National Technical University of Athens
School of Electrical and Computer Engineering
Division of Signals, Control and Robotics

Improving Projected Gradient Descent based Adversarial Attacks

Diploma Thesis

of

Nikolaos Antoniou

Supervisor: Alexandros Potamianos
Associate Professor, NTUA

Athens, September 2022

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Signals, Control and Robotics

Improving Projected Gradient Descent based Adversarial Attacks

Diploma Thesis

of

Nikolaos Antoniou

Supervisor: Alexandros Potamianos
Associate Professor, NTUA

Approved by the examination committee on 14th September 2022.

(Signature)	*(Signature)*	*(Signature)*
.....	.....	.....
Alexandros Potamianos	Stefanos Kollias	Constantinos Tzafestas
Associate Professor, NTUA	Professor, NTUA	Associate Professor, NTUA

Athens, September 2022

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Signals, Control and Robotics

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism.

(Signature)

.....
Nikolaos Antoniou

14th September 2022

Περίληψη

Η παρούσα διπλωματική εργασία καταπιάνεται με το φαινόμενο των ανταγωνιστικών παραδειγμάτων (adversarial examples) που εμφανίστηκε πριν μερικά χρόνια στην βιβλιογραφία, και έκτοτε έχει αλλάξει άρδην τον τρόπο που αντιλαμβανόμαστε τα Νευρωνικά Δίκτυα. Τα ανταγωνιστικά παραδείγματα, για παράδειγμα στο πρόβλημα της κατηγοριοποίησης εικόνων, συμπίπτουν με εικόνες οι οποίες έχουν νοθευτεί με προσεκτικά σχεδιασμένες διαταραχές (perturbations), πολύ μικρές μεν για να γίνουν αντιληπτές από έναν ανθρώπινο παρατηρητή, αρκετά επιδραστικές δε ώστε να κατευθύνουν τα Νευρωνικά Δίκτυα να κατηγοριοποιούν τις εικόνες σε οποιαδήποτε (αυθαίρετη) κλάση, επηρεάζοντας αρνητικά την απόδοση σε εισόδους που το ανθρώπινο μάτι μπορεί πολύ εύκολα να αναθέσει την σωστή ετικέτα. Η ανάδειξη ενός τέτοιου φαινομένου ανησύχησε, και επομένως, κινητοποίησε την επιστημονική κοινότητα να αναζητήσει τρόπους ωστέ να βελτιώσει την ευρωστία (robustness) των Νευρωνικών Δικτύων, έναντι στην εν δυνάμει εμφάνιση τέτοιων παραδειγμάτων. Οι μέθοδοι που αποσκοπούν στο να αμβλύνουν τις συνέπειες αυτού του φαινομένου ονομάζονται Ανταγωνιστικές Άμυνες (Adversarial Defenses).

Έπειτα από την πάροδο των χρόνων, έχει γίνει αντιληπτό πως η αξιολόγηση των προτεινόμενων αμυνών πάσχει από ένα ιδιαίτερα σημαντικό ζήτημα: την υπερεκτίμηση ευρωστίας (Robustness Overestimation). Για να αποφανθούμε αν μια μέθοδος άμυνας μπορεί πραγματικά να αυξήσει την ευρωστία, πρέπει να λύσουμε ένα πρόβλημα βελτιστοποίησης το οποίο είναι ανέφικτο στον χώρο των Νευρωνικών Δικτύων. Ωστόσο, καταφεύγουμε σε προσεγγιστικές λύσεις αυτού του προβλήματος βελτιστοποίησης (όπως ακριβώς κάνουμε και όταν τα εκπαιδεύουμε, βλ. τον αλγόριθμο Gradient Descent). Η υπερεκτίμηση ευρωστίας αναφέρεται στην περίπτωση που ο αμυνόμενος δεν καταφέρνει να λύσει (προσεγγιστικά) αυτό το πρόβλημα επαρκώς καλά, επομένως αποκτά μια ψευδή αίσθηση πως η μέθοδος του προσφέρει ευρωστία, ενώ η πραγματικότητα διαφέρει.

Ένας από τους πιο δημοφιλής τρόπους για την προσεγγιστική λύση του προβλήματος που σχετίζεται με την αξιολόγηση Νευρωνικών Δικτύων είναι ο αλγόριθμος Projected Gradient Descent (συντομ. PGD). Μια από τις σχεδιαστικές επιλογές του αλγόριθμου PGD είναι αυτή της αντικειμενικής συνάρτησης (objective function) που βελτιστοποιεί ο αλγόριθμος, για την οποία η σχετική βιβλιογραφία έχει προτείνει μια πληθώρα από εναλλακτικές. Ωστόσο, έστω και μικρές διαφοροποιήσεις στην μαθηματική έκφραση της συνάρτησης κόστους έχουν την δυνατότητα να επηρεάσουν σε μεγάλο βαθμό την απόδοση του αλγορίθμου, λαμβάνοντας υπόψη την εξαιρετικά πολύπλοκη γεωμετρία στο πεδίο βελτιστοποίησης (optimization landscape) σε τόσο μεγάλες διαστάσεις. Στην εργασία αυτή, θέτουμε αυτήν την παρατήρηση (η οποία υποστηρίζεται εμπειρικά από προηγούμενα αποτελέσματα) ως το κύριο κίνητρο της δουλειάς μας, και αναζητούμε τρόπους ανάμειξης διαφορετικών συναρτήσεων κόστους ώστε να αποκομίσουμε καλύτερη επίδοση. Τα πειράματα μας επιδεικνύουν εμπειρικά ότι μια αρκετά απλοϊκή μέθοδος, δηλαδή η αλλαγή συνάρτησης κόστους στα μέσα του επαναληπτικού αλγορίθμου PGD, βοηθά τον αλγόριθμο να φθάσει καλύτερες λύσεις, με συνέπεια, καθώς το εύρημά μας γενικεύεται για 15 διαφορετικές μεθόδους άμυνας.

Λέξεις Κλειδιά: Βαθιά Νευρωνικά Δίκτυα, Όραση Υπολογιστών, Κατηγοριοποίηση Εικόνων, Ανταγωνιστικά Παραδείγματα/ Επιθέσεις και Άμυνες, Υπερεκτίμηση Ευρωστίας, Projected Gradient Descent

Abstract

This Diploma Thesis delves into the phenomenon of *Adversarial Examples*, which appeared some years ago in the literature and since then has radically changed our perception about Neural Networks. Adversarial examples, in the task of Image Classification for instance, refer to images that have been tampered with carefully designed perturbations, small enough to remain undetected from a human observer, though exerting great influence to the final output, steering the model towards predicting the image as belonging to any arbitrary class, whereas in the meantime, humans can readily assign the correct label to the distorted image. The emergence of this peculiar phenomenon concerned, and subsequently, motivated researchers to explore ways of enhancing the robustness of Neural Networks, against the prospect of confronting such adversarial inputs. Approaches that attempt to mitigate the downsides of this behaviour are called *Adversarial Defenses*.

In the course of time, it has become evident that evaluating the robustness of proposed defenses is plagued by a paramount issue: this of *Robustness Overestimation*. Deciding whether a method can really improve robustness, requires the defender to solve an optimization problem which, in the space of Neural Networks, is infeasible. Hence, we must resort to approximate solutions of this problem (exactly akin to the training procedure of such networks, c.f. the Gradient Descent Algorithm). Robustness Overestimation refers to the case where the defender fails to (approximately) solve the problem sufficiently well, hence acquiring a false sense about the true effectiveness of his method.

One of the most popular algorithms of obtaining approximate solutions for the optimization problem that pertains to the evaluation of Neural Networks' robustness is Projected Gradient Descent (PGD). Among several designing choices, the objective function considered during the iterative PGD process is quite influential, with the literature proposing various alternatives. However, even subtle changes in the mathematical expression of this objective may non-trivially affect the obtained results, given the highly complex geometry of such high-dimensional optimization landscapes. In this work, we set this observation (backed up by strong empircal evidence) as the focal point of our research, seeking methods of combining different objectives, hoping to reap benefits in the obtained performance. Our experiments empirically demonstrate that a rather simplistic approach, i.e. switching loss functions during PGD, urges the algorithm to yield better final solutions with pronounced constancy, since our findings generalize across 15 different adversarial defenses.

Key Terms: Deep Neural Networks, Computer Vision, Image Classification, Adversarial Examples/ Attacks and Defenses, Robustness Overestimation, Projected Gradient Descent

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή Αλέξανδρο Ποταμιάνο, ο οποίος με φιλοξένησε στην ομάδα του για να εκπονήσω την διπλωματική μου εργασία. Στα πλαίσια αυτά, μου έδωσε την ελευθερία να ασχοληθώ με το θέμα της επιλογής μου, χωρίς περιορισμούς, κάτι που αποτέλεσε εξαιρετική ώθηση για να ολοκληρώσω την εργασία μου με το καλύτερο δυνατό αποτέλεσμα. Η μεταξύ μας επικοινωνία στάθηκε κίνητρο ώστε να βελτιωθούν αρκετές ακατέργαστες πτυχές μου όπως η ικανότητα μου να παρουσιάζω την δουλειά μου με σαφήνεια και να διατυπώνω τις ιδέες μου για τα μελλοντικά ερευνητικά βήματα πιο θεμελιωμένα.

Επίσης, ευχαριστώ θερμά τον Υ.Δ. Ευθύμη Γεωργίου, για την διαθεσιμότητα του κατά την διάρκεια της χρονιάς ώστε να με βοηθήσει, όσο περνούσε από το χέρι του, στα διάφορα θέματα που παρουσιάζονταν. Επιπλέον, με βοήθησε καθοριστικά στην συγγραφή της δημοσίευσης που προέκυψε από την έρευνα της διπλωματικής. Του εύχομαι καλή τύχη και επιτυχία στην συνέχεια του διδακτορικού του.

Στους γονείς μου, Παναγιώτα και Χρήστο, που κατέβαλαν αδιάκοπη προσπάθεια ώστε εγώ και τα αδέρφια μου να σπουδάσουμε χωρίς περισπασμούς. Και στα αδέρφια μου, Βαγγέλη και Γιώργο, που τους είχα πρότυπα μεγαλώνοντας. Τους ευχαριστώ για την υποστήριξη.

Τέλος, καταλήγω πως είναι αρκετά δύσκολο (και εξαντλητικό) να εκφράσω ξεχωριστά την ευγνωμοσύνη μου σε όσους με βοήθησαν κατά την διάρκεια αυτών των χρόνων. Για τους φίλους, επιλέγω την εύκολη οδό να τους ευχαριστήσω όλους μαζί. Αυτούς που γνώριζα από πριν, και ομολογουμένως τύχαινε να μοιραζόμαστε μαζί πιο ξέγνοιαστες στιγμές. Και τους συμφοιτητές μου, που γνώρισα κατά την διάρκεια των σπουδών, δίχως τους οποίους τα φοιτητικά χρόνια θα ήταν ατελείωτα. Τους εύχομαι οι περιστάσεις να τους βοηθήσουν (ή τουλάχιστον ας μην τους σταθούν εμπόδιο) να πετύχουν όσα επιθυμούν.

Νίκος Αντωνίου
Αθήνα, Σεπτέμβρης 2022

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AT	Adversarial Training
AA	Auto Attack
PGD	Projected Gradient Descent
CE	Cross Entropy
CW	Carlini-Wagner loss
MD Attack	Margin Decomposition Attack
GAMA-PGD	Guided Adversarial Margin Attack
CAA	Composite Adversarial Attack
DLR	Difference of Logits Ratio
ASR	Attack Success Rate
DNN	Deep Neural Networks
FGSM	Fast Gradient Sign Method
FGM	Fast Gradient Method
NF	Normalizing Flows
VAE	Variational Autoencoders

Chapter 1

Εκτεταμένη Ελληνική Περίληψη

1.1 Εισαγωγή

Η παρούσα διπλωματική εργασία ασχολείται με το φαινόμενο των *ανταγωνιστικών παραδειγμάτων* (adversarial examples) στα Βαθιά Νευρωνικά Δίκτυα. Στην περίπτωση της κατηγοριοποίησης εικόνων, τα ανταγωνιστικά παραδείγματα μπορούν να χαρακτηριστούν εν συντομία ως μικρές διαταραχές που προστίθενται σε καθαρές εικόνες, οι οποίες αρχικά κατηγοριοποιούνται σωστά από το σύστημα, που έχουν την δυνατότητα να αλλάξουν ριζικά την έξοδο του συστήματος. Για την ακρίβεια, οι διαταραχές αυτές είναι τόσο μικρές που ένας ανθρώπινος παρατηρητής δεν μπορεί να αντιληφθεί την διαφορά μεταξύ καθαρής και νοθευμένης εικόνας, όπως φαίνεται στην Εικόνα 1.1. Το φαινόμενο αυτό, στην περιοχή των βαθιών νευρωνικών δικτύων, έγινε για πρώτη φορά αντιληπτό από τους Szegedy et al. [Sze+14] το 2014.

Figure 1.1. *Προσθέτοντας κατάλληλα επιλεγμένο θόρυβο στην αρχική εικόνα, μπορούμε να δημιουργήσουμε μια νέα είσοδο στην οποία το δίκτυο αποτυγχάνει να αναθέσει την σωστή ετικέτα. Ένας άνθρωπος, ωστόσο, μπορεί χωρίς δυσκολία να κατηγοριοποιήσει σωστά την νέα εικόνα.*

Όπως είναι φυσικό, η ανακάλυψη αυτής της ιδιόμορφης συμπεριφοράς δημιούργησε αρκετή συζήτηση στην επιστημονική κοινότητα. Τα νευρωνικά δίκτυα χρησιμοποιούνται σε πολλές εφαρμογές στις οποίες η ασφάλεια είναι επιτακτική, π.χ. στην Αυτόνομη Οδήγηση όπου το να αποτύχει το σύστημα να αναγνωρίσει επιτυχώς μια πινακίδα Stop, μπορεί να προκαλέσει ανεπανόρθωτες συνέπειες. Οι ερευνητές, έκτοτε, προσπαθούν να εξηγήσουν τους παράγοντες που καθιστούν τόσο ασταθή τα Νευρωνικά Δίκτυα και κυρίως, έχουν θέσει ως πρωταρχικό στόχο την εξάλειψη αυτής της αστάθειας, προσπαθώντας να δημιουργήσουν εύρωστα δίκτυα, ανεπηρέαστα σε τέτοια ανταγωνιστικά παραδείγματα. Οι μέθοδοι που στοχεύουν να βελτιώσουν την ευρωστία των συστημάτων απέναντι σε τέτοιες εισόδους ονομάζονται Ανταγωνιστικές Άμυνες (Adversarial Defenses). Αποφεύγοντας για την ώρα να μπούμε σε περαιτέρω λεπτομέρειες οι οποίες θα ήταν δυσνόητες σε κάποιον που δεν είναι οικείος με την εν λόγω περιοχή, αναφέρουμε περιληπτικά πως η αξιολόγηση της αποτελεσματικότητας των αμυνών

γίνεται μέσω της επιστράτευσης κάποιας ανταγωνιστικής επίθεσης (Adversarial Attacks), δηλαδή ενός αλγορίθμου ο οποίος λαμβάνει μια εικόνα και επιστρέφει ένα ανταγωνιστικό παράδειγμα. Ο αμυνόμενος, όταν διεξάγει αξιολόγηση της μεθόδου του, χρησιμοποιεί μια επίθεση για να παράξει ανταγωνιστικά παραδείγματα τα οποία εν συνεχεία περνούν από το σύστημα και δίνουν μια εκτίμηση της ευρωστίας της προτεινόμενης τεχνικής. Η δουλειά μας συνεισφέρει σε ένα ιδιαίτερα σημαντικό ζήτημα που ελλοχεύει στην αξιολόγηση αμυνών: το ζήτημα της Υπερεκτίμησης Ευρωστίας (Robustness Overestimation), όπου ο αμυνόμενος αποτυγχάνει να χρησιμοποιήσει την καταλληλότερη επίθεση, επομένως η λανθασμένη αξιολόγηση τον παραπλανεί σχετικά με την πραγματική αποτελεσματικότητα της μέθοδο του.

Σε αυτό το σημείο αναφέρουμε πως στην εν λόγω εργασία, μελετάμε το πρόβλημα των ανταγωνιστικών παραδειγμάτων από την σκοπιά της κατηγοριοποίησης εικόνων. Ωστόσο, η παρουσία τέτοιων εισόδων έχει εμφανιστεί σε οποιοδήποτε πρόβλημα το οποίο προσεγγίζεται μέσω των Νευρωνικών Δικτύων: για παράδειγμα σε Αυτόματη Αναγνώριση Φωνής [CW18] ή σε εφαρμογές πού έχουν να κάνουν με Φυσική Γλώσσα [Zha+20]. Επομένως, τα ανταγωνιστικά παραδείγματα δεν πρέπει να θεωρούνται ως ένα ζήτημα αποκλειστικά συνδεδεμένο με την φύση των εικόνων, αλλά πιο σωστά ως ένα ψεγάδι των Νευρωνικών Δικτύων, ανεξάρτητα από το πεδίο που προέρχονται τα δεδομένα.

1.2 Απαραίτητο Υπόβαθρο

1.2.1 Ορισμός του προβλήματος εύρεσης ανταγωνιστικών παραδειγμάτων

Στην προηγούμενη συζήτησή μας, αναφέραμε πως τα ανταγωνιστικά παραδείγματα πρέπει, σε σχέση με τις αντίστοιχές καθαρές εικόνες, να μην έχουν καμία φαινομενική διαφορά (οπτικά) ως προς έναν ανθρώπινο παρατηρητή. Μία τέτοια συνθήκη είναι δύσκολο να διατυπωθεί μαθηματικά, δηλαδή ποιες διαταραχές θεωρούνται ανεπαίσθητες για το ανθρώπινο μάτι και ποιες όχι. Οι πρώτές δουλειές στον χώρο θεώρησαν την ℓ_p-νόρμα ως μια καλή προσέγγιση οπτικής ομοιότητας μεταξύ δυο εικόνων: Για μια είσοδο $\mathbf{x}$ θεωρούμε ότι τα ανταγωνιστικά παραδείγματα πρέπει αναγκαστικά να βρίσκονται εντός μιας ορισμένης απόστασης, όπως αυτή ορίζεται από την ℓ_p-νόρμα. Εκφράζοντας μαθηματικά αυτήν την συνθήκη, το σύνολο αναζήτησης επιθέσεων για την είσοδο $\mathbf{x}$ ορίζεται ως η ℓ_p-μπάλα ακτίνας ϵ (μικρή σταθερά) γύρω από την εικόνα:

$$\Delta(\mathbf{x}) = \{\mathbf{x}' \in \mathcal{X} : \|\mathbf{x}' - \mathbf{x}\|_p \leq \epsilon\} \tag{1.1}$$

Φυσικά, μια τέτοια υπόθεση δεν είναι ούτε αναγκαία, αλλά ούτε και ικανή συνθήκη που εξασφαλίζει την οπτική ομοιότητα: Δεν είναι αναγκαία γιατί μπορεί δύο φαινομενικά όμοιες εικόνες να έχουν σχετικά μεγάλη απόσταση (βάσει της ℓ_p-νόρμας) και δεν είναι ικανή γιατί δύο εικόνες με σχετικά μικρή ℓ_p-απόσταση μπορεί να έχουν ορατές διαφορές στο περιεχόμενο τους [WB09]. Παρόλα αυτά, η ιστορία μας έχει δείξει πως οι αλγόριθμοι που δημιουργούν διαταραχές φραγμένης ℓ_p-νόρμας είναι όντως μη αντιληπτές, ενώ το γεγονός πως δεν είναι αναγκαία σημαίνει πως ασχολούμαστε με ένα πιο εύκολο πρόβλημα το οποίο θα μπορούσε να λυθεί πιο αποτελεσματικά, ωστόσο ακόμα και αυτό το υποπρόβλημα κρύβει πολλές δυσκολίες. Επομένως, η μελέτη του προβλήματος μέσα από την σκοπιά αυτών των υποθέσεων σαφώς και έχει αξία για την επιστημονική κοινότητα.
Υιοθετώντας αυτήν την υπόθεση για το εφικτό σύνολο αναζήτησης ανταγωνιστών, το πρόβλημα της εύρεσης τέτοιων εισόδων για ένα μοντέλο $f_{\boldsymbol{\theta}}$ (όπου $\boldsymbol{\theta}$: οι παράμετροι του) μπορεί να εκφραστεί ξεκάθαρα ως εξής:

$$\mathbf{x}' : f(\mathbf{x}') \neq f(\mathbf{x}) \quad , \quad \text{s.t.} \quad \mathbf{x}' \in \Delta(\mathbf{x}) \tag{1.2}$$

Για να χειριστούμε το πρόβλημα εύρεσης ανταγωνιστών ως πρόβλημα βελτιστοποίησης, είναι πιο βολικό να θέσουμε την παραπάνω διατύπωση ως την αναζήτηση της εισόδου $\mathbf{x}'$ που ανήκει στο $\Delta(\mathbf{x})$, και τέτοια ώστε να μεγιστοποιεί την συνάρτηση κόστους $\mathcal{L}(f(\mathbf{x}'), y)$, π.χ. στην κατηγοριοποίηση εικόνας μια κατάλληλη επιλογή είναι το cross-entropy, η οποία μετρά πόσο ικανά το σύστημα αναθέτει την ετικέτα y στην είσοδο $\mathbf{x}'$:

$$\mathbf{x}' : \max_{\mathbf{x}'} \mathcal{L}(f(\mathbf{x}'), y) \quad , \quad \text{s.t.} \quad \mathbf{x}' \in \Delta(\mathbf{x}) \tag{1.3}$$

Το παραπάνω πρόβλημα βελτιστοποίησης έχει την δυσκολία του ότι είναι μη κυρτό, καθώς η έξοδος του νευρωνικού $f_{\boldsymbol{\theta}}(\mathbf{x})$ είναι μη κυρτή ως προς την είσοδο, ωστόσο μπορούμε να το λύσουμε προσεγγιστικά αναζητώντας κάποιο τοπικό ακρότατο της αντικειμενικής συνάρτησης. Όταν εκπαιδεύουμε νευρωνικά δίκτυα, ένας τρόπος να λύσουμε τέτοια προβλήματα είναι μέσω του αλγορίθμου Gradient Descent. Οι δύο διαφορές είναι πως, στην περίπτωση μας, θέλουμε να βελτιστοποιήσουμε την αντικειμενική συνάρτηση ως προς την είσοδο $\mathbf{x}'$ αντί των παραμέτρων $\boldsymbol{\theta}$ του δικτύου και επίσης επιθυμούμε η τελική λύση να βρίσκεται στο σύνολο $\Delta(\mathbf{x})$. Αυτό μπορεί εύκολα να επιτευχθεί μέσω του, κλασικού στην κυρτή βελτιστοποίηση, αλγορίθμου Projected Gradient Descent (PGD), όπου κάθε βήμα ως προς την κατεύθυνση που υποδεικνύει το gradient ακολουθείται από ένα βήμα προβολής στο σύνολο των εφικτών λύσεων. Ο αλγόριθμος PGD μπορεί μαθηματικά να εκφραστεί μέσω της ακόλουθης εξίσωσης:

$$\mathbf{x}_{t+1} = \mathcal{P}_{\Delta(\mathbf{x})}\Big(\mathbf{x}_t + \eta \cdot \text{norm}_p(\nabla_{\mathbf{x}_t}\mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}_t), y))\Big) \tag{1.4}$$

όπου $\mathcal{P}_{\Delta(\mathbf{x}))}$: ο τελεστής προβολής, που προβάλλει το όρισμά μέσα στην ℓ_p-μπάλα ακτίνας ϵ γύρω από το $\mathbf{x}$, norm_p: ένας τελεστής κανονικοποίησης, ο οποίος κανονικοποιεί το όρισμα του ώστε να έχει μοναδιαία ℓ_p-νόρμα και α: το μέγεθος του βήματος. Αυτός ο αλγόριθμος είναι ένας από τους πιο απλούς και θεμελιώδεις τρόπους που μπορεί να κανείς να χρησιμοποιήσει ώστε να βρει ανταγωνιστικά παραδείγματα: οι Goodfellow et al. [Goo+15] έδειξαν πως μονάχα ένα βήμα του PGD δημιουργεί εικόνες που μπορούν να παραπλανήσουν τα νευρωνικά με μεγάλη επιτυχία. Οι παραλλαγές του PGD με πολλαπλά βήματα έχουν ακόμα ισχυρότερη επίδοση, όπως φάνηκε σε 2 ξεχωριστές ερευνητικές δουλειές [Mad+18; Kur+17]. Βεβαίως, στην βιβλιογραφία έχουν συσταθεί αρκετοί άλλοι τρόποι δημιουργίας ανταγωνιστικών παραδειγμάτων, αλλά σχεδόν όλοι μπορούν να ιδωθούν από την σκοπιά του αλγόριθμου PGD.

