in our method. The ADMM attack in (Zhao *et al.*, 2019) has no public implementation, so we were not able to implement it in our experimental framework. The publicly available implementation of the StrAttack (Xu *et al.*, 2019) contains several differences with the original paper regarding hyper-parameters and update rules for the auxiliary variables. Therefore, we

contacted the authors of both papers (of which several are in common) regarding the lack of public implementation, and discrepancies between paper and code, but did not get any answer. Therefore, we did not include these attacks in our experiments. It should be noted that StrAttack's (Xu *et al.*, 2019) implementation is based on the C&W ℓ_2 attack, but adds a sparsity objective, which tends to increase the perturbation size in terms of ℓ_2 norm compared to the vanilla C&W ℓ_2 attack.

2. CIEDE2000

The CIEDE2000 color difference formula is complex, so we advise the reader to look at the original work (Sharma *et al.*, 2005). This formula is calculated using the CIELAB color space. However, most image datasets are provided in an RGB format. Therefore, to use the CIEDE2000 color difference formula, we must first convert the images from RGB to the CIELAB color space. We need to use a first conversion between RGB and XYZ color spaces. For this step, we need to know the RGB working space and the reference white. However, we do not have that information available, as we do not know how the images were captured in the first place. As a consequence, we make the assumption of the sRGB working space with an Illuminant D65 white reference. With these assumptions, the formula to convert from RGB (with values in $[0, 1]$) to XYZ is the following:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124564 & 0.3575761 & 0.1804375 \\ 0.2126729 & 0.7151522 & 0.0721750 \\ 0.0193339 & 0.1191920 & 0.9503041 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{A III-3}$$

Once we have the colors represented in the XYZ color space, we need to convert to the CIELAB color space. The conversion is the following:

$$\begin{aligned} L^\star &= 116 f\left(\frac{Y}{Y_n}\right) - 16 \\ a^\star &= 500\left(f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right)\right) \\ b^\star &= 200\left(f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right)\right) \end{aligned} \tag{A III-4}$$

where:

$$f(t) = \begin{cases} \sqrt[3]{t} & \text{if} \quad t > \boldsymbol{\delta}^3 \\ \frac{t}{3\boldsymbol{\delta}^2} + \frac{4}{29} & \text{otherwise} \end{cases} \tag{A III-5}$$

with $\boldsymbol{\delta} = \frac{6}{29}$. Under the Illuminant D65 white reference, we have $X_n = 95.0489$, $Y_n = 100$ and $Z_n = 108.8840$.

3. Modified DLR loss

The original DLR loss proposed in (Croce & Hein, 2020b) is formulated as follows:

$$\mathrm{DLR}(z, y) = -\frac{z_y - \max\limits_{i \neq y} z_i}{z_{\pi_1} - z_{\pi_3}} \tag{A III-6}$$

where $z = f(\boldsymbol{x})$ and π is the ordering of the element of z in decreasing order. If a sample $\boldsymbol{x}$ is correctly classified, we have $\mathrm{DLR}(z, y) \in [-1, 0]$ and $\boldsymbol{x}$ is misclassified only if $\mathrm{DLR}(z, y) > 0$. Croce *et al*. also propose a variant for untargeted attacks with a targeted objective. In some cases, performing a targeted attack against each class proved to be more successful at finding untargeted adversarial examples than simply performing an untargeted attack. The targeted variant is:

$$\mathrm{tDLR}(z, y) = -\frac{z_y - z_t}{z_{\pi_1} - (z_{\pi_3} + z_{\pi_4})/2} \tag{A III-7}$$

where t is the target class. For this variant, we can have no guarantee as to what $\boldsymbol{x}$ is classified as, simply by looking at the value of tDLR.

For our optimization problem, we need to have a loss that is negative only when the misclassification or targeted classification is achieved. This way, we can formulate the misclassification or targeted classification constraint as $g(x) < 0$. To this end, we modify DLR by taking the negative as follows:

$$\mathrm{DLR}^+(z, y) = \frac{z_y - \max\limits_{i \neq y} z_i}{z_{\pi_1} - z_{\pi_3}} \tag{4.4}$$

For targeted attack, we modify tDLR as follows:

$$\mathrm{tDLR}^+(z, y) = \frac{\max\limits_{i \neq t} z_i - z_t}{z_{\pi_1} - (z_{\pi_3} + z_{\pi_4})/2} \tag{A III-8}$$

With these modifications, the misclassification and targeted classification constraints are respected only when DLR^+ and tDLR^+ are negative. Conversely, the constraints are violated when these losses are positive, hence the $^+$ superscript.

4. Penalty functions

The four penalty functions plotted in Figure 4.1 are taken from (Birgin *et al.*, 2005) and defined as follows:

$$\mathrm{PHR}(y, \rho, \mu) = \frac{1}{2\rho}(\max\{0, \mu + \rho y\}^2 - \mu^2) \tag{A III-9}$$

$$P_1(y, \rho, \mu) = \begin{cases} \mu y + \frac{1}{2}\rho y^2 + \rho^2 y^3 & \text{if} \quad y \geq 0 \\ \mu y + \frac{1}{2}\rho y^2 & \text{if} \quad -\frac{\mu}{\rho} \leq y \leq 0 \\ -\frac{1}{2\rho}\mu^2 & \text{if} \quad y \leq -\frac{\mu}{\rho} \end{cases} \tag{A III-10}$$

$$P_2(y, \rho, \mu) = \begin{cases} \mu y + \mu\rho y^2 + \frac{1}{6}\rho^2 y^3 & \text{if} \quad y \geq 0 \\ \frac{\mu y}{1-\rho y} & \text{if} \quad y \leq 0 \end{cases} \tag{A III-11}$$

$$P_3(y, \rho, \mu) = \begin{cases} \mu y + \mu\rho y^2 & \text{if} \quad y \geq 0 \\ \frac{\mu y}{1-\rho y} & \text{if} \quad y \leq 0 \end{cases} \tag{A III-12}$$

5. Choice of α

For the experiments, we change the value of α according to the number of iterations. In our attack, α is a smoothing parameter. However, too much smoothing can degrade the performance of the attack for lower numbers of iterations. Therefore, we recommend values between 0.5 and 0.9 for numbers of iterations between 100 and 1 000, and to keep $\alpha = 0.9$ for more than 1 000 iterations. Since our attack aims to find minimal adversarial perturbations, lower numbers of iterations are not recommended.

6. Hyper-parameter ϵ values

Table III-1 reports the different ϵ used for each distance function in our experiments.

Table-A III-1 Initial values of ϵ for each distance

Distance	ϵ
ℓ_1	0.5
ℓ_2	0.1
SSIM	3×10^{-5}
CIEDE2000	0.05
LPIPS	1×10^{-3}

7. Additional results with SSIM

We also tested our ALMA attack with the SSIM (Wang, Bovik, Sheikh & Simoncelli, 2004) for which, to the best of our knowledge, no gradient-based attack currently exists. The only related work on adversarial attacks and SSIM is a black-box method (Gragnaniello *et al.*, 2021). SSIM is a similarity function, so identical images have a SSIM of 1. Therefore, we minimize the quantity $1 - \text{SSIM}$ and report this instead of the SSIM. The SSIM metric is defined between two gray-level images. Several modifications exist for color images, however, we simply considered the average SSIM over the color channels. Tables III-8 and III-13 report the results for all models on CIFAR10 and ImageNet respectively.

8. Detailed experimental results

Tables III-2, III-3, III-4, III-5, III-8, III-6, III-7, III-9, III-10, III-13, III-11 and III-12 report the detailed results for each dataset, model and attack. Results from Tables 4.1 and 4.2 are calculated from these tables using the geometric mean over the models. For the CIFAR10 and ImageNet models, RN stands for ResNet and WRN for Wide ResNet. For ImageNet, the targeted variant of FAB is used in the experiments denoted by a T superscript (see section 4.4 for details).

9. Robust Accuracy curves

Figures III-1, III-2, III-3, III-4, III-5, III-7, III-8 and III-9 present the robust accuracy curves for each dataset and model against the attacks considered for each distance. The dotted line represent the reduced budget versions of the attack, as reported in the corresponding tables.

Table-A III-2 Performance of the ℓ_1 attacks on the MNIST dataset for each model

Model	Attack	ASR (%)	Median ℓ_1	Forwards / Backwards
SmallCNN	EAD 9×100 (Chen *et al.*, 2018b)	100	8.41	870 / 439
	EAD 9×1 000 (Chen *et al.*, 2018b)	100	7.90	3 810 / 1 909
	FAB ℓ_1 100 (Croce & Hein, 2020a)	100	6.31	201 / 1 000
	FAB ℓ_1 1 000 (Croce & Hein, 2020a)	100	6.20	2 001 / 10 000
	FMN ℓ_1 100 (Pintor *et al.*, 2021)	95.02	6.50	100 / 100
	FMN ℓ_1 1 000 (Pintor *et al.*, 2021)	95.19	6.39	1 000 / 1 000
	ALMA ℓ_1 100	100	6.77	100 / 100
	ALMA ℓ_1 1 000	100	6.23	1 000 / 1 000
SmallCNN DDN	EAD 9×100 (Chen *et al.*, 2018b)	100	17.41	990 / 499
	EAD 9×1 000 (Chen *et al.*, 2018b)	100	16.35	5 010 / 2 509
	FAB ℓ_1 100 (Croce & Hein, 2020a)	100	16.53	201 / 1 000
	FAB ℓ_1 1 000 (Croce & Hein, 2020a)	100	15.42	2 001 / 10 000
	FMN ℓ_1 100 (Pintor *et al.*, 2021)	99.97	16.09	100 / 100
	FMN ℓ_1 1 000 (Pintor *et al.*, 2021)	99.97	15.67	1 000 / 1 000
	ALMA ℓ_1 100	100	14.73	100 / 100
	ALMA ℓ_1 1 000	100	14.02	1 000 / 1 000
SmallCNN TRADES	EAD 9×100 (Chen *et al.*, 2018b)	100	14.80	967 / 486
	EAD 9×1 000 (Chen *et al.*, 2018b)	100	12.15	6 409 / 3 208
	FAB ℓ_1 100 (Croce & Hein, 2020a)	99.22	36.60	201 / 1 000
	FAB ℓ_1 1 000 (Croce & Hein, 2020a)	99.35	32.37	2 001 / 10 000
	FMN ℓ_1 100 (Pintor *et al.*, 2021)	50.30	42.02	100 / 100
	FMN ℓ_1 1 000 (Pintor *et al.*, 2021)	98.30	8.28	1 000 / 1 000
	ALMA ℓ_1 100	100	6.16	100 / 100
	ALMA ℓ_1 1 000	100	5.32	1 000 / 1 000
CROWN IBP	EAD 9×100 (Chen *et al.*, 2018b)	89.17	106.95	509 / 258
	EAD 9×1 000 (Chen *et al.*, 2018b)	91.32	86.45	5 210 / 2 609
	FAB ℓ_1 100 (Croce & Hein, 2020a)	99.99	147.79	201 / 1 000
	FAB ℓ_1 1 000 (Croce & Hein, 2020a)	99.99	110.96	2 001 / 10 000
	FMN ℓ_1 100 (Pintor *et al.*, 2021)	49.89	–	100 / 100
	FMN ℓ_1 1 000 (Pintor *et al.*, 2021)	88.36	3.50	1 000 / 1 000
	ALMA ℓ_1 100	99.59	27.94	100 / 100
	ALMA ℓ_1 1 000	100	5.65	1 000 / 1 000

Table-A III-3 Performance of the ℓ_2 attacks on the MNIST dataset for each model

Model	Attack	ASR (%)	Median ℓ_2	Forwards / Backwards
SmallCNN	C&W ℓ_2 9×1 000 (Carlini & Wagner, 2017)	99.98	1.35	9 000 / 9 000
	C&W ℓ_2 9×10 000 (Carlini & Wagner, 2017)	99.77	1.35	90 000 / 90 000
	DDN 100 (Rony *et al.*, 2019)	100	1.39	100 / 100
	DDN 1 000 (Rony *et al.*, 2019)	100	1.37	1 000 / 1 000
	FAB ℓ_2 100 (Croce & Hein, 2020a)	100	1.37	201 / 1 000
	FAB ℓ_2 1 000 (Croce & Hein, 2020a)	100	1.36	2 001 / 10 000
	FMN ℓ_2 100 (Pintor *et al.*, 2021)	82.04	1.53	100 / 100
	FMN ℓ_2 1 000 (Pintor *et al.*, 2021)	96.61	1.39	1 000 / 1 000
	$\text{APGD}^{\text{T}}_{\text{DLR}}$ ℓ_2 (Croce & Hein, 2020b)	100	1.31	13 082 / 13 062
	ALMA ℓ_2 100	100	1.38	100 / 100
	ALMA ℓ_2 1 000	100	1.32	1 000 / 1 000
SmallCNN DDN	C&W ℓ_2 9×1 000 (Carlini & Wagner, 2017)	99.96	2.76	9 000 / 9 000
	C&W ℓ_2 9×10 000 (Carlini & Wagner, 2017)	99.59	2.69	90 000 / 90 000
	DDN 100 (Rony *et al.*, 2019)	100	2.74	100 / 100
	DDN 1 000 (Rony *et al.*, 2019)	100	2.66	1 000 / 1 000
	FAB ℓ_2 100 (Croce & Hein, 2020a)	100	2.74	201 / 1 000
	FAB ℓ_2 1 000 (Croce & Hein, 2020a)	100	2.71	2 001 / 10 000
	FMN ℓ_2 100 (Pintor *et al.*, 2021)	99.95	2.67	100 / 100
	FMN ℓ_2 1 000 (Pintor *et al.*, 2021)	100	2.67	1 000 / 1 000
	$\text{APGD}^{\text{T}}_{\text{DLR}}$ ℓ_2 (Croce & Hein, 2020b)	100	2.58	13 410 / 13 390
	ALMA ℓ_2 100	100	2.68	100 / 100
	ALMA ℓ_2 1 000	100	2.59	1 000 / 1 000
SmallCNN TRADES	C&W ℓ_2 9×1 000 (Carlini & Wagner, 2017)	99.99	3.32	9 000 / 9 000
	C&W ℓ_2 9×10 000 (Carlini & Wagner, 2017)	99.97	2.28	90 000 / 90 000
	DDN 100 (Rony *et al.*, 2019)	99.69	2.17	100 / 100
	DDN 1 000 (Rony *et al.*, 2019)	100	1.91	1 000 / 1 000
	FAB ℓ_2 100 (Croce & Hein, 2020a)	99.88	1.77	201 / 1 000
	FAB ℓ_2 1 000 (Croce & Hein, 2020a)	99.90	1.74	2 001 / 10 000
	FMN ℓ_2 100 (Pintor *et al.*, 2021)	86.41	2.24	100 / 100
	FMN ℓ_2 1 000 (Pintor *et al.*, 2021)	99.83	1.99	1 000 / 1 000
	$\text{APGD}^{\text{T}}_{\text{DLR}}$ ℓ_2 (Croce & Hein, 2020b)	100	3.36	13 919 / 13 899
	ALMA ℓ_2 100	100	1.74	100 / 100
	ALMA ℓ_2 1 000	100	1.55	1 000 / 1 000
CROWN IBP	C&W ℓ_2 9×1 000 (Carlini & Wagner, 2017)	2.61	–	9 000 / 9 000
	C&W ℓ_2 9×10 000 (Carlini & Wagner, 2017)	2.63	–	90 000 / 90 000
	DDN 100 (Rony *et al.*, 2019)	94.34	1.46	100 / 100
	DDN 1 000 (Rony *et al.*, 2019)	99.27	0.97	1 000 / 1 000
	FAB ℓ_2 100 (Croce & Hein, 2020a)	99.98	5.19	201 / 1 000
	FAB ℓ_2 1 000 (Croce & Hein, 2020a)	99.98	3.34	2 001 / 10 000
	FMN ℓ_2 100 (Pintor *et al.*, 2021)	67.80	2.14	100 / 100
	FMN ℓ_2 1 000 (Pintor *et al.*, 2021)	89.08	1.34	1 000 / 1 000
	$\text{APGD}^{\text{T}}_{\text{DLR}}$ ℓ_2 (Croce & Hein, 2020b)	99.94	3.57	9 286 / 9 273
	ALMA ℓ_2 100	98.90	4.96	100 / 100
	ALMA ℓ_2 1 000	100	1.26	1 000 / 1 000

Table-A III-4 Performance of the ℓ_1 attacks on the CIFAR10 dataset for each model

Model	Attack	ASR (%)	Median ℓ_1	Forwards / Backwards
WRN 28-10	EAD 9×100 (Chen *et al.*, 2018b)	100	1.79	530 / 269
	EAD 9×1 000 (Chen *et al.*, 2018b)	100	1.62	4 910 / 2 459
	FAB ℓ_1 100 (Croce & Hein, 2020a)	92.3	1.27	200 / 1 000
	FAB ℓ_1 1 000 (Croce & Hein, 2020a)	98.8	1.07	2 000 / 10 000
	FMN ℓ_1 100 (Pintor *et al.*, 2021)	99.7	1.01	100 / 100
	FMN ℓ_1 1 000 (Pintor *et al.*, 2021)	99.5	0.98	1 000 / 1 000
	ALMA ℓ_1 100	100	1.26	100 / 100
	ALMA ℓ_1 1 000	100	1.02	1 000 / 1 000
WRN 28-10 Carmon *et al.* (2019)	EAD 9×100 (Chen *et al.*, 2018b)	100	6.62	600 / 304
	EAD 9×1 000 (Chen *et al.*, 2018b)	100	6.07	3 760 / 1 884
	FAB ℓ_1 100 (Croce & Hein, 2020a)	97.8	5.57	200 / 1 000
	FAB ℓ_1 1 000 (Croce & Hein, 2020a)	98.2	5.07	2 000 / 10 000
	FMN ℓ_1 100 (Pintor *et al.*, 2021)	100	4.70	100 / 100
	FMN ℓ_1 1 000 (Pintor *et al.*, 2021)	100	4.64	1 000 / 1 000
	ALMA ℓ_1 100	100	5.20	100 / 100
	ALMA ℓ_1 1 000	100	4.75	1 000 / 1 000
RN-50 Augustin *et al.* (2020)	EAD 9×100 (Chen *et al.*, 2018b)	100	19.18	590 / 299
	EAD 9×1 000 (Chen *et al.*, 2018b)	100	16.39	4 260 / 2 134
	FAB ℓ_1 100 (Croce & Hein, 2020a)	99.8	10.95	200 / 1 000
	FAB ℓ_1 1 000 (Croce & Hein, 2020a)	99.8	10.09	2 000 / 10 000
	FMN ℓ_1 100 (Pintor *et al.*, 2021)	100	10.21	100 / 100
	FMN ℓ_1 1 000 (Pintor *et al.*, 2021)	100	9.79	1 000 / 1 000
	ALMA ℓ_1 100	100	12.15	100 / 100
	ALMA ℓ_1 1 000	100	10.35	1 000 / 1 000

Table-A III-5 Performance of the ℓ_2 attacks on the CIFAR10 dataset for each model

Model	Attack	ASR (%)	Median ℓ_2	Forwards / Backwards
WRN 28-10	C&W ℓ_2 9×1 000 (Carlini & Wagner, 2017)	100	0.10	9 000 / 9 000
	C&W ℓ_2 9×10 000 (Carlini & Wagner, 2017)	100	0.10	90 000 / 90 000
	DDN 100 (Rony *et al.*, 2019)	100	0.11	100 / 100
	DDN 1 000 (Rony *et al.*, 2019)	100	0.11	1 000 / 1 000
	FAB ℓ_2 100 (Croce & Hein, 2020a)	100	0.09	201 / 1 000
	FAB ℓ_2 1 000 (Croce & Hein, 2020a)	100	0.09	2 001 / 10 000
	FMN ℓ_2 100 (Pintor *et al.*, 2021)	99.7	0.12	100 / 100
	FMN ℓ_2 1 000 (Pintor *et al.*, 2021)	99.5	0.09	1 000 / 1 000
	$\text{APGD}^{\text{T}}_{\text{DLR}}$ ℓ_2 (Croce & Hein, 2020b)	100	0.09	4 336 / 4 312
	ALMA ℓ_2 100	100	0.09	100 / 100
	ALMA ℓ_2 1 000	100	0.09	1 000 / 1 000
WRN 28-10 Carmon *et al.* (2019)	C&W ℓ_2 9×1 000 (Carlini & Wagner, 2017)	100	0.70	7 502 / 7 500
	C&W ℓ_2 9×10 000 (Carlini & Wagner, 2017)	100	0.70	71 602 / 71 600
	DDN 100 (Rony *et al.*, 2019)	100	0.72	100 / 100
	DDN 1 000 (Rony *et al.*, 2019)	100	0.71	1 000 / 1 000
	FAB ℓ_2 100 (Croce & Hein, 2020a)	100	0.71	201 / 1 000
	FAB ℓ_2 1 000 (Croce & Hein, 2020a)	100	0.71	2 001 / 10 000
	FMN ℓ_2 100 (Pintor *et al.*, 2021)	100	0.69	100 / 100
	FMN ℓ_2 1 000 (Pintor *et al.*, 2021)	100	0.70	1 000 / 1 000
	$\text{APGD}^{\text{T}}_{\text{DLR}}$ ℓ_2 (Croce & Hein, 2020b)	100	0.68	5 683 / 5 659
	ALMA ℓ_2 100	100	0.70	100 / 100
	ALMA ℓ_2 1 000	100	0.67	1 000 / 1 000
RN-50 Augustin *et al.* (2020)	C&W ℓ_2 9×1 000 (Carlini & Wagner, 2017)	100	0.96	7 515 / 7 513
	C&W ℓ_2 9×10 000 (Carlini & Wagner, 2017)	100	0.95	73 869 / 73 867
	DDN 100 (Rony *et al.*, 2019)	100	0.97	100 / 100
	DDN 1 000 (Rony *et al.*, 2019)	100	0.96	1 000 / 1 000
	FAB ℓ_2 100 (Croce & Hein, 2020a)	100	1.01	201 / 1 000
	FAB ℓ_2 1 000 (Croce & Hein, 2020a)	100	1.00	2 001 / 10 000
	FMN ℓ_2 100 (Pintor *et al.*, 2021)	100	0.95	100 / 100
	FMN ℓ_2 1 000 (Pintor *et al.*, 2021)	100	0.96	1 000 / 1 000
	$\text{APGD}^{\text{T}}_{\text{DLR}}$ ℓ_2 (Croce & Hein, 2020b)	100	0.91	6 198 / 6 174
	ALMA ℓ_2 100	100	0.98	100 / 100
	ALMA ℓ_2 1 000	100	0.92	1 000 / 1 000