Αξιολόγηση Επιθέσεων. Σε αυτό το σημείο είναι σημαντικό να αναφέρουμε πως μπορεί κανείς να συγκρίνει την επίδοση των αλγορίθμων επίθεσης, δεδομένου του συστήματος f στο οποίο εφαρμόζουμε αυτές τις επιθέσεις. Στις ℓ_p-φραγμένες επιθέσεις, όπου το ζητούμενο είναι η επίθεση να αλλάζει την απόφαση του συστήματος και συγχρόνως η παραγόμενη εικόνα να κείτεται εντός της ℓ_p-μπάλας ακτίνας ϵ γύρω από την καθαρή, η πιο συνηθισμένη μετρική είναι ο Δείκτης Επιτυχίας Επίθεσης (Attack Success Rate, abbr. ASR) σε ένα σύνολο απο δεδομένα $\mathcal{D}_{\text{test}} = (\mathbf{x}_i, y_i)_{i=1}^N$ που το σύστημα δεν έχει δει κατά την εκπαίδευσή του. Αυτή η μετρική, για τον αλγόριθμο επιθέσεων A, μπορεί να εκφραστεί ως:

$$\text{ASR}_A = \frac{1}{N}\sum_i^N \mathbb{1}[f(A(\mathbf{x}_i)) \neq y_i]$$

1.2.2 Μια ταξινομία των Ανταγωνιστικών Επιθέσεων

Προτού προχωρήσουμε περαιτέρω στην παρουσίαση του απαραίτητου υπόβαθρου, είναι αναγκαίο να αποσαφηνίσουμε το εξής: στην προηγούμενη συζήτηση μας, όπου αναφέραμε τον αλγόριθμο PGD,

θεωρήσαμε ότι κάποιος που θέλει να επιτεθεί σε ένα σύστημα έχει πρόσβαση στα gradients του μοντέλου. Ωστόσο, η εικασία ότι τέτοιου είδους πληροφορία θα είναι πάντα διαθέσιμη σε πραγματικές εφαρμογές δεν είναι εντελώς εύστοχη. Τα συστήματα του πραγματικού κόσμου διατηρούν κρυφές τέτοιου είδες πληροφορίες, και η γνώση του επιτιθέμενου είναι αρκετά περιορισμένη. Επομένως, οι ανταγωνιστικές επιθέσεις γενικά μπορούν να διχοτομηθούν σε δύο μεγάλες κατηγορίες: Τις επιθέσεις *πλήρους γνώσης* (White-box Attacks) και τις επιθέσεις *μαύρου κουτιού* (Black-box Attacks). Στην πρώτη κατηγορία, ο επιτιθέμενος γνωρίζει τα πάντα για το απειλούμενο σύστημα: τα βάρη του, το σύνολο δεδομένων εκπαίδευσης κλπ. Η δεύτερη όμως, μελετά περιπτώσεις όπου ο επιτιθέμενος μπορεί μόνο να εκμεταλλευθεί το σύστημα μέσω ερωτημάτων (queries), δίνοντας ως είσοδο μια εικόνα $\mathbf{x}$ και λαμβάνοντας την έξοδο του δικτύου $f_{\boldsymbol{\theta}}(\mathbf{x})$.
Ποια όμως είναι τα κίνητρα να ερευνήσουμε κάθε μια από τις δύο περιπτώσεις ; Από άποψη πραγματικών εφαρμογών, η κατηγορία επιθέσεων τύπου black-box είναι πιο ρεαλιστική, αναδεικνύοντας τους κινδυνεύουν που μπορεί να αντιμετωπίσει ένα μοντέλο που δίνει ανοικτή πρόσβαση σε ελάχιστη πληροφορία. Ωστόσο, οι επιθέσεις white-box είναι πιο ενδιαφέρουσες για την μελέτη ευρωστίας: Αν καταφέρουμε να δημιουργήσουμε κάποια άμυνα η οποία έχει αντοχή απέναντι σε τέτοιους ανταγωνιστές, τότε αναμένουμε πως η προσφερόμενη ευρωστία θα γενικεύεται και απέναντι σε επιθέσεις που έχουν φτιαχτεί μέσω πολύ πιο φειδωλής γνώσης.
Στην παρούσα εργασία, εμείς θα ασχοληθούμε με τις επιθέσεις τύπου white-box, καθώς σκοπός μας είναι να εντρυφήσουμε στην διαδικασία αξιολόγησης των ανταγωνιστικών αμυνών και στην ιδιαίτερα προβληματική τους υπόσταση. Ωστόσο, ο αναγνώστης μπορεί στο Κεφάλαιο 3 να βρει μια εισαγωγή στο πως κανείς μπορεί να προσεγγίσει την δημιουργία επιθέσεων όταν δεν γνωρίζει τα gradients (ως προς την είσοδο) του συστήματος.

1.2.3 Ανταγωνιστικές Άμυνες

Η ανακάλυψη του φαινομένου των ανταγωνιστικών παραδειγμάτων κέντρισε την προσοχή των ερευνητών, οι οποίοι έκτοτε προσπαθούν πυρετωδώς να αμβλύνουν την ζημία που μπορούν να προκαλέσουν τέτοιες επιθέσεις στα Νευρωνικά Δίκτυα. Ο απώτερος σκοπός, φυσικά, είναι η κατασκευή εύρωστων μοντέλων, τα οποία δεν είναι τόσο ευαίσθητα σε μικρές διαταραχές των εισόδων τους. Σε αυτό το κεφάλαιο, επιχειρούμε να κάνουμε μία σύντομη εισαγωγή στην περιοχή των Ανταγωνιστικών Αμυνών, διευκρινίζοντας αρχικά τις διαφορές μεταξύ του να εκπαιδεύουμε μοντέλα, που απλά επιθυμούμε να γενικεύουν καλά σε καινούρια δεδομένα, και του να εκπαιδεύουμε εύρωστα μοντέλα, που επιπλέον πρέπει να είναι ομαλά απέναντι σε ℓ_p-φραγμένες διαταραχές.

Στην απλή περίπτωση, η εκπαίδευση των νευρωνικών δικτύων γίνεται μέσω της αναζήτησης των παραμέτρων $\boldsymbol{\theta}$ οι οποίες ελαχιστοποιούν του αναμενόμενο ρίσκο, πάνω στην κατανομή των δεδομένων:

$$\boldsymbol{\theta}^* : \min_{\boldsymbol{\theta}} \ \mathbb{E}_{(\mathbf{x},y)\sim p_{\text{data}}}\Big[\mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}), y; \boldsymbol{\theta})\Big] \tag{1.5}$$

Ωστόσο, αυτός ο τρόπος εκπαίδευσης βρίσκει παραμέτρους που αντιστοιχούν σε άκρως ευάλωτα, απέναντι σε ℓ_p-φραγμένες διαταραχές, συστήματα. Ένας τρόπος να προσεγγίσουμε το πρόβλημα της εκπαίδευσης νευρωνικών δικτύων που να είναι εύρωστα απέναντι σε τέτοιες επιθέσεις, είναι να μάθουμε παραμέτρους οι οποίες μειώνουν το αναμενόμενο ρίσκο απέναντι σε ℓ_p-φραγμένες επιθέσεις, ως εξής:

$$\boldsymbol{\theta}^* : \min_{\boldsymbol{\theta}} \ \mathbb{E}_{(\mathbf{x},y)\sim p_{\text{data}}}\Big[\max_{\mathbf{x}':\|\mathbf{x}-\mathbf{x}'\|_p\leq\epsilon} \mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}'), y; \boldsymbol{\theta})\Big] \tag{1.6}$$

Η εν λόγω αντικειμενική συνάρτηση έχει το πλεονέκτημα του ότι έχει μια αρκετά διαισθητική βάση: Αν επιθυμούμε να προσδώσουμε ℓ_p-φραγμένη ευρωστία στο δίκτυο, τότε το δίκτυο πρέπει να ΄μάθει΄

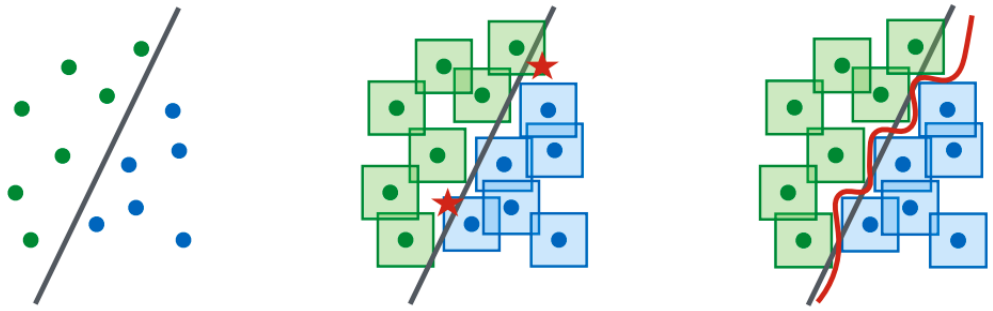


Figure 1.2. *Οπτικοποίηση των οριογραμμών απόφασης (decision boundaries). Η απλή ευθεία γραμμή δεν είναι αρκετή για να εξασφαλίσει καλή απόδοση απέναντι σε ℓ_p-φραγμένες επιθέσεις.*

αντιμετωπίζοντας τέτοιες επιθέσεις κατά την εκπαίδευσή του. Η παραπάνω τυποποίηση απαρτίζεται από 2 επιμέρους προβλήματα: το εξωτερικό (ελαχιστοποίησης) και το εσωτερικό (μεγιστοποίησης). Το εσωτερικό μπορεί να λυθεί προσεγγιστικά με οποιονδήποτε τρόπο βρίσκει ανταγωνιστικές επιθέσεις, π.χ. όπως τον αλγόριθμο του PGD, ενώ το εξωτερικό μπορεί κατά τα γνωστά να λυθεί όπως και όταν εκπαιδεύουμε νευρωνικά με τον καθιερωμένο τρόπο π.χ μέσω Stochastic Gradient Descent. Πράγματι, οι Madry et al. [Mad+18] εκμεταλλευθήκαν την παραπάνω τυποποίηση για να δημιουργήσουν εύρωστα δίκτυα, λύνοντας το εσωτερικό πρόβλημα μεγιστοποίησης μέσω του PGD, θέτοντας μικρό αριθμό βημάτων. Η τεχνική αυτή ονομάστηκε Ανταγωνιστική Εκπαίδευση (Adversarial Training) και αποτελεί, ως και σήμερα, έναν εκ των πιο καρποφόρων τρόπων για να βελτιωθεί η ευρωστία των νευρωνικών.

Γενικότερα, η ανταγωνιστική εκπαίδευση φιλοδοξεί να λύσει ένα αρκετά πολυπλοκότερο πρόβλημα, όπως μπορεί να απεικονίσει παραστατικά η Εικόνα 1.2: Στα γραμμικά διαχωρίσιμα δεδομένα της εικόνας, ένας κατηγοριοποιητής που είναι ευθεία γραμμή δεν αρκεί πλέον να λύσει το πρόβλημα εύρωστα. Αντίθετα, η νέα οριογραμμή απόφασης, σύμφωνα με το δεξί σχήμα, πρέπει να γίνει αρκετά πιο σύνθετη ώστε τα ανταγωνιστικά παραδείγματα των δεδομένων εκπαίδευσης να κατηγοριοποιούνται σωστά. Επομένως, η ευρωστία απαιτεί γενικά πιο πολύπλοκα μοντέλα, με μεγαλύτερη χωρητικότητα (capacity), από τα μοντέλα που χρειαζόμαστε στην απλή κατηγοριοποίηση εικόνας.

1.2.4 Αξιολόγηση ℓ_p-φραγμένης Ευρωστίας Νευρωνικών Δικτύων

Η αξιολόγηση της πραγματικής ℓ_p-ευρωστίας ενός νευρωνικού δικτύου είναι μια διαδικασία άκρως προβληματική. Γενικά, η ευρωστία ενός συστήματος, θεωρητικά, μπορεί να εκτιμηθεί μέσω της επόμενης εξίσωσης, απέναντι σε ένα σύνολο από δεδομένα $\mathcal{D}_{\text{test}} = (\mathbf{x}_i, y_i)_{i=1}^N$ που το μοντέλο δεν έχει δει κατά την εκπαίδευσή του:

$$\text{RobAcc}(f) = \frac{1}{N}\sum_{i=1}^{N} \min_{\mathbf{x}_i' : \|\mathbf{x}_i - \mathbf{x}_i'\|_p \leq \epsilon} \mathbb{1}[f(\mathbf{x}_i') = y_i] \tag{1.7}$$

Όπως φαίνεται, η αξιολόγηση απαιτεί την λύση ενός προβλήματος βελτιστοποίησης μέσα στον χώρο αναζήτησης των επιθέσεων (που είναι το συνεχές σύνολο της ℓ_p-μπάλας), το οποίο όμως δεν μπορεί να λυθεί ακριβώς για παραμετρικά μοντέλα που αναπαρίστανται μέσω βαθιών νευρωνικών δικτύων. Ο μόνος τρόπος να προσεγγίσει κάποιος το πρόβλημα είναι ο εξής: επιστρατεύοντας κάποιον αλγόριθμο που δημιουργεί ℓ_p-φραγμένες επιθέσεις, το πρόβλημα ελαχιστοποίησης μπορεί να λυθεί προσεγγιστικά μέσω της χρήσης των τοπικών ακροτάτων που επιστρέφει ο αλγόριθμος αυτός. Αφού επιτευχθεί αυτό,

ο αμυνόμενος μπορεί να εκτιμήσει την ευρωστία του συστήματός του μόνο μέσω ενός άνω φράγματος, το οποίο ελπίζει να μην απέχει πολύ από την πραγματική ευρωστία.

Η παραπάνω συζήτηση αναδεικνύει ένα ιδιαίτερα σημαντικό ζήτημα: η εκτίμηση της ευρωστίας ενός νευρωνικού δικτύου μπορεί να γίνει μόνο προσεγγιστικά, και η ποιότητα της εκτίμησης που λαμβάνουμε είναι άρρηκτα συνδεδεμένη με την ισχύ του αλγορίθμου που χρησιμοποιούμε για να λύσουμε το πρόβλημα βελτιστοποίησης. Αδύναμοι αλγόριθμοι είναι φυσικό πως θα προσφέρουν αδύναμες λύσεις, επομένως τα άνω φράγματα που απορρέουν από χρήση τέτοιων αλγορίθμων είναι αρκετά χαλαρά και ο αμυνόμενος παραπλανείται, έχοντας την ψευδαίσθηση πως η μέθοδος του προσδίδει ευρωστία στο σύστημα. Η δυσκολία, φυσικά, έγκειται στο γεγονός πως η ισχύς ενός αλγορίθμου έχει συσχέτιση με το ποιο σύστημα είναι υπό αξιολόγηση: Πράγματι, κάθε σύστημα έχει τις δικές του λύσεις σε αυτό το πρόβλημα ελαχιστοποίησης, τα δικά του ανταγωνιστικά παραδείγματα. Δυστυχώς, δεν υπάρχει κάποια καθολική επίθεση που να λύνει αυτό το πρόβλημα ελαχιστοποίησης το ίδιο καλά για όλα τα συστήματα. Το ζήτημα αυτό, εν κατακλείδι, είναι γνωστό στην βιβλιογραφία ως **υπερεκτίμηση ευρωστίας** (*robustness overestimation*) και όπως γίνεται αντιληπτό καθιστά ένα δριμύτατο εμπόδιο στην έρευνα που σχετίζεται με τις ανταγωνιστικές άμυνες.

Το ζήτημα της υπερεκτίμησης ευρωστίας αποτέλεσε το επίκεντρο πολλών δουλειών [Ath+18; Ues+18; Tra+20] στο οποίο οι αντίστοιχοι συγγραφείς κατάφεραν να παρακάμψουν ανταγωνιστικές άμυνες, παρ΄ότι φαινομενικά οι άμυνες αυτές (στις αρχικές δημοσιεύσεις τους) ανέφεραν μεγάλα ποσοστά ευρωστίας. Οι άμυνες που στοχοποιήθηκαν μέσα σε αυτές τις τρείς δουλειές ακολουθούσαν ένα μοτίβο: Για να προσφέρουν ευρωστία στο σύστημα, χρησιμοποιούσαν κάποιο τρόπο ο οποίος εν τέλει κατέστρεφε τα gradients ως προς την είσοδο, με αποτέλεσμα αλγόριθμοι που ακολουθούσαν τα gradients, όπως π.χ. ο PGD, αποτύγχαναν οικτρά να βρουν ανταγωνιστικές επιθέσεις. Ενώ πολλές άμυνες αποδείχθηκαν εκ των υστέρων ανούσιες, προσφέροντας μηδενική ευρωστία, η Ανταγωνιστική Άμυνα έχει αντέξει στο πέρασμα του χρόνου, αποτελώντας έναν από τους βασικότερους (και ασφαλέστερους τρόπους) που έχει κανείς στην φαρέτρα του ώστε να βελτιώσει την ευρωστία του συστήματός του.

Εμβαθύνοντας περαιτέρω στο θέμα αυτό, πρέπει να γίνει κατανοητό πως οποιαδήποτε επίθεση επιστρέφει εκτιμήσεις που πάσχουν από αυτό το πρόβλημα (καθώς εμείς λαμβάνουμε μόνο άνω φράγματα της πραγματικής μετρικής): το ζητούμενο είναι να βρούμε αξιόπιστους αλγόριθμους επίθεσης, που επιστρέφουν όσο πιο στενά άνω φράγματα γίνεται. Το πόσο αξιόπιστος είναι ένας αλγόριθμος για μια άμυνα είναι κάτι που μπορεί να αξιολογηθεί μόνο από το πέρασμα του χρόνου: Αν πολλές προσπάθειες εύρεσης πιο δυνατών επιθέσεων αποτυγχάνουν, τότε σημαίνει ότι ο αλγόριθμος έχει βρει αρκετά ικανοποιητικές εκτιμήσεις. Εδώ αξίζει να αναλύσουμε πιο διεξοδικά στις εκδοχές του προβλήματος της υπερεκτίμησης ευρωστίας που μπορεί να παρατηρήσει κάποιος στην αξιολόγηση μιας ανταγωνιστικής άμυνας. Πρώτον, υπάρχουν οι άμυνες που δεν είναι καθόλου εύρωστες και το μόνο που καταφέρνουν είναι να έχουν ισχυρή απόδοση απέναντι στις επιθέσεις που βασίζονται στα gradients. Για αυτές τις άμυνες, η βιβλιογραφία έχει προτείνει μια πληθώρα από τρόπους ανίχνευσης τους, π.χ. οι Athalye et al. [Ath+18] συστήνουν πως τέτοιες άμυνες μπορούν να ανιχνευθούν αν ο αλγόριθμος επίθεσης δεν καταφέρνει να βρει ανταγωνιστικά παραδείγματα ακόμα και αν μεγαλώσουμε ανεξέλεγκτα το φράγμα αναζήτησης ϵ ή οι αλγόριθμοι black-box είναι πιο επιτυχείς από τους white-box. Οι εν λόγω άμυνες δεν παρουσιάζουν κανένα ενδιαφέρον, αφού πλέον είναι εύκολα διαχωρίσιμες, επιστρατεύοντας αυτές τις τεχνικές ανίχνευσης. Εμείς, στην εργασία αυτή, θα ασχοληθούμε με την δεύτερη κατηγορία αμυνών: αυτές που προσφέρουν κάποιο βαθμό ευρωστίας, απλά βρίσκοντας ισχυρότερους αλγόριθμους επιθέσεις μπορούμε να προσφέρουμε πιο στενά άνω φράγματα της ℓ_p-ευρωστίας τους και επομένως, έχουμε την δυνατότητα να τις ταξινομήσουμε ορθότερα μεταξύ τους.

RobustBench Evaluation. Το προηγούμενο σκέλος της παρουσίασης αναδεικνύει εμφατικά το πόσο περίπλοκη διαδικασία είναι η αξιολόγηση της ℓ_p-ευρωστίας. Το σημαντικότερο βήμα προς την συστηματοποιήση της διαδικασίας αυτής έγινε από την δουλειά των Croce et al. [Cro+21]. Στην μελέτη αυτή, οι εν λόγω ερευνητές χτίζουν το RobustBench benchmark, όπου η αξιολόγηση των αμυνών γίνεται μέσω της επίθεσης AutoAttack, που έχει προταθεί από τους Croce and Hein [CH20b]. Η επίθεση AutoAttack, είναι μια συλλογή από τρεις white-box και μια black-box επιμέρους επιθέσεις. Σε πειράματα μεγάλης κλίμακας, αναδείχθηκε εμπειρικά πως αυτή η μέθοδος κατάφερνε συνεπώς να επιστρέψει πιο αξιόπιστα φράγματα για μια τεράστια συλλογή μοντέλων. Αυτή η συνέπεια στην ποιότητα της αξιολόγησης, ανεξαρτήτως του μοντέλου υπό αξιολόγηση, προσφέρθηκε ως ισχυρό εμπειρικό διαπιστευτήριο για την ικανότητα της μεθόδου να βρίσκει πιο δυνατές ανταγωνιστικές επιθέσεις, και επομένως να μπορεί να αξιοποιηθεί ως ασφαλέστερο εργαλείο αξιολόγησης της ℓ_p-ευρωστίας. Είναι σημαντικό, ωστόσο, να αποσαφηνίσουμε πως παρά την γενικότερη αποδοχή της αξιολόγησης μέσω της επίθεσης AutoAttack, είναι πιθανό η μέθοδος αυτή να αποτυγχάνει για ορισμένες άμυνες, π.χ. γι αυτές που εκμεταλλεύονται κάποιο αθέμιτο μέσο για να κερδίσουν εικονική ευρωστία, όπως π.χ. να χαλάσουν τα gradients που επιστρέφει το δίκτυο. Γι΄ αυτό ακριβώς τον λόγο, δεν πρέπει να θεωρούμε το AutoAttack ως πανάκεια, που μπορεί να βρίσκει πάντα αρκετά στενά φράγματα, ανεξαρτήτως μοντέλου, αλλά πιο πολύ ως μια αρκετά αξιόπιστη μέθοδο για να υποβάλλουμε την άμυνα μας από έναν πρώτο γύρο αξιολόγησης.

Εν κατακλείδι, η τυπική διαδικασία για την αξιολόγηση μιας νέας προτεινόμενης ανταγωνιστικής άμυνας πλέον περιλαμβάνει την υποβολή της άμυνας στο RobustBench benchmark. Οι ανταγωνιστικές άμυνες ταξινομούνται ως προς τον βαθμό ανθεκτικότητας τους απέναντι στην επίθεση AutoAttack, συνιστώντας έναν πίνακα βαθμολογίας στο επίσημο site.

1.3 Η συνεισφορά μας

Το επίκεντρο της μελέτης μας είναι ο αλγόριθμος Projected Gradient Descent (PGD) και η αξιοποίηση αυτού για την δημιουργία ℓ_p-φραγμένων ανταγωνιστών. Ο αλγόριθμος PGD απαρτίζεται από αρκετές σχεδιαστικές επιλογές οι οποίες δύνανται να επηρεάσουν σε μεγάλο βαθμό το τελικό αποτέλεσμα, ως προς την ισχύ των ανταγωνιστικών παραδειγμάτων. Οι 4 κύριες σχεδιαστικές επιλογές είναι:

- Το μέγεθος του βήματος η,
- Ο βελτιστοποιητής (optimizer),
- Η μέθοδος αρχικοποίησης του αλγορίθμου, δηλαδή από ποιο σημείο ξεκινά η αναζήτηση για την εύρεση ανταγωνιστικού παραδείγματος, και
- Η συνάρτηση κόστους που βελτιστοποιεί ο αλγόριθμος, η οποία αντικαθιστά την πραγματική μετρική που θέλουμε να ελαχιστοποιήσουμε, δηλ. την μη συνεχή μετρική 0-1, που παίρνει την τιμή 0 όταν το τελικό αποτέλεσμα κατηγοριοποιείται με διαφορετική ετικέτα από αυτήν της αρχικής εικόνας.

Το κύριο μέλημα μας είναι να κατανοήσουμε τον ρόλο της τέταρτης επιλογής, δηλαδή την επίδραση που ασκεί η αντικαταστάτρια αντικειμενική συνάρτηση στην απόδοση του αλγόριθμου. Ένα άκρως σημαντικό εύρημα, στο οποίο στηρίζουμε την πειραματική μας δουλειά, είναι αυτό των Croce and Hein [CH20b], όπου εξετάζουν την αποτελεσματικότητα 3 διαφορετικών συναρτήσεων κόστους, και το συμπέρασμα που μπορεί να εξαχθεί είναι πως καμία από τις τρείς επιλογές δεν είναι εκ των προτέρων ανώτερη από τις υπόλοιπες: Η επίδοση τους εξαρτάται από πολλούς παράγοντες, όπως το μοντέλο

υπό εξέταση, το σύνολο δεδομένων εκπαίδευσης κλπ. Οι τρεις διαφορετικές συναρτήσεις που χρησιμοποιούν οι Croce and Hein είναι: η cross-entropy (CE), η Carlini-Wagner (CW, γνωστή και ως margin) [CW17] και η Difference of Logits Ratio (DLR) που προτάθηκε από τους ίδιους [CH20b]. Οι συναρτήσεις αυτές δίνονται από τους παρακάτω τύπους:

$$\begin{aligned} \mathrm{CE}(\mathbf{z}, y) &= -\log p(y|\mathbf{x}) = -\mathbf{z}_y + \log \sum_{j=1}^{C} \exp(\mathbf{z}_j) \\ \mathrm{CW}(\mathbf{z}, y) &= -\mathbf{z}_y + \max_{j \neq y} \mathbf{z}_j \\ \mathrm{DLR}(\mathbf{z}, y) &= -\frac{\mathbf{z}_y + \max_{j \neq y} \mathbf{z}_j}{\mathbf{z}_{\pi_1} - \mathbf{z}_{\pi_3}} \end{aligned} \tag{1.8}$$

όπου $\mathbf{z}_\pi$: είναι το διάνυσμα των logits του δικτύου (δηλ. η αναπαράσταση του τελευταίου στρώματος στο δίκτυο), ταξινομημένη σε φθίνουσα σειρά.
Θέτοντας στο στόχαστρο μας αυτήν την αδυναμία μιας μοναδικής αντικαταστάτριας συνάρτησης να προσεγγίσει επαρκώς καλά την 0-1 μετρική με καθολικό τρόπο, δηλαδή για όλα τα διαφορετικά μοντέλα, η κινητήρια ιδέα της μελέτης μας είναι να συνδυάσουμε διαφορετικές αντικειμενικές συναρτήσεις στον αλγόριθμο PGD. Κατά αυτόν τον τρόπο, η διαδικασία θα γίνει πιο ανθεκτική απέναντι σε τυχόν επιβλαβείς παράγοντες, συνδεδεμένους με την γεωμετρία των επιμέρους συναρτήσεων, που μπορεί να επιδράσουν αρνητικά στο τελικό αποτέλεσμα.

Ο συνδυασμός διαφορετικών συναρτήσεων κόστους μπορεί να πραγματοποιηθεί με μια πληθώρα τρόπων, ωστόσο η δουλεία αυτής της διπλωματικής βασίζεται σε μια απλή ιδέα. Αντί να χρησιμοποιήσουμε μόνο μια συνάρτηση κόστους σε όλη την διάρκεια του PGD, χωρίζουμε τον αριθμό των επαναλήψεων T σε K (ίσα ως προς την διάρκεια) στάδια. Σε κάθε στάδιο, ο αλγόριθμος PGD βελτιστοποιεί και μια διαφορετική συνάρτηση κόστους, ξεκινώντας από το σημείο όπου τελείωσε το προηγούμενο στάδιο. Η διαδικασία αυτή, μπορεί να εκφραστεί εξής για K στάδια:

$$\mathcal{L}(\mathbf{x}, y) = \begin{cases} \mathcal{L}_1(\mathbf{x}, y), & \text{if } t < \frac{T}{K} \\ \mathcal{L}_2(\mathbf{x}, y), & \text{if } \frac{T}{K} \leq t < \frac{2T}{K} \\ \quad \vdots & \\ \mathcal{L}_K(\mathbf{x}, y), & \text{if } \frac{(K-1)T}{K} \leq t < T \end{cases}$$

Στην δουλειά μας θα θεωρήσουμε τις περιπτώσεις όπου $K = 2$ ή $K = 3$, χρησιμοποιώντας ως αντικειμενικές συναρτήσεις τις πιο κοινές επιλογές στην βιβλιογραφία, δηλαδή τις συναρτήσεις CE, CW και DLR.

Από εδώ και στο εξής, η μέθοδος αλλαγής αντικειμενικής συνάρτησης θα συμβολίζεται ως εξής: $\mathrm{PGD}_{\mathcal{L}_1 \& \mathcal{L}_2 \& \ldots \& \mathcal{L}_K}$, επομένως αν ο αλγόριθμος χρησιμοποιεί μόνο το CE loss θα αναφέρεται ως $\mathrm{PGD}_{\mathrm{CE}}$, ενώ αν αλλάζουμε το loss απο CE σε CW θα αναφέρουμε την μέθοδο ως $\mathrm{PGD}_{\mathrm{CE\&CW}}$.

1.3.1 Πειραματικό Σκέλος

Προτού εμβαθύνουμε στο πειραματικό σκέλος, θα παρουσιάσουμε την συλλογή από μοντέλα στα οποία ελέγξαμε την απόδοση των προτεινόμενων μεθόδων. Τα πειράματα μας διεξάγονται πάνω σε 15 μοντέλα τα οποία έχουν εκπαιδευτεί για να είναι εύρωστα απέναντι σε ℓ_∞-φραγμένες επιθέσεις, όπου ο επιτιθέμενος μπορεί να προσθέσει θόρυβο στην εικόνα με μέγιστη ℓ_∞-νόρμα ίση με $\epsilon = 8/255$. Τα μοντέλα που είναι προεκπαιδευμένα και εύκολα διαθέσιμα μέσω της βιβλιοθήκης RobustBench, προέρχονται από τις εξής ερευνητικές δουλειές: [Eng+19; Car+19; Hen+19; Zha+19a; Zha+19b; Wu+20;

Seh+22; AF20; Dai+22; Gow+21; Hua+21; Zha+21; RM21; Add+21; Seh+20]. Επειδή κάθε μια από τις παραπάνω δημοσιεύσεις μπορεί να συνδέεται με πολλά διαφορετικά μοντέλα, π.χ. επειδή ελέγχουν την εκάστοτε μέθοδο σε πολλές αρχιτεκτονικές ή σε διαφορετικά budget, στον παρακάτω πίνακα παραθέτουμε ακριβώς τα μοντέλα που χρησιμοποιούμε, συμπεριλαμβάνοντας το αναγνωριστικό τους (ModelID) από το οποίο συνοδεύονται στην βιβλιοθήκη RobustBench.

#	Paper	Model ID in RobustBench leaderboard	Standard Acc. (%)
1	[Eng+19]	Engstrom2019Robustness	87.03
2	[Car+19]	Carmon2019Unlabeled	89.69
3	[Hen+19]	Hendrycks2019Using	87.11
4	[Zha+19a]	Zhang2019You	87.20
5	[Zha+19b]	Zhang2019Theoretically	84.92
6	[Wu+20]	Wu2020Adversarial	85.36
7	[Seh+22]	Sehwag2021Proxy_R18	84.59
8	[AF20]	Andriushchenko2020Understanding	79.84
9	[Dai+22]	Dai2021Parameterizing	87.02
10	[Gow+21]	Gowal2021Improving_28_10_ddpm_100m	87.50
11	[Hua+21]	Huang2021Exploring_ema	91.23
12	[Zha+21]	Zhang2020Geometry	89.36
13	[RM21]	Rade2021Helper_R18_extra	89.02
14	[Add+21]	Addepalli2021Towards_RN18	80.24
15	[Seh+20]	Sehwag2020Hydra	88.98

Table 1.1. *Αντιστοίχιση των ερευνητικών δουλειών με τα ModelID των μοντέλων που χρησιμοποιούμε. Παραθέτουμε επίσης το ποσοστό επιτυχίας του κάθε μοντέλου στο καθαρό test-set του CIFAR-10.*

Σε αυτό το σημείο κρίνουμε σκόπιμο να παρουσιάσουμε ένα συνθετικό παράδειγμα, έχοντας ως στόχο να θεμελιώσουμε πιο διαισθητικά την μέθοδο μας και πως αυτή μπορεί να βελτιώσει τον κλασικό αλγόριθμο PGD για την εύρεση ανταγωνιστικών παραδειγμάτων. Ας θεωρήσουμε ένα παράδειγμα 2 διαστάσεων που σκοπός είναι να διαχωριστούν τα δεδομένα σε 3 κλάσεις. Θεωρούμε ένα γραμμικό μοντέλο κατηγοριοποίησης, το οποίο παίρνει εισόδους $\mathbf{x} = (x_1, x_2)^T$ και δίνει ως έξοδο το διάνυσμα των logits $\mathbf{z} = (z_1, z_2, z_3)^T$, που δείχνουν πόσο πιθανό είναι η είσοδος να ανήκει στην καθεμιά από τις 3 κλάσεις. Το γραμμικό μοντέλο υπολογίζει το διάνυσμα εξόδου μέσω ενός γραμμικού μετασχηματισμού: $\mathbf{z} = \mathbf{W}\mathbf{x}$, όπου:

$$\mathbf{W} = \begin{bmatrix} 0.3 & -0.3 \\ 1 & -0.01 \\ -0.25 & 0.75 \end{bmatrix}$$

Το σύστημα δέχεται ως είσοδο το σημείο $\mathbf{x} = (-0.45, -0.8)$, το οποίο προέρχεται από την κλάση y_1. Μέσω της γραμμικής σχέσης υπολογίζουμε πως $\mathbf{z} = (0.105, -0.442, -0.4875)$, επομένως το σύστημα κατηγοριοποιεί σωστά το δεδομένο εφόσον $z_1 > \max(z_2, z_3)$. Στόχος μας τώρα είναι να δημιουργήσουμε μια διαταραχή, που να απέχει το πολύ κατά 0.4 όσον αφορά την ℓ_2-νόρμα από την είσοδο $\mathbf{x}$, και να είναι τέτοια ώστε η νέα (διαταραγμένη) είσοδος που προκύπτει να αλλάζει την απόφαση του γραμμικού ταξινομητή. Για να το πετύχουμε αυτό, εκτελούμε τον αλγόριθμο PGD, χρησιμοποιώντας 50 επαναλήψεις με μέγεθος βήματος $\eta = 2 * 0.4 = 0.8$. Αρχικά, θεωρούμε 2 περιπτώσεις: Στην πρώτη, εκτελούμε τον αλγόριθμο χρησιμοποιώντας την αντικειμενική συνάρτηση CE, ενώ στην δεύτερη χρησιμοποιούμε την συνάρτηση CW. Για λόγους πληρότητας, σχεδιάζουμε την μορφή των level sets αυτών τον 2 συναρτήσεων στο πάνω μέρος της Εικόνας 1.3.

Το κάτω αριστερά σχήμα της Εικόνας 1.3 δείχνει τα αποτελέσματα της εκτέλεσης του αλγόριθμου PGD. Όπως φαίνεται, ενώ η συνάρτηση CE καταφέρνει να βρει μια διαταραχή που αλλάζει την απόφαση του classifier, η άλλη επιλογή αντικειμενικής συνάρτησης αποτυγχάνει καθώς τα gradients δείχνουν συνεχώς ως προς μια κατεύθυνση, η οποία επιστρέφει ένα σημείο που εξακολουθεί να κατηγοριοποιείται

σωστά. Αυτό είναι ένα απλό παράδειγμα του πως μπορεί να αποτύχει ο PGD όταν χρησιμοποίει μονάχα μια συνάρτηση κατά την διάρκεια της βελτιστοποίησης.

Όταν όμως εκτελούμε τον αλγόριθμο PGD σύμφωνα με την μέθοδο εναλλαγής που προτείνουμε (κάτω δεξιά σχήμα στην Εικόνα 1.3), η διαδικασία βρίσκει επιτυχώς μια διαταραχή που παραπλανεί το σύστημα. Αυτό συμβαίνει παρόλο που για το πρώτο μισό των επαναλήψεων ο αλγόριθμος χρησιμοποιεί το προβληματικό loss CW. Αυτό το παράδειγμα έρχεται να δείξει πιο απλουστευμένα πως η μέθοδος που προτείνουμε είναι ένας τρόπος να καταστήσουμε τον αλγόριθμο PGD πιο ανθεκτικό απέναντι στις υπερπαραμέτρους του, όπως είναι η αντικειμενική συνάρτηση που βελτιστοποιεί.

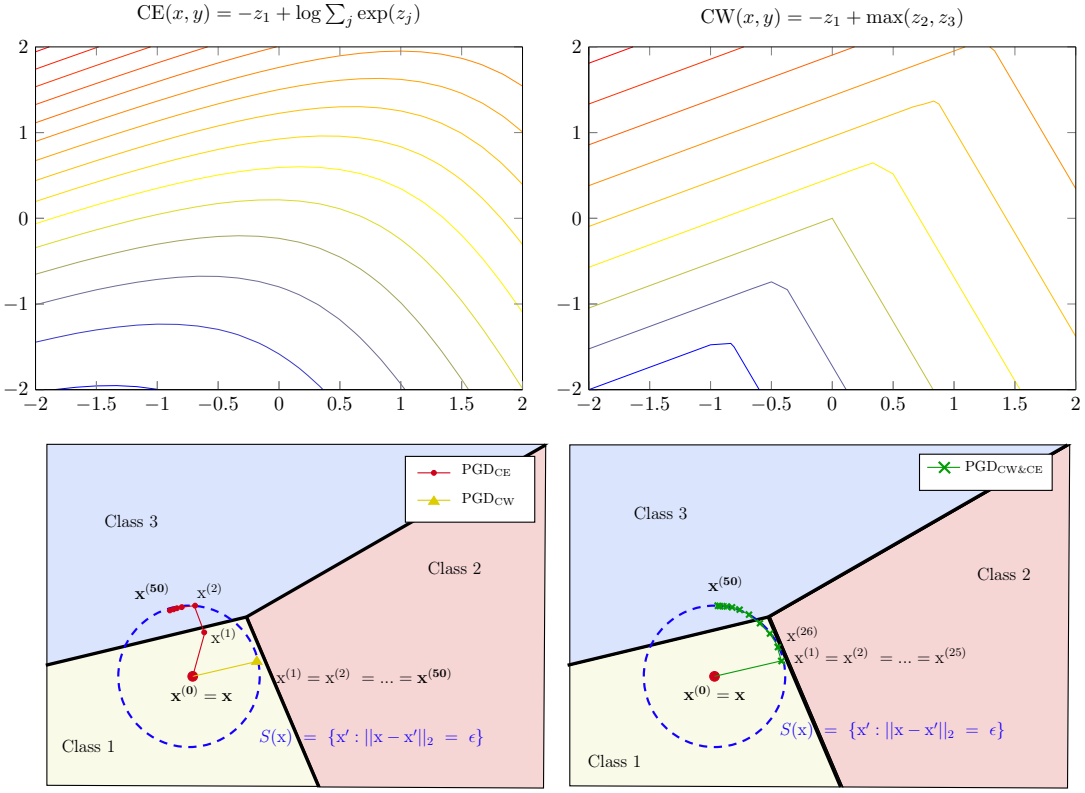


Figure 1.3. Επάνω σειρά: *Τα level sets των συναρτήσεων CE και CW, θεωρώντας ως πραγματική κλάση την* y_1. Κάτω σειρά: *(Αριστερά) Η αλληλουχία σημείων που επιστρέφει το PGD, δείχνοντας την πορεία της βελτιστοποίησης, για τις περιπτώσεις όπου εκτελούμε τον PGD με μονάχα ένα loss. Οι κόκκινοι κύκλοι συμβολίζουν την εκτέλεση με CE, ενώ το κίτρινο τρίγωνο αναπαριστά την εκτέλεση με CW. (Δεξιά) Η πορεία της βελτιστοποίησης του PGD όταν εναλλάσσουμε αντικειμενική συνάρτηση στα μέσα της διαδικασίας.*

Σαν πρώτο βήμα της πειραματικής ανάλυσης, εξετάζουμε την απολεσματικότητα της προτεινόμενης μεθόδου εναλλαγής αντικειμενικής συνάρτησης κατά την διάρκεια του PGD, σε σχέση με τις τυπικές εκδοχές του αλγορίθμου που χρησιμοποιούν μονάχα μια συνάρτηση. Τα αποτελέσματα φαίνονται στον Πίνακα 1.2, όπου αναδεικνύεται εμφανώς το πλεονέκτημα της μεθόδου μας. Συγκεκριμένα, όταν συνδυάζουμε την συνάρτηση CE με την CW (στήλη PGD$_{\mathrm{CE\&CW}}$) ή την DLR (στήλη PGD$_{\mathrm{CE\&DLR}}$) ή και τις 2 (στήλη PGD$_{\mathrm{CE\&CW\&DLR}}$), ο αλγόριθμος επιστρέφει εμφανώς καλύτερα αποτελέσματα από τα αντίστοιχα των εκδοχών που βελτιστοποιούν το ίδιο loss σε όλη την διάρκεια του αλγόριθμου.

Επεκτείνουμε την σύγκριση της μεθόδους μας με τις πιο δημοφιλείς ℓ_∞-φραγμένες επιθέσεις που υπάρχουν στην βιβλιογραφία. Στο πρώτο πείραμα, συγκρίνουμε την καλύτερη επίθεση μας (όπου κατά μέσο όρο είναι η PGD$_{\mathrm{CE\&CW\&DLR}}$) με τις white-box συνιστώσες της επίθεσης AutoAttack [CH20b], δηλαδή τις επιμέρους επιθέσεις APGD$_{\mathrm{CE}}$, APGD$_{\mathrm{DLR}}$ και την FAB [CH20a]. Σε αυτό το πείραμα, επίσης, κρατάμε σταθερό το βήμα μεγέθους όταν εκτελούμε την μέθοδο μας, ενώ ο optimizer μένει

	K=1			*K=2*			*K=3*
Model	PGD_{CE}	PGD_{CW}	PGD_{DLR}	$\text{PGD}_{\text{CE\&CW}}$	$\text{PGD}_{\text{CE\&DLR}}$	$\text{PGD}_{\text{CW\&DLR}}$	$\text{PGD}_{\text{CE\&CW\&DLR}}$
[Eng+19]	52.24	52.59	53.55	50.29	**50.22**	52.63	50.27
[Car+19]	62.09	60.86	61.16	60.00	60.00	60.88	**59.97**
[Hen+19]	57.38	56.61	57.47	55.41	55.37	56.55	**55.35**
[Zha+19a]	46.28	47.44	47.97	45.33	**45.32**	47.42	**45.32**
[Zha+19b] †	55.47	54.21	54.39	53.45	53.43	54.23	**53.41**
[Wu+20]	59.05	56.93	57.02	56.47	56.44	56.94	**56.42**
[Seh+22]	58.68	57.22	57.89	56.06	**56.05**	57.21	56.06
[AF20]	47.14	46.62	47.62	44.56	44.53	46.62	**44.50**
[Dai+22]	63.98	63.23	63.83	61.80	**61.76**	63.23	61.77
[Gow+21]	65.79	65.20	65.76	63.86	63.85	65.20	**63.84**
[Hua+21]	64.95	64.15	64.64	63.09	**63.03**	64.12	63.06
[Zha+21]	66.67	60.40	60.59	59.78	**59.69**	60.37	**59.69**
[RM21]	61.48	58.51	58.56	57.77	**57.74**	58.51	**57.74**
[Add+21]	56.00	51.88	51.97	51.45	51.43	51.86	**51.41**
[Seh+20]	59.86	58.41	58.57	57.66	**57.61**	58.40	**57.61**

Table 1.2. *Συγκρίνουμε τα αποτελέσματα του αλγορίθμου PGD όταν χρησιμοποιείται μονάχα ένα loss ($K = 1$) σε σχέση με την μέθοδο εναλλαγής που προτείνουμε. (†) Αυτό το μοντέλο επιτίθεται με φράγμα $\epsilon = 0.031$.*

και αυτός ίδιος με το αρχικό setting (δηλαδή το απλό gradient χωρίς momentum. Παραθέτουμε τα αποτελέσματα αυτής της σύγκρισης στον Πίνακα 1.3, όπου φαίνεται ξεκάθαρα πως η επίθεση $\text{PGD}_{\text{CE\&CW\&DLR}}$ δημιουργεί πιο δυνατά ανταγωνιστικά παραδείγματα, παρά το γεγονός πως την εκτελούμε με default υπερπαραμέτρους. Αυτή η σύγκριση επιδεικνύει την προοπτική συμπερίληψης της επίθεσής μας σε συλλογές επιθέσεων, όπως το AutoAttack, που συχνά χρησιμοποιούνται για την αξιολόγηση της ευρωστίας νευρωνικών δικτύων.

Model	APGD_{CE}	APGD_{DLR}	FAB	$\text{PGD}_{\text{CE\&CW\&DLR}}$	Δ
[Eng+19]	51.72	52.67	50.67	**50.27**	-0.40
[Car+19]	61.74	60.67	60.88	**59.97**	-0.70
[Hen+19]	57.23	57.03	55.55	**55.35**	-0.20
[Zha+19a]	46.15	47.39	45.83	**45.32**	-0.51
[Zha+19b] †	55.28	53.52	53.92	**53.41**	-0.11
[Wu+20]	58.90	56.68	56.82	**56.42**	-0.26
[Seh+22]	58.38	57.37	56.27	**56.06**	-0.21
[AF20]	46.93	47.08	44.72	**44.50**	-0.22
[Dai+22]	63.93	63.44	62.27	**61.77**	-0.50
[Gow+21]	65.63	65.14	64.14	**63.84**	-0.30
[Hua+21]	64.55	64.14	64.45	**63.06**	-1.08
[Zha+21]	66.37	60.19	59.97	**59.69**	-0.28
[RM21]	61.40	58.41	58.42	**57.74**	-0.67
[Add+21]	55.80	51.56	51.93	**51.41**	-0.15
[Seh+20]	59.60	58.29	58.29	**57.61**	-0.68

Table 1.3. *Σύγκριση της μεθόδου $PGD_{CE\&CW\&DLR}$ με τις white-box συνιστώσες της επίθεσης AutoAttack. Η στήλη Δ δείχνει την διαφορά στην επίδοση μεταξύ της μεθόδου μας και της καλύτερης επίθεσης από τις υπόλοιπες 3. (†) Αυτό το μοντέλο επιτίθεται με φράγμα $\epsilon = 0.031$.*

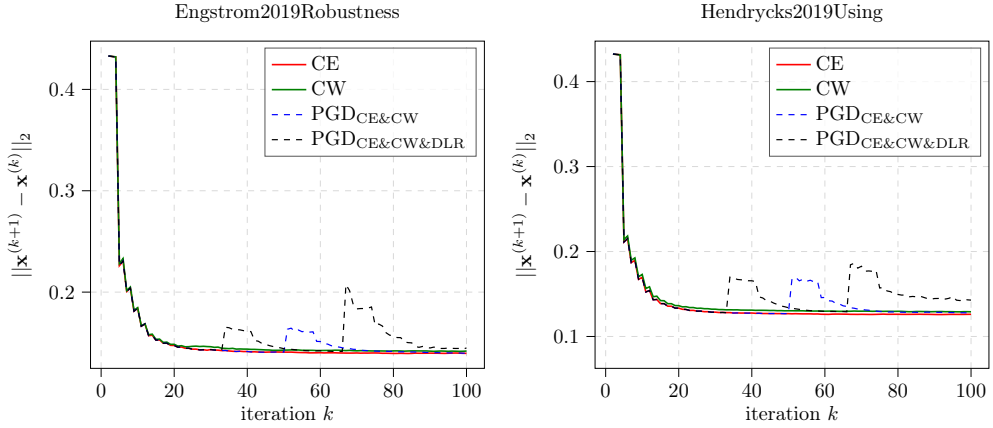
Στο δεύτερο πείραμα σύγκρισης της μεθόδου μας με δυνατά baselines της βιβλιογραφίας, επικεντρωνόμαστε σε 2 επιθέσεις, την επίθεση GAMA-PGD [Sri+20] και την επίθεση Margin Decomposition (MD) [Ma+20]. Επιλέγουμε αυτές τις 2 επιθέσεις διότι πετυχαίνουν state-of-the-art αποτελέσματα στο υπολογιστικό budget των $T = 100$ επαναλήψεων. Για να εγγυηθούμε ότι αυτή η σύγκριση γίνεται με πιο δίκαιους όρους, αλλάζουμε το βήμα μεγέθους στην μέθοδό μας ακριβώς όπως το κάνουν σε αυτές τις 2 επιθέσεις. Στον Πίνακα 1.4 παρουσιάζουμε την σύγκριση της επίθεσης μας $\text{PGD}_{\text{CE\&CW\&DLR}}$ με αυτά τα 2 baselines. Τα αποτελέσματα δείχνουν πως η επίθεσή μας καταφέρνει

να ξεπεράσει και τις 2 μεθόδους στην πλειοψηφία των μοντέλων. Συγκεκριμένα, είναι καλύτερη από την μέθοδο GAMA-PGD σε 11 εκ των 15 μοντέλων, ενώ νικάει την επίθεση MD σε 13 από τα 15 δίκτυα. Έπειτα από την διεξαγωγή αυτών των 2 συγκρίσεων, γίνεται εμφανής η αποτελεσματικότητα της εναλλαγής αντικειμενικών συναρτήσεων κατά την διαδικασία βελτιστοποίησης του PGD.