Table-A III-6 Performance of the CIEDE2000 attacks on the CIFAR10 dataset for each model

Model	Attack	ASR (%)	Median CIEDE2000	Forwards / Backwards
WRN 28-10	C&W CIEDE2000 9×1 000	100	0.23	7 741 / 7 740
	Perc-AL 100 (Zhao *et al.*, 2020)	100	0.86	201 / 100
	Perc-AL 1 000 (Zhao *et al.*, 2020)	100	0.72	2 001 / 1 000
	ALMA CIEDE2000 100	100	0.18	100 / 100
	ALMA CIEDE2000 1 000	100	0.14	1 000 / 1 000
WRN 28-10 Carmon *et al.* (2019)	C&W CIEDE2000 9×1 000	100	2.12	6 243 / 6 240
	Perc-AL 100 (Zhao *et al.*, 2020)	100	5.69	201 / 100
	Perc-AL 1 000 (Zhao *et al.*, 2020)	100	5.82	2 001 / 1 000
	ALMA CIEDE2000 100	100	3.65	100 / 100
	ALMA CIEDE2000 1 000	100	2.08	1 000 / 1 000
RN-50 Augustin *et al.* (2020)	C&W CIEDE2000 9×1 000	100	1.63	6 303 / 6 300
	Perc-AL 100 (Zhao *et al.*, 2020)	100	4.85	201 / 100
	Perc-AL 1 000 (Zhao *et al.*, 2020)	100	4.83	2 001 / 1 000
	ALMA CIEDE2000 100	99.8	1.94	100 / 100
	ALMA CIEDE2000 1 000	99.8	1.58	1 000 / 1 000

Table-A III-7 Performance of the LPIPS variant of ALMA on the CIFAR10 dataset for each model. ‡A binary search is performed on each sample to get a minimal perturbation attack (Equation 4.2)

Model	Attack	ASR (%)	Median LPIPS $\times 10^{-2}$	Forwards / Backwards
WRN 28-10	C&W LPIPS 9×1 000	100	0.32	4 565 / 4 560
	LPA‡ (Laidlaw *et al.*, 2021)	100	4.81	1 129 / 1 119
	ALMA LPIPS 100	100	0.29	100 / 100
	ALMA LPIPS 1 000	100	0.12	1 000 / 1 000
WRN 28-10 Carmon *et al.* (2019)	C&W LPIPS 9×1 000	100	0.50	7 981 / 7 980
	LPA‡ (Laidlaw *et al.*, 2021)	100	5.06	1 092 / 1 082
	ALMA LPIPS 100	100	6.76	100 / 100
	ALMA LPIPS 1 000	100	1.01	1 000 / 1 000
RN-50 Augustin *et al.* (2020)	C&W LPIPS 9×1 000	100	0.64	8 101 / 8 100
	LPA‡ (Laidlaw *et al.*, 2021)	100	6.42	1 133 / 1 123
	ALMA LPIPS 100	99.9	7.66	100 / 100
	ALMA LPIPS 1 000	100	1.82	1 000 / 1 000

Table-A III-8 Performance of the SSIM variant of ALMA on the CIFAR10 dataset for each model

Model	Attack	ASR (%)	Median $1-\text{SSIM} \times 10^{-4}$	Forwards / Backwards
WRN 28-10	ALMA SSIM 100	100	0.4	100 / 100
	ALMA SSIM 1 000	100	0.1	1 000 / 1 000
WRN 28-10 Carmon *et al.* (2019)	ALMA SSIM 100	100	7.5	100 / 100
	ALMA SSIM 1 000	100	2.8	1 000 / 1 000
ResNet-50 Augustin *et al.* (2020)	ALMA SSIM 100	100	4.1	100 / 100
	ALMA SSIM 1 000	100	2.0	1 000 / 1 000

Table-A III-9 Performance of the ℓ_1 attacks on the ImageNet dataset for each model

Model	Attack	ASR (%)	Median ℓ_1	Forwards / Backwards
RN-50	EAD 9×100 (Chen *et al.*, 2018b)	100	6.70	437 / 222
	EAD 9×1 000 (Chen *et al.*, 2018b)	100	6.08	4 510 / 2 259
	FAB^T ℓ_1 100 (Croce & Hein, 2020a)	74.4	9.01	1 810 / 900
	FAB^T ℓ_1 1 000 (Croce & Hein, 2020a)	81.0	4.82	18 010 / 9 000
	FMN ℓ_1 100 (Pintor *et al.*, 2021)	95.6	3.72	100 / 100
	FMN ℓ_1 1 000 (Pintor *et al.*, 2021)	94.5	3.43	1 000 / 1 000
	ALMA ℓ_1 100	100	8.47	100 / 100
	ALMA ℓ_1 1 000	100	4.25	1 000 / 1 000
RN-50 ℓ_2-AT	EAD 9×100 (Chen *et al.*, 2018b)	100	62.21	458 / 233
	EAD 9×1 000 (Chen *et al.*, 2018b)	100	55.16	3 450 / 1 729
	FAB^T ℓ_1 100 (Croce & Hein, 2020a)	98.5	31.33	1 810 / 900
	FAB^T ℓ_1 1 000 (Croce & Hein, 2020a)	93.2	33.58	18 010 / 9 000
	FMN ℓ_1 100 (Pintor *et al.*, 2021)	100	36.68	100 / 100
	FMN ℓ_1 1 000 (Pintor *et al.*, 2021)	100	30.52	1 000 / 1 000
	ALMA ℓ_1 100	100	61.37	100 / 100
	ALMA ℓ_1 1 000	100	40.41	1 000 / 1 000
RN-50 ℓ_∞-AT	EAD 9×100 (Chen *et al.*, 2018b)	100	6.40	582 / 295
	EAD 9×1 000 (Chen *et al.*, 2018b)	100	6.29	3 410 / 1 709
	FAB^T ℓ_1 100 (Croce & Hein, 2020a)	95.6	4.36	1 810 / 900
	FAB^T ℓ_1 1 000 (Croce & Hein, 2020a)	93.6	4.33	18 010 / 9 000
	FMN ℓ_1 100 (Pintor *et al.*, 2021)	87.8	4.16	100 / 100
	FMN ℓ_1 1 000 (Pintor *et al.*, 2021)	87.7	4.16	1 000 / 1 000
	ALMA ℓ_1 100	100	14.90	100 / 100
	ALMA ℓ_1 1 000	100	10.33	1 000 / 1 000

Table-A III-10 Performance of the ℓ_2 attacks on the ImageNet dataset for each model

Model	Attack	ASR (%)	Median ℓ_2	Forwards / Backwards
RN-50	C&W ℓ_2 9×1 000 (Carlini & Wagner, 2017)	100	0.21	8 775 / 8 775
	C&W ℓ_2 9×10 000 (Carlini & Wagner, 2017)	100	0.21	82 668 / 82 667
	DDN 100 (Rony *et al.*, 2019)	99.8	0.18	100 / 100
	DDN 1 000 (Rony *et al.*, 2019)	99.9	0.17	1 000 / 1 000
	FAB^T ℓ_2 100 (Croce & Hein, 2020a)	99.3	0.10	1 810 / 900
	FAB^T ℓ_2 1 000 (Croce & Hein, 2020a)	98.0	0.10	18 010 / 9 000
	FMN ℓ_2 100 (Pintor *et al.*, 2021)	98.9	0.12	100 / 100
	FMN ℓ_2 1 000 (Pintor *et al.*, 2021)	99.3	0.10	1 000 / 1 000
	$APGD^T_{DLR}$ ℓ_2 (Croce & Hein, 2020b)	100	0.09	4 866 4 838
	ALMA ℓ_2 100	100	0.10	100 / 100
	ALMA ℓ_2 1 000	100	0.10	1 000 / 1 000
RN-50 ℓ_2-AT	C&W ℓ_2 9×1 000 (Carlini & Wagner, 2017)	99.9	1.17	6 260 / 6 256
	C&W ℓ_2 9×10 000 (Carlini & Wagner, 2017)	99.9	1.17	57 004 / 52 000
	DDN 100 (Rony *et al.*, 2019)	99.5	1.09	100 / 100
	DDN 1 000 (Rony *et al.*, 2019)	99.7	1.10	1 000 / 1 000
	FAB^T ℓ_2 100 (Croce & Hein, 2020a)	100	0.81	1 810 / 900
	FAB^T ℓ_2 1 000 (Croce & Hein, 2020a)	99.3	0.81	18 010 / 9 000
	FMN ℓ_2 100 (Pintor *et al.*, 2021)	99.6	0.84	100 / 100
	FMN ℓ_2 1 000 (Pintor *et al.*, 2021)	99.9	0.82	1 000 / 1 000
	$APGD^T_{DLR}$ ℓ_2 (Croce & Hein, 2020b)	100	0.80	7 005 / 6 977
	ALMA ℓ_2 100	100	0.85	100 / 100
	ALMA ℓ_2 1 000	100	0.84	1 000 / 1 000
RN-50 ℓ_∞-AT	C&W ℓ_2 9×1 000 (Carlini & Wagner, 2017)	99.6	0.76	6 933 / 6 930
	C&W ℓ_2 9×10 000 (Carlini & Wagner, 2017)	99.6	0.76	65 203 / 65 200
	DDN 100 (Rony *et al.*, 2019)	99.8	0.67	100 / 100
	DDN 1 000 (Rony *et al.*, 2019)	100	0.66	1 000 / 1 000
	FAB^T ℓ_2 100 (Croce & Hein, 2020a)	99.8	0.55	1 810 / 900
	FAB^T ℓ_2 1 000 (Croce & Hein, 2020a)	99.4	0.55	18 010 / 9 000
	FMN ℓ_2 100 (Pintor *et al.*, 2021)	99.8	0.57	100 / 100
	FMN ℓ_2 1 000 (Pintor *et al.*, 2021)	99.7	0.57	1 000 / 1 000
	$APGD^T_{DLR}$ ℓ_2 (Croce & Hein, 2020b)	100	0.54	6 647 / 6 619
	ALMA ℓ_2 100	100	0.62	100 / 100
	ALMA ℓ_2 1 000	100	0.54	1 000 / 1 000

Table-A III-11 Performance of the CIEDE2000 attacks on the CIFAR10 dataset for each model

Model	Attack	ASR (%)	Median CIEDE2000	Forwards / Backwards
RN-50	C&W CIEDE2000 9×1 000	100	0.80	4 505 / 4 500
	Perc-AL 100 (Zhao *et al.*, 2020)	100	1.31	2 01 / 100
	Perc-AL 1 000 (Zhao *et al.*, 2020)	100	1.07	2 001 / 1 000
	ALMA CIEDE2000 100	100	0.17	100 / 100
	ALMA CIEDE2000 1 000	100	0.13	1 000 / 1 000
RN-50 ℓ_2-AT	C&W CIEDE2000 9×1 000	100	1.64	6 303 / 6 300
	Perc-AL 100 (Zhao *et al.*, 2020)	99.9	5.87	201 / 100
	Perc-AL 1 000 (Zhao *et al.*, 2020)	99.9	6.07	2 001 / 1 000
	ALMA CIEDE2000 100	100	1.51	100 / 100
	ALMA CIEDE2000 1 000	100	1.34	1 000 / 1 000
RN-50 ℓ_∞-AT	C&W CIEDE2000 9×1 000	100	2.05	6 303 / 6 300
	Perc-AL 100 (Zhao *et al.*, 2020)	99.8	5.82	201 / 100
	Perc-AL 1 000 (Zhao *et al.*, 2020)	99.9	6.12	2 001 / 1 000
	ALMA CIEDE2000 100	100	1.61	100 / 100
	ALMA CIEDE2000 1 000	100	1.46	1 000 / 1 000

Table-A III-12 Performance of the LPIPS variant of ALMA on the ImageNet dataset for each model. ‡A binary search is performed on each sample to get a minimal perturbation attack (Equation 4.2)

Model	Attack	ASR (%)	Median LPIPS $\times 10^{-2}$	Forwards / Backwards
RN-50	C&W LPIPS 9×1 000	100	2.39	2 332 / 2 325
	LPA‡ (Laidlaw *et al.*, 2021)	100	5.02	1 159 / 1 149
	ALMA LPIPS 100	100	0.34	100 / 100
	ALMA LPIPS 1 000	100	0.24	1 000 / 1 000
RN-50 ℓ_2-AT	C&W LPIPS 9×1 000	100	2.22	7 701 / 7 700
	LPA‡ (Laidlaw *et al.*, 2021)	100	6.83	1 257 / 1 247
	ALMA LPIPS 100	100	3.96	100 / 100
	ALMA LPIPS 1 000	100	2.90	1 000 / 1 000
RN-50 ℓ_∞-AT	C&W LPIPS 9×1 000	100	1.68	6 753 / 6 750
	LPA‡ (Laidlaw *et al.*, 2021)	100	5.66	1 218 / 1 208
	ALMA LPIPS 100	100	2.99	100 / 100
	ALMA LPIPS 1 000	100	2.11	1 000 / 1 000

Table-A III-13 Performance of the SSIM variant of ALMA on the ImageNet dataset for each model

Model	Attack	ASR (%)	Median $1-\text{SSIM} \times 10^{-5}$	Forwards / Backwards
RN-50	ALMA SSIM 100	100	0.44	100 / 100
	ALMA SSIM 1 000	100	0.05	1 000 / 1 000
RN-50 ℓ_2-AT	ALMA SSIM 100	100	16.13	100 / 100
	ALMA SSIM 1 000	100	5.58	1 000 / 1 000
RN-50 ℓ_∞-AT	ALMA SSIM 100	100	8.69	100 / 100
	ALMA SSIM 1 000	100	2.77	1 000 / 1 000

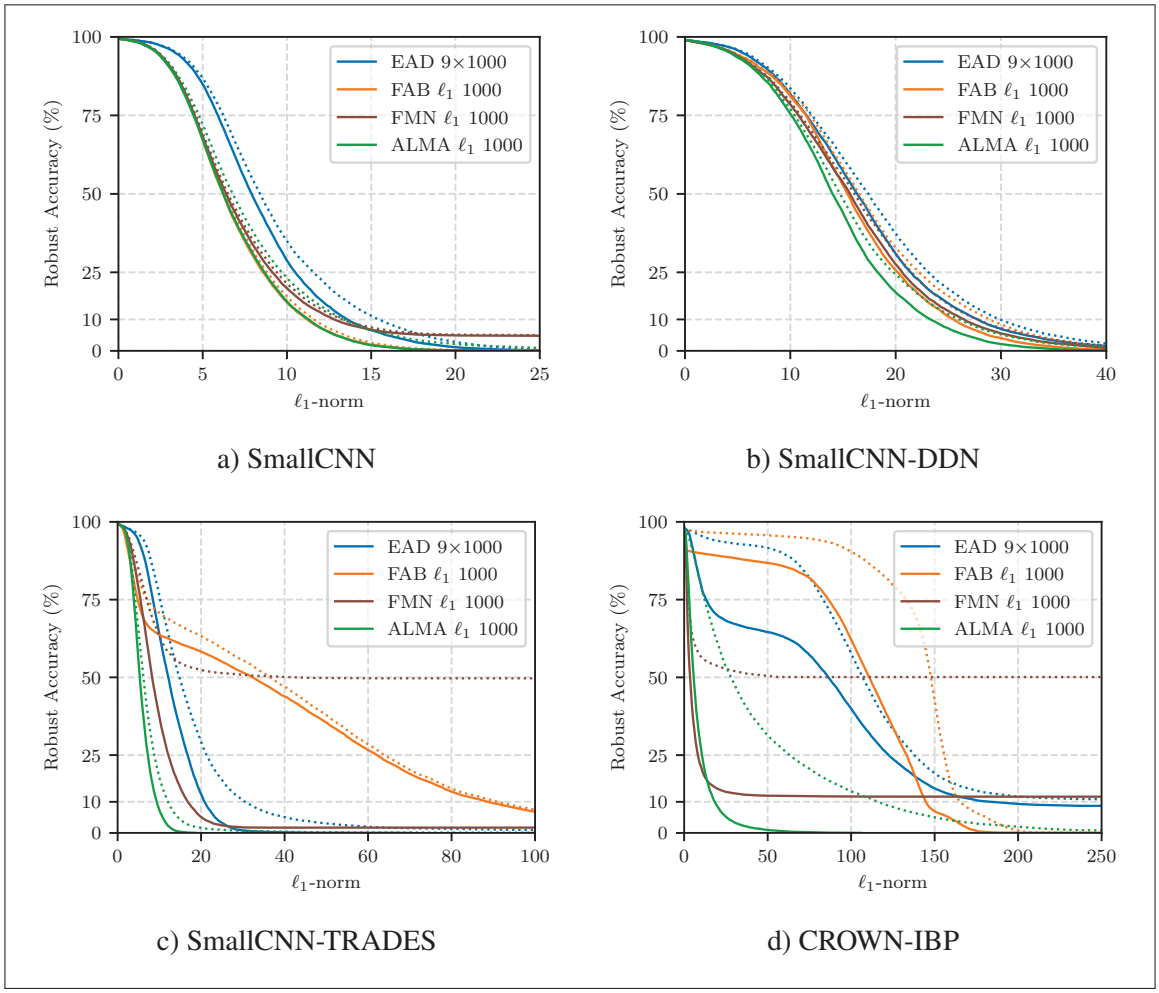


Figure-A III-1 Robust accuracy curves for MNIST models against ℓ_1 attacks

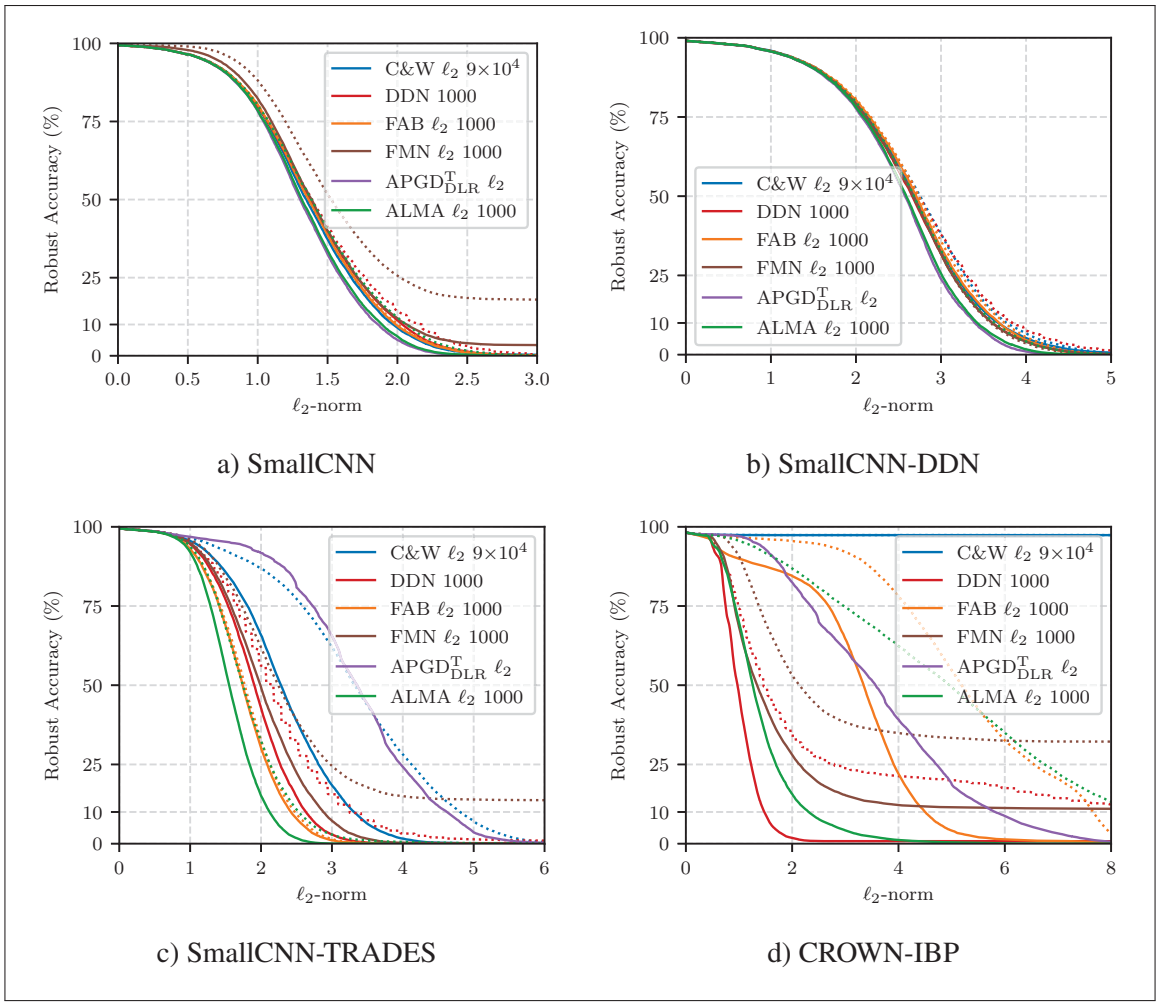
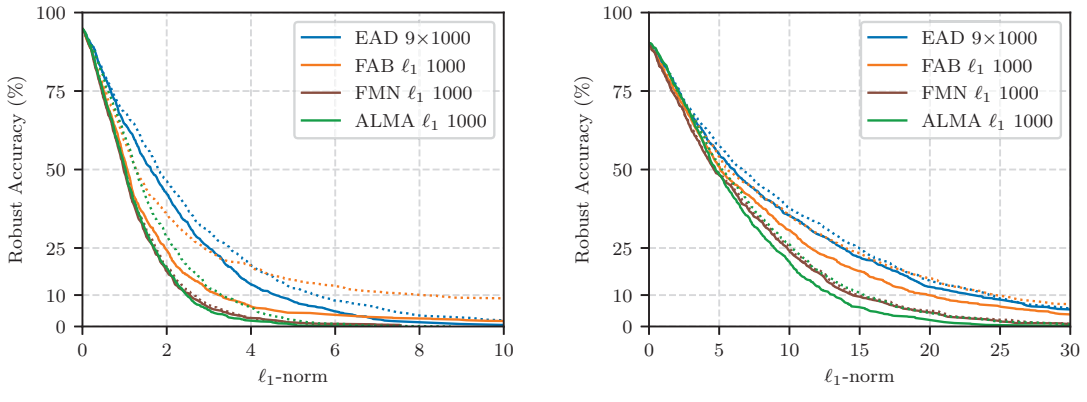
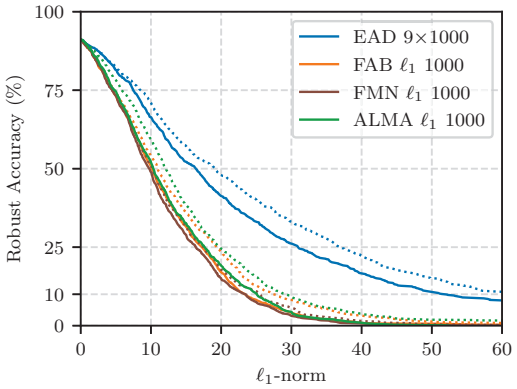


Figure-A III-2 Robust accuracy curves for MNIST models against ℓ_2 attacks



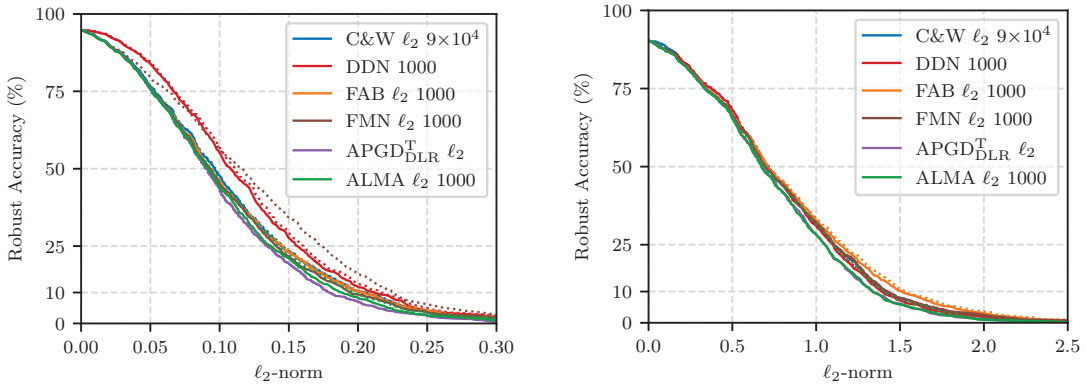
a) Wide ResNet 28-10

b) Wide ResNet 28-10 (Carmon *et al.*, 2019)



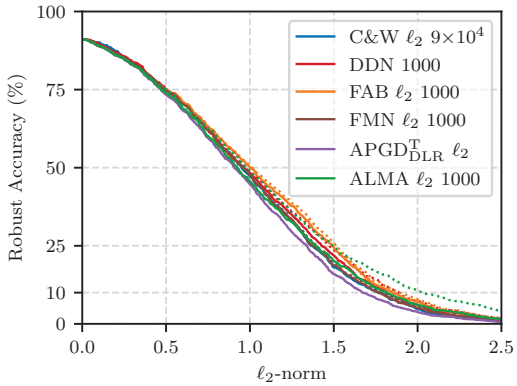
c) ResNet-50 (Augustin *et al.*, 2020)

Figure-A III-3 Robust accuracy curves for CIFAR10 models against ℓ_1 attacks



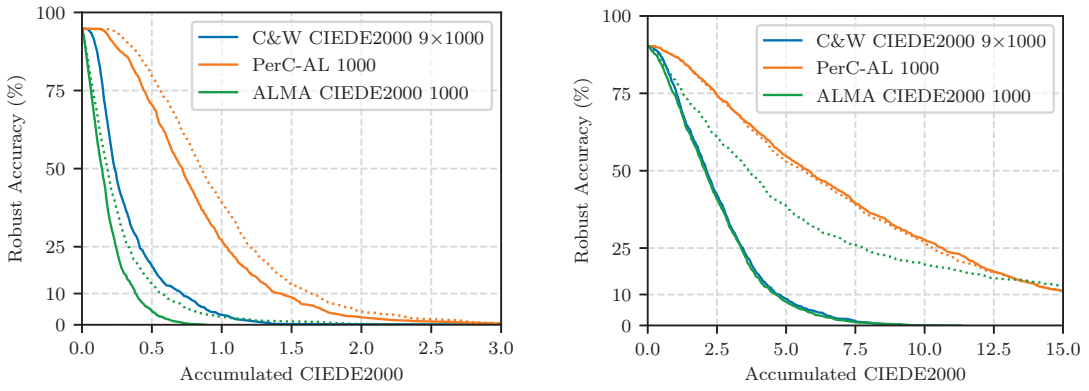
a) Wide ResNet 28-10

b) Wide ResNet 28-10 (Carmon *et al.*, 2019)



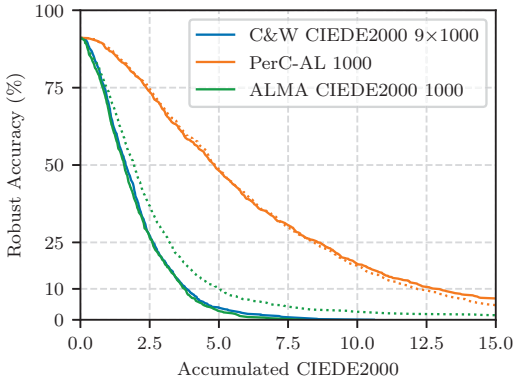
c) ResNet-50 (Augustin *et al.*, 2020)

Figure-A III-4 Robust accuracy curves for CIFAR10 models against ℓ_2 attacks



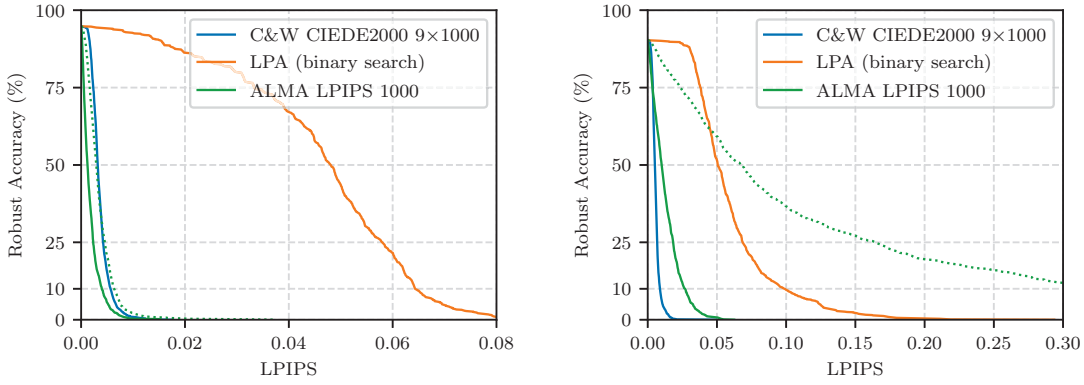
a) Wide ResNet 28-10

b) Wide ResNet 28-10 (Carmon *et al.*, 2019)



c) ResNet-50 (Augustin *et al.*, 2020)

Figure-A III-5 Robust accuracy curves for CIFAR10 models against CIEDE2000 attacks



a) Wide ResNet 28-10

b) Wide ResNet 28-10(Carmon *et al.*, 2019)

c) ResNet-50(Augustin *et al.*, 2020)

Figure-A III-6 Robust accuracy curves for CIFAR10 models against LPIPS attacks

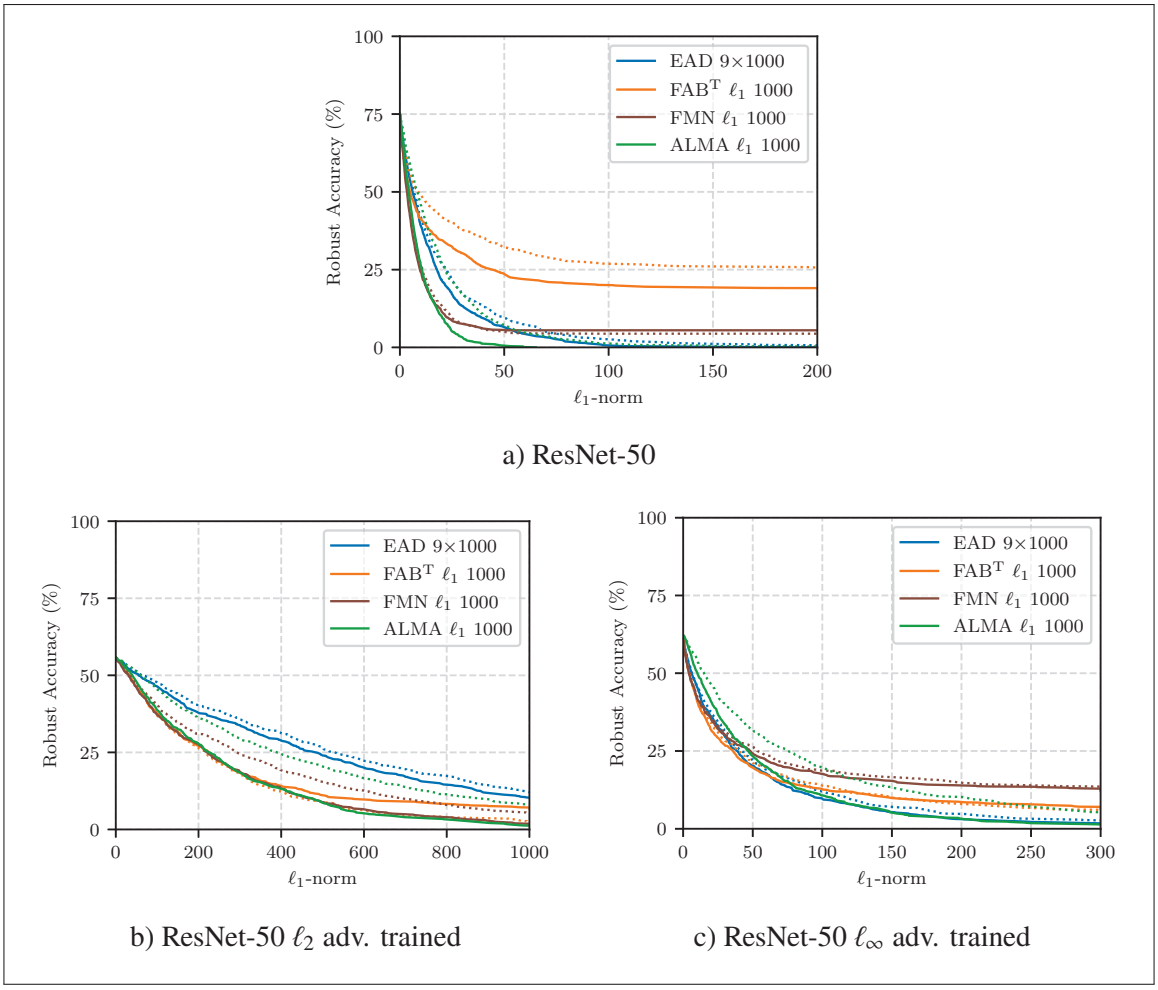
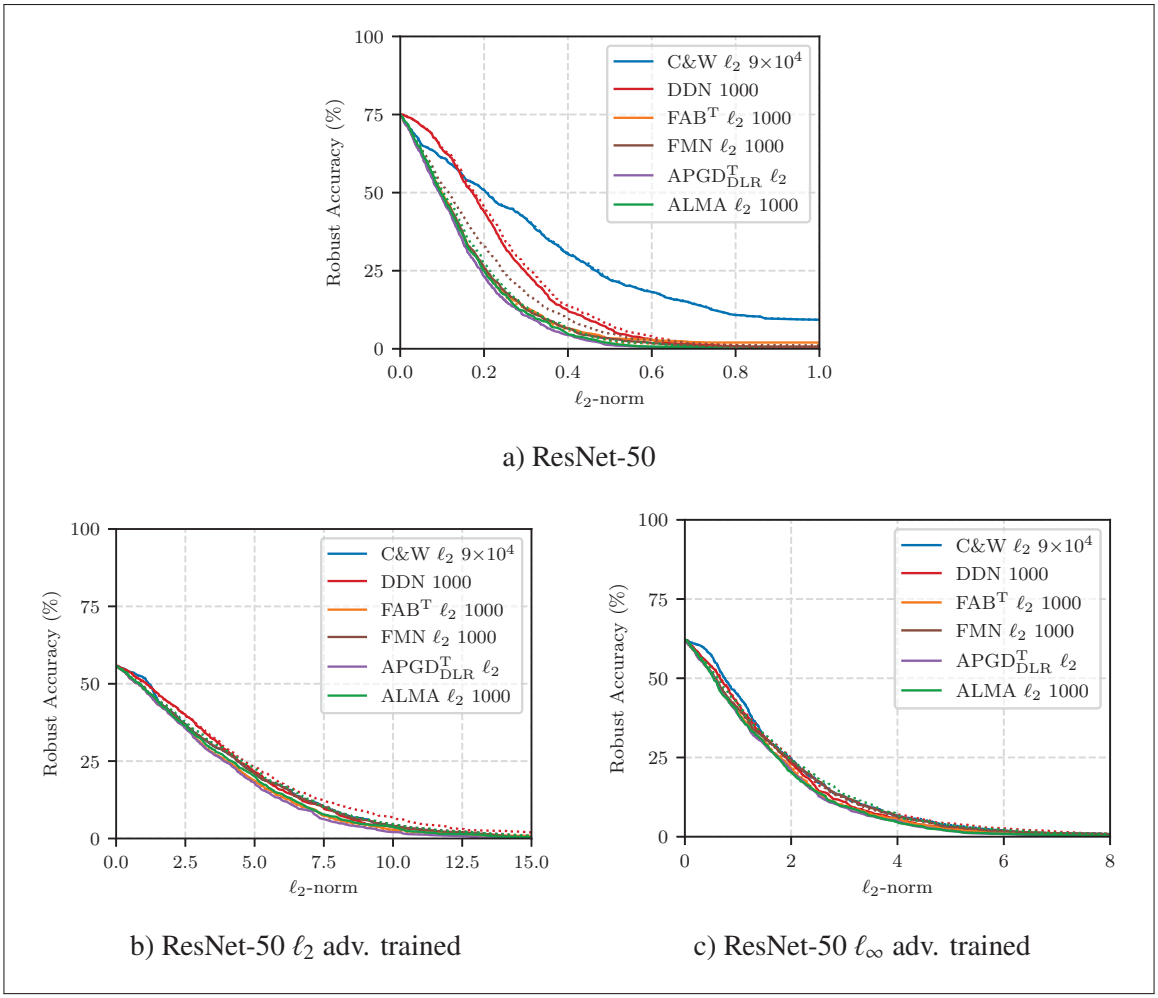