Model	GAMA-PGD [Sri+20]	$\text{PGD}_{\text{CE\&CW\&DLR}}$ (GAMA-PGD sch.)	Δ	MD Attack [Ma+20]	$\text{PGD}_{\text{CE\&CW\&DLR}}$ (MD sched.)	Δ
[Eng+19]	50.05	**49.88**	-0.17	50.34	**49.87**	-0.47
[Car+19]	59.84	**59.78**	-0.06	59.83	**59.72**	-0.11
[Hen+19]	**55.22**	55.26	+0.04	**55.15**	55.20	+0.05
[Zha+19a]	45.32	**45.20**	-0.12	45.49	**45.17**	-0.32
[Zha+19b]†	53.29	53.29	0	53.36	**53.26**	-0.10
[Wu+20]	56.30	56.30	0	56.28	**56.26**	-0.02
[Seh+22]	56.01	**55.95**	-0.06	55.92	**55.89**	-0.03
[AF20]	44.42	**44.41**	-0.01	44.57	**44.44**	-0.13
[Dai+22]	61.94	**61.74**	-0.20	61.99	**61.72**	-0.27
[Gow+21]	63.78	**63.72**	-0.06	63.94	**63.73**	-0.21
[Hua+21]	**62.87**	62.89	+0.02	62.93	**62.86**	-0.07
[Zha+21]	60.72	**59.62**	-1.10	59.73	**59.58**	-0.15
[RM21]	57.78	**57.73**	-0.05	57.74	**57.72**	-0.02
[Add+21]	51.43	**51.26**	-0.17	51.30	**51.25**	-0.05
[Seh+20]	57.49	**57.43**	-0.06	**57.31**	57.45	+0.14

Table 1.4. *Σύγκριση της επίθεσής μας, $PGD_{CE\&CW\&DLR}$ με τα δυνατότερα baselines της βιβλιογραφίας, δηλ. τις επιθέσεις GAMA-PGD και Margin Decomposition. Η στήλη Δ υποδηλώνει την διαφορά επίδοσης μεταξύ της επίθεσής μας και του εκάστοτε baseline. (†) Αυτό το μοντέλο επιτίθεται με φράγμα $\epsilon = 0.031$.*

Εκτός των παραπάνω πειραμάτων, που έχουν σκοπό να κατατάξουν την επίθεση μας ανάμεσα σε άλλες δυνατές επιθέσεις της βιβλιογραφίας, επεκτείνουμε την πειραματική μας ανάλυση ώστε να καταλάβουμε σε μεγαλύτερο εύρος τους λόγους για τους οποίους η μέθοδός μας καταφέρνει να πετύχει καλά αποτελέσματα. Συγκεκριμένα, διεξάγουμε το εξής ποιοτικό πείραμα: Σχεδιάζουμε την ℓ_2-απόσταση μεταξύ των διαδοχικών σημείων που επισκέπτεται ο PGD κατά την εκτέλεση του. Παραθέτουμε αυτό το σχήμα για 4 περιπτώσεις αντικειμενικών συναρτήσεων: Την CE, την CW και τις 2 που προτείνουμε εμείς, δηλαδή την εναλλαγή συναρτήσεων, CE&CW και CE&CW&DLR. Το πείραμα αυτό πραγματοποιείται για 4 διαφορετικούς classifiers, δίνοντας το οπτικό αποτέλεσμα που φαίνεται στην Εικόνα 1.4. Από εκεί, αυτό που συμπεραίνουμε είναι ότι η εναλλαγή συναρτήσεων πιθανώς είναι ωφέλιμη καθώς παρακινεί τον αλγόριθμο να ψάξει πιο εκτενώς τον χώρο, και να επισκεφτεί σημεία που είναι πιο μακρινά μεταξύ τους.



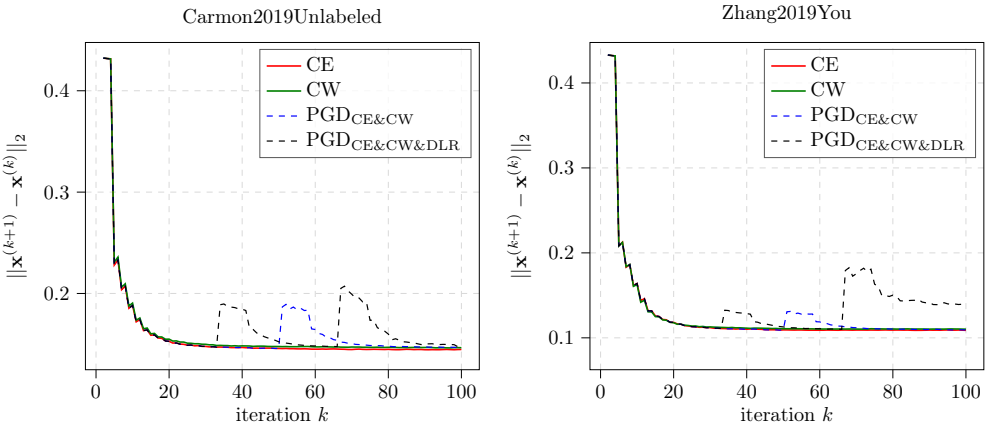


Figure 1.4. *Σχεδίαση της ℓ_2-απόστασης μεταξύ διαδοχικών σημείων που βρίσκει ο PGD. Κάθε σχήμα αναπαριστά αυτήν την απόσταση συναρτήσει των επαναλήψεων του αλγόριθμου, για 4 διαφορετικούς classifiers (Παραθέτουμε το ModelID κάθε classifier σαν τίτλο σε κάθε σχήμα.*

Chapter 2

Introduction

2.1 Motivation

The dawn of Deep Learning triggered a paradigm shift to the approaches deployed to solve some of the most interesting and challenging technological tasks. Deep Neural Networks (DNNs) have consolidated as the de facto standard to tackle problems originating from every possible knowledge domain, including data of visual, linguistic or acoustic nature (among others). The universal adoption of Deep Learning, of course, is justified by both their exceptional performance and the fact that they are conceptually simple, constituting of few building blocks (neurons/connections, gradient descent, backpropagation) which makes it easy for someone to manipulate them without a great amount of effort.

Despite their numerous advantages, Deep Learning models operate in counter-intuitive ways which prevents us from truly grasping their inner mechanisms sufficiently well and indicate important blind spots that create a chasm between our understanding and their genuine functionality. One rather worrisome phenomenon associated with them is the presence of *adversarial examples*, discovered by Szegedy et al. [Sze+14]. In their seminal work, these researchers demonstrated that a tiny amount of carefully designed noise, unintelligible to a human observer, can steer the system towards classifying the image as belonging to a wrong class, whereas a human can effortlessly assign the correct label to the image. The susceptibility of DNNs against such perturbations casts doubt about their potential in performing more elaborate tasks that require reasoning.

An immediate corollary of this intriguing finding is that researchers steered their attention towards the direction of increasing the resilience of systems against such examples. The general term *Adversarial Defense* connotes systems that aspire to overcome the obstacle of adversarial examples. *Adversarial Attacks*, on the other hand, is a term that refers to the algorithms that produce such examples. Attacks and Defenses participate in an arms race, where one attempts to enhance the robustness of its model against current attacks by exploiting them and subsequently, novel attacks are devised in order to circumvent these defenses.

Recall that typical adversarial examples must have the property of being indistinguishable with their unperturbed counterparts. Researchers modelled this property through the ℓ_p-bounded threat model, where the attacker's search space $\Delta(\mathbf{x})$ is reduced to the ℓ_p-ball around the clean data point x of radius ϵ:

$$\Delta(\mathbf{x}) = \{\mathbf{x}' \in \mathcal{X} : \|\mathbf{x}' - \mathbf{x}\|_p \leq \epsilon\}$$

Measuring the degree of ℓ_p-bounded robustness, for the classifier f, can be done through the

calculation of *robust accuracy*, where $\mathcal{D}$ denotes a held-out test set:

$$\text{RobAcc}(f) = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \min_{\mathbf{x}'_i : \|\mathbf{x}_i - \mathbf{x}'_i\|_p \leq \epsilon} \mathbb{1}[f(\mathbf{x}'_i) = y_i]$$

The exact solution of this equation, however, is intractable for complex function classes as those represented by Deep Neural Networks. One needs to resort in adversarial attacks to obtain local minimizers of this expression, hopefully yielding tight enough upper bounds. The issue of *robustness overestimation* [Ath+18; Ues+18; Tra+20] refers to the scenario where the defender chooses an inappropriate algorithm to evaluate robustness, thus the reported robust accuracy is unreasonably high and doesn't faithfully reflects the genuine effectiveness of the proposed method. Ultimately, it becomes evident that finding powerful adversarial attacks is of paramount importance, enabling us to reliably estimate whether a defense algorithm is valuable.

Projected Gradient Descent (PGD) [Kur+17; Mad+18] is the most popular attack algorithm to evaluate the ℓ_p-bounded robustness of deep networks against adversarial examples. In short, PGD can be formulated as follows:

$$\mathbf{x}^{(t+1)} = \mathcal{P}_{\Delta(\mathbf{x})}\left[\mathbf{x}^{(t)} + \eta^{(t)}\boldsymbol{\delta}^{(t)}\right] \tag{2.1}$$

where $\mathbf{x}^{(t)}$: the iterate, $\eta^{(t)}$: step size, $\boldsymbol{\delta}^{(t)}$: update rule of t-th iteration and $\mathcal{P}_{\mathcal{S}}$: the projection operation, which maps the updated iterate into the feasible region $\mathcal{S}$, which in our case is the ℓ_p-ball of radius ϵ around $\mathbf{x}$. For a more thorough analysis on the aforementioned expression, we refer the reader to the study of Gowal et al. [Gow+19] which contains a highly illustrative pseudoalgorithm.

2.2 Thesis Contribution

The performance of Projected Gradient Descent (PGD) [Kur+17; Mad+18], in the context of adversarial examples' generation, is influenced by 4 hyperparameters (assuming fixed computational budget): The step size, the initialization strategy, the optimizer, and the surrogate loss. In our thesis, we aspire to gain further understanding on the impact of surrogate loss. Our work is empirically underpinned by a remarkable observation whose significance has been neglected in previous related research. Croce and Hein [CH20b] illustrate that there is no option for the objective function which is able of delivering equally good results, indiscriminately of which classifier is evaluated. We build on this finding to propose the following view: Combining different objectives in the same run of PGD could benefit optimization since it would "robustify" the method against poor selection of surrogate loss. In our study, we experiment with three ways of aggregating different objectives (in the form of ablation experiments) and the results indicate that a simple loss alternation during PGD is highly performant. The importance of our proposed method's performance is primarily studied through three baseline experiments in 15 different ℓ_∞-bounded CIFAR-10 robust models: First, alternating surrogate losses during PGD is significantly better than using a single loss. Then, we compare our method with the three white-box components from AutoAttack [CH20b], that is APGD with CE and DLR losses [CH20b] and FAB attack [CH20a], where we observe that our attack is consistently better across the entirety of our model collection. Finally, we find that our adversarial attack is better than two of the strongest attacks on the literature (for the budget of our study, that is $T = 100$ iterations): GAMA-PGD [Sri+20] and Margin Decomposition (MD) attack [Ma+20]. Additionally, we provide further qualitative analysis indicating that

the alternation of objectives may be advantageous because it motivates PGD to extend its search space, visiting more distant intermediate points during optimization.

2.3 Thesis Outline

The remaining content of this present dissertation has been divided into four chapters:

- In **Chapter** 3 we represent the most essential notions revolving the field of Machine Learning, demarcating the minimum background that the reader should possess in order to comfortably follow the concepts discussed in following chapters (experienced readers may skip this),
- In **Chapter** 4 we delve into Adversarial Machine Learning, which constitutes the broader topic of this thesis, introducing the main work associated with Adversarial Attacks and Defenses,
- In **Chapter** 5 we demonstrate the main contribution of this thesis. Essentially, we represent an extensive experimental analysis which provides enough empirical evidence to verify the effectiveness of our proposal of alternating surrogates,
- In **Chapter** 6 we summarize the implications of our findings and provide additional discussion about several subsequent research steps which emerge from our work.

Chapter 3

Machine Learning

3.1 Introduction

Machine Learning (ML) studies the development of systems which are endowed with the ability to perform a wide variety of tasks through learning algorithms: Instead of the basic paradigm of algorithms, where the computer program is performing a task through executing a sequence of basic instructions, learning algorithms is a different term that is used to describe the process wherein the system assimilates natural data and learns to operate, without the need for human intervention, based on the experience it has gained through getting familiar with these data. In an effort to prescribe a compact definition for ML, we find it expedient to quote Tom Mitchell's definition [Mit97]:

> *A computer program is said to learn from experience E with respect to some class of tasks T, and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*

The above definition makes it clear that there are many different flavours of ML: Based on the nature of the task at hand, the performance measure and the type of data which we leverage to make our system learn, a distinct subfield of ML emerges.

One of the most strong appeals of studying ML is the numerous applications, on a diverse array of scientific fields, where it has given substantial solutions: In Computer Vision, ML algorithms have enabled computers to recognize objects, classify them into different categories or even produce novel instances of images that seem to be completely natural. In Speech Processing, ML has successfully been used to transcribe speech in challenging auditory conditions, while in Natural Language Processing (NLP), ML systems can learn to semantically parse huge quantities of text or even translate text from one language to another. Those are only a grain of the innumerable applications where ML has attained to induce unprecedented progress.

The vast majority of those intriguing practical applications were only successfully tackled with the advent of Deep Learning (DL). Deep Learning is a new scientific field which experienced an outburst around a decade ago, when advances in parallel computing made it possible to train massive models, triggering an unprecedented surge of interest. DL studies the development of models where their functionality has the Artificial Neural Network (ANN) (and its variants) as its main building block which we will analyze it to a great extent. An insightful perspective of DL models is that of representation learning: they ingest natural, high-dimensional data and they learn to project them to a compact representation of significantly lower dimensions.

3.2 Types of Learning Algorithms

In this section, we attempt to separate the different kinds of problems that one can come across in Machine Learning. In general, ML could be divided into 3 major categories: Supervised, Unsupervised and Reinforcement Learning. The discriminating factors are mainly associated with: (a) the nature of data that the algorithms of each category exploit in order to learn about the world and (b) their ultimate objective .

3.2.1 Supervised Learning

The ultimate paradigm of ML is that of Supervised Learning. The term "supervision" is inextricably linked with the type of data that the learner has at its disposal. Supervised Learning tackles problems where there is an input $\mathbf{x} \in \mathcal{X}$ and output variable $\mathbf{y} \in \mathcal{Y}$, where $\mathcal{X}, \mathcal{Y}$ are the spaces where the input and output to "live". Presumably, for any given task, there is a true mapping $f_{\text{tr}} : \mathcal{X} \to \mathcal{Y}$ that links every input to the respective output. The end goal of supervised tasks is to learn that mapping, by approximating it through an estimator $f : \mathcal{X} \to \mathcal{Y}$. The learning procedure (also called training) is driven from the training set $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_N, \mathbf{y}_N)\}$, which essentially is an array of humanly labeled input-output pairs and resembles the notion of experience which appears in the definition that we mentioned earlier.

3.2.2 Unsupervised Learning

On the other hand, Unsupervised Learning algorithms manipulate data that they are not labeled; the observed inputs are simple vectors of some natural signal: $\mathcal{D} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$. In Supervised Learning, the goal is much more definite to articulate; one desires to estimate the true function that maps the input to the corresponding output. Here, this type of learning encompasses different kinds of purposes about what is the desired goal. A vague phrase that could partially explain the aim of Unsupervised Learning algorithms is that they aspire to make sense of the data. Just to satisfy the reader's curiosity, this "umbrella" phrase includes applications such as data clustering, identifying latent factors of variation in the inputs, or even building probabilistic models that learn a probability density function for the data.

3.2.3 Reinforcement Learning

The third most essential instance of ML research is Reinforcement Learning (RL). The discipline of RL is concerned with the construction of systems, which are often called agents, that interact with their environment. Essentially, the agent learns a policy function that informs him about the most favourable action that it can take, based on its current state. RL agents learn by receiving data which associate each state,action pair with a reward, quantifying the appropriateness of its move. RL has been explored in many exciting applications, with the most vivid one being this of creating systems that have the ability to play Atari games in human-like performance.

3.3 Basic Concepts in Machine Learning

This section is devoted in shedding light to some of the most essential body of knowledge that someone should possess when it comes to ML research. The discussion will be primarily seen, for the sake of representational clarity, through the lens of Supervised Learning, which is the most

basic example of ML applications, but that doesn't mean that those concepts are only restricted to this kind of problems.

3.3.1 Loss Function

The learning of an estimator f can be posed as an optimization problem, by introducing the notion of the loss function. Assuming that the estimator is a parametric function $f(\mathbf{x};\boldsymbol{\theta})$ (where $\boldsymbol{\theta}$: its parameters) and $\hat{\mathbf{y}} = f(\mathbf{x};\boldsymbol{\theta})$ is the prediction of the actual output, then the loss function (also called per-example loss) $\ell(\mathbf{y},\hat{\mathbf{y}};\boldsymbol{\theta})$ is a metric that measures the ability of the predictor to reliably estimate the true output $\mathbf{y}$. The choice of an appropriate loss function depends on the respective task and will be fully understood in the ensuing conversation, in the context of specific instances of learning problems. Ideally, if we had access to the true data distribution p_{data} that generates the data, learning the estimator f would amount to minimizing the *expected risk*, which is the expected per-example loss over the true distribution. However, since in practical scenarios we only have access to a finite number of samples from this distribution, learning the estimator f is equivalent with optimizing the *empirical risk*, which is a Monte Carlo approximation of expected risk:

$$\mathcal{L}(\mathbf{x},\mathbf{y};\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{(\mathbf{x},y)\sim p_{\text{data}}}[\ell(f(\mathbf{x};\boldsymbol{\theta}),\mathbf{y})] \approx \frac{1}{N}\sum_{i=1}^{N}\ell(f(\mathbf{x}_i;\boldsymbol{\theta}),\mathbf{y}_i)$$

Then the learning problem can be viewed as finding the parameters $\boldsymbol{\theta}$ which minimize the empirical risk:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \frac{1}{N}\sum_{i=1}^{N}\ell(f(\mathbf{x}_i;\boldsymbol{\theta}),y_i)$$

3.3.2 Generalization, Underfitting and Overfitting

The preceding paragraph clarified how it is possible to exploit the training set in order to learn a reliable estimator. However, even if the training procedure yields parameters that achieve adequately low empirical risk, the aspiration of ML algorithms is to develop models that perform well on unseen inputs. Those unseen inputs are encompassed in a held-out set which is not present during training, colloquially termed as the test set. The ability to perform well on those unprecedented instances is called *generalization*. Previously, we mentioned that the learning problem can be deemed as optimizing the empirical risk; however, this is not the entire picture. Our goal is to simultaneously achieve both low training and test error. Therefore, based on our desire to enhance our model's ability to perform well on the test set, we identify two particularly concerning scenarios, where their emergence in practice indicates that we should improve our learning algorithm:

- The *underfitting* problem, where our model fails to achieve a sufficiently low train error.
- The *overfitting* problem, where our model's parameters induce a large gap between train and test error.

Both of these problems are tightly intertwined with the rather abstract notion of *capacity*. The capacity of an ML model roughly represents the ability of the model to learn arbitrarily complex functions. In practice, we can increase our model's capacity by inflating the number of parameters. The presence of underfitting implies that our model has too small capacity, whereas

the manifestation of overfitting informs us that our model has an exaggerated ability to learn which led it to perfectly memorize the training set. Regulating the amount of model capacity to reach the sweet spot where the generalization gap minimizes is a nuisance in ML research. Figure 3.1 perfectly visualizes the underfitting-overfitting trade-off which is induced by altering the model capacity.

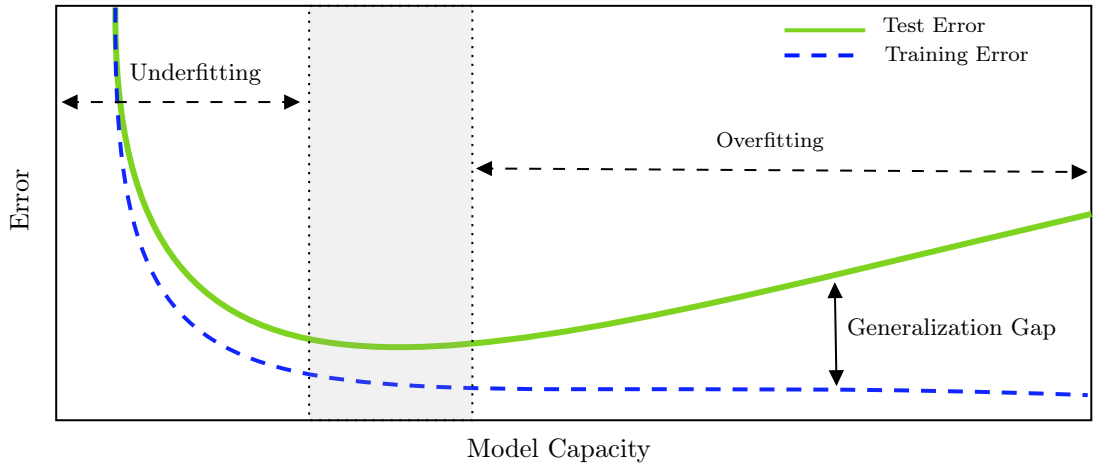


Figure 3.1. *The shade area indicates a "sweet" interval where the generalization gap is sufficiently low. Increasing the capacity beyond that area means that we enter the overfitting regime, whereas lower capacity means that our model suffers from the problem of underfitting.*

3.3.3 Regularization

An efficient remedy to the problem of overfitting is *regularization*. As Goodfellow et al. [Goo+16] put it: *"Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error"*. An expedient way to perform modifications to the learning algorithm is through its training objective; instead of solely optimizing the training error, the objective is augmented with an additional penalty term $\mathcal{R}(\boldsymbol{\theta})$, yielding the following minimization problem:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) + \lambda \mathcal{R}(\boldsymbol{\theta})$$

where λ adjust the strength of the regularization term. The most popular regularization function choices arise from ℓ_p euclidean norms, i.e. $\mathcal{R}(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_p$. L_1 regularization (or LASSO) produces sparse solutions, whereas L_2 regularization (or weight decay) restricts the parameters to have small magnitude. From a Bayesian Learning perspective, these two regularization terms can be considered as obtaining a Maximum a Posteriori (MAP) estimate for the parameter vector $\boldsymbol{\theta}$, where we have imposed $p(\boldsymbol{\theta})$ to be distributed according to the Laplacian (in the L1 case) or the Gaussian density (in the L2 case). In the context of Deep Learning, as we shall mention later on, there are even more ploys which someone can leverage in order to perform regularization, such as Dropout and Early Stopping.