Figure-A III-7 Robust accuracy curves for ImageNet models against ℓ_1 attacks



a) ResNet-50

b) ResNet-50 ℓ_2 adv. trained

c) ResNet-50 ℓ_∞ adv. trained

Figure-A III-8 Robust accuracy curves for ImageNet models against ℓ_2 attacks

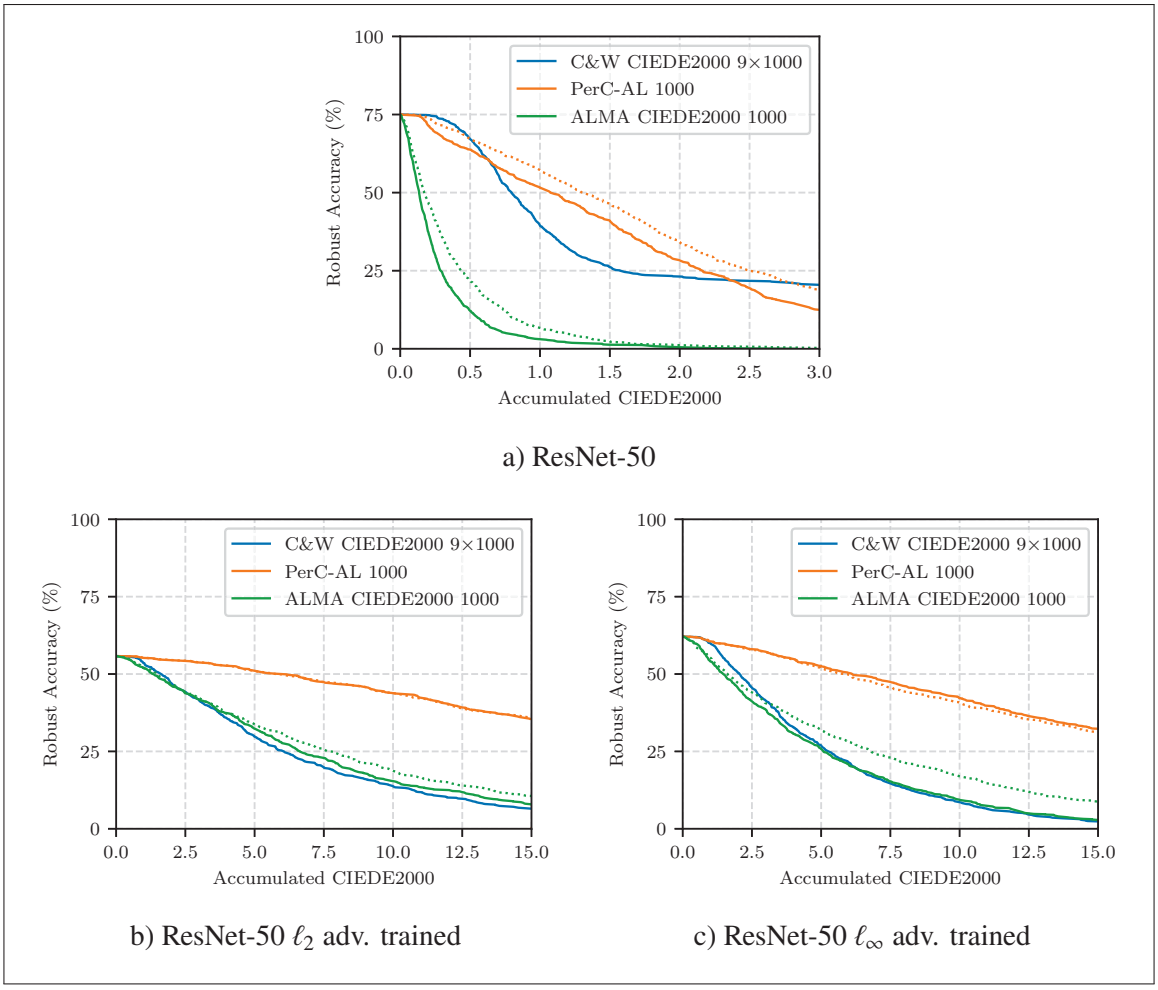
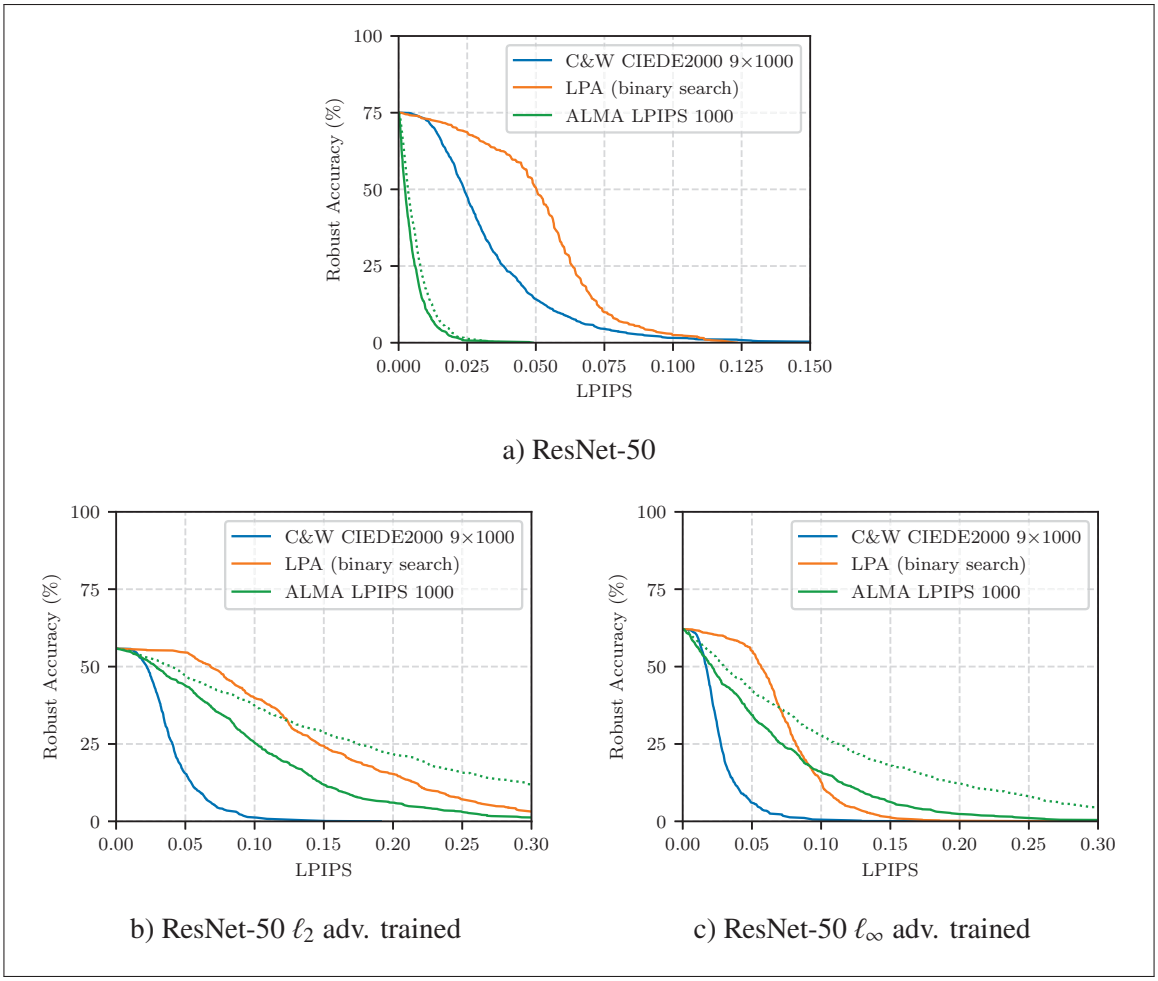


Figure-A III-9 Robust accuracy curves for ImageNet models against CIEDE2000 attacks



a) ResNet-50

b) ResNet-50 ℓ_2 adv. trained

c) ResNet-50 ℓ_∞ adv. trained

Figure-A III-10 Robust accuracy curves for ImageNet models against LPIPS attacks

APPENDIX IV

SUPPLEMENTARY MATERIAL FOR THE PAPER TITLED *PROXIMAL SPLITTING ADVERSARIAL ATTACK FOR SEMANTIC SEGMENTATION*

1. Dense Adversary Generation attack

Algorithm-A IV-1 DAG attack

Input: Classifier f, original image $\boldsymbol{x} \in [0,1]^{C\times H\times W}$, true or target label $\boldsymbol{y} \in \mathbb{N}^{H\times W}$, binary mask $\boldsymbol{m} \in \{0,1\}^{H\times W}$

Input: Step size η, maximum number of iterations N, threshold of pixel success rate ν

1 Initialize $\boldsymbol{\delta}^{(0)} \leftarrow \mathbf{0}$

2 If targeted attack: $\mu \leftarrow -1$ else $\mu \leftarrow 1$

3 **for** $t \leftarrow 1, \ldots, N$ **do**

4 $\quad \tilde{\boldsymbol{x}}^{(t)} \leftarrow \mathcal{P}_{[0,1]}(\boldsymbol{x} + \boldsymbol{\delta}^{(t-1)})$; // $\in [0,1]^{C\times H\times W}$

5 $\quad \boldsymbol{z} \leftarrow f(\tilde{\boldsymbol{x}}^{(t)})$; // $\in \mathbb{R}^{K\times H\times W}$

6 $\quad$ **for** $i \leftarrow 1, \ldots, d$ **do**

7 $\quad\quad \Delta z_i = \mu(z_{\boldsymbol{y}_i,i} - \max_{j\neq \boldsymbol{y}_i} z_{j,i})$; // Difference of logits

8 $\quad$ **end for**

9 $\quad r \leftarrow \frac{\boldsymbol{m}^\top[\Delta \boldsymbol{z}<0]}{\|\boldsymbol{m}\|_1}$; // Pixel success rate

10 $\quad$ **if** $r \geq \nu$ **then**

11 $\quad\quad$ **return** $\tilde{\boldsymbol{x}}^{(t)}$; // Stop the attack

12 $\quad \mathcal{L} \leftarrow \boldsymbol{m}^\top \max\{0, \Delta \boldsymbol{z}_i\}$

13 $\quad \boldsymbol{g} \leftarrow \nabla_{\boldsymbol{\delta}} \mathcal{L}$

14 $\quad \boldsymbol{\delta}^{(t)} \leftarrow \boldsymbol{\delta}^{(t-1)} - \frac{\eta}{\|\boldsymbol{g}\|_\infty} \boldsymbol{g}$; // Normalized gradient step

15 **end for**

In this section, we provide the algorithm of the Dense Adversary Generation (DAG) attack from (Xie *et al.*, 2017). The main difference with the original algorithm proposed in (Xie *et al.*, 2017) is the stopping criterion based on the pixel success rate in steps 9 to 11. In the original method published in (Xie *et al.*, 2017), the attack is supposed to stop once all the constraints are satisfied (*i.e.* all pixels in the mask $\boldsymbol{m}$ are adversarial). However, since this criterion is rarely satisfied, even after hundreds of iterations on a dataset like Cityscapes, the actual implementation made available in https://github.com/cihangxie/DAG uses the stopping criterion described in

Algorithm IV-1: the attack stops once a threshold of pixel success rate set to 99% is reached. The threshold value used is thus identical to the one used in the experiments of this paper.

2. Proof of Proposition 1

Proof. Problem (5.12) amounts to minimizing function

$$\begin{aligned}\Phi\colon (\boldsymbol{p},\beta) \mapsto \frac{1}{2}\left\|\boldsymbol{p}-\boldsymbol{\delta}\right\|_2^2 + \lambda\beta \; &+ \iota_{[0,+\infty[^{Cd}}(\beta\mathbf{1}_{Cd}-\boldsymbol{p}) \\ &+ \iota_{[0,+\infty[^{Cd}}(\boldsymbol{p}+\beta\mathbf{1}_{Cd}) + \iota_{\Lambda}(\boldsymbol{p}),\end{aligned} \tag{A IV-1}$$

defined on $\mathbb{R}^{Cd}\times\mathbb{R}$. This function is convex since it is a sum of elementary convex functions of $(\boldsymbol{p},\beta)$. It follows that its marginal function

$$\underline{\Phi}\colon \beta \mapsto \inf_{\boldsymbol{p}\in\mathbb{R}^{Cd}} \Phi(\boldsymbol{p},\beta) \tag{A IV-2}$$

is convex. Since, for any given $\beta\in[0,1]$, $\boldsymbol{p}\mapsto\Phi(\boldsymbol{p},\beta)$ is strongly convex and proper, it admits a unique minimizer $\boldsymbol{p}_\beta$. More precisely, we have, for every $\beta\in\mathbb{R}$,

$$\underline{\Phi}(\beta) = \begin{cases} \frac{1}{2}\left\|\boldsymbol{p}_\beta-\boldsymbol{\delta}\right\|_2^2 + \lambda\beta & \text{if } \beta\in[0,1] \\ +\infty & \text{otherwise.}\end{cases} \tag{A IV-3}$$

The above function admits a unique minimizer $\beta^\star\in[0,1]$ since $\boldsymbol{p}^\star=\boldsymbol{p}_{\beta^\star}$ is uniquely defined (as it is the proximity point of a proper lower-semicontinuous convex function) and $\beta^\star=\|\boldsymbol{p}_{\beta^\star}\|_\infty$.

Let $\boldsymbol{p}^\star=\operatorname{prox}_{\lambda\|\cdot\|_\infty+\iota_\Lambda}(\boldsymbol{\delta})$ be the minimizer of (5.12) and let $\boldsymbol{\delta}_\Lambda=\mathcal{P}_\Lambda(\boldsymbol{\delta})$. We have

$$\frac{1}{2}\left\|\boldsymbol{p}^\star-\boldsymbol{\delta}\right\|_2^2 + \lambda\left\|\boldsymbol{p}^\star\right\|_\infty \le \frac{1}{2}\left\|\boldsymbol{\delta}_\Lambda-\boldsymbol{\delta}\right\|_2^2 + \lambda\left\|\boldsymbol{\delta}_\Lambda\right\|_\infty \tag{A IV-4}$$

Since $\boldsymbol{\delta}_\Lambda=\mathcal{P}_\Lambda(\boldsymbol{\delta})=\underset{\boldsymbol{y}\in\Lambda}{\arg\min}\,\|\boldsymbol{y}-\boldsymbol{\delta}\|_2^2$, we also have

$$\left\|\boldsymbol{\delta}_\Lambda-\boldsymbol{\delta}\right\|_2^2 \le \left\|\boldsymbol{p}^\star-\boldsymbol{\delta}\right\|_2^2 \tag{A IV-5}$$

Thus

$$\begin{aligned}\lambda \left\|\boldsymbol{p}^\star\right\|_\infty &\leq \underbrace{\frac{1}{2}(\|\boldsymbol{\delta}_\Lambda - \boldsymbol{\delta}\|_2^2 - \left\|\boldsymbol{p}^\star - \boldsymbol{\delta}\right\|_2^2)}_{\leq 0} + \lambda \|\boldsymbol{\delta}_\Lambda\|_\infty \\ &\leq \lambda \|\boldsymbol{\delta}_\Lambda\|_\infty\end{aligned} \tag{A IV-6}$$

We deduce that $\boxed{\beta^\star = \left\|\boldsymbol{p}^\star\right\|_\infty \leq \|\boldsymbol{\delta}_\Lambda\|_\infty}$. □

3. ALMA prox attack algorithm

Algorithm IV-2 details the complete algorithm of the attack.

4. Image size and performance of the models

For all models, except DeepLabV3 DDC-AT, the images of Pascal VOC 2012 are resized so that the smaller side is of length 512 while keeping the aspect ratio, and for Cityscapes, the images keep their original size of 2 048×1 024. For DeepLabV3 DDC-AT from (Xu *et al.*, 2021), the images of Pascal VOC 2012 are resized so that the longer side is of length 512 while keeping the aspect ratio, and for Cityscapes, the images are resized to 1 024 × 512.

Table-A IV-1 Performance of the models used in the experiments on the validation sets. Numbers were obtained from our evaluation; subscripts correspond to the difference with the original evaluation protocol. For DeepLabV3 DDC-AT from (Xu *et al.*, 2021), the pixel accuracy was not reported

Dataset	Model	mIoU (%)	Pixel Accuracy (%)
Pascal VOC 2012 (+Aug)	DeepLabV3+ ResNet-50	$77.4_{+0.8}$	$94.9_{+0.2}$
	DeepLabV3+ ResNet-101	$78.8_{+0.1}$	$95.3_{+0.1}$
	FCN HRNetV2 W48	$76.4_{+0.2}$	$94.7_{+0.1}$
	DeepLabV3 DDC-AT	$75.2_{+0.0}$	94.4
Cityscapes	DeepLabV3+ ResNet-50	$80.1_{-0.2}$	$96.4_{-0.1}$
	FCN HRNetV2 W48	$80.5_{-0.2}$	$96.6_{-0.1}$
	SegFormer MiT-B0	$76.4_{-0.1}$	$95.9_{-0.0}$
	SegFormer MiT-B3	$81.8_{-0.0}$	$96.7_{-0.1}$
	DeepLabV3 DDC-AT	$71.0_{-0.3}$	95.0

5. Cityscapes target label

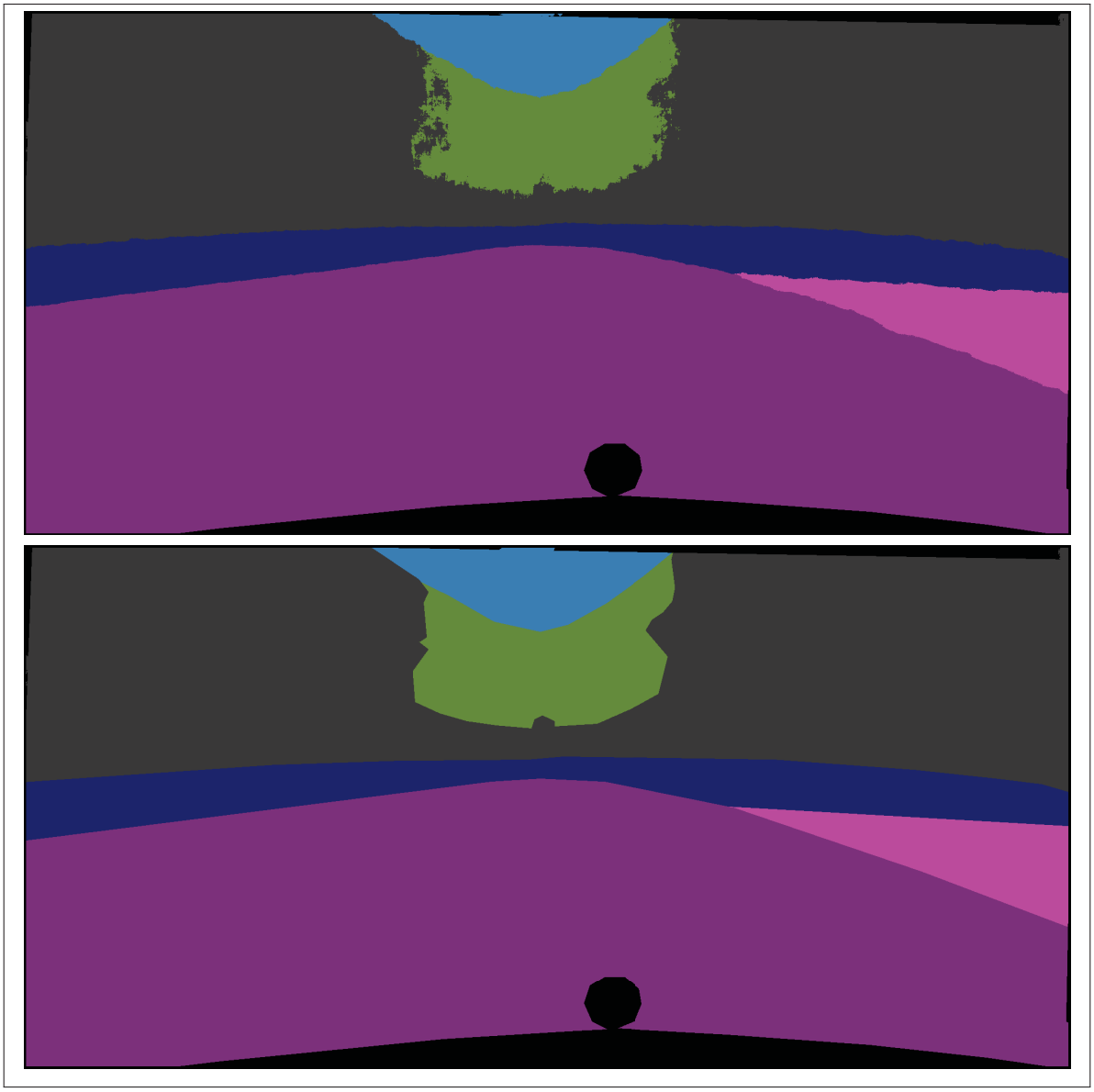


Figure-A IV-1 Cityscapes target segmentation used in our experiments. Top is the pixel majority label, bottom is the smoothed version. Classes in the target are: *road* (purple), *sidewalk* (pink), *car* (blue), *building* (dark gray), *vegetation* (green), *sky* (cyan) and *no label* (black)

Algorithm-A IV-2 ALMA prox attack (untargeted)

Input: Classifier f, original image $\boldsymbol{x} \in [0,1]^{C\times H\times W}$, true or target label $\boldsymbol{y} \in \mathbb{N}^{H\times W}$, binary mask $\boldsymbol{m} \in \{0,1\}^{H\times W}$
Input: Threshold of pixel success rate ν
Input: Penalty function P, initial multiplier $\boldsymbol{\mu}^{(0)} \in \mathbb{R}_{++}^{H\times W}$, initial penalty parameter $\boldsymbol{\rho}^{(0)} \in \mathbb{R}_{++}^{H\times W}$
Input: Minimum scale $w_{\min}$, scale adjustment rate $\gamma_w > 0$
Input: Number of iterations N, initial step size $\lambda^{(0)}$, penalty parameter increase rate $\gamma > 1$, constraint improvement rate τ, M number of steps between ρ increase, α smoothing parameter

Initialize $\boldsymbol{\delta}^{(0)} \leftarrow \mathbf{0}$, $\boldsymbol{v}^{(0)} \leftarrow 0$, $w^{(0)} \leftarrow 1$
for $t \leftarrow 1, \dots, N$ **do**
  $\tilde{\boldsymbol{x}}^{(t)} \leftarrow \boldsymbol{x} + \boldsymbol{\delta}^{(t-1)}$; // $\in [0,1]^{C\times H\times W}$
  $\boldsymbol{d}^{(t)} \leftarrow \mathrm{DLR}^{+}(f(\tilde{\boldsymbol{x}}^{(t)}), \boldsymbol{y})$; // $\in \mathbb{R}^{H\times W}$
  if $\frac{\boldsymbol{m}^\top[\boldsymbol{d}^{(t)} \le 0]}{\|\boldsymbol{m}\|_1} < \tau$ **then**
    $\hat{w} \leftarrow \frac{w^{(t-1)}}{1-\gamma_w}$; // Increase scale
  else
    $\hat{w} \leftarrow \frac{w^{(t-1)}}{1+\gamma_w}$; // Decrease scale
  $w^{(t)} \leftarrow \mathcal{P}_{[w_{\min},1]}(\hat{w})$
  $\xi^{(t)} \leftarrow (1-(1-\nu)\frac{t-1}{N-1})$-percentile of $\boldsymbol{d}^{(t)}$
  $\tilde{\boldsymbol{m}}^{(t)} \leftarrow [\boldsymbol{d}^{(t)} \le \xi^{(t)}]$; // $\in \{0,1\}^{H\times W}$
  $\hat{\boldsymbol{\mu}} \leftarrow \nabla_{\boldsymbol{d}} \left((\tilde{\boldsymbol{m}}^{(t)})^\top P(w^{(t)}\boldsymbol{d}^{(t)}, \boldsymbol{\rho}^{(t-1)}, \boldsymbol{\mu}^{(t-1)}) \right)$
  $\boldsymbol{\mu}^{(t)} \leftarrow \mathcal{P}_{[\mu_{\min},\mu_{\max}]}\left(\alpha\boldsymbol{\mu}^{(t-1)} + (1-\alpha)\hat{\boldsymbol{\mu}}\right)$; // $\in \mathbb{R}_{++}^{H\times W}$
  for $i \leftarrow 1, \dots, d$ **do**
    if $t \bmod M = 0$ ***and*** $\tilde{\boldsymbol{m}}_i^{(t)} = 1$ ***and*** $\left(\exists j \in \{0,\dots,M-1\} : \boldsymbol{d}_i^{(t-j)} \le 0$ ***or*** $\boldsymbol{d}_i^{(t)} \le \tau \boldsymbol{d}_i^{(t-M)}\right)$ **then**
      $\boldsymbol{\rho}_i^{(t)} \leftarrow \boldsymbol{\rho}_i^{(t-1)}$; // Constraint improved or satisfied
    else
      $\boldsymbol{\rho}_i^{(t)} \leftarrow \gamma\boldsymbol{\rho}_i^{(t-1)}$
  end for
  $\mathcal{L} \leftarrow (\tilde{\boldsymbol{m}}^{(t)})^\top P(w^{(t)}\boldsymbol{d}^{(t)}, \boldsymbol{\rho}^{(t)}, \boldsymbol{\mu}^{(t)})$; // $\in \mathbb{R}$
  $\boldsymbol{g}^{(t)} \leftarrow \nabla_{\boldsymbol{\delta}}\mathcal{L}$; // $\in \mathbb{R}^{C\times H\times W}$
  $\boldsymbol{v}^{(t)} \leftarrow \alpha\boldsymbol{v}^{(t-1)} + (1-\alpha)(\boldsymbol{g}^{(t)})^2$; // $\in \mathbb{R}_{+}^{C\times H\times W}$
  $\mathsf{H} \leftarrow \mathrm{Diag}\left(\sqrt{\frac{\boldsymbol{v}^{(t)}}{1-\alpha^t}} + \varepsilon\right)$
  $\boldsymbol{\delta}^{(t)} \leftarrow \mathrm{prox}^{\mathsf{H}}_{\lambda^{(t)}\|\cdot\|_\infty + \iota_\Lambda}(\boldsymbol{\delta}^{(t-1)} - \lambda^{(t)}\mathsf{H}^{-1}\boldsymbol{g})$; // VMFB
end for
return $\tilde{\boldsymbol{x}}^{(t)}$ *that is adversarial and has the smallest norm*

6. Masking strategies for ALMA prox and PDPGD

In this section, we detail the modifications brought to PDPGD (Matyasko & Chau, 2021) and study the effect of the masking strategies on PDPGD and ALMA prox. Besides the addition of perturbation tracking logic (*i.e.* monitoring the best smallest perturbations during the optimization), we also incorporate the masking needed for the unlabeled regions.

6.1 Modifications for PDPGD

For PDPGD, the dual variable $\lambda \in \mathbb{R}$ is replaced by a vector $\boldsymbol{\lambda} \in \mathbb{R}^d$. In (Matyasko & Chau, 2021), the gradient ascent on the dual variable is performed in the log domain, and is projected onto the 1-simplex (section IV of (Matyasko & Chau, 2021)). For the segmentation variant, this translates into adding 1 to the denominator of the softmax function used to project onto the $(d-1)$-simplex. This is equivalent to padding $\boldsymbol{\lambda}$ with a 0 and projecting it on the d-simplex $\Delta^d \subset [0,1]^{d+1}$. The weight of the norm $\|\boldsymbol{\delta}\|$ in Equation (10) of (Matyasko & Chau, 2021) becomes 1 minus the sum of the weights of the constraints. Adding the mask $\boldsymbol{m}$, this results in the following computations (introducing variables not described in (Matyasko & Chau, 2021)):

$$\begin{aligned} \boldsymbol{\lambda}_\Delta &= \frac{\boldsymbol{m} \odot \exp \boldsymbol{\lambda}}{1 + \boldsymbol{m}^\top \exp \boldsymbol{\lambda}} \quad \in [0,1]^d \\ \mathbb{L}_{\boldsymbol{\delta}}(\boldsymbol{\delta}, \boldsymbol{\lambda}) &= (1 - \boldsymbol{m}^\top \boldsymbol{\lambda}_\Delta) \, \|\boldsymbol{\delta}\| + \boldsymbol{\lambda}_\Delta^\top \mathcal{L}(\boldsymbol{x} + \boldsymbol{\delta}, \boldsymbol{y}). \end{aligned} \tag{A IV-7}$$

By replacing $\boldsymbol{m}$ by $\tilde{\boldsymbol{m}}^{(t)}$ in each iteration, we obtain the adaptive constraint masking described in subsection 5.4.1.

The step-size is set to 0.01 for the primal variables and 0.1 for the dual variables with the same exponential and linear decays respectively, as in (Matyasko & Chau, 2021). The dual variable is initialized such that the ratio of the weight of the norm term and the constraints terms is 1. From

the above equations, we can derive the initial $\boldsymbol{\lambda}^{(0)} \in \mathbb{R}^d$ from the ratio $r \in \mathbb{R}_{++}$:

$$\begin{aligned}
\frac{1 - \boldsymbol{m}^\top \boldsymbol{\lambda}_\Delta^{(0)}}{\boldsymbol{m}^\top \boldsymbol{\lambda}_\Delta^{(0)}} = r &\Leftrightarrow 1 - \boldsymbol{m}^\top \boldsymbol{\lambda}_\Delta^{(0)} = r \boldsymbol{m}^\top \boldsymbol{\lambda}_\Delta^{(0)} \\
&\Leftrightarrow (1 + r) \boldsymbol{m}^\top \boldsymbol{\lambda}_\Delta^{(0)} = 1 \\
&\Leftrightarrow (1 + r) \frac{\boldsymbol{m}^\top \exp \boldsymbol{\lambda}^{(0)}}{1 + \boldsymbol{m}^\top \exp \boldsymbol{\lambda}^{(0)}} = 1 \\
&\Leftrightarrow (1 + r) \boldsymbol{m}^\top \exp \boldsymbol{\lambda}^{(0)} = \\
&\qquad\qquad 1 + \boldsymbol{m}^\top \exp \boldsymbol{\lambda}^{(0)} \\
&\Leftrightarrow \boldsymbol{m}^\top \exp \boldsymbol{\lambda}^{(0)} = \frac{1}{r}
\end{aligned} \tag{A IV-8}$$

Assuming that $\boldsymbol{\lambda}^{(0)} = \omega \mathbf{1}_d$ with $\omega \in \mathbb{R}$, we get:

$$\begin{aligned}
\boldsymbol{m}^\top \exp \boldsymbol{\lambda}^{(0)} = \frac{1}{r} &\Leftrightarrow \|\boldsymbol{m}\|_1 \exp \omega = \frac{1}{r} \\
&\Leftrightarrow \omega = -\log(r \|\boldsymbol{m}\|_1)
\end{aligned} \tag{A IV-9}$$

6.2 Ablation

To test the effect of these modifications on PDPGD and ALMA prox, we perform an ablation study on the masking strategy. We perform this experiment with DeepLabV3+ ResNet-50 on Cityscapes. We compare three different constraint masking strategies:

- masking only the unlabeled regions, denoted by $\boldsymbol{m}$;
- masking the unlabeled regions and $(100 - \nu)\%$ of the largest constraints, denoted by $\tilde{\boldsymbol{m}}$;
- masking the unlabeled regions and linearly decreasing the fraction of constraints to reach $\nu\%$ at the last iteration, corresponding to strategy described in Equation (5.6), denoted by $\tilde{\boldsymbol{m}}^{(t)}$.

The results of this experiment are provided in Figure IV-2. While the effect is small for this particular model and dataset, the $\nu\tilde{m}$ and $\nu\tilde{m}^{(t)}$ strategies do result in smaller perturbations. However, PDPGD does not seem to benefit from the more advanced masking strategies. It obtains the best results when masking only the unlabeled regions. This issue comes from the projection of the dual variables onto the d-simplex (A IV-7): as different constraints get

discarded in subsequent iterations, their relative weights vary drastically, leading to oscillations of the dual variables. This phenomenon does not occur with an Augmented Lagrangian based attacks, as the penalty multipliers are not projected together. This does not create a dependency between the multipliers, resulting in a more stable optimization. Therefore, for the experiments, ALMA prox is used with the adaptive masking strategy with a linear decay, whereas PDPGD is used with the unlabeled region masking only.

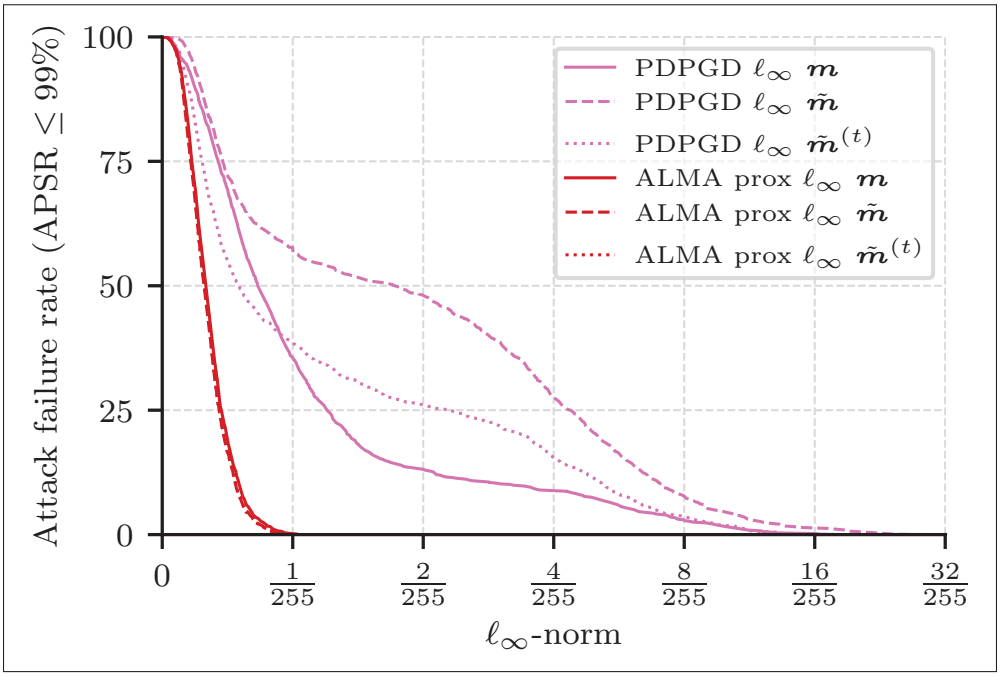


Figure-A IV-2 Influence of the constraint masking strategies on PDPGD and ALMA prox for untargeted attacks on Cityscapes with DeepLabV3+ ResNet-50. $\boldsymbol{m}$ corresponds to masking the pixels with no labels, $\tilde{\boldsymbol{m}}$ corresponds to additionally masking the top $(100 - \nu)\%$ constraints, and $\tilde{\boldsymbol{m}}^{(t)}$ corresponds to the strategy in Equation (5.6). The curves for PDGD and ALMA prox $\tilde{\boldsymbol{m}}$ and $\tilde{\boldsymbol{m}}^{(t)}$ overlap

7. Attack complexities and run-times

Tables IV-2, IV-3 and IV-4 report the average complexities (in terms of forward and backward) and run-time per sample for the attacks. The run-times for the targeted attacks are similar to their targeted variant, as the only difference lies in the loss, which is far from being the most computationally expensive part of the attacks. These results show that ALMA prox has slightly higher run-times compared to the other attacks with similar budgets, while being more effective. Note that some variance come from the fact that the experiments were run on a shared compute cluster.

Table-A IV-2 Average complexity of the attacks in terms of number of forward and backward propagations. All the attacks, except I-FGSM, MI-FGSM and PGD have a 500 iteration budget. DAG is the only attack that uses an early stopping criterion

Attack	Forwards	Backwards
I-FGSM 13×20 and MI-FGSM	260	260
PGD 13×40 and $13 \times 4 \times 10$	520	520
DAG $\eta = 0.003$	63	62
DAG $\eta = 0.001$	155	154
FMN ℓ_∞	500	500
PDPGD ℓ_∞	500	500
ALMA prox	500	500

8. Complete attack results

Tables IV-5 and IV-6 report the median and average ℓ_∞-norm (multiplied by 255 for readability) of the perturbations produced by the attacks for all regular models. As in section 5.6, the perturbation norm is considered to be 1 for unsuccessful attacks. This means that attack with less than 50% success have a 1 (*i.e.* 255) median ℓ_∞-norm in the table. Additionally, Figures IV-3, IV-4 and IV-5 show the percentage of unsuccessful attacks on Pascal VOC 2012 and Cityscapes for all models considered.

Table-A IV-3 Average run-times per image, in seconds, for the attacks on Pascal VOC 2012

	Attack	DeepLabV3+ ResNet-50	DeepLabV3+ ResNet-101	FCN HRNetV2 W48	DeepLabV3 DDC-AT
Untargeted	I-FGSM 13×20	13.3	19.2	28.8	–
	MI-FGSM 13×20	13.3	19.2	29.1	–
	PGD CE 13×40	26.4	38.2	57.0	–
	PGD CE $13 \times 4 \times 10$	26.6	38.3	57.2	–
	PGD DLR 13×40	27.7	39.3	60.9	–
	PGD DLR $13 \times 4 \times 10$	27.9	39.4	57.6	–
	DAG $\eta = 0.003$	2.1	5.2	6.0	7.0
	DAG $\eta = 0.001$	4.8	12.9	14.8	12.7
	FMN ℓ_∞	24.4	32.0	41.3	24
	PDPGD ℓ_∞	28.1	35.5	41.0	25.5
	ALMA prox	28.1	36.1	43.4	28
Targeted	I-FGSM 13×20	13.3	19.2	29.8	–
	MI-FGSM 13×20	13.3	19.4	28.8	–
	PGD CE 13×40	27.0	38.1	56.8	–
	PGD CE $13 \times 4 \times 10$	26.7	38.4	56.5	–
	PGD DLR 13×40	27.7	39.4	58.9	–
	PGD DLR $13 \times 4 \times 10$	27.8	39.8	58.7	–
	DAG $\eta = 0.003$	0.8	1.5	1.9	2.2
	DAG $\eta = 0.001$	2.1	4.2	4.6	4.8
	FMN ℓ_∞	24.4	32.1	41.1	25.4
	PDPGD ℓ_∞	27.6	36.9	39.0	26.8
	ALMA prox	28.4	36.3	43.0	28.5

Table-A IV-4 Average run-times per image, in seconds, for the attacks on Cityscapes

	Attack	DeepLabV3+ ResNet-50	FCN HRNetV2 W48	SegFormer MiT-B0	SegFormer MiT-B3	DeepLabV3 DDC-AT
Untargeted	I-FGSM 13×20	87.0	43.2	31.5	83.4	–
	MI-FGSM 13×20	87.6	41.5	31.1	83.2	–
	PGD CE 13×40	171.9	81.9	61.7	164.6	–
	PGD CE $13 \times 4 \times 10$	174.1	81.9	61.6	164.5	–
	PGD DLR 13×40	179.5	88.1	67.6	170.7	–
	PGD DLR $13 \times 4 \times 10$	179.3	88.2	67.6	170.6	–
	DAG $\eta = 0.003$	19.5	14.4	9.2	49.2	22.4
	DAG $\eta = 0.001$	45.5	37.3	23.4	107.3	44.7
	FMN ℓ_∞	155.2	79.1	59.1	158.2	64.5
	PDPGD ℓ_∞	161.0	84.5	64.1	162.4	64.7
	ALMA prox	161.9	96.8	77.6	176.3	69.8
Targeted	I-FGSM 13×20	87.5	41.4	31.1	83.0	–
	MI-FGSM 13×20	87.8	41.4	31.1	83.1	–
	PGD CE 13×40	171.9	81.9	61.4	165.0	–
	PGD CE $13 \times 4 \times 10$	171.9	82.0	61.5	164.3	–
	PGD DLR 13×40	179.6	88.2	68.4	172.0	–
	PGD DLR $13 \times 4 \times 10$	179.4	88.2	69.7	171.0	–
	DAG $\eta = 0.003$	47.6	18.6	21.9	54.1	44.5
	DAG $\eta = 0.001$	111.7	44.9	52.0	132.9	61.0
	FMN ℓ_∞	155.1	79.2	58.7	157.9	60.4
	PDPGD ℓ_∞	158.8	81.4	64.1	159.1	63.2
	ALMA prox	161.9	97.2	78.4	179.5	67.6

Table-A IV-5 Median and average $\|\boldsymbol{\delta}\|_\infty \times 255$ for each adversarial attack on **Pascal VOC 2012** for the regular models

	Attack	DeepLabV3+ ResNet-50		DeepLabV3+ ResNet-101		FCN HRNetV2 W48	
Untargeted	I-FGSM 13×20	146.32	136.55	124.14	131.34	227.11	145.69
	MI-FGSM 13×20	195.35	145.96	188.37	150.13	255.00	157.63
	PGD CE 13×40	80.10	120.93	100.42	123.17	152.08	136.05
	PGD CE $13 \times 4 \times 10$	24.90	74.15	20.05	30.28	66.93	118.76
	PGD DLR 13×40	7.22	26.42	9.00	21.29	11.11	23.85
	PGD DLR $13 \times 4 \times 10$	4.42	24.99	6.41	11.02	10.46	29.75
	DAG $\eta = 0.003$	5.69	6.63	6.65	8.74	8.80	10.61
	DAG $\eta = 0.001$	5.23	8.22	6.17	21.14	8.49	14.61
	FMN ℓ_∞	0.46	38.34	0.56	51.60	0.91	46.39
	PDPGD ℓ_∞	0.73	1.77	1.17	3.49	1.52	2.43
	ALMA prox	**0.32**	**0.34**	**0.37**	**0.41**	**0.51**	**0.56**
Targeted	I-FGSM 13×20	0.47	0.50	0.65	0.67	0.59	0.64
	MI-FGSM 13×20	0.59	0.66	0.77	0.82	0.77	0.85
	PGD CE 13×40	0.37	0.43	0.50	0.55	0.50	0.54
	PGD CE $13 \times 4 \times 10$	0.50	0.51	0.62	0.70	0.62	0.65
	PGD DLR 13×40	0.62	0.68	0.81	0.92	0.81	0.94
	PGD DLR $13 \times 4 \times 10$	0.87	1.07	1.18	2.03	1.12	1.28
	DAG $\eta = 0.003$	4.21	4.50	4.84	5.09	5.32	5.66
	DAG $\eta = 0.001$	3.92	4.21	4.55	5.80	5.07	5.36
	FMN ℓ_∞	0.42	0.45	0.46	0.56	0.47	0.49
	PDPGD ℓ_∞	0.28	0.36	0.31	0.40	0.35	0.46
	ALMA prox	**0.25**	**0.26**	**0.29**	**0.30**	**0.32**	**0.34**

Table-A IV-6 Median and average $\|\boldsymbol{\delta}\|_\infty \times 255$ for each adversarial attack on **Cityscapes** for the regular models

	Attack	DeepLabV3+ ResNet-50		FCN HRNetV2 W48		SegFormer MiT-B0		SegFormer MiT-B3	
Untargeted	I-FGSM 13×20	255.00	242.15	255.00	194.98	106.34	121.12	255.00	208.97
	MI-FGSM 13×20	255.00	244.43	255.00	218.91	202.15	159.52	255.00	228.43
	PGD CE 13×40	255.00	236.77	255.00	176.94	9.31	48.10	255.00	188.15
	PGD CE $13 \times 4 \times 10$	255.00	244.92	255.00	216.38	34.77	42.49	255.00	231.51
	PGD DLR 13×40	17.75	24.23	13.29	16.15	23.53	32.34	84.22	102.80
	PGD DLR $13 \times 4 \times 10$	255.00	227.89	12.08	45.17	22.37	35.49	110.82	134.41
	DAG $\eta = 0.003$	6.30	7.51	8.42	9.91	6.41	6.48	8.83	9.02
	DAG $\eta = 0.001$	5.95	9.39	8.20	20.29	6.08	13.57	8.59	53.65
	FMN ℓ_∞	1.97	96.57	0.97	35.12	0.58	1.16	1.08	6.42
	PDPGD ℓ_∞	1.06	2.82	1.69	2.75	0.87	1.17	1.39	3.05
	ALMA prox	**0.24**	**0.26**	**0.40**	**0.41**	**0.26**	**0.26**	**0.33**	**0.33**
Targeted	I-FGSM 13×20	255.00	255.00	115.76	128.65	5.91	31.96	51.73	88.24
	MI-FGSM 13×20	4.26	55.35	4.23	4.47	2.48	2.53	2.54	2.60
	PGD CE 13×40	3.49	3.71	3.49	3.62	2.30	2.34	1.87	1.92
	PGD CE $13 \times 4 \times 10$	255.00	255.00	255.00	255.00	255.00	255.00	255.00	255.00
	PGD DLR 13×40	8.37	67.86	7.41	7.52	4.79	4.96	3.98	4.09
	PGD DLR $13 \times 4 \times 10$	255.00	255.00	255.00	255.00	255.00	255.00	255.00	255.00
	DAG $\eta = 0.003$	11.34	12.96	9.87	10.49	8.05	8.44	9.82	10.06
	DAG $\eta = 0.001$	10.96	40.28	9.43	14.42	255.00	145.61	11.53	119.47
	FMN ℓ_∞	255.00	254.36	2.25	2.34	255.00	255.00	255.00	255.00
	PDPGD ℓ_∞	14.51	14.41	16.71	16.75	17.93	17.87	19.26	19.20
	ALMA prox	**1.15**	**1.17**	**1.11**	**1.12**	**0.70**	**0.70**	**0.65**	**0.66**