3.4 Examples of Learning Algorithms

In this section, the reader will be succinctly introduced to some basic examples of ML, spanning from the supervised to the unsupervised setting.

3.4.1 Classification

It is practically infeasible to speak for Supervised Learning without discussing about Classification. In this setting, our goal is the categorization of the inputs to some category; For example, we may want to build a system that learns to classify images into the correct class among a predefined set of classes, or a system that ingests text and decides whether this chunk of text expresses positive or negative thoughts. Essentially, in the classification context, the output space $\mathcal{Y}$ is identical to a discrete set containing the K possible classes: $\mathcal{Y} = \{1, ..., K\}$. Here, the loss function that we are interested to minimize is the $0-1$ loss: $\ell_{0-1}(y, \hat{y}) = \mathbb{1}[y = \hat{y}]$.

A nice way to become acquainted with classification is the paradigm of the binary case, where the goal is to classify each input as positive (assuming $y = 1$) or negative (assuming $y = 0$). Rosenblatt [Ros58] proposed a seminal idea to solve the binary classification, introducing the notion of perceptron. The perceptron learns to classify each input $\mathbf{x} \in \mathbb{R}^d$ by linearly combining each element of the input vector with the parameters $\mathbf{w}$ and then applying a threshold function, hence the classifier is formally expressed as:

$$f(\mathbf{x}, \mathbf{w}) = \text{sgn}(\mathbf{w}^T\mathbf{x} + b)$$

The expression inside the sign function characterizes a hyperplane with normal vector $\mathbf{w}$ with a distance of b from the reference point. The vector $\mathbf{w}$ can be randomly initialized and then iteratively refined based on the update rule: $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t(\hat{y}_t - y)\mathbf{x}_i$, where η_t is the learning rate or the step size and (x_t, y_t) is the input-output pair presented at the algorithm in $t-$th timestep. The limitation of this approach is that it can only provide adequate solutions for data that are linearly separable, something that it is untrue for modern, high dimensional data like images or speech signals.

3.4.2 Linear Regression

Regression is another exemplar of Supervised Learning. In this task, our goal is to transform the input into a single number $y \in \mathbb{R}$; for example, we may want to predict the projected GPA of a master student (this is the output variable) based on its undergraduate GPA, TOEFL and GRE scores. For regression, the most dominant option for the loss function is the ℓ_2 loss: $\ell_2(y, \hat{y}) = (y - \hat{y})^2$. The empirical risk when using the ℓ_2 loss is called the Mean Square Error (MSE). Linear Regression is a well-studied mathematical problem and it refers to the scenario where we assume that the inputs can be linearly combined to predict the outputs: Consider that we dispose n training inputs $\mathbf{x} \in \mathbb{R}^d$ alongside their respective outputs $\mathbf{y} \in \mathbb{R}$. The inputs are stacked together to the training data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, whereas the training outputs are stacked into the output vector $\mathbf{y} \in \mathbb{R}^n$. In Linear Regression, our model is represented through the parameter vector $\mathbf{w} \in \mathbb{R}^d$ and the predicted output is calculated as $y = \mathbf{w}^T\mathbf{x}$. The optimal parameters can be found as the global minimum of the MSE objective:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n}(y - \mathbf{w}^T\mathbf{x})^2 = \arg\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$$

The above optimization problem can be analytically solved to yield the following solution:

$$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

The linear dependence on the input vector significantly restricts the capacity of the model. The easiest way to bypass this unreasonable restriction is to project the data into a higher dimensional space through a function $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^m$, obtaining the transformed data matrix $\boldsymbol{\Phi} \in \mathbb{R}^{n \times m}$. In this case, we treat the problem the same way as before, i.e. we consider that the outputs can be expressed as: $y = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$, where $\mathbf{w} \in \mathbb{R}^m$ acts linearly on the transformed input. Analytically, it follows that even in this case, the optimal parameters are easily calculated as:

$$\mathbf{w}^* = (\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \mathbf{y}$$

The increased capacity may inevitably induce the problem of overfitting. As we already discussed, an effective countermeasure is to regularize our solution, for instance with the weight decay term. In this case, the parameters can be found from the solution of the following problem:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \Big\{ \frac{1}{n} \sum_{i=1}^{n} (y - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}))^2 + \lambda \mathbf{w}^T \mathbf{w} \Big\} = \arg\min_{\mathbf{w}} \{ \|\mathbf{y} - \boldsymbol{\Phi}\mathbf{w}\|_2^2 + \lambda \mathbf{w}^T \mathbf{w} \}$$

The analytical solution is still straightforward, without being greatly affected by the introduction of the regularization term:

$$\mathbf{w}^* = \Big[(\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T + \lambda \mathbf{I} \Big] \mathbf{y}$$

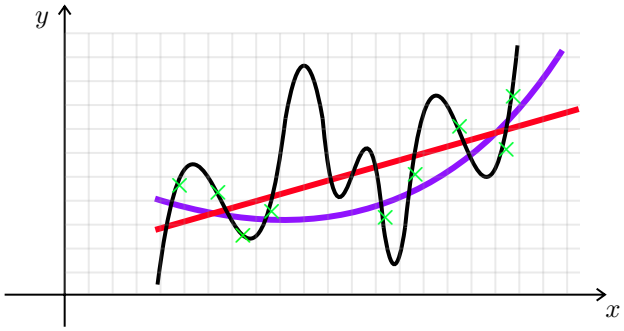


Figure 3.2. *The red line interpolates the data with a simple line $y = ax + b$, whereas the purple curve fits the data through a quadratic polynomial. Increasing the model capacity (black curve) leads to a solution that perfectly fits the data points but it is very unlikely to interpolate unseen data smoothly.*

3.4.3 Clustering

Data clustering is maybe the most instructional way to introduce someone to Unsupervised Learning. Here, the goal is to partition the input space into groups, where each group ideally contains inputs that are semantically similar. The most simplistic approach to perform data clustering is the K-Means algorithm: The clusters in this case are represented by K centroid points, randomly initialized, which are then iteratively refined based on the inputs belonging to the cluster in that specific timestep. This algorithm performs hard clustering, where it is presumed that each data point can only be member of exactly one cluster. The Gaussian Mixture Model (GMM) fits to the training data a mixture of Gaussian distributions. The optimal mixture parameters are obtained through Maximum Likelihood Estimation and once they are calculated, a data point can be stochastically assigned to each cluster through a probability that represents the likelihood of that point belonging to a specific cluster.

3.5 Deep Learning

Deep Learning has taken the world by storm in the last decade since the work of Krizhevsky et al. [Kri+12], when they were the first to achieve the training of a large deep neural network that led to state-of-the-art results in the ILSVRC2012 image classification competition. Since then, Deep Learning holds the lion share of research in the ML community and it has been adopted to tackle challenging problems with exceptional success. This section is dedicated to represent the nuts and bolts of DL; from the underlying architectures to the training procedure that efficiently fits massive amounts of data to DL models.

3.5.1 Deep Learning Basics through the Paradigm of MLP

Multi-Layer Perceptrons

In preceding sections, we discussed about the perceptron introduced by Rosenblatt [Ros58]. This algorithm produces an output by multiplying each element x_i of the input vector $\mathbf{x} \in \mathbb{R}^d$ with the respective weight w_i of the parameter vector $\mathbf{w}$, then adding a bias b and applying a threshold function. In the case where we want to produce an output vector $\mathbf{y} \in \mathbb{R}^C$ instead of a single number (as it happened in the binary classification case), we can apply this linear operation C times (where C: the number of outputs), with different weights for each output element. This operation can be compactly expressed through the weight matrix $\mathbf{W} \in \mathbb{R}^{C \times d}$ and the bias vector $\mathbf{b} \in \mathbb{R}^C$: $\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b}$. Visually, the above operation can be demonstrated as follows:

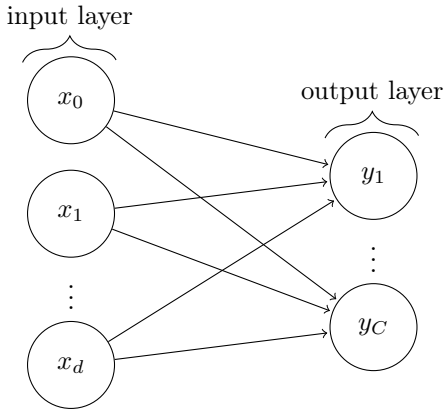


Figure 3.3. *A perceptron with many outputs. Figure was obtained from David Stutz's blogpost*

In the above structure, each unit (or node) is called *neuron* (a term loosely based on neuroscience) because it receives input from many other units and produces its own activation value which is the result of the linear operation. In Multi-Layered Perceptrons (MLPs), neurons are grouped together in layers and those that belong in the same layer are unable to interact with each other. For instance, notice the lack of connections between neurons in the input layer in Figure 3.3. The simple structure of the above figure assumes that the inputs and outputs are immediately connected. However, it is possible to introduce many *hidden layers* between the input and output layers to increase the expressive capabilities of our models (c.f. Figure 3.4). Considering L hidden layers, each layer can be represented by the vector of its hidden units $\mathbf{y}^{(i)} \in \mathbb{R}^{m^{(i)}}$, $i = 1, ..., L$. The activations of hidden units can be obtained as in the case of two layers, where the input is received from the neurons of the preceding layer: $\mathbf{y}^{(i)} = \mathbf{W}^{(i)}\mathbf{y}^{(i-1)} + \mathbf{b}^{(i)}$. The final output is

differentiable w.r.t. to the parameter vector $\boldsymbol{\theta} = \{\mathbf{W}^{(i)}, \mathbf{b}^{(i)}\}_{i=1}^{L}$ and it can be obtained through the following expression:

$$\mathbf{y} = \mathbf{W}^{(L)}(\mathbf{W}^{(L-1)}(...(\mathbf{W}^{1}\mathbf{x} + \mathbf{b}^{(1)})...) + \mathbf{b}^{(L-1)}) + \mathbf{b}^{(L)}$$

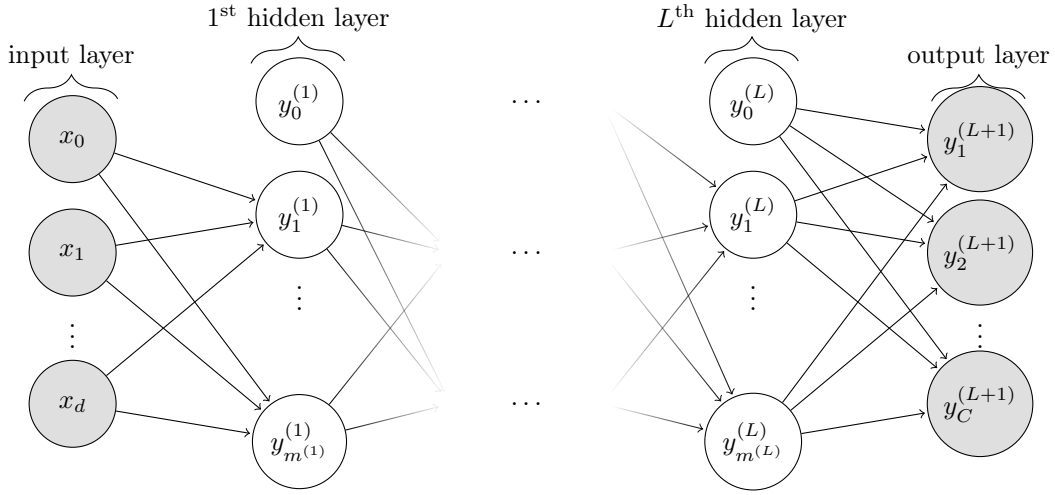


Figure 3.4. *The depiction of a multi-layer FFN. Figure was obtained from David Stutz's blogpost*

To sum it up, an MLP model ingest an input $\mathbf{x} \in \mathbb{R}^d$ and produces the output $\mathbf{y} \in \mathbb{R}^C$, based on its current parameters $\boldsymbol{\theta}$. The activations in each hidden layer can be calculated in a parallel fashion through a matrix multiplication. Once they are obtained, these activations can be propagated forward to obtain the respective activations of the next layer. This process is called **forward propagation**.

Activation Function

An omnipresent notion associated with DL models is that of activation function. In MLPs, the activations of a hidden layer are mapped through an affine transformation to the activations of the next hidden layer's units. In practice, it is common to apply some element-wise function $\phi : \mathbb{R} \to \mathbb{R}$ to the results of such affine mappings. This transformation is colloquially termed as the *activation function*. Therefore, with the introduction of the activation function, the units in hidden layer are obtained as: $\mathbf{y}^{(i)} = \boldsymbol{\phi}(\mathbf{W}^{(i)}\mathbf{y}^{(i-1)} + \mathbf{b}^{(i)})$ (where $\boldsymbol{\phi}$: the element-wise application of ϕ). Earlier works on the field mostly considered Sigmoid and Hyperbolic Tangent (Tanh) functions as the most suitable candidates:

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}}$$
$$\text{Tanh}(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

However, in Deep Neural Networks those activation functions deteriorate the training performance of the model. The most dominant choice is the Rectified Linear Unit (ReLU) function:

$$\text{ReLU}(x) = \max(0, x)$$

In the classification setting, we desire to transform the output layer (also called the logit vector) into a probability vector, where each element represents the confidence that the model assigns to each class for any given input. An easy way to obtain such a probability vector $\mathbf{p} = [p_1, ..., p_C]$ through the logits $\mathbf{y} = [y_1, ..., y_c]$ is through the Softmax function:

$$p_i = \text{Softmax}(\mathbf{y})_i = \frac{exp(\mathbf{y}_i)}{\sum_{j=1}^{C} exp(\mathbf{y}_j)}$$

Gradient-Based Learning

The previous paragraph presented how a simple MLP model predicts the output through matrix multiplications and applications of the activation function, from an algorithm called forward propagation. Initially, there is no way of predict the optimal parameter vector; one has to randomly initialize it. Of course, these random parameters will produce rather unsatisfying predictions about the outputs. **Gradient Descent** provides a principled approach to iteratively update the parameters in order to improve the model's performance.

First, we shall invoke the notion of loss function from earlier sections: For example, in the classification setting, one wants to obtain the parameters that minimizes the empirical risk. The 0-1 loss is not suitable for gradient-based learning due to its non-differentiability hence we will resort to some surrogate loss. The cross-entropy loss is the most widely used option for the surrogate loss in classification literature:

$$\text{CE}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{C} y_i \log \hat{y}_i$$

The introduction of this differentiable loss makes it possible to connect every parameter with a single scalar (the loss value) that quantifies how well they perform at estimating the output. The gradient $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta})$ of this scalar with respect to the parameters $\boldsymbol{\theta}$, indicates the best direction that we can move our parameter vector in order to minimize the loss function (this follows from the first-order Taylor approximation of $\mathcal{L}$). In the optimization literature, this is known as the Gradient Descent algorithm:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \eta_t \nabla_{\boldsymbol{\theta}} \mathbb{E}_{(\mathbf{x},y)\sim p_{\text{data}}} [\ell(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta})]$$

Recall that all we have at our disposal is a finite number of samples from the data distribution. There are many different variants of Gradient Descent that we can exploit to minimize the empirical risk, but in DL, the most prevalent is Stochastic Gradient Descent (SGD): Instead of approximating the true gradient by averaging through the entire dataset, we're sampling a subset $\mathcal{B}$ of the dataset $\mathcal{D}$, called *batch*, and the average is taken over this set:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \eta_t \nabla_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{|\mathcal{B}|} \ell(f(\mathbf{x}_i; \boldsymbol{\theta}), \mathbf{y}_i)$$

SGD offers an affordable way to approximate the true gradient; using the entire dataset would be prohibitively expensive both in terms of runtime and memory consumption. Inspecting the two ends of the batch size spectrum, if $|\mathcal{B}| = 1$ then the estimated gradient is extremely noisy but the parameter updates are more frequent, whereas if $|\mathcal{B}| = N$, each paramater update uses the true gradient but it is considerably slower, sometimes without even providing significant performance gains. In our above discussion, we presented the vanilla version of SGD where the update rule is simply a scaled version of the gradient. Polyak [Pol64] proposed Momentum methods that were

devised to accelerate learning, in settings where the optimization objective is characterized by high curvature. The update rule is slightly modified with the introduction of the velocity vector $\mathbf{u}$ and it can be formulated as:

$$\mathbf{u}_t = \alpha \mathbf{u}_{t-1} - \eta_t \nabla_{\boldsymbol{\theta}} \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \ell(f(\mathbf{x}_i; \boldsymbol{\theta}), \mathbf{y}_i)$$

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \mathbf{u}_t$$

where α: a hyperparameter that regulates how quickly the contribution of past gradients is decaying. Another category of optimization strategies with increasing reputation is Adaptive Methods. The essence of adaptive methods is that the learning rate associated to each weight is refined in every iteration based on the history of its past gradients w.r.t that specific weight. Kingma and Ba [KB15] introduced Adam, which unequivocally can be labeled as the most popular variant of such optimizers. The conventional wisdom surrounding these different optimization strategies is that adaptive methods converge significantly faster than SGD, but SGD yields better generalization capabilities.

Backpropagation

By this point, it should be clear how Gradient Descent enables us to iteratively adjust the parameters of a model in order to minimize the empirical risk. An important question that is yet to be answered is how one can efficiently obtain the gradients of the loss w.r.t every network parameter in the first place. This can be done through the *Backpropagation* algorithm (abbr. as Backprop), proposed by Rumelhart et al. [Rum+86].

The functionality of Backprop hinges on the chain rule. Chain rule enables us to compute the gradients of a compositional function of the form $\mathbf{f}(\mathbf{g}(\mathbf{x}))$ where $\mathbf{f} : \mathbb{R}^m \to \mathbb{R}^n, \mathbf{g} : \mathbb{R}^k \to \mathbb{R}^m, \mathbf{x} \in \mathbb{R}^k$, by multiplying the respective gradients (assume that $\mathbf{x}_1 = \mathbf{g}(\mathbf{x})$):

$$\nabla_{\mathbf{x}} \mathbf{f}(\mathbf{g}(\mathbf{x})) = \mathbf{J}_{\mathbf{f}}(\mathbf{x}_1) \mathbf{J}_{\mathbf{g}}(\mathbf{x})$$

where: $\mathbf{J}_{\mathbf{f}} \in \mathbb{R}^{n \times m}, \mathbf{J}_{\mathbf{g}} \in \mathbb{R}^{m \times k}$ the Jacobian matrices of $\mathbf{f}$ and $\mathbf{g}$. In the MLP case, the output layer is the result of composing multiple affine transformations followed by non-linearities. Denoting as $\mathbf{x}^{(i)}$ the $i-$th layer activations and $\mathbf{f}_i$ the $i-$th layer transformation (note that for sake of clarity we omit to include the parameters $\boldsymbol{\theta}_i$ of each layer), it follows that:

$$\mathbf{y} = \mathbf{x}^{(L)} = \mathbf{f}_L(\mathbf{f}_{L-1}(...(\mathbf{f}_1(\mathbf{x}))))$$

Hence, the per-example loss $l(\mathbf{y}, \hat{\mathbf{y}})$ gradients can be computed w.r.t the activations of l-th layer-and hence w.r.t to its parameters $\boldsymbol{\theta}^{(l)}$- as follows:

$$\nabla_{\mathbf{x}^{(l)}} l(\mathbf{y}, \hat{\mathbf{y}}) = \nabla_{\mathbf{x}^{(L)}} l(\mathbf{y}, \hat{\mathbf{y}}) \cdot \mathbf{J}_{\mathbf{x}^{(L-1)}}(\mathbf{x}^{(L)}) \cdot ... \cdot \mathbf{J}_{\mathbf{x}^{(l)}}(\mathbf{x}^{(l+1)})$$

This is the gist of Backpropagation. Essentially, its computational efficiency arises from the fact that the first term in the RHS of the above expression is a vector (since the per-loss example is a scalar) therefore we are sequentially performing vector-matrix products. The matrix-matrix products would've added a significant computational burden, had we applied the chain rule from first to last layer.

Other Components in Deep Learning Models

Normalization Layers. Stacking multiple layers culminates into models of massive depth that are hard to train due to problems such *vanishing* or *exploding gradients*. An effective solution, that increases the resilience of DL models against such problems is the addition *Batch Normalization* (BN) layers [IS15]. BN layers standarize the statistics of hidden units, yielding activations with zero mean and unit variance. The term batch stems from the fact that the hidden layer statistics are computed in each batch of training data. The standardization is also followed by an affine transformation of learnable scale and shift element-wise vectors.

Addressing Overfitting. Deep Neural Networks are egregiously overparameterized, meaning that their number of parameters (and the model capacity) is more than enough to learn the training data. This could inevitably lead to the overfitting problem. *Data Augmentation* is a technique to reduce the effect of this issue, by adding new training data (so the mismatch between training data and learnable parameters is reduced). This can be artificially achieved by injecting transformations of the existing data into the train set. Another trick that aids the prevention of overfitting is *Dropout* [Sri+14]. Dropout essentially refers to the procedure where during training, each unit (and all the connections associated with it) is stochastically disabled with a probability p. Intuitively, dropout semantically coincides with the training of an ensemble of exponentially many models. Another popular strategy to ameliorate overfitting is *Early Stopping*, where instead of returning the parameters of the model in the end of the training procedure, the model's final parameters are set to those that yielded the smallest training-validation error gap.

3.5.2 Architectures of Deep Learning models

Convolutional Neural Networks

Convolutional Neural Networks [Fuk79; Lec89] (abbr. as CNNs) are networks designed to efficiently proccess data with a grid-like structure, such as images. Their functionality hinges on the convolution operation, which in the case of 2D-signals like images can be formulated as follows:

$$\begin{aligned} I'[i,j] = (I * W)[i,j] &= \sum_m \sum_n I[m,n] W[i-m, j-n] \\ &= \sum_m \sum_n I[i-m, j-n] W[m,n] \end{aligned}$$

This is exactly how convolutional layers compute the activations of the next layer. Instead of applying affine transformations, those layers convolve their input with a weight matrix, also called *kernel*, to produce the output, also called *feature map*. Images are 3D tensors since, other than height and width, they also have a third dimension associated with their number of color channels. Hence, the input $I \in \mathbb{R}^{H \times W \times C}$ in convolutional layers is convolved channel-wise with a kernel $W \in \mathbb{R}^{B_1 \times B_2 \times C}$ and then the results is added across all those channel-wise convolutions. The output spatial dimensions depend on whether the input is padded or how much the kernel shifts in every convolution operation (also called stride). Usually, in a convolutional layer, the input is convolved with many different kernels to produce an output with many channels. There are certain properties of CNNs that render them greatly appealing: (1) They have sparse interactions, i.e. each output node interacts only with a local neighbourhood of the input, (2) Their parameter

sharing, referring to the property where the same weight kernel is used across the whole image; this property has as an immediate corollary that CNNs are equivariant to translation.

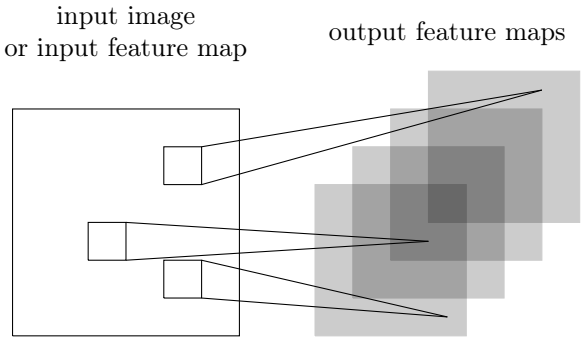


Figure 3.5. *Illustration of a single convolutional layer. Figure adapted from David Stutz's blogpost*

CNNs also include another type of layer that transform its input feature maps based on their local neighborhood, called *pooling layers*. For example, a max-pooling layer outputs the maximum activation unit among the square neighborhood of its input, whereas the average-pooling layer calculates the average of the activations in such square-shaped vicinities. Pooling layers confer translation invariance to CNNs, which is particularly beneficial in many applications. A typical CNN is comprised of alternating convolutional and pooling layers, yielding an output that has the spatial structure of a 3D-tensor. In Image Classification, for instance, it is common to apply a global average pooling to this output, then flatten the result of this operation into a simple 1D vector and passing this holistic image embedding to an MLP in order to obtain a label prediction.

Recurrent Neural Networks

Recurrent Neural Networks (also called RNNs) is another specialized kind of architectrues, specifically designed to handle sequential data, much like CNNs are invented to perform computations in grid-like data. The computational process in RNNs can be easily summarized by Figure 3.6. The hidden layer in RNNs is comprised of neurons that are called the *hidden state*. The hidden state at $t-$th timestep is a vector $\mathbf{h}^{(t)}$ that has accumulated information from all previous timesteps and it can be combined with the input $\mathbf{x}^{(t)}$ to produce the updated version of the hidden state for the next time step. This procedure can be formalized as: $\mathbf{h}^{(t)} = f(\mathbf{x}^{(t)}, \mathbf{h}^{(t-1)}; \boldsymbol{\theta})$. Similar to MLPs, this function f may be the application of a non-linearity of top of an affine transformation. The dashed arrows above the hidden layer's units depict the prospect of adding multiple layers in our network, such as an output layer which receives the current hidden vector to infer the output of that specific timestep. The forward propagation in RNNs, for a sequence of length τ, is an inherently sequential procedure hence it has computational cost of $O(\tau)$. The gradients in such networks are computed through an algorithm called Back-Propagation through Time (BPTT).

This is the most basic form of RNNs and is hardly used to solve any applications that require sequential computing. In recent years, RNNs have evolved into many different variants and we succinctly mention the most significant ones:

- Bi-directional RNNs [SP97], which process information in both directions. In the vanilla RNN, the input is treated as having a causal structure, where it is implied that the hidden state can carry information only from previous timesteps. However, in some applications, it may be expedient to exploit information in the reverse direction. Bi-directional RNNs

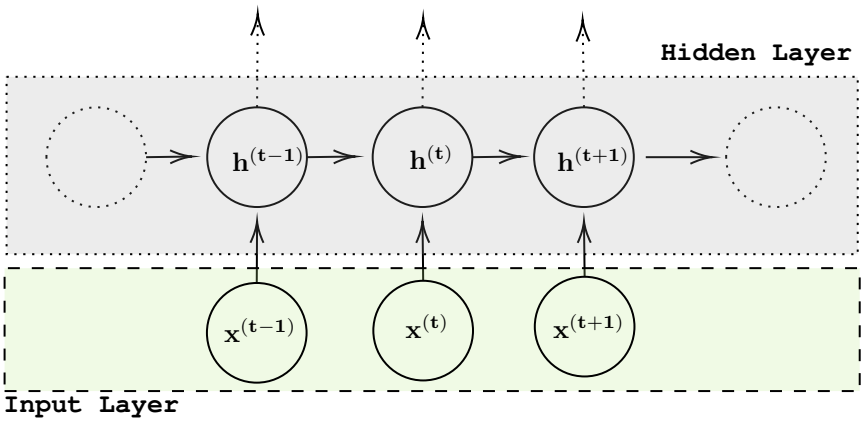


Figure 3.6. *An illustration of the vanilla RNN*

achieve this by manipulating the input both from left to right (with a hidden state $\overrightarrow{\mathbf{h}}^{(i)}$) and from right to left (with a hidden state $\overleftarrow{\mathbf{h}}^{(i)}$), combining those "dual" hidden states in each timestep to produce their output.

- Long-Short Term Memory (LSTM) models, introduced by Hochreiter and Schmidhuber [HS97]. The key module of such models is the LSTM cell. Each cell is characterized by its cell state vector $\mathbf{c}^{(t)}$ and its three gate vectors $\mathbf{f}^{(t)}, \mathbf{i}^{(t)}, \mathbf{o}^{(t)}$ (forget, input, output gate respectively). The forget gate regulates the amount of the previous cell state that is preserved in the new one, the input gate controls how the new input information will affect the cell state, whereas the output state updates the regular hidden state with the information that is carried into the cell state.

Transformers

Transformer, proposed by Vaswani et al. [Vas+17] is another type of architecture that has gained a lot of reputation recently. The input to the the transformer is a sequence of tokens $\mathbf{x} = (\mathbf{x}_1, ..., \mathbf{x}_n)$, where $\mathbf{x}_i \in \mathbb{R}^d$. The transformer processes those tokens in a way that allows them to exchange information, through the self-attention operation, to produce the output embeddings $\mathbf{z} = (\mathbf{z}_1, ..., \mathbf{z}_n), \mathbf{z}_i \in \mathbb{R}^d$. In its most simplest form, the transformer projects the input tokens into three sequences of vectors Q,K,V standing for Queries,Keys and Values respectively. Then, the similarity between each query-key pair in the sequence is measured through the scaled dot-product module. Those similarities are stacked into the attention matrix, which is then multiplied with the values' sequence V to yield the sequence of output tokens. This procedure can be readily understood by inspecting the visualization in Figure 3.7. However, in reality, this attention module is repeated many times, yielding the Multi-Head Attention structure, where each head is performing this type of attention operation. Then, the output sequences of all heads are concatenated and passed through an additional dense layer to project the output in the space of dimensionality that matches the input.

Typically, Transformer architectures are consisted of two parts: the Encoder and the Decoder. The encoder is responsible for mapping the input into a meaningful representation, where the interactions between different parts of the input have been properly encoded into the output sequence of tokens. A typical encoder consists of multiple Multi-Head Attention modules followed by Normalization and Feed-Forward layers. The decoder module is quite similar to the decoder (structure-wise) and learns how to decode the attended input representation to produce the desired

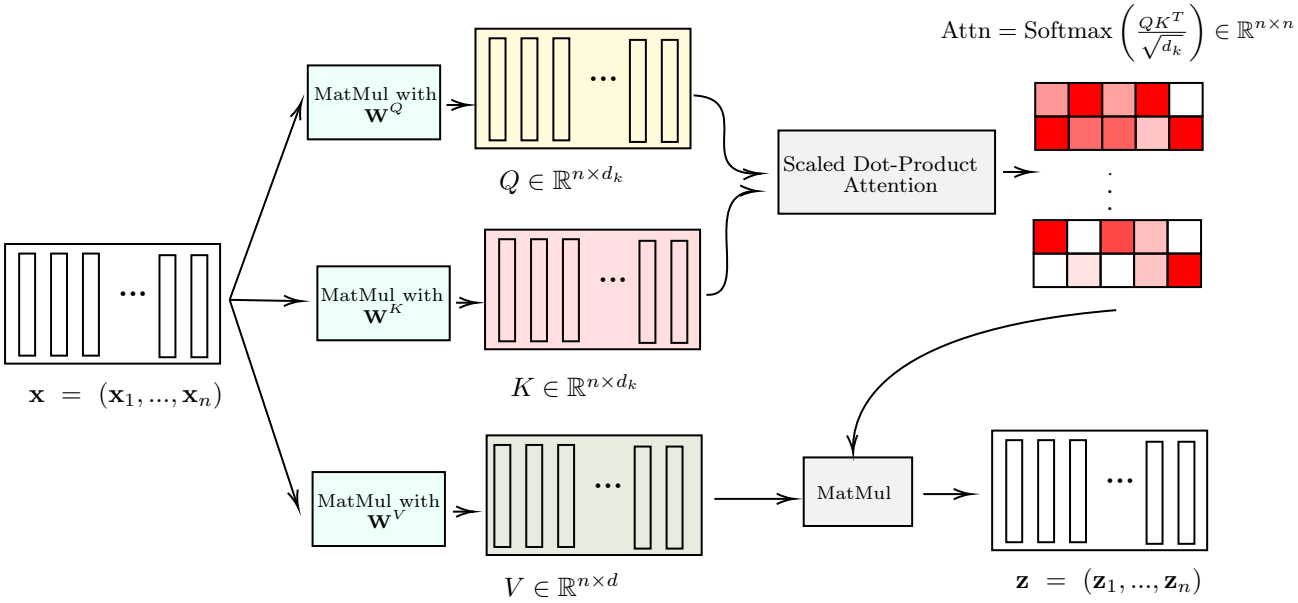


Figure 3.7. *An illustration of Transformer architecture, with only one attention head.*

output. A deeper understading and description of the inner workings of the Transformer architecture transcends the intended scope of this thesis. We should mention, however, that this novel architecture has been widely adopted in NLP with exceptional success, see e.g the BERT model [Dev+19] and later it also demonstrated similar performance with CNNs to Computer Vision tasks, see e.g the Vision Transformer (ViT) [Dos+20].

3.6 Deep Generative Models

Generative Modelling is a chapter of monumental importance in ML community. Our previous discussion was primarily focused on the classification paradigm, where the networks designed to solve that specific task can be considered as attempting to model the conditional distribution $p(\mathbf{y}|\mathbf{x})$. However, Artificial Intelligence (AI) and the quest of building machines that can perform reliable decision making can't be content with that information alone. Deep Generative Models (DGMs) are studied in order to broaden the general capabilities of Machine Learning. If we had to sum it up in a few words, we'd say that such models aspire to find a way of representing the data probability distribution $p(\mathbf{x})$. Some instances of DGMs aim to achieve this directly whereas others approach this problem implicitly, by offering the opportunity to perform other useful operations, such as drawing samples from $p(\mathbf{x})$.

In this section, our goal is to introduce the most representative instances of DGMs, namely: Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) which only implicitly model the data distribution and Normalizing Flows (NFs) which explicitly learn to approximately represent this distribution. All those instances of DGMs have one major shared component: they all are *latent variable models*, i.e. their operation hinges on the premise that there is a (lower) dimensional latent space that compactly captures hidden factors in data and those data are generated as follows:

$$\begin{aligned} \mathbf{z} &\sim p(\mathbf{z}) \\ \mathbf{x} &\sim p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) \end{aligned} \tag{3.1}$$

3.6.1 Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) [Goo+14] are comprised of two major components, the generator G and the discriminator D, which compete each other in the following sense: The generator learns to map random noise $\mathbf{z} \sim p(\mathbf{z})$ (where $\mathbf{z} \in \mathbb{R}^k$) to the data space $G(\mathbf{z}, \boldsymbol{\theta}_g) \in \mathbb{R}^d$. The discriminator learns to discern between fake and real images, yielding as output a single scalar $D(\mathbf{x}, \boldsymbol{\theta}_d)$ which denotes the probability of $\mathbf{x}$ being an image from the real data distribution p_{data}. Both of these components are represented through Deep Neural Networks e.g. MLPs in their simplest form. The training of GANs amounts to a two-player minimax game that can be formulated as:

$$\min_G \min_D V(G, D) = \min_G \min_D \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})}[log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})}[log(1 - D(G(\mathbf{z})))]$$

Thus, the discriminator simultaneously maximizes the log probability of real and generator data being identified as real and fake respectively, whereas the generator minimizes the latter objective. At the end of training, we can discard the discriminator and sample new data from the generator as: $\mathbf{z} \sim p(\mathbf{z}) : \mathbf{x}_{\text{new}} = G(\mathbf{z})$.

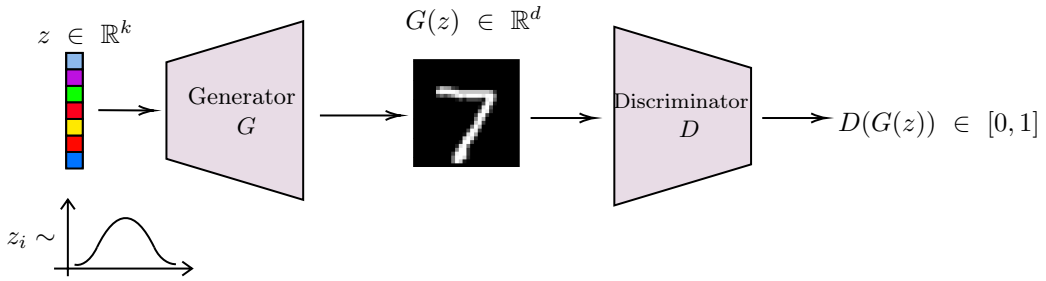


Figure 3.8. *A high-level demonstration of typical GAN architecture.*

GANs can produce samples with incredible visual plausibility, however the standard training procedure is rather delicate and unstable. Besides that, standard GANs also suffer from the problem of mode collapse, where generators draw samples from only a tiny proportion of modes in the distribution, hence the synthesized samples are lacking diversity. An important step towards providing effective solutions against such problems was prescribed by the introduction of Wasserstein GANs (WGANs) by Arjovsky et al. [Arj+17], which establish a slightly different training objective and WGAN-GP [Gul+17], an improved variant that improves training stability by penalizing the input gradients' norm of the critic/discriminator.

3.6.2 Variational Autoencoders (VAEs)

Variational Autoencoders (VAEs) [KW14] is a probabilistc version of determnistic autoencoders. This class of models fits to the mathematical framework of Variational Inference, where one tries to approximate a given distribution through a family of parameterized distributions. When adopting the generative process of Equation 3.1, calculating the true density $p_{\boldsymbol{\theta}}(\mathbf{x})$ and the true posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ is intractable since: $p_{\boldsymbol{\theta}}(\mathbf{x}) = \int_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) p_{\boldsymbol{\theta}}(\mathbf{z}) d\boldsymbol{\theta}$. VAEs resort to approximating the true posterior through $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ (called approximate posterior). Overall, they are constituted of two parts:

- The decoder, which models the distribution $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$. For binary data, this distribution is modelled as a Bernoulli distribution of unknown parameters. The decoder maps an input

latent code to the probability vector of this Bernoulli distribution

- The encoder (or recognition network), that models the distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$. For example, we can assume that $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is a Gaussian distribution, thus the encoder receives as input the data point $\mathbf{x}$ and outputs the distribution's statistics, i.e. mean and variance vectors $\mu_{\boldsymbol{\phi}}(\mathbf{x}), \sigma_{\boldsymbol{\phi}}(\mathbf{x})$.

The training of VAEs is done through the maximizaiton of Evidence Lower Bound (ELBO), which lower bounds the data likelihood:

$$\begin{aligned} \log p_{\boldsymbol{\theta}}(\mathbf{x}) \geq \text{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}) &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\Big[-\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) + \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\Big] \\ &= -\text{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z})) + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\big[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\big] \end{aligned} \tag{3.2}$$

where the prior $p_{\boldsymbol{\theta}}(\mathbf{z})$ is usually set to the normal distribution $\mathcal{N}(0, \mathbb{I})$. The KL divergence term can be analytically computed for normal distributions, whereas the second term can be approximated through Monte Carlo sampling and quantifies our desire of having high log-likelihood for latent codes drawn from the approximate posterior. Notably, the gradients w.r.t encoder's parameters $\boldsymbol{\phi}$ of the latter expression in Equation 3.2 exhibit extremely high variance, with Kingma and Welling [KW14] proposing the repameterization trick to reduce it. This trick considers the latent vector as the result of a deterministic mapping, exploiting a property associated with Gaussian random variables, i.e.:

$$\boldsymbol{\epsilon} \sim \mathcal{N}(0, I) : \mathbf{z} = \mu_{\boldsymbol{\phi}}(\mathbf{x}) + \boldsymbol{\epsilon} \odot \sigma_{\boldsymbol{\phi}}(\mathbf{x}) \Rightarrow \mathbf{z} \sim \mathcal{N}(\mu_{\boldsymbol{\phi}}(\mathbf{x}), \sigma_{\boldsymbol{\phi}}(\mathbf{x})I)$$

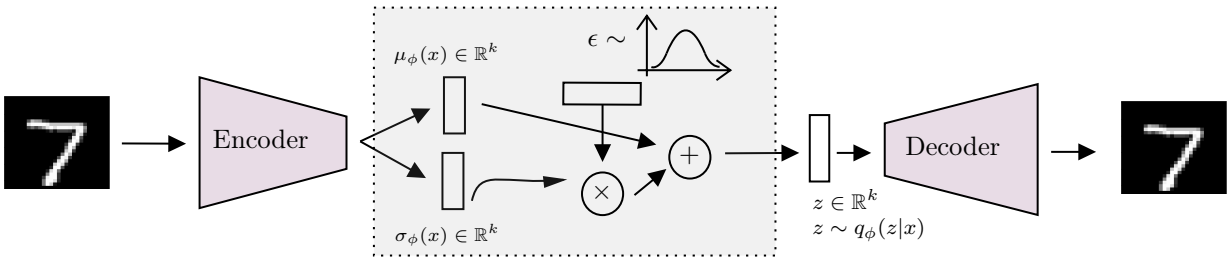


Figure 3.9. *A high level illustration of a typical VAE architecture. The dashed rectangle depicts the procedure of performing the reparameterization trick.*

3.6.3 Normalizing Flows (NFs)

Normalizing flows (NFs) [TT13] provide a general way of constructing flexible probability distributions over continuous random variables. NFs express real data $\mathbf{x} \in \mathcal{X}$, $\mathcal{X} \subseteq \mathbb{R}^d$ as a transformation $f : \mathcal{Z} \to \mathcal{X}$ of a real vector $\mathbf{z} \in \mathcal{Z}$ that is distributed according to $p_Z(\mathbf{z})$. If we restrict this transformation to be invertible (with inverse $g = f^{-1}, g : \mathcal{X} \to \mathcal{Z}$) and differentiable (a.k.a. diffeormorphisms), then the change-of-variables formula enables us to calculate $p(\mathbf{x})$:

$$p_X(\mathbf{x}) = p_Z(g(\mathbf{x})) \cdot |\det \mathbf{J}_{\mathbf{x}}(g(\mathbf{x}))| \tag{3.3}$$

where $\mathbf{J}_{\mathbf{x}}(g(\mathbf{x}))$ is the Jacobian $d \times d$ matrix of all partial derivatives of g. Essentially, NFs model such transformations through neural networks. Typically, those transformations can be designed as the composition of many individual diffeormophisms g_i: $g = g_K \circ g_{K-1} \circ ... \circ g_1$, where g_i

transforms $\mathbf{z}_i$ into $\mathbf{z}_{i-1}$, assuming $\mathbf{x} = \mathbf{z}_K, \mathbf{z} = \mathbf{z}_0$. In this case, the Jacobian of g can be calculated as the product of the individual Jacobians, hence the change-of-variables formula can be expressed as follows (simultaneously, we apply the log operator):

$$\log p(\mathbf{x}) = \log p(g(\mathbf{x})) + \sum_{i=1}^{K} \log \left| \det \mathbf{J}_{\mathbf{z}_i}(g_i(\mathbf{z}_{i-1})) \right| \tag{3.4}$$

NFs essentially refers to the scenario where this diffeomorphism g is constructed through Deep Neural Networks. The training procedure maximizes the log-likelihood expression of Equation 3.4. The most essential part when building NFs is to carefully design those transformations in order to obtain tractable Jacobian determinant terms; For an arbitrary $d \times d$ matrix, calculating the determinant has a time cost of $\mathcal{O}(d^3)$. However, such a computational cost is prohibitively large for high-dim data and thus it should be decreased to linear complexity $\mathcal{O}(d)$.

In the literature of NFs, there has been proposed a plethora of different ways to efficiently construct such mappings and this class of DGMs has been used to provide novel solutions for many applications across many different modalities i.e. images, speech, text. For a comprehensive review, we highly recommend the perusal of [Pap+21]. However, we shall point our direction towards three works: NICE [Din+15], RealNVP [Din+17] and Glow [KD18]. The former two papers are considered as a predecessor of the latter, which constitutes one of the most popular works in this field; hence, they make a strong candidate for an introduction on NFs.

Generally, NFs have some enticing properties that GANs and VAEs do not possess: (1) They can perform exact inference, exactly mapping the input $\mathbf{x}$ to a single latent code $\mathbf{z}$ (VAEs only perform approximate inference, whereas GANs can only be inverted through Gradient Descent) and (2) They can serve as density estimators, providing the capability to exactly calculate $p(\mathbf{x})$ (VAEs can only compute a lower bound of this quantity).

Affine Coupling Layers. Affine Coupling Layers were initially proposed in Dinh et al. [Din+15] to create expressive diffeomorphisms through neural networks with tractable Jacobian terms. Coupling layers works as follows: The input $\mathbf{x} \in \mathbb{R}^d$ is split into two components, $\mathbf{x}_1 \in \mathbb{R}^{d_1}, \mathbf{x}_2 \in \mathbb{R}^{d-d_1}$ (usually $d_1 = d/2$). Then, the first component remains intact $\mathbf{z}_1 = \mathbf{x}_1$, whereas in the second component we apply an affine transformation mapping : $\mathbf{z}_2 = \exp(s(\mathbf{x}_1)) \odot \mathbf{x}_2 + t(\mathbf{x}_1)$, where $s(\mathbf{x}_1), t(\mathbf{x}_1) \in \mathbb{R}^{d-d_1}$ have been generated through a NN with parameters $\boldsymbol{\theta}$. This type of layer models an invertible mapping since:

$$\left.\begin{aligned} \mathbf{z_1} &= \mathbf{x}_1 \\ \mathbf{z}_2 &= \exp(s(\mathbf{x}_1)) \odot \mathbf{x}_2 + t(\mathbf{x}_1) \end{aligned}\right\} \Rightarrow \begin{cases} \mathbf{x}_1 &= \mathbf{z_1} \\ \mathbf{x}_2 &= \Big(\mathbf{z}_2 - t(\mathbf{x}_1)\Big) \exp(-s(\mathbf{x}_2)) \end{cases}$$

The Jacobian of this mapping has an lower triangular form:

$$\mathbf{J} = \frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \begin{bmatrix} \mathbb{I}_{d_1} & \mathbf{0}_{d-d_1} \\ \frac{\partial \mathbf{z}_2}{\partial \mathbf{x}_1} & \operatorname{diag}(\exp(s(\mathbf{x}_1))) \end{bmatrix} \tag{3.5}$$

Hence, the log-determinant of this Jacobian amounts to the sum: $\sum_{j=1}^{d-d_1} s(\mathbf{x}_1)_j$. Figure 3.10 visualizes the inner workings of a coupling layer.

Permutation Layers. Since the coupling layer leaves a part of its input unchanged, it is necessary to apply some permutation transformation in order to modify the unchanged part. This can

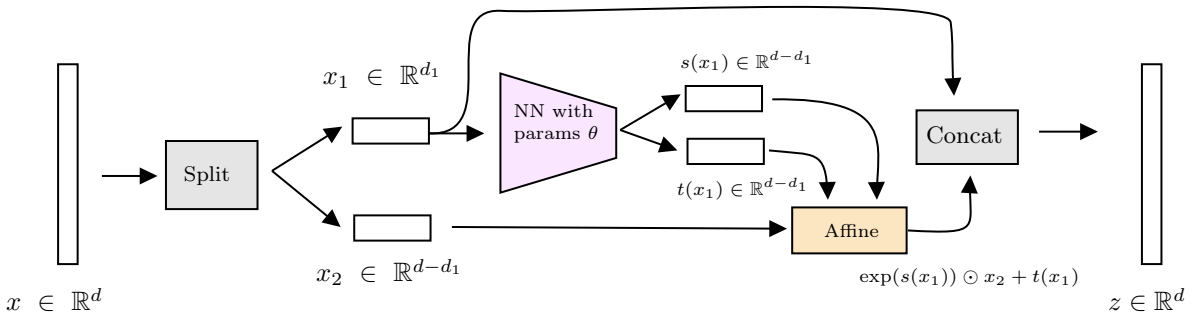


Figure 3.10. *A high level illustration of a standard affine coupling layer.*

be done through a binary mask $\mathbf{b}$, hence the coupling layer transformation is compactly written as:

$$\mathbf{z} = \mathbf{b} \odot \mathbf{x} + (1 - \mathbf{b}) \odot (\mathbf{x} \odot (\exp(s(\mathbf{b} \odot \mathbf{x})) + t(\mathbf{b} \odot \mathbf{x}))$$

Dinh et al. [Din+17] proposed two masking strategies: (1) a checkerboard masking scheme, which alternates the 0s and 1s after each coupling layer and (2) a channel-wise masking, which masks out half of the channel dimensions.