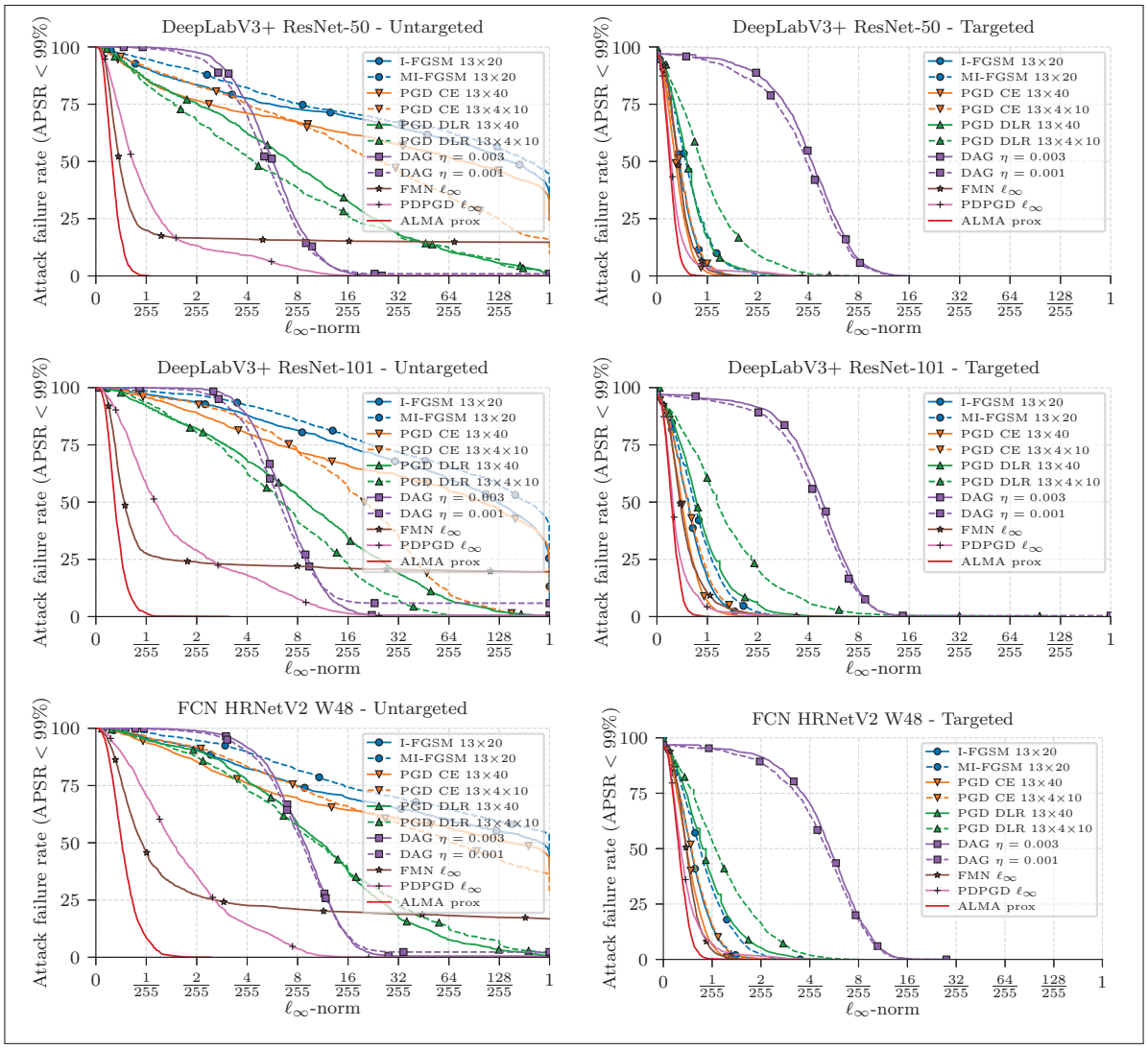


Figure-A IV-3 Percentage of unsuccessful ℓ_∞ attacks on **Pascal VOC 2012** (*i.e.* with APSR $\leq$ 99%). A stronger attack has a lower curve; a more robust model has a higher curve. Horizontal axis is linear on $[0, 2/255]$ and logarithmic on $[2/255, 1]$

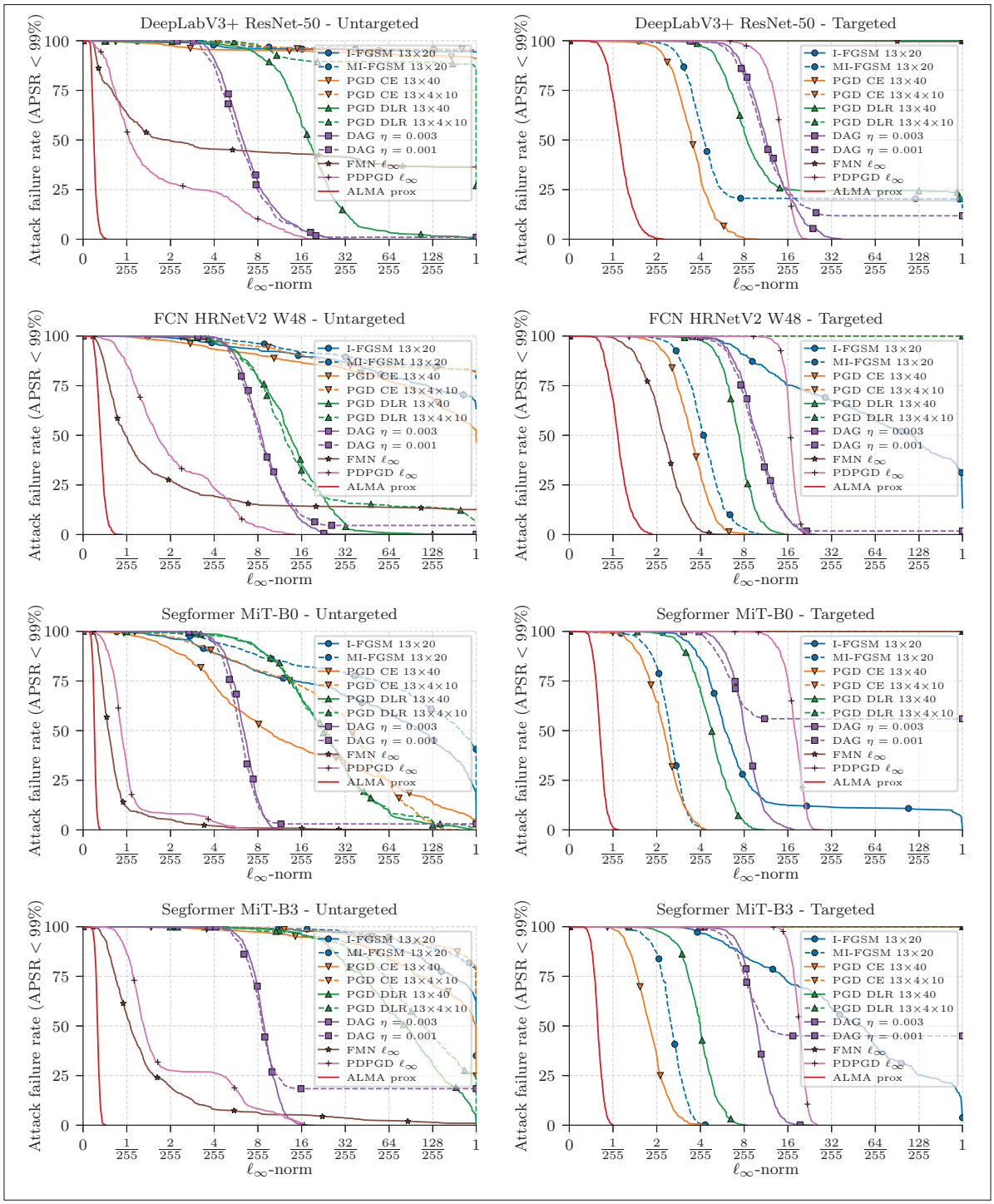


Figure-A IV-4 Percentage of unsuccessful ℓ_∞ attacks on **Cityscapes** (*i.e.* with APSR $\leq$ 99%). A stronger attack has a lower curve; a more robust model has a higher curve. Horizontal axis is linear on $[0, 2/255]$ and logarithmic on $[2/255, 1]$

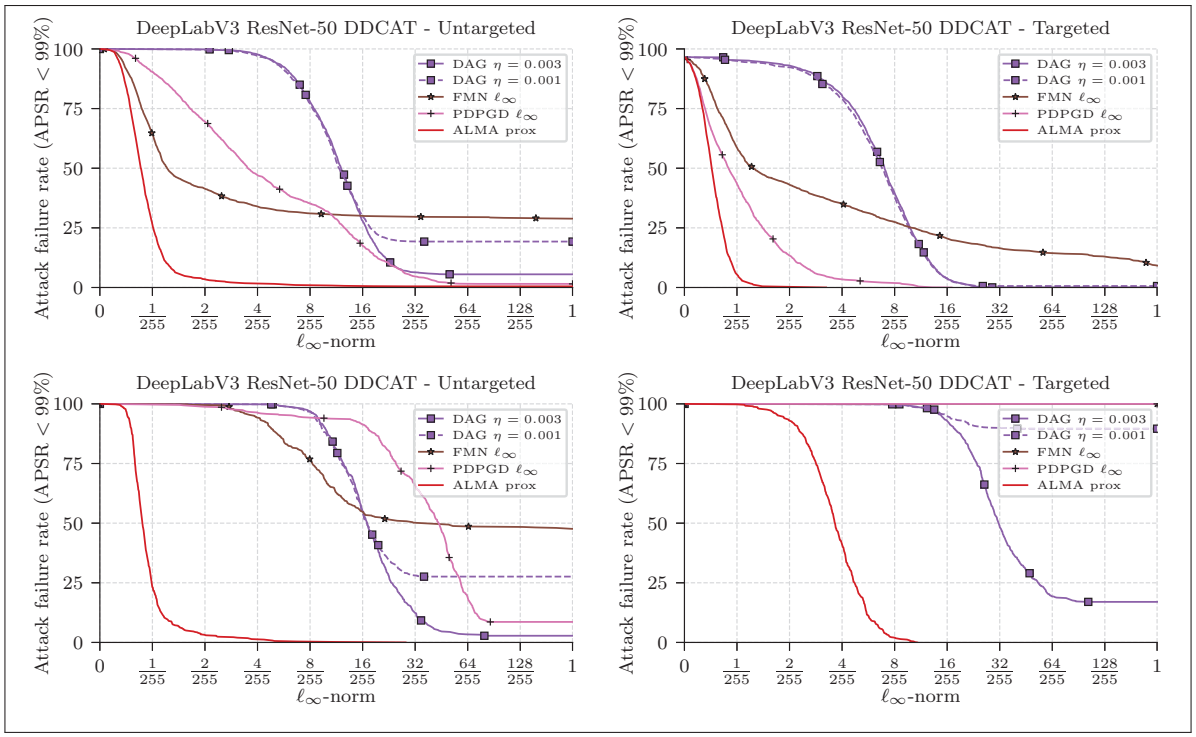


Figure-A IV-5 Percentage of unsuccessful ℓ_∞ attacks on Pascal VOC 2012 and Cityscapes (*i.e.* with APSR $\leq$ 99%) for the robust model DeepLabV3-DDCAT. A stronger attack has a lower curve; a more robust model has a higher curve. Horizontal axis is linear on $[0, 2/255]$ and logarithmic on $[2/255, 1]$

APPENDIX V

SUPPLEMENTARY MATERIAL FOR THE PAPER TITLED *CLASS ADAPTIVE NETWORK CALIBRATION*

1. Penalty functions for ALM

Here, we provide the requirements for a penalty function in Augmented Lagrangian Multiplier (ALM) method.

A function $P : \mathbb{R} \times \mathbb{R}_{++} \times \mathbb{R}_{++} \to \mathbb{R}$ is a Penalty-Lagrangian function such that $P'(z, \rho, \lambda) = \frac{\partial}{\partial y} P(z, \rho, \lambda)$ exists and is continuous for all $z \in \mathbb{R}$, $\rho \in \mathbb{R}_{++}$ and $\lambda \in \mathbb{R}_{++}$. In addition, it should satisfy the following four axioms (Birgin *et al.*, 2005):

Axiom 1: $P'(z, \rho, \lambda) \geq 0 \quad \forall z \in \mathbb{R}, \rho \in \mathbb{R}_{++}, \lambda \in \mathbb{R}_{++}$

Axiom 2: $P'(z, \rho, \lambda) = \lambda \quad \forall \rho \in \mathbb{R}_{++}, \lambda \in \mathbb{R}_{++}$

Axiom 3: If, for all $j \in \mathbb{N}$, $0 < \lambda_{\min} \leq \lambda^{(j)} \leq \lambda_{\max} < \infty$, then: $\lim_{j\to\infty} \rho^{(j)} = \infty$ and $\lim_{j\to\infty} z^{(j)} > 0$ imply that $\lim_{j\to\infty} P'(z^{(j)}, \rho^{(j)}, \lambda^{(j)}) = \infty$

Axiom 4: If, for all $j \in \mathbb{N}$, $0 < \lambda_{\min} \leq \lambda^{(j)} \leq \lambda_{\max} < \infty$, then: $\lim_{j\to\infty} \rho^{(j)} = \infty$ and $\lim_{j\to\infty} z^{(j)} < 0$ imply that $\lim_{j\to\infty} P'(z^{(j)}, \rho^{(j)}, \lambda^{(j)}) = 0$

where the first two axioms guarantee the derivative of the Penalty-Lagrangian function P w.r.t. z is positive and equals to λ when $z = 0$, while the last two axioms guarantee that the derivative tends to infinity when the constraint is not satisfied, and zero when the constraint holds.

There is a large number of valid penalty functions (Birgin *et al.*, 2005). In this paper we adopt the PHR function suggested by (Birgin *et al.*, 2005) and confirmed by our empirical results in Section 5 of the main text. We also empirically compare with another two popular choices, *i.e.* P2 and P3 (Birgin *et al.*, 2005), as shown in Figure 3 of the main text. The formulations of the above three penalty functions are as follows:

$$\mathrm{PHR}(z, \rho, \lambda) = \begin{cases} \lambda z + \frac{1}{2}\rho z^2 & \text{if} \quad \lambda + \rho z \geq 0; \\ -\frac{\lambda^2}{2\rho} & \text{otherwise.} \end{cases} \tag{A V-1}$$

$$P_2(z,\rho,\lambda) = \begin{cases} \lambda z + \lambda\rho z^2 + \frac{1}{6}\rho^2 z^3 & \text{if} \quad z \geq 0 \\ \frac{\lambda z}{1-\rho z} & \text{if} \quad z \leq 0 \end{cases} \tag{A V-2}$$

$$P_3(z,\rho,\lambda) = \begin{cases} \lambda z + \lambda\rho z^2 & \text{if} \quad z \geq 0 \\ \frac{\lambda z}{1-\rho z} & \text{if} \quad z \leq 0 \end{cases} \tag{A V-3}$$

2. Dataset description with implementation details

Tiny-ImageNet (Deng *et al.*, 2009) is a standard benchmark for image classification and commonly used in the calibration literature (Mukhoti *et al.*, 2020; Liu *et al.*, 2022a). It includes 64×64 dimensional images across 200 classes, with 500 images per class in the train set and 50 per class in the validation set. Following (Mukhoti *et al.*, 2020), we split out a validation set by randomly choose 50 samples per class from the train set, while the original validation set is used as the test set. We train ResNet-50 (He *et al.*, 2016a) model by SGD optimizer with a batch size of 128, and the number of epochs is set to 100. A multi-step learning rate scheduling strategy is used, *i.e*. learning rate 0.1 for the first 40 epochs, 0.01 for the next 20 epochs and 0.001 for the rest.

ImageNet (Deng *et al.*, 2009) is a large-scaled image classification benchmark. We use the version of ILSVRC-2012 (or ImageNet-1K) in our experiments (referred as ImageNet in this paper). It consists of 1K object classes with 1.2M images for training and 5K for validation. The average resolution of an image is 469×387. We follow (Minderer *et al.*, 2021) for evaluating calibration performance on ImageNet, *i.e*. reserving 20% for validation and the remaining 80% for testing. Besides ResetNet-50 (He *et al.*, 2016a), we also train state-of-the-art transformer based network, *i.e*. SwinV2-T (Liu *et al.*, 2022b), on this dataset. AdamW (Loshchilov & Hutter, 2019) optimizer is applied, and a cosine learning rate scheduler (Loshchilov & Hutter, 2017) with an initial learning rate of 0.001 is used. The number of training epochs is set to 200 and 300 for ResNet-50 and SwinV2-T respectively. The input size is 224×224 for ResNet-50 and 256×256 for SwinV2-T, while the batch size is 1024 for training both networks. Regular data

augmentation techniques like random resized crop, random horizontal flips, random color jitter, and random pixel erasing are applied on the training samples.

ImageNet-LT (Liu *et al.*, 2019) is truncated from ImageNet by sampling a subset so that the labels of the training set follow a long-tailed distribution. Overall, it has 115.8K images belonging to 1K classes, and the number of samples per class ranges from 5 to 1280. Both the validation and test sets are balanced, where the validation set includes 20 images per class and the original validation set in ImageNet is employed as the test set. Regarding the networks and training details, we use the same settings as those on ImageNet.

PASCAL VOC2012 (Everingham *et al.*, 2012) is a natural semantic segmentation benchmark including 20 foreground object classes and an additional background class. As the original test set is not publicly released and it is unable to evaluate the calibration performance via the official evaluation server, we split out a validation set by randomly selecting 20% images from the training set and treat the original validation set as our test set. Overall, the training/validation/test split contains $1171/293/1449$ images. For segmentation model training, we employ DeepLabV3 (Chen *et al.*, 2017) implemented by the popular public library[16], where we use ResNet-34 as encoder initialized with pre-trained weights on ImageNet, and the decoder is trained from scratch. The batch size is set to 8 and AdamW optimizer is used with an initial learning rate of 0.001 alongside a cosine learning rate scheduler. Finally, the maximum training epoch is set to 100.

20 Newsgroups (Lang, 1995). To evaluate the generalization of the proposed method, we include a non-vision dataset, *i.e*. 20 Newsgroups, which is a text classification benchmark and also used in previous calibration papers (Mukhoti *et al.*, 2020; Liu *et al.*, 2022a). It contains $20K$ news articles from 20 different groups according to the content, *e.g*. rec.motorcycles, rec.autos, sci.space, etc. We use the standard data split setting : $15,098$ documents for training, 900 for validation and $3,999$ for testing. The Glove word embedding (Pennington, Socher & Manning, 2014) is used to encode the text and then a Global Pooling Convolutional Network (GPCN) (Lin *et al.*, 2014) is trained. During training, we use Adam optimizer with an initial learning rate of

[16] https://github.com/qubvel/segmentation_models.pytorch

0.001. We train the model for 100 epochs, where the learning rate is decayed by a factor of 0.1 after the first 50 epochs.

3. Additional results

Table-A V-1 Calibration performance (ECE in %) when adding post temperature scaling (best T value for each method in subscript). The architecture is fixed to ResNet-50 for the vision datasets and GPCN for 20 News dataset.

	TinyImageNet		ImageNet		ImageNet-LT		20 News	
Method	Pre-TS	Post-TS	Pre-TS	Post-TS	Pre-TS	Post-TS	Pre-TS	Post-TS
CE	3.73	$1.86_{1.1}$	9.19	$3.88_{1.6}$	28.12	$3.72_{1.7}$	22.75	$3.01_{3.1}$
LS	3.17	$1.79_{0.9}$	2.57	$2.57_{1.0}$	10.46	$3.32_{1.3}$	8.07	$3.69_{1.2}$
FL	2.96	$1.74_{0.9}$	1.60	$1.60_{1.0}$	18.37	$2.52_{1.5}$	10.80	$3.33_{1.4}$
FLSD	2.91	$1.74_{0.9}$	2.08	$2.08_{1.0}$	17.77	$3.40_{1.4}$	10.87	$4.10_{1.4}$
CPC	4.88	$2.66_{1.5}$	3.66	$2.00_{1.1}$	16.00	$3.22_{1.2}$	9.46	$4.35_{1.4}$
MbLS	1.64	$1.64_{1.0}$	4.44	$2.07_{1.1}$	6.16	$2.60_{1.1}$	5.40	$2.09_{1.1}$
CALS-HR	2.50	$1.82_{0.9}$	5.63	$1.68_{1.4}$	2.83	$2.83_{1.0}$	6.99	$3.14_{1.1}$
CALS-ALM	**1.54**	$\mathbf{1.54}_{1.0}$	**1.46**	$\mathbf{1.28}_{1.1}$	**2.15**	$\mathbf{1.81}_{0.9}$	**2.04**	$\mathbf{1.86}_{1.1}$

Table V-1 reports the results of post-training temperature scaling (post-TS) on the outputs of the trained models (Guo *et al.*, 2017). Since this post-process technique is orthogonal to training based methods, we also present the results of applying it to our method, as well as the related works. We can see that our method without temperature scaling (pre-TS) outperforms previous methods, even post-TS, across all the benchmarks. Additionally, the ECE of our model is further reduced with post-TS in some cases, for instance on ImageNet (1.46% $\rightarrow$ 1.28%) and ImageNet-LT (2.04% $\rightarrow$ 1.81%).

Table V-2 reports the performance on the two natural image datasets, *i.e*. ImageNet and ImageNet-LT, in terms of Class-wise Calibration Errors (CWCE) (Maier-Hein *et al.*, 2022), which is a class-wise extension of ECE. Our method consistently achieve the best scores, with relative improvements of 25.0% on ImageNet and 23.3% on ImageNet-LT.

In Table V-3, we present results on the out-of-distribution (OOD) scenario (Minderer *et al.*, 2021). It is shown nn both settings, our method achieves the lowest ECE on the target domain. These results confirm the effectiveness of our method in the OOD scenario.

Table-A V-2 Class-wise Calibration Error (CWCE in %) computed for different approaches on ImageNet and ImageNet-LT. The architecture is fixed to ResNet-5. Best method is highlighted in bold.

Method	ImageNet	ImageNet-LT
CE	0.036	0.090
LS	0.029	0.072
FL	0.030	0.087
FLSD	0.029	0.087
CPC	0.049	0.078
MbLS	0.030	0.072
CALS-HR	0.029	0.071
CALS-ALM	**0.027**	**0.069**

Table-A V-3 ECE (%) on the out-of-distribution dataset, *i.e*. ImageNet-C (Gaussian noise corruption with severity level 5), for models trained on in-distribution datasets, *i.e*. ImageNet and ImageNetLT.

	CE	LS	FL	MbLS	**Ours**
ImageNet → ImageNet-C	26.25	24.00	23.73	26.55	**22.52**
ImageNet-LT → ImageNet-C	19.99	27.51	15.80	15.40	**12.91**

4. Visualization of learned classwise multipliers

Figure V-1 shows the evolution of learned multipliers λ_k on ImageNet for the three classes with the highest average and the three classes with the lowest average. This highlights the advantages of our method: 1) assigning distinct penalty weights for different classes; 2) adaptively updating the weight for each class throughout the training process.

5. Reliability diagram

Figure V-2 presents the reliability diagrams for different models trained on ImageNet and ImageNet-LT, which is a standard way of visualizing calibration performance. The curve of a perfectly calibrated model in the reliability diagram should match the dashed red line, where the prediction confidence perfectly reflects the accuracy of the model. It is shown that the models

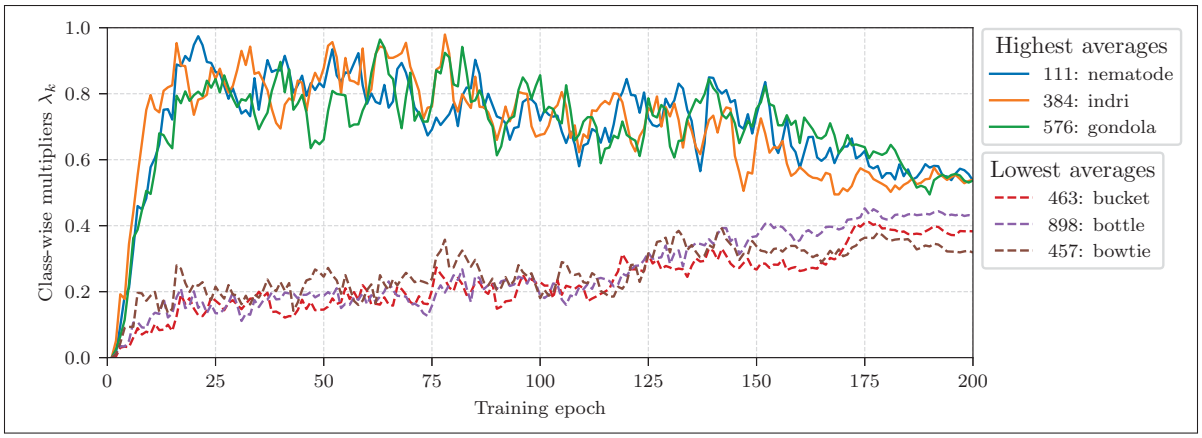


Figure-A V-1 Visualization of learned multipliers λ_k during the training of the ResNet-50 model on ImageNet. We show classes with the highest average (*Solid lines*) and the lowest average (*dashed lines*).

trained with CE (*left-most plots*) are over-confident, with accuracy mostly lower than confidence. Our method, CALS, is the most effective one to pull the curves closer to the expected lines, showing nearly perfect calibration performances. In particular, the improvement on ImageNet-LT is substantial compared to the other methods like LS and FL, which further demonstrates that the proposed class adaptive learning method could address the class imbalance issue in the long-tailed dataset. On ImageNet, LS and FL also present strong calibration performance, but decrease the final accuracy as shown in Table 1 of the main text. Overall, our method achieves the best compromise between calibration and accuracy. It is noted that the observation from Figure V-2 is supported by the quantitative scores reported in Table 1 of the main text.

6. Hyper-parameter setting

Table V-4 gives details of the hyper-parameter settings in our method, *i.e*. CALS-ALM. Note, the margin values are set by following (Liu *et al.*, 2022a), *i.e*. 10 for all the vision tasks including classification and segmentation, and 6 for the text classification on 20 Newsgroups.

Regarding the related methods reported in Table 1 of the main text, we set their hyper-parameters by following previous works, except that the values for MMCE (Kumar *et al.*, 2018) and CPC

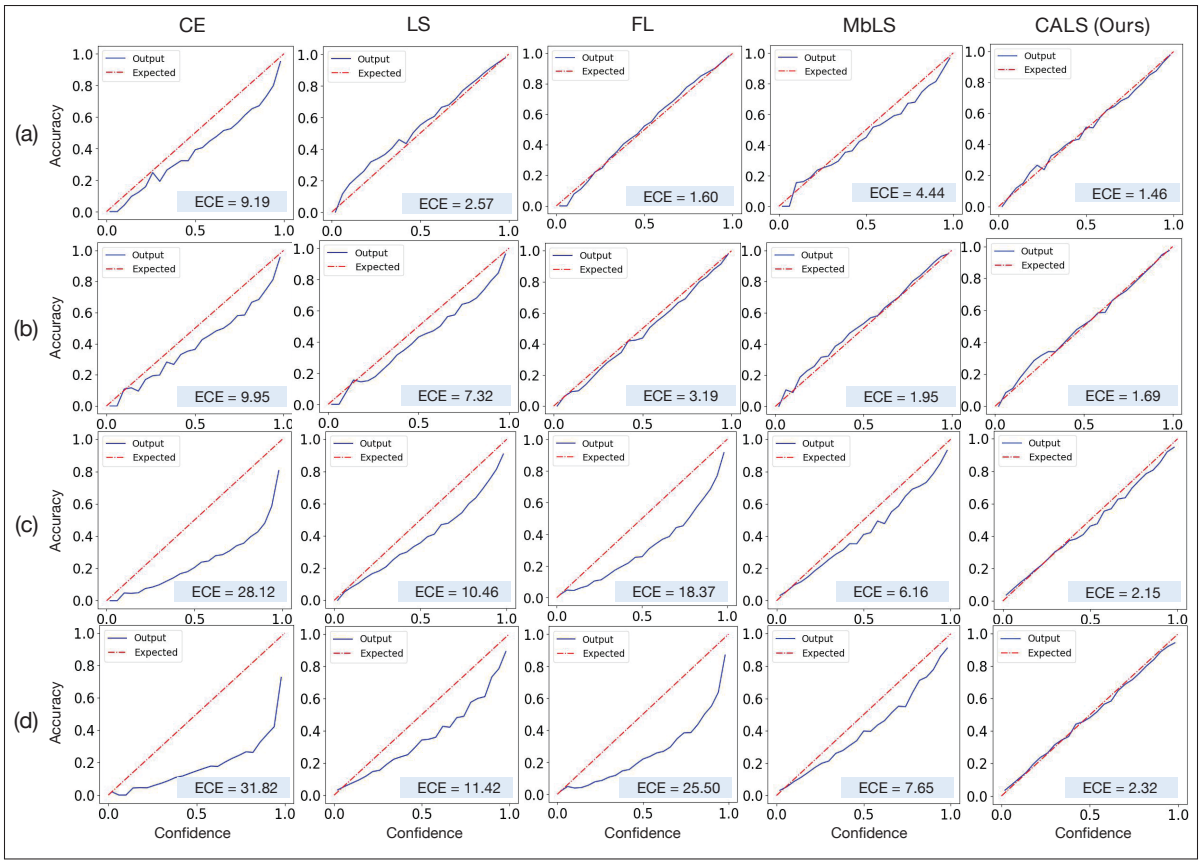


Figure-A V-2 **Calibration visualizations: (a) ImageNet (ResNet-50), (b) ImageNet (SwinV2-T), (c) ImageNet-LT (ResNet-50), and (d) ImageNet-LT (SwinV2-T) .** We present the reliability diagrams of our method (CALS), compared with those of baselines and closely related works. The number of bins to plot reliability diagrams is set to 25.

Table-A V-4 Hyper-parameters for our CALS-ALM method

Hyper-parameter	value
Margin m (all vision tasks)	10
Margin m (text classification)	6
Initial multiplier $\boldsymbol{\lambda}^{(0)}$	$10^{-6} \cdot \mathbf{1}_K$
Initial Penalty parameter $\boldsymbol{\rho}^{(0)}$	$\mathbf{1}_K$
Penalty increasing factor γ	1.2
Constraint improvement factor τ	0.9
Period of penalty parameter update	10

(Cheng & Vasconcelos, 2022) are empirically set according to our implementation . Detailed hyper-parameter settings for each method are as follows:

- MMCE (Kumar *et al.*, 2018): balancing weight $\lambda = 0.1$.
- ECP (Pereyra *et al.*, 2017): balancing weight $\lambda = 0.1$.
- LS (Szegedy *et al.*, 2016): smoothing factor $\alpha = 0.05$.
- FL (Mukhoti *et al.*, 2020): scaling factor $\gamma = 3$
- FLSD (Mukhoti *et al.*, 2020): scaling factor γ is set to 5 for $s_k \in [0, 0.2)$ and 3 for $s_k \in [0.2, 1)$, where k is the right class for the sample.
- CPC (Cheng & Vasconcelos, 2022): balancing weights for the binary discrimination penalty and binary exclusion penalty are set to 10 and 1 respectively. It is noted that we re-implement CPC since the official code is not publicly available.
- MbLS (Liu *et al.*, 2022a): balancing weight $\lambda = 0.1$, margin $m = 10$ for all the vision tasks, and $m = 6$ for the text classification task.

BIBLIOGRAPHY

Arnab, A., Miksik, O. & Torr, P. H. (2018). On the robustness of semantic segmentation models to adversarial attacks. *IEEE Conference on Computer Vision and Pattern Recognition*.

Athalye, A., Carlini, N. & Wagner, D. (2018). Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. *International Conference on Machine Learning*.

Attouch, H., Bolte, J. & Svaiter, B. F. (2013). Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized Gauss–Seidel methods. *Mathematical Programming*, 137(1), 91–129.

Augustin, M., Meinke, A. & Hein, M. (2020). Adversarial Robustness on In-and Out-Distribution Improves Explainability. *European Conference on Computer Vision*.

Bach, F., Jenatton, R., Mairal, J., Obozinski, G. et al. (2012). Optimization with sparsity-inducing penalties. *Foundations and Trends® in Machine Learning*, 4(1), 1–106.

Bauschke, H. H., Combettes, P. L. et al. (2011). *Convex analysis and monotone operator theory in Hilbert spaces*. Springer.

Beck, A. & Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1), 183–202.

Becker, S. & Fadili, J. (2012). A quasi-Newton proximal splitting method. *Advances in Neural Information Processing Systems*.

Bertsekas, D. P. (2014). *Constrained Optimization and Lagrange Multiplier Methods*. Academic press.

Bertsekas, D. P. (2016). *Nonlinear Programming: 3rd Edition*. Athena Scientific.

Biggio, B. & Roli, F. (2018). Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84, 317-331.

Birgin, E. G. & Martínez, J. M. (2014). *Practical augmented Lagrangian methods for constrained optimization*. SIAM.

Birgin, E. G., Castillo, R. A. & Martínez, J. M. (2005). Numerical Comparison of Augmented Lagrangian Algorithms for Nonconvex Problems. *Computational Optimization and Applications*, 31(1), 31–55.

Bonnet, B., Furon, T. & Bas, P. (2020). What If Adversarial Samples Were Digital Images? *Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security*.

Boudiaf, M., Rony, J., Ziko, I. M., Granger, E., Pedersoli, M., Piantanida, P. & Ben Ayed, I. (2020). A unifying mutual information view of metric learning: cross-entropy vs. pairwise losses. *European Conference on Computer Vision*.

Boyd, S., Boyd, S. P. & Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Brendel, W., Rauber, J., Kümmerer, M., Ustyuzhaninov, I. & Bethge, M. (2019). Accurate, reliable and fast robustness evaluation. *Advances in Neural Information Processing Systems*.

Brendel, W., Rauber, J., Kurakin, A., Papernot, N., Veliqi, B., Mohanty, S. P., Laurent, F., Salathé, M., Bethge, M., Yu, Y. et al. (2020). Adversarial vision challenge. *The NeurIPS'18 Competition: From Machine Learning to Intelligent Conversations*.

Buckman, J., Roy, A., Raffel, C. & Goodfellow, I. (2018). Thermometer Encoding: One Hot Way To Resist Adversarial Examples. *International Conference on Learning Representations*.

Cakir, F., He, K., Xia, X., Kulis, B. & Sclaroff, S. (2019). Deep metric learning to rank. *IEEE Conference on Computer Vision and Pattern Recognition*.

Carlini, N. & Wagner, D. (2017). Towards evaluating the robustness of neural networks. *IEEE Symposium on Security and Privacy*.

Carlini, N., Athalye, A., Papernot, N., Brendel, W., Rauber, J., Tsipras, D., Goodfellow, I., Madry, A. & Kurakin, A. (2019). *On Evaluating Adversarial Robustness*.

Carmon, Y., Raghunathan, A., Schmidt, L., Duchi, J. C. & Liang, P. S. (2019). Unlabeled Data Improves Adversarial Robustness. *Advances in Neural Information Processing Systems*.

Chambolle, A. & Dossal, C. H. (2015). On the convergence of the iterates of the "fast iterative shrinkage/thresholding algorithm". *Journal of Optimization theory and Applications*, 166(3), 968–982.

Chen, L.-C., Papandreou, G., Schroff, F. & Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*.

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F. & Adam, H. (2018a). Encoder-decoder with atrous separable convolution for semantic image segmentation. *European Conference on Computer Vision*.

Chen, P.-Y., Sharma, Y., Zhang, H., Yi, J. & Hsieh, C.-J. (2018b). EAD: Elastic-Net Attacks to Deep Neural Networks via Adversarial Examples. *Association for the Advancement of Artificial Intelligence*.

Chen, T., Kornblith, S., Norouzi, M. & Hinton, G. (2020). A Simple Framework for Contrastive Learning of Visual Representations. *International Conference on Machine Learning*.

Cheng, J. & Vasconcelos, N. (2022). Calibrating deep neural networks by pairwise constraints. *IEEE Conference on Computer Vision and Pattern Recognition*.

Chouzenoux, E., Pesquet, J.-C. & Repetti, A. (2014). Variable metric forward–backward algorithm for minimizing the sum of a differentiable function and a convex function. *Journal of Optimization Theory and Applications*, 162(1), 107–132.

Cisse, M. M., Adi, Y., Neverova, N. & Keshet, J. (2017). Houdini: Fooling Deep Structured Visual and Speech Recognition Models with Adversarial Examples. *Advances in Neural Information Processing Systems*.

Cohen, J., Rosenfeld, E. & Kolter, Z. (2019). Certified Adversarial Robustness via Randomized Smoothing. *International Conference on Machine Learning*.

Combettes, P. L. & Pesquet, J.-C. (2011). Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering* (pp. 185–212). Springer.

Combettes, P. L. & Vũ, B. C. (2014). Variable metric forward–backward splitting with applications to monotone inclusions in duality. *Optimization*, 63(9), 1289–1318.

Combettes, P. L. & Wajs, V. R. (2005). Signal recovery by proximal forward-backward splitting. *Multiscale modeling & simulation*, 4(4), 1168–1200.

Combettes, P. L., Dũng, Đ. & Vũ, B. C. (2010). Dualization of signal recovery problems. *Set-Valued and Variational Analysis*, 18(3), 373–404.

Conn, A., Gould, N. & Toint, P. (1997). A Globally Convergent Lagrangian Barrier Algorithm for Optimization with General Inequality Constraints and Simple Bounds. *Mathematics of Computation*, 66(217), 261–288.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S. & Schiele, B. (2016). The Cityscapes Dataset for Semantic Urban Scene Understanding. *IEEE Conference on Computer Vision and Pattern Recognition*.

Croce, F. & Hein, M. (2020a). Minimally distorted Adversarial Examples with a Fast Adaptive Boundary Attack. *International Conference on Machine Learning*.

Croce, F. & Hein, M. (2020b). Reliable Evaluation of Adversarial Robustness with an Ensemble of Diverse Parameter-free Attacks. *International Conference on Machine Learning*.

Croce, F., Andriushchenko, M., Sehwag, V., Debenedetti, E., Flammarion, N., Chiang, M., Mittal, P. & Hein, M. (2021). RobustBench: a standardized adversarial robustness benchmark. *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Davis, J. V., Kulis, B., Jain, P., Sra, S. & Dhillon, I. S. (2007). Information-theoretic metric learning. *International Conference on Machine Learning*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *IEEE Conference on Computer Vision and Pattern Recognition*.

Dhillon, G. S., Azizzadenesheli, K., Bernstein, J. D., Kossaifi, J., Khanna, A., Lipton, Z. C. & Anandkumar, A. (2018). Stochastic activation pruning for robust adversarial defense. *International Conference on Learning Representations*.

Ding, Z., Han, X., Liu, P. & Niethammer, M. (2021). Local temperature scaling for probability calibration. *International Conference on Computer Vision*.

Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X. & Li, J. (2018). Boosting adversarial attacks with momentum. *IEEE Conference on Computer Vision and Pattern Recognition*.

Engstrom, L., Ilyas, A., Salman, H., Santurkar, S. & Tsipras, D. (2019). Robustness (Python Library). Retrieved from: https://github.com/MadryLab/robustness.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J. & Zisserman, A. (2012). The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. Retrieved from: http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

Fischer, V., Kumar, M. C., Metzen, J. H. & Brox, T. (2017). Adversarial Examples for Semantic Image Segmentation. *International Conference on Learning Representations, Workshop Track Proceedings*.

Fletcher, R. (2013). *Practical Methods of Optimization*. John Wiley & Sons.

Gal, Y. & Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *International Conference on Machine Learning*.

Ge, W. (2018). Deep metric learning with hierarchical triplet loss. *European Conference on Computer Vision*.

Goldberger, J., Hinton, G. E., Roweis, S. T. & Salakhutdinov, R. R. (2005). Neighbourhood components analysis. *Advances in Neural Information Processing Systems*.

Goodfellow, I., Shlens, J. & Szegedy, C. (2015). Explaining and Harnessing Adversarial Examples. *International Conference on Learning Representations*.