A typical processing block in RealNVP is comprised of three coupling layers with alternating checkerboard masks, then they perform a squeezing operation, and finally apply three more coupling layers with alternating channel-wise masking. The squeezing operation trades spatial size for channel dimensionality.

Multi-Scale Architectures. Instead of propagating the vector $\mathbf{x} \in \mathbb{R}^d$ through all the coupling layers, Dinh et al. [Din+17] suggest that it would be expedient to employ a multiscale strategy: first, the vector $\mathbf{x}$ is passed through a block of transformations and half of the $\mathbf{z}$ dimensions are factored out, without applying further processing. Then, the remaining dimensions are passed again through such a block, with another chunk of $\mathbf{z}$ being factored out and so forth and so on. This provides a major benefit memory-wise, since the smaller vector can be processed for many more layers than the full-dimension vector.

Invertible 1×1 Convolution. In general, the Glow model [KD18] resembles vividly the RealNVP architecture, but they devise a new type of layer that lent further performance gains. Kingma and Dhariwal [KD18] dispense with the permutation layers and instead propose to use 1×1 convolutional layers. Convolving the $h \times w \times c$ tensor $\mathbf{h}$ with a $c \times c$ matrix $\mathbf{W}$ has the following log-determinant:

$$\log \left| \det \left(\frac{\partial \text{conv2D}(\mathbf{h}, \mathbf{W})}{\partial \mathbf{h}} \right) \right| = h \cdot w \cdot \log |\det(\mathbf{W})|$$

The cost of computing $\det(\mathbf{W})$ is $\mathcal{O}(c^3)$, but the LU decomposition trick reduces it to $\mathcal{O}(c)$. They model $\mathbf{W}$ as follows:

$$\mathbf{W} = \mathbf{PL}(\mathbf{U} + \text{diag}(\mathbf{s}))$$

where $\mathbf{P}, \mathbf{U}, \mathbf{L}$ are a permutation matrix, a lower triangular matrix with ones on the diagonal and an upper triangular matrix with zeros on the diagonal. The log-determinant is computed as: $\log |\det(\mathbf{W})| = \sum_j \log |\mathbf{s}|_j$. The permutation matrix remains fixed during training, whereas the other three components are trained.

Chapter 4

Adversarial Machine Learning

This chapter delves into the field of *Adversarial Machine Learning*, which essentially consists the primary research topic of this present dissertation. Here, our goal is to thoroughly introduce the core ideas of this field through providing the proper background, as well as to present a multitude of research byproducts that helped the community to gain a deeper understanding for Deep Learning models.

4.1 Introduction

Deep Learning based models have been established as the de facto standard to tackle a plethora of tasks across many different domains. The universal adoption of such models was triggered by the seminal work of Krizhevsky et al. [Kri+12], where they managed to exploit parallel-computing hardware to train CNNs with extreme efficiency and obtain unprecedented results in the ILSVRC-2012 competition. However, Szegedy et al. [Sze+14] demonstrated that DL-based image classifiers are vulnerable to imperceptibly modified inputs, colloquially known as *adversarial examples*, that have been maliciously generated in order to change the system's decision making. This phenomenon is fully comprehended through the examination of Figure 4.1, where adding a humanly undetectable amount of noise to the clean image leads to a new image which is erroneously classified from the system.

Figure 4.1. *Adding impercetible noise dramatically alters the classifier's decision.*

This counter-intuitive property brings forth important questions that seek for answers. Such models has been used in a variety of applications where reliable and secure decision-making should be an imperative trait of their deployment, e.g. in Autonomous Driving failing to correctly detect a stop sign could lead to pernicious consequences. It is needless to say that the discovery of this intriguing phenomenon urged researchers to study it in a great extent, both for illuminating the

reasons of its existence and for building credible systems that are robust to such discontinuities. The procedure of producing such misleading examples is known as *Adversarial Attacks*, whereas enhancing the stability of DL models against such inputs is referred as *Adversarial Defenses*.

At this point we should mention that adversarial attacks are ubiquitous in the DL literature and have appeared in various domains such as images, audio signals or text documents. However, we will study this phenomenon from the image classification perspective, where the attacks are generated to deceive image classifiers.

4.2 Adversarial Attacks

This section initially dedicates its content to lay the proper mathematical ground for crafting attacks and will help the reader to readily follow the presentation of ensuing related work. Besides that, we attempt to separate algorithms for generating adversarial attacks based on different discriminating factors and present the most accepted prescribed interpretations that explain their existence.

4.2.1 Threat Models

The most important concept, prior to mathematically formulating the problem of finding adversarial attacks, is the allowable set of perturbations, also called the *threat model*. Ideally, for a clean input $\mathbf{x} \in \mathcal{X}$, where $\mathcal{X}$: the image space $[0, 1, ...255]^d$, a neat definition of the threat model would be:

$$O(\mathbf{x}) = \{\mathbf{x}' \in \mathcal{X} | \, \mathbf{x}', \mathbf{x} \text{ are visually (to the human eye) identical}\}$$

The ultimate aspiration of studying robustness in the DL setting is to build models that behave uniformly for every input $\mathbf{x}' \in O(\mathbf{x})$. However, this formulation is impractical since there is no principled approach to express such a human "oracle" mathematically. The most prevalent model, which replaces the visual imperceptibility constraint with euclidean distances, is the ℓ_p-norm threat model, where the permissible adversaries are confined within the ℓ_p-ball of radius ϵ around $\mathbf{x}$:

$$\Delta(\mathbf{x}) = \{\mathbf{x}' \in \mathcal{X} : \|\mathbf{x}' - \mathbf{x}\|_p \leq \epsilon\}$$

where ϵ: a small constant, expressing our desire of producing adversaries $\mathbf{x}' \in \Delta(\mathbf{x})$ that are ostensibly identical to their clean versions. The ℓ_p-norm adversaries constitute the most studied threat model across the literature, implicitly introduced by Goodfellow et al. [Goo+15]. Different values of p suggest distinct types of constraints that we impose to the adversary: If $p = \infty$, we restrict the perturbation's maximum element to ϵ, whereas if $p = 0$ the adversary can only change a small amount of pixels in the image. The most common choices for the value of p are $p = \infty$ or $p = 2$.

Despite its wide adoption, the ℓ_p-bounded threat model has been scrutinized since close proximity in terms of ℓ_p-norm doesn't necessarily translates to visual imperceptibility, nor it is required for visually identical pairs of clean-perturbed inputs to have small ℓ_p distance [Sha+18]. However, in practice, algorithms that craft ℓ_p-bounded attacks generate adversaries which cannot be visually distinguished from their respective clean inputs. An important issue related to this threat model is that classifiers that are trained to perform well against such adversaries are unable to generalize their resilience against unseen (during training) threat models; other forms of distortions, such as spatial transformations or image recolorings can degrade the model's performance to trivial percentages.

Nevertheless, studying the ℓ_p-norm threat model, both from the attack and defense perspectives, can be bolstered by two important arguments: (1) It is mathematically well-defined, which facilitates the task of devising new defenses and (2) It can be considered as a subset of the original robustness problem, where failing to adequately solve it would probably mean that we should not have high expectations of achieving general robustness in the DL setting.

In the following discussion, we will frequently mention that some attack method leads to stronger adversaries than another; The ranking between different methods for a given classifier is quantified through the Attack Success Rate (ASR) on the test set: For a given adversarial attack algorithm, ASR denotes the percentage of test set examples that were successfully perturbed by the algorithm.

4.2.2 A taxonomy of Adversarial Attacks

Here, we will discern the methods that produce adversarial attacks based on different criteria, associated with the amount of knowledge that the adversary has at its disposal and the requirements that the crafted outputs should satisfy. This brief discussion is helpful since it introduces the most basic terminology that everyone should come across when delving into this research field.

White-box vs Black-Box. Adversaries can be separated into White Box and Black Box, depending on the available information. There are two extreme cases: The adversary has full access to every aspect of the model e.g. both its architecture, weights and the training dataset. This class of attacks is referred with the term "white box" and the most apparent way of harnessing this information is to calculate the model's gradients w.r.t the input in order to guide the generation process. At the other end of the knowledge spectrum lies the "black box" attacks, where the adversary can only query the threatened model, obtaining the most probable class for the input with its associated confidence or even the full logit vector. Black box attacks, in their own right, are divided into different categories based on how they opt to approach the lack of knowledge (See Section).

Generally, for a given classifier, white-box adversaries should give rise to more powerful attacks than black-box methods; if this general principle is not true, it indicates that the classifier most likely impedes the gradient-based approaches e.g. because it contains a randomization module. This behaviour, as we shall discuss later in depth, is the root of major controversy in the literature of Adversarial ML and it constitutes the problem of *gradient masking*.

Targeted vs Untargeted. Recall that in the image classification setting, the attacker aims to change the classifier's predicted label for a given input. Adversarial attacks can be partitioned based on whether we desire the system to classify the input as belonging to a certain class (target class) or not. In the former case, the adversary generates *targeted* attacks, whereas in the latter case the attacks are called *untargeted*. However, the transition between the targeted and untargeted versions of an algorithm is trivial most of the times e.g. in white-box attacks, instead of following the ascent directions to minimize the true label's prediction, the generation can be posed as the maximization of the target label's confidence.

As in the case of white vs black box methods, the untargeted version of an adversary should yield stronger attacks than its targeted variant (with randomly chosen target classes), for a given classifier. Observing the opposite behaviour is probably an indicator that the model returns noisy gradients hence the gradient-based methods are performing poorly.

Beyond the ℓ_p-bounded threat model. The ℓ_p-norm constraint has been imposed in order to preserve intact the visual content of the image, as perceived by a human observer. There are many other ways to produce imperceptible perturbations without being limited to small euclidean magnitude: Xiao et al. [Xia+18] produced adversarial perturbations through constructing adversarial pixel displacements (optical flow); Engstorm et al. [Eng+19b] demonstrated that small rotations and translations suffice to deceive image classifiers. Also, GAN-based approaches [Zha+18; Son+18a] have managed to exploit the representational power of GAN generators to create outputs that are misclassified by the system, while a human judge can readily classify them to their ground truth class. Another line of work [HP18; LF19] managed to construct adversarial examples by slightly recoloring the content of an image. All of these aforementioned methods have in common that the generated perturbation has unbounded magnitude; Such approaches are dubbed as *unrestricted attacks*.

4.2.3 White Box Attacks

Finding adversarial examples can be stated in various ways. Consider a classifier $f_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}^C$ (where $\boldsymbol{\theta}$: its parameters, which sometimes we may omit for clarity purposes) which maps the inputs $\mathbf{x} \in [0,1]^d$ to the logit vector of C scores $f_{\boldsymbol{\theta}}(\mathbf{x}) \in \mathbb{R}^C$. A natural way to describe the generation of an adversary $\mathbf{x}'$ for the input $\mathbf{x}$ is:

$$\begin{aligned} \min_{\mathbf{r}} \|\mathbf{r}\|_p \quad \text{s.t.} \quad & \arg\max f(\mathbf{x}+\mathbf{r}) \neq y \\ & \mathbf{x}+\mathbf{r} \in [0,1]^d \end{aligned} \tag{4.1}$$

The non-linear constraint is impractical when it comes to optimizing this expression. In the following part of this section, we will analyze the alternatives that were prescribed by researchers to overcome this obstacle. In general, methods that are (loosely) based on this formulation generate minimum norm perturbations. There is a distinct formulation, fostering the search of solutions (inside the ℓ_p-ball of radius ϵ around $\mathbf{x}'$) which maximally decrease the ground truth label's confidence and can be stated as follows:

$$\begin{aligned} \max_{\mathbf{x}'} \mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}'), y) \quad \text{s.t.} \quad & \|\mathbf{x}'-\mathbf{x}\|_p \leq \epsilon \\ & \mathbf{x}' \in [0,1]^d \end{aligned} \tag{4.2}$$

where $\mathcal{L}$: a smooth,differentiable loss function that quantifies the classifier's ability to assign the label y to input $\mathbf{x}'$, e.g. cross-entropy for image classification, which acts like a surrogate alternative of the discontinuous 0-1 loss.

L-BFGS. Szegedy et al. [Sze+14] in their pioneering work, considered the task of finding adversaries through the lens of Equation 4.1. The non-linear constraint was replaced by a penalty term using a loss function such as cross-entropy (in similar fashion with Equation 4.1). Then, they sought for minimizers to the following objective through the L-BFGS numerical optimization method:

$$\min_{\mathbf{r}} \|\mathbf{r}\|_p - \mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}+\mathbf{r}), y) \quad \text{s.t.} \quad \mathbf{x}+\mathbf{r} \in [0,1]^d$$

Despite its novelty, a grave defect of this approach is the excessive computational overhead of L-BFGS which is the main reason it has been deprecated.

FGSM. A straightforward strategy of creating adversaries was introduced by Goodfellow et al. [Goo+15]. If we consider Equation 4.2 without the constraints, then the gradient w.r.t the input $\mathbf{x}$, $\nabla_{\mathbf{x}} L(f_{\boldsymbol{\theta}}(\mathbf{x}), y)$ indicates the optimal direction (based on first-order information) to move towards in order to minimize $\mathcal{L}$. Taking into account the desire of restricting the perturbation's $\ell_\infty-$norm, they introduce a normalization step:

$$\mathbf{x}' = \mathbf{x} + \epsilon \cdot \text{sgn}(\nabla_{\mathbf{x}} \mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}), y))$$

This method is called Fast Gradient Sign Method (FGSM). Notice that it can be extended easily to the case of ℓ_2-norm by rescaling the gradient to have unit magnitude:

$$\mathbf{x}' = \mathbf{x} + \epsilon \cdot \frac{\nabla_{\mathbf{x}} \mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}), y)}{\|\nabla_{\mathbf{x}} \mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}), y)\|_2}$$

In subsequent works, FGSM was evolved into R+FGSM [Tra+18] which added a random initialization step before applying the gradient-based displacement, BIM (Basic Iterative Method) [Kur+17] which executed multiple smaller gradient steps and MI-FGSM (Momentum Iterative FGSM) [Don+18] which proposed to integrate a momentum term into the iterative method.

PGD. The iterative (and more powerful) variant BIM, introduced by Kurakin et al. [Kur+17], can alternatively framed through the classical convex optimization method of Projected Gradient Descent (PGD). PGD is deployed when we aim to enforce Gradient Descent to find solutions that are contained into a certain set. In the case of adversarial attacks, this set coincides with the ℓ_p-ball of radius ϵ around the clean input $\mathbf{x}$. Viewing the process of crafting adversaries through the PGD perspective was first proposed by Madry et al. [Mad+18] and it is expressed as follows:

$$\mathbf{x}_{t+1} = \mathcal{P}_{\Delta(\mathbf{x})}\Big(\mathbf{x}_t + \eta \cdot \text{norm}_p(\nabla_{\mathbf{x}_t} \mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}_t), y))\Big) \tag{4.3}$$

where η: the step size which is typically smaller than the radius ϵ, norm_p: a normalizing operator which adjusts the input gradient in order to have unit ℓ_p-norm and $\mathcal{P}_{\Delta(\mathbf{x}))}$: the projection operator which maps the updated iterate into the feasible solution set. Specifically, if we denote as $\mathbf{y}_{t+1}$ prior to the projection step, then the projection proceeds as follows (for $p = \infty, 2$):

$$p = \infty \begin{cases} \mathbf{r}_{t+1} & = \mathbf{y}_{t+1} - \mathbf{x} \\ \mathbf{r}'_{t+1,i} & = \max(\min(\mathbf{r}_{t+1,i}, \epsilon), -\epsilon) \\ \mathbf{x}_{t+1} & = \mathbf{x} + \mathbf{r}'_{t+1} \end{cases}$$

$$p = 2 \begin{cases} \mathbf{r}_{t+1} & = \mathbf{y}_{t+1} - \mathbf{x} \\ \mathbf{r}'_{t+1} & = \epsilon \cdot \frac{\mathbf{r}_{t+1}}{\max\{\|\mathbf{r}_{t+1}\|_2, \epsilon\}} \\ \mathbf{x}_{t+1} & = \mathbf{x} + \mathbf{r}'_{t+1} \end{cases}$$

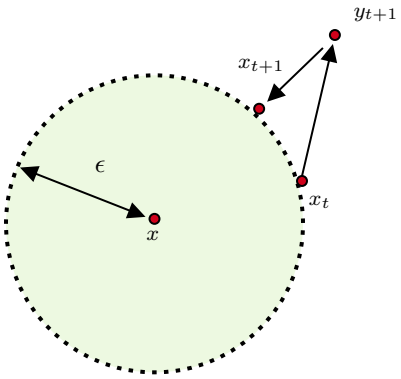


Figure 4.2. *A 2D example of how a single update works in the PGD algorithm, in the ℓ_2-norm case.*

Finally, the adversary is obtained either from selecting the last iterate $\mathbf{x}_T$, where T: the number of iterations or alternatively, the iterate that reached the lowest objective value (in an early-stopping manner). Whenever it is computationally affordable, instead of initializing the starting point as the clean example, a random perturbation (uniform or normal) is added to $\mathbf{x}$ and the procedure is repeated multiple times from different initial points. It is worth mentioning that PGD has prevailed as the most standard algorithm of generating adversaries, in

the case of ℓ_∞-bounded attacks. It forms the basis for even more elaborate techniques of attacking image classifiers, and also, it can be exploited in order to increase the robustness of neural networks.

C&W. Carlini and Wagner [CW17] replaced the non-linear constraint of Equation 4.1 through the introduction of a smooth loss function $\mathcal{L}(\mathbf{x}, y)$ endowed with the following property: $\mathcal{L}(\mathbf{x}, y) < 0$ iff $\arg\max f(\mathbf{x}+\mathbf{r}) \neq y$, where $\mathbf{r}$: the additive perturbation. They present multiple formulations of objective functions with this exact property, but the experiments demonstrated the margin loss (also called CW loss) as the best option:

$$\mathcal{L}_{\text{CW}}(f(\mathbf{x}), y) = f(\mathbf{x})_y - \max_{i \neq y} f(\mathbf{x})_y$$

Notice that if $\mathcal{L}_{\text{CW}}(\mathbf{x}, y) < 0$ then $f_y(\mathbf{x}) < \max_{i \neq y} f_i(\mathbf{x})$ thus the network f misclassifies the input $\mathbf{x}$. An objective function with this type of property is exploited in the following way: Equation 4.1 can be alternatively expressed through a Lagrangian relaxation:

$$\min_{\mathbf{r}} \|\mathbf{r}\|_p + c \cdot \mathcal{L}(f(\mathbf{x}+\mathbf{r}), y) \quad \text{s.t} \quad \mathbf{x}+\mathbf{r} \in [0,1]^d \tag{4.4}$$

The box constraint is completely dismissed through the change of variables formula, where $\mathbf{x}$ and $\mathbf{r}$ are connected as: $\mathbf{x}+\mathbf{r} = \frac{1}{2}(\tanh \mathbf{w} + \mathbf{1}_d), \mathbf{w} \in \mathbb{R}^d$. This modification allows to search for adversaries by optimizing the following expression (for $p = 2$):

$$\min_{\mathbf{w}} \left\| \frac{1}{2}(\tanh \mathbf{w} + \mathbf{1}_d) - \mathbf{x} \right\|_2^2 + c \cdot \mathcal{L}\Big(f\big(\frac{1}{2}(\tanh \mathbf{w} + \mathbf{1}_d)\big), y\Big) \tag{4.5}$$

The constant c determines the importance of the term associated with misclassifiaction; greatly increasing it will most surely result to adversaries but with larger ℓ_2-distortion which may be visible. Ideally, we'd like to set it to the smallest possible constant which successfully finds an adversary; in practice, they perform grid search into an interval ranging from 0.01 to 100. Equation 4.5 is best solved by the Adam optimizer.

We purposely omit to present how the C&W attack is extended to other ℓ_p-norms, since, besides the increased complexity due to the non-differentiability of other ℓ_p-norms, that method is mostly used for the $p = 2$ setting. In the $p = \infty$ case, the default attack choice is the PGD algorithm.

JSMA. Papernot et al. [Pap+15] introduced Jacobian Saliency-Map Attack (JSMA) which is a method of generating targeted attacks of bounded ℓ_0-norm. The ℓ_0-norm constraint essentially limits the amount of pixels that can be changed, rather than their intensity. An essential part of JSMA is the calculation of the Jacobian matrix: $\nabla_{\mathbf{x}} f(\mathbf{x}) = [\frac{\partial f(\mathbf{x})}{\partial x_1}, ..., \frac{\partial f(\mathbf{x})}{\partial x_d}] \in \mathbb{R}^{K \times d}$, where $f(\mathbf{x}) \in \mathbb{R}^K$ is the logit vector of input $\mathbf{x} \in \mathbb{R}^d$, which is done through forward propagation. Then, the Jacobian information is used to construct a saliency map $S(\mathbf{x})$, where $S(\mathbf{x})_i$ quantifies the significance of *increasing* i-th pixel to the score of class t:

$$S(\mathbf{x})_i = \begin{cases} 0, & \frac{\partial f_t(\mathbf{x})}{\partial x_i} < 0 \quad \text{or} \quad \sum_{j \neq t} \frac{\partial f_j(\mathbf{x})}{\partial x_i} > 0 \\ \Big(\frac{\partial f_t(\mathbf{x})}{\partial x_i}\Big) \Big| \sum_{j \neq t} \frac{\partial f_j(\mathbf{x})}{\partial x_i} \Big|, & \text{else} \end{cases}$$

Hence, the i-th pixel is considered important based on both how much it contributes to the increase of target class' score and the cumulative decrease in the confidence of all remaining classes. The most important pixel, according to the saliency map, is altered and the iterative algorithm proceeds until the perturbed image is classified as belonging to class t.

Other Attacks. Considering that we presented some of the most fundamental methods for generating adversarial attacks, we further extend our discussion in a more concise manner to introduce some other works too. DeepFool [Moo+16], a geometrically-inspired method, approximates the non-linear decision boundaries of DNNs through their first-order approximation and "sends" the current adversary towards the nearest decision boundary linearization. Croce and Hein [CH20a] proposed the Fast Adaptive Boundary (FAB) attack, which can be regarded as an evolution of DeepFool that simultaneously seeks for adversaries that minimize the distance from the clean input. The DDN method [Ron+19] is another instance of minimum-norm attacks, where the attacker follows the gradient direction in each step with a perturbation budget ϵ. The bound ϵ is constantly adjusted in order to find its minimum value that permits the finding of adversaries. Another line of work is involved with lending performance gains to the PGD algorithm. Gowal et al. [Gow+19] suggest that instead of investing a large number of iterations to PGD, the attacker should rather perform fewer iterations with many restarts, wherein each restart the adversary targets a different label (MultiTargeted Attack). Croce and Hein [CH20b] devise a novel optimizer, dubbed as AutoPGD, which has the appeal of being parameter-free and hence is less vulnerable to low performance due to poor hyperparameter tuning. Sriramanan et al. [Sri+20] find that the addition of an MSE regularization term to the standard margin loss enhances the performance of PGD.

4.2.4 Black Box Attacks

We referred to black box attacks as the setting where the threat model has limited information at its disposal. In our presentation, we will assume that the target network can be used as an oracle, i.e. the attacker can query the model to obtain the associated logit vector (or for some cases just the classifier's decision). We also presume that the training dataset is available to the malicious user; The real-world applicability of this conjecture can be criticized, since most of the deployed ML models do not expose that type of information. There is research that tackles the problem of adversarial attacks through a more realistic perspective. However, our aim is to superficially (and succinctly) cover the most essential body of work for black box attacks, since our contribution (See Section 5) is solely related to the white box scenario.

Transfer-Based Attacks. There is an intriguing feature associated with adversarial examples: They demonstrate *transferability* [Sze+14; Goo+15; Pap+16], meaning that the attacks generated for some network might be used to fool another one. This behaviour enables a simplistic yet powerful approach, suggested by Papernot et al. [Pap+16], to produce adversaries without knowledge of the network: (1) Design a source network S, (2) Create a synthetic dataset, where each input $\mathbf{x}$ from the original training dataset is labeled according to the output of the target network T, (3) Train S in this exact dataset (knowledge distillation) and finally (4) Perform white-box attack in the target network T and transfer those perturbations to the classifier S. Despite the effectiveness of this method on networks that are trained in a standard way, its performance is inferior against classifiers that have been trained to be robust (Section 4.3) and depends heavily on the architectural similarity between the two classifiers.