Gragnaniello, D., Marra, F., Verdoliva, L. & Poggi, G. (2021). Perceptual quality-preserving black-box attack against deep learning image classifiers. *Pattern Recognition Letters*, 147, 142-149.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J. et al. (2018). Recent advances in convolutional neural networks. *Pattern Recognition*, 77, 354-377.

Guo, C., Pleiss, G., Sun, Y. & Weinberger, K. Q. (2017). On calibration of modern neural networks. *International Conference on Machine Learning*.

Guo, C., Rana, M., Cisse, M. & van der Maaten, L. (2018). Countering Adversarial Images using Input Transformations. *International Conference on Learning Representations*.

Hadsell, R., Chopra, S. & LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. *IEEE Conference on Computer Vision and Pattern Recognition*.

He, K., Zhang, X., Ren, S. & Sun, J. (2016a). Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition*.

He, K., Zhang, X., Ren, S. & Sun, J. (2016b). Identity mappings in deep residual networks. *European Conference on Computer Vision*.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P. & Girshick, R. (2022). Masked autoencoders are scalable vision learners. *IEEE Conference on Computer Vision and Pattern Recognition*.

He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J. & Li, M. (2019). Bag of tricks for image classification with convolutional neural networks. *IEEE Conference on Computer Vision and Pattern Recognition*.

Hein, M. & Andriushchenko, M. (2017). Formal Guarantees on the Robustness of a Classifier against Adversarial Manipulation. *Advances in Neural Information Processing Systems*.

Hendrycks, D. & Dietterich, T. (2019). Benchmarking Neural Network Robustness to Common Corruptions and Perturbations.

Hermans, A., Beyer, L. & Leibe, B. (2017). *In defense of the triplet loss for person re-identification*.

Hestenes, M. R. (1969). Multiplier and gradient methods. *Journal of optimization theory and applications*, 4(5), 303–320.

Ioffe, S. & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning*.

Jensen, P. A. & Bard, J. F. (2002). *Operations research models and methods*. John Wiley & Sons.

Jia, X., Song, S., He, W., Wang, Y., Rong, H., Zhou, F., Xie, L., Guo, Z., Yang, Y., Yu, L. et al. (2018). *Highly scalable deep learning training system with mixed-precision: Training imagenet in four minutes*.

Kang, X., Song, B., Du, X. & Guizani, M. (2020). Adversarial Attacks for Image Segmentation on Multiple Lightweight Models. *IEEE Access*, 8, 31359-31370.

Karandikar, A., Cain, N., Tran, D., Lakshminarayanan, B., Shlens, J., Mozer, M. C. & Roelofs, R. (2021). Soft Calibration Objectives for Neural Networks.

Katz, G., Barrett, C., Dill, D. L., Julian, K. & Kochenderfer, M. J. (2017). Reluplex: An efficient SMT solver for verifying deep neural networks. *International Conference on Computer Aided Verification*, pp. 97–117.

Kaya, M. & Bilge, H. Ş. (2019). Deep metric learning: A survey. *Symmetry*, 1066.

Kedem, D., Tyree, S., Sha, F., Lanckriet, G. R. & Weinberger, K. Q. (2012). Non-linear metric learning. *Advances in Neural Information Processing Systems*.

Kervadec, H., Dolz, J., Yuan, J., Desrosiers, C., Granger, E. & Ben Ayed, I. (2022). Constrained deep networks: Lagrangian optimization via log-barrier extensions. *2022 30th European Signal Processing Conference (EUSIPCO)*.

Kim, W., Goyal, B., Chawla, K., Lee, J. & Kwon, K. (2018). Attention-based ensemble for deep metric learning. *European Conference on Computer Vision*.

Kingma, D. P. & Ba, J. (2015). Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*.

Kort, B. W. & Bertsekas, D. P. (1976). Combined Primal–Dual and Penalty Methods for Convex Programming. *SIAM Journal on Control and Optimization*, 14(2), 268–294.

Krause, J., Stark, M., Deng, J. & Fei-Fei, L. (2013). 3d object representations for fine-grained categorization. *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*.

Kumar, A., Sarawagi, S. & Jain, U. (2018). Trainable calibration measures for neural networks from kernel mean embeddings. *International Conference on Machine Learning*.

Kurakin, A., Goodfellow, I. & Bengio, S. (2017a). Adversarial examples in the physical world. *International Conference on Learning Representations (workshop track)*.

Kurakin, A., Goodfellow, I. J. & Bengio, S. (2017b). Adversarial Machine Learning at Scale. *International Conference on Learning Representations*.

Laidlaw, C., Singla, S. & Feizi, S. (2021). Perceptual Adversarial Robustness: Defense Against Unseen Threat Models. *International Conference on Learning Representations*.

Lang, K. (1995). Newsweeder: Learning to filter netnews. *International Conference on Machine Learning*.

Lin, M., Chen, Q. & Yan, S. (2014). Network in network. *International Conference on Learning Representations*.

Lin, T.-Y., Goyal, P., Girshick, R., He, K. & Dollár, P. (2017). Focal loss for dense object detection. *IEEE Conference on Computer Vision and Pattern Recognition*.

Lions, P.-L. & Mercier, B. (1979). Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16(6), 964–979.

Liu, B., Ben Ayed, I., Galdran, A. & Dolz, J. (2022a). The devil is in the margin: Margin-based label smoothing for network calibration. *IEEE Conference on Computer Vision and Pattern Recognition*.

Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B. & Song, L. (2017). Sphereface: Deep hypersphere embedding for face recognition. *IEEE Conference on Computer Vision and Pattern Recognition*.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S. & Guo, B. (2021). Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. *International Conference on Computer Vision*.

Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., Ning, J., Cao, Y., Zhang, Z., Dong, L. et al. (2022b). Swin transformer v2: Scaling up capacity and resolution. *IEEE Conference on Computer Vision and Pattern Recognition*.

Liu, Z., Luo, P., Qiu, S., Wang, X. & Tang, X. (2016). DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations. *IEEE Conference on Computer Vision and Pattern Recognition*.

Liu, Z., Miao, Z., Zhan, X., Wang, J., Gong, B. & Yu, S. X. (2019). Large-scale long-tailed recognition in an open world. *IEEE Conference on Computer Vision and Pattern Recognition*.

Loshchilov, I. & Hutter, F. (2017). SGDR: Stochastic Gradient Descent with Warm Restarts. *International Conference on Learning Representations*.

Loshchilov, I. & Hutter, F. (2019). Decoupled Weight Decay Regularization. *International Conference on Learning Representations*.

Lowe, D. G. (1995). Similarity metric learning for a variable-kernel classifier. *Neural computation*, 7(1), 72–85.

Ma, X. & Blaschko, M. B. (2021). Meta-cal: Well-controlled post-hoc calibration by ranking. *International Conference on Machine Learning*.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D. & Vladu, A. (2018). Towards Deep Learning Models Resistant to Adversarial Attacks. *International Conference on Learning Representations*.

Maier-Hein, L., Reinke, A., Christodoulou, E., Glocker, B., Godau, P., Isensee, F., Kleesiek, J., Kozubek, M., Reyes, M., Riegler, M. A. et al. (2022). *Metrics reloaded: Pitfalls and recommendations for image analysis validation*.

Matyasko, A. & Chau, L.-P. (2021). *PDPGD: Primal-Dual Proximal Gradient Descent Adversarial Attack*. Retrieved from: https://arxiv.org/abs/2106.01538.

Melchior, P., Joseph, R. & Moolekamp, F. (2019). *Proximal Adam: robust adaptive update scheme for constrained optimization*. Retrieved from: https://arxiv.org/abs/1910.10094.

Minderer et al. (2021). Revisiting the Calibration of Modern Neural Networks. *Advances in Neural Information Processing Systems*.

MMSegmentation Contributors. (2020). MMSegmentation: OpenMMLab Semantic Segmentation Toolbox and Benchmark. Retrieved from: https://github.com/open-mmlab/mmsegmentation.

Moosavi-Dezfooli, S.-M., Fawzi, A. & Frossard, P. (2016). DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. *IEEE Conference on Computer Vision and Pattern Recognition*.

Movshovitz-Attias, Y., Toshev, A., Leung, T. K., Ioffe, S. & Singh, S. (2017). No fuss distance metric learning using proxies. *International Conference on Computer Vision*.

Mukhoti, J., Kulharia, V., Sanyal, A., Golodetz, S., Torr, P. & Dokania, P. (2020). Calibrating deep neural networks using focal loss.

Müller, R., Kornblith, S. & Hinton, G. E. (2019). When does label smoothing help?

Musgrave, K., Belongie, S. & Lim, S.-N. (2020). A metric learning reality check. *European Conference on Computer Vision*.

Naeini, M. P., Cooper, G. & Hauskrecht, M. (2015). Obtaining well calibrated probabilities using bayesian binning. *Association for the Advancement of Artificial Intelligence*.

Nakayama, H., Sayama, H. & Sawaragi, Y. (1975). A generalized Lagrangian function and multiplier method. *Journal of Optimization Theory and Applications*, 17, 211–227.

Narasimhan, M. & Bilmes, J. (2005). A submodular-supermodular procedure with applications to discriminative structure learning. *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence (UAI)*.

Nesterov, Y. (1983). A method for solving the convex programming problem with convergence rate $O(1/k^2)$. *Proceedings of the USSR Academy of Sciences*, 269, 543-547.

Nocedal, J. & Wright, S. (2006). *Numerical Optimization*. Springer Science & Business Media.

Oh Song, H., Jegelka, S., Rathod, V. & Murphy, K. (2017). Deep metric learning via facility location. *IEEE Conference on Computer Vision and Pattern Recognition*.

Oord, A. v. d., Li, Y. & Vinyals, O. (2018). *Representation learning with contrastive predictive coding*.

Opitz, M., Waltner, G., Possegger, H. & Bischof, H. (2017). Bier-boosting independent embeddings robustly. *International Conference on Computer Vision*.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J. V., Lakshminarayanan, B. & Snoek, J. (2019). Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. *Advances in Neural Information Processing Systems*.

Ozbulak, U., Van Messem, A. & De Neve, W. (2019). Impact of adversarial examples on deep learning models for biomedical image segmentation. *International Conference on Medical Image Computing and Computer-Assisted Intervention*.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B. & Swami, A. (2016a). The limitations of deep learning in adversarial settings. *IEEE Symposium on Security and Privacy*.

Papernot, N., McDaniel, P., Wu, X., Jha, S. & Swami, A. (2016b). Distillation as a defense to adversarial perturbations against deep neural networks. *IEEE Symposium on Security and Privacy*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. & Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc.

Pennington, J., Socher, R. & Manning, C. D. (2014). Glove: Global vectors for word representation. *Conference on Empirical Methods in Natural Language Processing*.

Pereyra, G., Tucker, G., Chorowski, J., Kaiser, L. & Hinton, G. (2017). Regularizing Neural Networks by Penalizing Confident Output Distributions. *International Conference on Learning Representations*.

Pintor, M., Roli, F., Brendel, W. & Biggio, B. (2021). Fast Minimum-norm Adversarial Attacks through Adaptive Norm Constraints. *Advances in Neural Information Processing Systems*.

Powell, M. J. (1969). A method for nonlinear constraints in minimization problems. *Optimization*, 283–298.

Raghunathan, A., Steinhardt, J. & Liang, P. (2018). Certified Defenses against Adversarial Examples. *International Conference on Learning Representations*.

Rauber, J., Brendel, W. & Bethge, M. (2017). Foolbox: A Python toolbox to benchmark the robustness of machine learning models. *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*.

Rolínek, M., Musil, V., Paulus, A., Vlastelica, M., Michaelis, C. & Martius, G. (2020). Optimizing Rank-based Metrics with Blackbox Differentiation. *IEEE Conference on Computer Vision and Pattern Recognition*.

Rony, J., Hafemann, L. G., Oliveira, L. S., Ben Ayed, I., Sabourin, R. & Granger, E. (2019). Decoupling Direction and Norm for Efficient Gradient-Based L2 Adversarial Attacks and Defenses. *IEEE Conference on Computer Vision and Pattern Recognition*.

Rony, J., Granger, E., Pedersoli, M. & Ben Ayed, I. (2021). Augmented Lagrangian Adversarial Attacks. *International Conference on Computer Vision*.

Salman, H., Li, J., Razenshteyn, I., Zhang, P., Zhang, H., Bubeck, S. & Yang, G. (2019). Provably robust deep learning via adversarially trained smoothed classifiers. *Advances in Neural Information Processing Systems*.

Sanakoyeu, A., Tschernezki, V., Buchler, U. & Ommer, B. (2019). Divide and conquer the embedding space for metric learning. *IEEE Conference on Computer Vision and Pattern Recognition*.

Sangalli, S., Erdil, E., Hötker, A., Donati, O. F. & Konukoglu, E. (2021). Constrained Optimization to Train Neural Networks on Critical and Under-Represented Classes. *Advances in Neural Information Processing Systems*.

Savanier, M., Chouzenoux, E., Pesquet, J.-C. & Riddell, C. (2022). Unmatched Preconditioning of the Proximal Gradient Algorithm. *IEEE Signal Processing Letters*, 1122-1126.

Schroff, F., Kalenichenko, D. & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. *IEEE Conference on Computer Vision and Pattern Recognition*.

Schultz, M. & Joachims, T. (2004). Learning a distance metric from relative comparisons. *Advances in Neural Information Processing Systems*.

Sharma, G., Wu, W. & Dalal, E. N. (2005). The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations. *Color Research & Application*, 30(1), 21–30.

Sohn, K. (2016). Improved deep metric learning with multi-class n-pair loss objective. *Advances in Neural Information Processing Systems*.

Song, H. O., Xiang, Y., Jegelka, S. & Savarese, S. (2016). Deep Metric Learning via Lifted Structured Feature Embedding. *IEEE Conference on Computer Vision and Pattern Recognition*.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. & Fergus, R. (2014). Intriguing properties of neural networks. *International Conference on Learning Representations*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2015). Going deeper with convolutions. *IEEE Conference on Computer Vision and Pattern Recognition*.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. *IEEE Conference on Computer Vision and Pattern Recognition*.

Tang, M., Marin, D., Ben Ayed, I. & Boykov, Y. (2019). Kernel cuts: Kernel and spectral clustering meet regularization. *International Journal of Computer Vision*, 127, 477–511.

Tieleman, T. & Hinton, G. (2012). Lecture 6.5-rmsprop, coursera: Neural Networks for Machine Learning.

Tomani, C., Gruber, S., Erdem, M. E., Cremers, D. & Buettner, F. (2021). Post-hoc uncertainty calibration for domain drift scenarios. *IEEE Conference on Computer Vision and Pattern Recognition*.

Tramer, F., Carlini, N., Brendel, W. & Madry, A. (2020). On adaptive attacks to adversarial example defenses.

Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D. & McDaniel, P. (2018). Ensemble Adversarial Training: Attacks and Defenses. *International Conference on Learning Representations*.

Tschannen, M., Djolonga, J., Rubenstein, P. K., Gelly, S. & Lucic, M. (2020). On Mutual Information Maximization for Representation Learning. *International Conference on Learning Representations*.

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A. & Madry, A. (2019). Robustness May Be at Odds with Accuracy. *International Conference on Learning Representations*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. & Polosukhin, I. (2017). Attention is all you need.

Wah, C., Branson, S., Welinder, P., Perona, P. & Belongie, S. (2011). *The Caltech-UCSD Birds-200-2011 Dataset* (Report n°CNS-TR-2011-001).

Wang, F., Cheng, J., Liu, W. & Liu, H. (2018a). Additive margin softmax for face verification. *IEEE Signal Processing Letters*, 25(7), 926–930.

Wang, H., Wang, Y., Zhou, Z., Ji, X., Gong, D., Zhou, J., Li, Z. & Liu, W. (2018b). Cosface: Large margin cosine loss for deep face recognition. *IEEE Conference on Computer Vision and Pattern Recognition*.

Wang, J., Zhou, F., Wen, S., Liu, X. & Lin, Y. (2017). Deep metric learning with angular loss. *International Conference on Computer Vision*.

Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X. et al. (2020). Deep high-resolution representation learning for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(10), 3349–3364.

Wang, M. & Sha, F. (2011). Information theoretical clustering via semidefinite programming. *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AIStats)*.

Wang, X., Hua, Y., Kodirov, E., Hu, G., Garnier, R. & Robertson, N. M. (2019a). Ranked list loss for deep metric learning. *IEEE Conference on Computer Vision and Pattern Recognition*.

Wang, X., Han, X., Huang, W., Dong, D. & Scott, M. R. (2019b). Multi-similarity loss with general pair weighting for deep metric learning. *IEEE Conference on Computer Vision and Pattern Recognition*.

Wang, Z., Bovik, A. C., Sheikh, H. R. & Simoncelli, E. P. (2004). Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4), 600–612.

Weinberger, K. Q. & Saul, L. K. (2009). Distance Metric Learning for Large Margin Nearest Neighbor Classification. *Journal of Machine Learning Research*, 10(9), 207–244.

Wen, Y., Zhang, K., Li, Z. & Qiao, Y. (2016). A discriminative feature learning approach for deep face recognition. *European Conference on Computer Vision*.

Wong, E. & Kolter, Z. (2018). Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope. *International Conference on Machine Learning*.

Wong, E., Schmidt, F. & Kolter, Z. (2019). Wasserstein Adversarial Examples via Projected Sinkhorn Iterations. *International Conference on Machine Learning*.

Wu, Z., Efros, A. A. & Yu, S. X. (2018). Improving generalization via scalable neighborhood component analysis. *European Conference on Computer Vision*.

Xie, C., Wang, J., Zhang, Z., Zhou, Y., Xie, L. & Yuille, A. (2017). Adversarial examples for semantic segmentation and object detection. *International Conference on Computer Vision*.

Xie, C., Wang, J., Zhang, Z., Ren, Z. & Yuille, A. (2018). Mitigating Adversarial Effects Through Randomization. *International Conference on Learning Representations*.

Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J. M. & Luo, P. (2021). SegFormer: Simple and efficient design for semantic segmentation with transformers. *Advances in Neural Information Processing Systems*.

Xing, E. P., Jordan, M. I., Russell, S. J. & Ng, A. Y. (2003). Distance metric learning with application to clustering with side-information. *Advances in Neural Information Processing Systems*.

Xu, K., Liu, S., Zhao, P., Chen, P.-Y., Zhang, H., Fan, Q., Erdogmus, D., Wang, Y. & Lin, X. (2019). Structured Adversarial Attack: Towards General Implementation and Better Interpretability. *International Conference on Learning Representations*.

Xu, X., Zhao, H. & Jia, J. (2021). Dynamic divide-and-conquer adversarial training for robust semantic segmentation. *International Conference on Computer Vision*.

Xuan, H., Souvenir, R. & Pless, R. (2018). Deep randomized ensembles for metric learning. *European Conference on Computer Vision*.

Xuan, H., Stylianou, A. & Pless, R. (2020). Improved embeddings with easy positive triplet mining. *IEEE Winter Conference on Applications of Computer Vision (WACV)*.

Yao, Z., Gholami, A., Xu, P., Keutzer, K. & Mahoney, M. W. (2019). Trust region based adversarial attack on neural networks. *IEEE Conference on Computer Vision and Pattern Recognition*.

Yuan, Y., Yang, K. & Zhang, C. (2017). Hard-aware deeply cascaded embedding. *IEEE Conference on Computer Vision and Pattern Recognition*.

Yuille, A. L. & Rangarajan, A. (2001). The concave-convex procedure (CCCP).

Zagoruyko, S. & Komodakis, N. (2016). Wide Residual Networks. *British Machine Vision Conference*.

Zhai, A. & Wu, H.-Y. (2019). Classification is a Strong Baseline for Deep Metric Learning. *British Machine Vision Conference*.

Zhang, H., Yu, Y., Jiao, J., Xing, E. P., Ghaoui, L. E. & Jordan, M. I. (2019a). Theoretically Principled Trade-off between Robustness and Accuracy. *International Conference on Machine Learning*.

Zhang, H., Chen, H., Xiao, C., Gowal, S., Stanforth, R., Li, B., Boning, D. & Hsieh, C.-J. (2019b). Towards Stable and Efficient Training of Verifiably Robust Neural Networks. *International Conference on Learning Representations*.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E. & Wang, O. (2018). The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. *IEEE Conference on Computer Vision and Pattern Recognition*.

Zhao, P., Xu, K., Liu, S., Wang, Y. & Lin, X. (2019). ADMM Attack: An Enhanced Adversarial Attack for Deep Neural Networks with Undetectable Distortions. *Proceedings of the 24th Asia and South Pacific Design Automation Conference*.

Zhao, Z., Liu, Z. & Larson, M. (2020). Towards Large yet Imperceptible Adversarial Image Perturbations with Perceptual Color Distance. *IEEE Conference on Computer Vision and Pattern Recognition*.

Zheng, W., Chen, Z., Lu, J. & Zhou, J. (2019). Hardness-aware deep metric learning. *IEEE Conference on Computer Vision and Pattern Recognition*.