Decision-Based Attacks. Here we will discuss about a class of gradient-free methods that only requires the classifier's predicted label (instead of the logit vector) to produce adversarial examples. The most pronounced instance of this category is the Boundary Attack, proposed by Brendel et al. [Bre+18]. This method is devised based on very simple geometric intuitions: Sample a random image $\mathbf{x}_0$ from a noise distribution, which is classified differently from the clean image

$\mathbf{x}$ and continually traverse across the decision boundary which separates the misclassified with the correctly classified region, till the algorithm reaches a point which has minimal distance with $\mathbf{x}$.

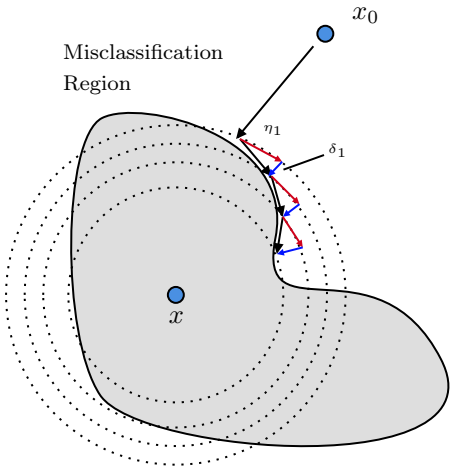


Figure 4.3. *The visualization of Boundary Decision attack.*

The traversal along the decision boundary can be approximated through a simple heuristic method comprised of two steps: A random perturbation $\boldsymbol{\eta}_{t+1}$ is sampled, such that if added to the current adversary $\mathbf{x}_t$ the ℓ_2-distance remains unaltered: $\|\mathbf{x} - \mathbf{x}_t\|_2 = \|\mathbf{x} - (\mathbf{x}_t + \boldsymbol{\eta}_{t+1})\|_2$, and then a new perturbation $\boldsymbol{\delta}_{t+1}$ is generated which can be regarded as a small nugde towards the direction of $\mathbf{x}$, i.e. $\boldsymbol{\delta}_{t+1} = \epsilon \cdot (\mathbf{x} - \mathbf{x}_t + \boldsymbol{\eta}_{t+1})$. This algorithm is vividly depicted in Figure 4.3. If the compound perturbation $\boldsymbol{\delta}_{t+1} + \boldsymbol{\eta}_{t+1}$ of $(t+1)$-th timestep leads to an image that is correctly classified then the algorithm dismisses it. Hence, this iterative procedures gets as near as possible to the original image, remaining in the misclassified region. Notice that, overall, this method only requires the classifier's decision to determine whether the perturbation is acceptable or not. Interestingly, this simple method achieved impressive results but it required many iterations in order to sufficiently decrease the visual distortion w.r.t the clean input.

Score-Based Attacks. This category contains the instances of black box attacks that only require to query the classifier, without having any information about the training dataset. The main ranking criterion between such methods is the average query rate that they need to yield adversaries. Basic white box methods cannot be applied when the classifier's structure is unknown since we are unable to obtain the input gradients. The most direct alternative to the issue of deficient information is to estimate the input gradients. Chen et al. [Che+17] propose Zero-th Order Optimization (ZOO), i.e. numerically estimate the gradients and Hessian's diagonal elements through the symmetric difference quotient formula:

$$\hat{g}_i = \frac{\partial f(\mathbf{x})}{\partial x_i} \approx \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x} - h\mathbf{e}_i)}{2h}$$
$$\hat{h}_i = \frac{\partial^2 f(\mathbf{x})}{\partial x_i^2} \approx \frac{f(\mathbf{x} + h\mathbf{e}_i) - 2f(\mathbf{x}) + f(\mathbf{x} - h\mathbf{e}_i)}{h^2}$$

where h: a small constant. Note that this approach only necessitates to query the target system f, but performing such a calculation for the entire input space is prohibitive in terms of compute. To bypass this restriction, they recommend to update only one coordinate per iteration, either by simply using the approximate gradient or even integrating the Hessian information like Newton's algorithm.

Uesato et al. [Ues+18] extended the infamous method of SPSA [Spa92] to the framework of black box attacks; Essentially, a random vector $\mathbf{u} \sim p(\mathbf{u})$ is sampled from the Rademacher distribution and the input gradients $\mathbf{g} = \nabla_{\mathbf{x}} f(\mathbf{x})$ are approximated as follows:

$$\hat{\mathbf{g}}(\mathbf{u}) = \big(f(\mathbf{x} + h\mathbf{u}) - f(\mathbf{x} - h\mathbf{u})\big) \odot \frac{1}{2h}\mathbf{u}^{-1}$$

where $\mathbf{u}^{-1}$: the element-wise inverse of vector $\mathbf{u}$. It can be proven that: $\mathbb{E}_{\mathbf{u} \sim p(\mathbf{u})}[\hat{\mathbf{g}}(\mathbf{u})] = \nabla_{\mathbf{x}} f(\mathbf{x})$, hence the gradient is estimated through the Monte Carlo approximation of the true expectation.

Another distinctive member of this black-box subcategory is the SquareAttack [And+20]. The idea behind this method is to generate random perturbations from a proposal distribution and move towards the proposed direction only if the resulting image yields higher loss value. The nature of such distribution is of paramount importance and they elaborately design it to produce noise perturbations that are localized in square-shaped regions (whose their size is constantly decaying) instead of being spread around on the image. The proposed perturbation is always drawn such that it nugdes the adversarial iterate to the surface of the ℓ_p-ball around $\mathbf{x}$, since it has been observed that successful adversarial perturbations satisfy this condition.

4.2.5 What is the reason of their existence?

The phenomenon of adversarial examples has reasonably sparked a surge of interest to the research community. The emergence of such phenomenon exposed significant holes in terms of our understanding about the inner mechanisms of Deep Neural Networks. Therefore, it is no wonder that a multitude of works invested substantial effort towards the assignment of illuminating the factors underlying this counterintuitive property. In this section, we will skim through the evolution of ideas related to such interpretations. Those works should not been necessarily regarded as mutually exclusive theses, but preferably more like non-overlapping premises that provide partial explanations.

Initially, Szegedy et al. [Sze+14] conducted an analysis on the Lipschitz constants of DNNs' layers, since adversarial examples are the exact opposite of the Lipschitz property: small input perturbations that lead to large output changes. However, they only provide upper bounds for the individual layers which are insufficient to formally explain the presence of adversaries. Goodfellow et al. [Goo+15] articulated the *linearity hypothesis*: adversarial examples, in linear classifiers, arise because the additive perturbation, even if bounded, is capable of drastically altering the activation: $\mathbf{w}^T\mathbf{x}' = \mathbf{w}^T\mathbf{x} + \mathbf{w}^T\boldsymbol{\eta}$ given that the input space is sufficiently high dimensional as is the case for natural images. They extend this explanation in the case of highly non-linear functions like that represented by DNNs, by advocating that in some regions, it may be the case that they behave too linearly.Another explanation [Son+18b; Sam+18] asserts that adversarial examples are essentially data points which lie off the regular data manifold. Song et al. [Son+18b] also demonstrate this empirically through estimating the density of adversarial attacks generated by common algorithms, where they discover that adversaries are concentrated in areas of low probability density. However, Stutz et al. [Stu+19] exhibit that it is also possible to find on-manifold adversaries. Arguably, the most convincing explanation originates from the work of Ilyas et al. [Ily+19]. In their interesting analysis, they propose to consider images as the union of two distinct sets of features/pixels: The *robust features* coincide with salient areas on an image that are of massive importance for humans to perform classification e.g. the eyes/ears/whiskers of a cat, and the *non-robust features* which essentially are trifling for the human perception. DNN classifiers learn to pick up the latter group of features during the standard training process because they are highly-predictive and more helpful in terms of generalization. Then, the adversary can modify the non-robust pixel values to flip the classifier's decision but the human perception concerning the image remains intact. Despite the fact that these ideas may induce skepticism at first, in their experimental section they carry out some fairly intuitive experiments that solidly demonstrate how these non-robust features, on their own, are capable to train classifiers with exceptional generalization ability.

4.3 Adversarial Defenses

The presence of adversarial examples indicate that the decision-making of Deep Neural Networks is unstable and extremely vulnerable, even to additive perturbations of minuscule magnitude. Over the years that succeeded the discovery of Szegedy et al. [Sze+14], researchers have been urged to provide meaningful solutions to alleviate this problematic behaviour. In this section, we will walk through the most fundamental approaches that can be found in the literature, which aim to build DNNs of increased robustness against malicious perturbations.

We should elucidate some important points related to our subsequent presentation: Over the years, there have been various methods of different "philosophy" that attempt to address the robustness problem: Someone can either alter the standard training process of the classifier, or *purify* the input, in the sense that the defender adds a module which aspires to remove adversarial noise from the image and then the classifier can confidently ingest it, or even construct an independent model that is responsible to assess the prospect that the input has been tampered with from a malevolent user, called the *detector*. Our presentation, generally, will have its content devoted to the first group of approaches and specifically for the case of ℓ_p-bounded adversaries.

Evaluating Adversarial Defenses. Before we delve into the evolved methods that improve robustness of image classifiers, it is necessary to clarify the evaluation process of such models. Given a classifier $f_{\boldsymbol{\theta}}$, where $\boldsymbol{\theta}$: its parameters, a proper metric which duly reflects the ability of $f_{\boldsymbol{\theta}}$ to perform well against ℓ_p-bounded adversaries is the *robust test error*:

$$\mathcal{R}_{\text{rob}}(f) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\Big[\max_{\mathbf{x}':\|\mathbf{x}-\mathbf{x}'\|_p\leq\epsilon} \mathbb{1}[f_{\boldsymbol{\theta}}(x') \neq y]\Big] \tag{4.6}$$

However, in practice, there two important notes to take into consideration: (1) The true expectation is approximated through the MC approximation, since we're typically in the finite-sample regimen and (2) The inner maximization problem, which requires the finding of the worst-case adversary inside an ℓ_p-ball, cannot be exactly solved for neural networks; hence, one must resort to approximate solutions by some adversarial attack algorithm e.g. PGD for $\ell_\infty-$bounded adversaries. Thus, we can only obtain an upper bound of the true robust error. Selecting the attack algorithm is crucial and an inappropriate choice can deceptively give a false sense of the true robustness; this is the problem of *robustness overestimation* which we shall thoroughly analyze in 4.3.2.

4.3.1 Building Robust Neural Networks

Adversarial Training. Recall that, when training deep neural networks, our aim is to search for the optimal parameters that yield the minimum expected risk:

$$\boldsymbol{\theta}^* : \min_{\boldsymbol{\theta}} \ \mathbb{E}_{(\mathbf{x},y)\sim p_{\text{data}}}\Big[\mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}), y; \boldsymbol{\theta})\Big] \tag{4.7}$$

However, when confronted with the prospect of ℓ_p-bounded adversarial attacks, it is necessary to adopt a distinct objective which will ultimately help to counteract the detrimental impact of such examples. The most straightforward objective to enforce ℓ_p-bounded robustness can be formulated as follows:

$$\boldsymbol{\theta}^* : \min_{\boldsymbol{\theta}} \ \mathbb{E}_{(\mathbf{x},y)\sim p_{\text{data}}}\Big[\max_{\mathbf{x}':\|\mathbf{x}-\mathbf{x}'\|_p\leq\epsilon} \mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}'), y; \boldsymbol{\theta})\Big] \tag{4.8}$$

The above equation asserts that training robust classifiers should involve the minimization of loss against the worst-case input that resides inside the ℓ_p-ball of radius ϵ around $\mathbf{x}$ (See how the need of improving ℓ_p-robustness Figure 4.4 affects the optimal decision boundaries). Indeed, such an approach was thoroughly explored in the seminal work of Madry et al. [Mad+18] and was theoretically motivated by Danskin's theorem [Dan66]. Equation 4.8 can be viewed as a saddle point (min-max) problem, comprised of two subproblems: the inner maximization and the outer minimization. The former is approximately solved through SGD, whereas the latter can be practically approached via the PGD algorithm. Despite its non-concavity, the inner problem can be solved adequately well in practice, even by first-order methods like PGD, according to the thorough analysis in [Mad+18]. Overall, this defense method is widely known as *Adversarial Training* (AT) and, alternatively, can be regarded as performing data augmentation in each batch, exploiting the worst-case perturbed inputs as the data transformation technique. The preceding work of Goodfellow et al. [Goo+15] also proposed to augment the training minibatches with adversarial examples, but they harnessed the evidently weaker FGSM algorithm. Adversarial Training is a powerful framework that has endured the test of time, since it is one of the most reliable options that the defender has at its disposal to increase resilience against ℓ_p-adversaries.

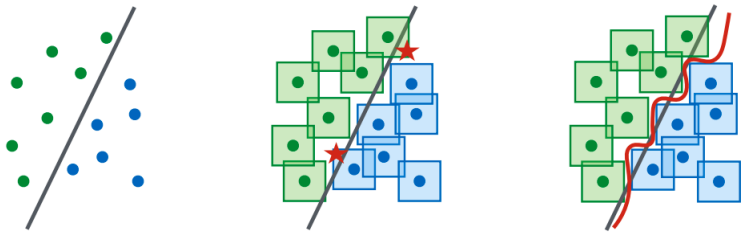


Figure 4.4. *Figure adapted from Madry et al. [Mad+18]. The simple decision boundary (middle figure) isn't enough to ensure that the system will not suffer from degraded performance versus $\ell_\infty-$bounded adversaries.*

A considerable drawback related to AT is the significant computational overhead that is imposed from the iterative PGD method. Subsequently, many variants have been developed with the end goal of accelerating AT. Free AT [Sha+19] proposes to use FGSM adversaries repeatedly on the same batch, without zeroing out the perturbations between the replays; that way, they manage to emulate the PGD algorithm but with interleaved parameter updates between PGD iterations. Despite the fact that Free AT is less time-consuming that standard AT, it is still rather slow compared to standard training. Wong et al. [Won+20] introduce *Fast* Adversarial Training: a method which exploits FGSM adversaries with random initializations and a number of other training tricks e.g. cycling learning rate schedules, that manage to reduce the total number of epochs. Fast AT matches the robust performance of the networks trained in Madry et al. [Mad+18], while simultaneously restricting the overall training time to similar levels with standard training. Rice et al. [Ric+20] observe that adversarially trained classifiers are plagued by the issue of *robust overfitting*, where overfitting in the training robust error induces a detrimental effect to the test robust error. Notably, this comes in stark contrast with the behaviour of overparameterized neural networks in the standard setting, where models are able to simultaneously overfit the training set and decrease test error. Following this remark, they identify early-stopping as the most effective way to mitigate this problem, which is as at least as effective as numerous other tricks that had been devised, prior to their work, in order to decrease robust test error.

TRADES. Aside from AT, the work of Zhang et al. [Zha+19c] inaugurated another promising "family" of defenses, called TRADES (TRadeoff-inspired Adversarial DEfense via Surrogate-loss minimization). They demonstrate that the robust error of Equation 4.6 can be decomposed as the sum of two constituent parts: The *natural classification error*, measuring the classifier's standard performance and the *boundary error* defined as:

$$\mathcal{R}_{\text{bdy}}(f) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\mathbb{1}\Big[\mathbf{x} \in \mathbb{B}(\text{DB}(f), \epsilon) : f(\mathbf{x}) = y\Big] \tag{4.9}$$

where $\text{DB}(f)$ denotes the set of points $\mathbf{x}$ that the decision boundary of f is consisted of, and $\mathbb{B}(\text{DB}(f), \epsilon)$ denotes the ϵ-neighborhood around these points. After replacing the 0-1 loss in the respective definitions with some surrogate loss $\mathcal{L}$ like cross-entropy, their theoretical analysis can be practically realized through a new training objective:

$$\min_{\boldsymbol{\theta}} \mathbb{E}\Big\{\mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}), y) + \frac{1}{\lambda}\max_{\mathbf{x}':\|\mathbf{x}'-\mathbf{x}\|_p\leq\epsilon} \mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}), f_{\boldsymbol{\theta}}(\mathbf{x}'))\Big\} \tag{4.10}$$

where λ: a regularization parameter. The first term of Equation 4.10 minimizes the standard classification error, whereas the insight behind the second term has a smoothing effect since it encourages the classifier to assign the same label both to $\mathbf{x}$ and the ℓ_p-ball around it. Essentially, this implies, in the finite-sample regimen, that the decision boundary will be pushed away from training set's data points due to this term. The solution of this maximization problem can be practically tackled by the PGD algorithm, akin to the AT framework.

More Data. Schmidt et al. [Sch+18] assert that robust generalization necessitates a greater amount of data compared with the standard setting. In addition, the second term of Equation 4.10 doesn't include any label information, since it only requires to generate the worst-case adversary $\mathbf{x}'$ for $\mathbf{x}$ based on the *pseudo-label* $f_{\boldsymbol{\theta}}(\mathbf{x})$. Based on these observations, many subsequent works managed to attain substantial robustness gains by utilizing unlabeled data [Ala+19; Car+19; Zha+19a].

Certified Defenses. The entirety of our previous discussion revolved around instances of a particular subset of defenses called *Empirical Defenses*. There is a "supplementary" subset to the empirical methods, dubbed as *Certified Defenses*. In the methods contained to the latter category, the defender provides provable robustness, in the sense that for a given data pair $(\mathbf{x}, y)$, the developed method answers exactly whether for this data point exists an ℓ_p-bounded perturbation that causes misclassification. We clarify beforehand that, despite the enticing property of proving robustness, certified methods are unable to match the performance of empirical defenses. Though this thesis' contribution is inextricably linked with the problem of robustness overestimation, which is only present on Empirical Defenses, we will briefly represent, for the sake of completeness, the most promising (and, conceptually, the most accessible) line of research in certified defenses, that is Randomized Smoothing [Coh+19].

Randomized Smoothing characterizes a smoothing transformation of a classifier $F : \mathcal{X} \to \mathcal{P}(\mathcal{Y})$, where $\mathcal{P}(\mathcal{Y})$: the set of probability distributions over $\mathcal{Y} = 1, ..., C$ (C: number of classes). This transformation amounts to convolving F with a Gaussian pdf $h(\mathbf{x}) = \mathcal{N}(\mathbf{x}; 0, \sigma^2\mathbf{I})$, resulting to the

smooth classifier G:

$$\begin{aligned} G(\mathbf{x}) = (F * h)(\mathbf{x}) &= \int_{\mathbf{t}\in\mathcal{X}} F(\mathbf{x}-\mathbf{t})h(\mathbf{t})d\mathbf{t} && \text{(h: symmetric w.r.t } \mathbf{t}\text{)} \\ &= \int_{\mathbf{t}\in\mathcal{X}} F(\mathbf{x}+\mathbf{t})h(\mathbf{t})d\mathbf{t} && \text{(h: gaussian pdf)} \\ &= \mathbb{E}_{\mathbf{t}\sim\mathcal{N}(0,\sigma^2\mathbf{I})}[F(\mathbf{x}+\mathbf{t})] \end{aligned}$$

In practice, the exact calculation of the above expression is infeasible and hence, we resort to MC approximations. The construction of the smooth version of F induces some nice properties that enable us to guarantee about the decision-making of G around a given point $\mathbf{x}$. Consider that for $\mathbf{x}$, a, b are the two most probable classes (in that order) according to the smooth classifier G, and p_a, p_b their associated confidences assigned by G. It can be deduced that the smooth classifier G assigns the same label to every point $\mathbf{x}'$ inside the ℓ_2-ball of radius $K = \frac{\sigma}{2}\Big(\Phi^{-1}(p_a) - \Phi^{-1}(p_b)\Big)$:

$$\forall \mathbf{x}' : \|\mathbf{x}-\mathbf{x}'\|_2 \leq \frac{\sigma}{2}\Big(\Phi^{-1}(p_a) - \Phi^{-1}(p_b)\Big) \Rightarrow \arg\max\, G(\mathbf{x}') = a$$

Notice that this property can be regarded as an equivalent of local Lipschitzness, where the inferred (from G) class of $\mathbf{x}$ remains invariant inside an ℓ_2-ball of radius that is dependant on the point $\mathbf{x}$ and how confidently can G classify it. Finally, the exact robustness of classifier G against ℓ_2-adversaries of maximum perturbation ϵ is equivalent to the fraction of test data points where the above constant K is at least ϵ.

A significant subtlety that needs to be understood is that in general, during training, the defender manipulates the base classifier F with the end goal of improving robustness for its smoothed counterpart G. If G isn't sufficiently competent at classification (i.e. p_a and p_b has small differences) then the yielded robustness certificates will hold for trivially low radii. Hence, the defender's main aspiration should be that the smoothed classifier performs well versus clean inputs. For example, standard training of F won't cut it; Cohen et al. [Coh+19] suggest that training F against Gaussian perturbed inputs yields good results. Though that is a reasonable heuristic to foster adequate performance for G, Salman et al. [Sal+19] argue that Gaussian data augmentation isn't quite the same with training in a way such that G yields low objective value. Instead, they recommend an alternative which explicitly aims at improving the performance of G, by using MC approximations to estimate the output of G (and, correspondingly, the plug-in estimator during backward propagation).

4.3.2 The problem of Robustness Overestimation

Previously, we mentioned that evaluating ℓ_p-robustness of image classifiers amounts to the computation of the robust test error, as it can be seen in Equation 4.6. Alternatively, robustness is also quantified with the *robust accuracy* metric, which essentially is the 0-1 equivalent of robust test error:

$$\text{RobAcc}(f) = \frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|} \min_{\mathbf{x}_i' : \|\mathbf{x}_i - \mathbf{x}_i'\|_p \leq \epsilon} \mathbb{1}[f(\mathbf{x}_i') = y_i] \tag{4.11}$$

The exact computation of such expressions is infeasible, in the context of Deep Learning. In practice, one obtains local minimizers that are found by first-order approximate attack algorithms such as PGD. However, there is an evident danger lurking: An inappropriate selection of the attack algorithm may lead to unduly high robust accuracy which doesn't properly reflect the true degree

of ℓ_p-robustness. This problematic behaviour is widely known as the issue of *robustness overestimation* and it is the main cause of major setbacks in the discipline of Adversarial Defenses. The primary factor that induces iterative gradient-based attacks to break down is relatively obvious: the defended network operates in some way that impedes gradient descent, a phenomenon called *gradient masking*. Robustness overestimation has been the epicentre of two works [Ath+18; Ues+18], where they managed to completely circumvent i.e. drop the robust accuracy to $\approx 0\%$, multiple adversarial defenses that had been published in the most prestigious ML venues, despite that the authors of the respective publications had reported great robustness results. These defenses, almost invariably, contained some stochastic or non-differentiable module hence the gradient-based attacks were doomed to fail.

Athalye et al. [Ath+18] managed to evade these defenses by improving gradient-based attacks through the deployment of some novel ploys such as the Backward Pass Differentiable Approximation (BPDA) method, where some not-usefully differentiable module of the threatened network is used only for forward propagation and during backprop is replaced by some differentiable approximation and Expectation over Transformation (EOT), particularly useful against stochastic networks, where the excessive amount of noise in the gradient signal is counteracted by feeding the input multiple times, and averaging the gradient. Uesato et al. [Ues+18] broke down defenses through black-box attacks, generated by the SPSA method, which we analyzed in 4.2.4. Besides their contribution on the side of breaking defenses, Athalye et al. [Ath+18] also discovered numerous indicators whose presence foreshadows that the defending model may suffer from gradient masking. In a similar vein, Tramer et al. [Tra+20] managed to evade another collection of ℓ_p-bounded defenses, by the construction of suitable *adaptive attacks*, that is, attacks which are specifically designed to fool a certain network by "reverse engineering" on its apparent weaknesses. Amazingly, this work demonstrated that defenders succumbed to the pitfall of robustness overestimation, even though the problem had been already brought to light by Athalye et al. [Ath+18]. All in all, these works increased the amount of skepticism associated with assertions about ℓ_p-bounded robustness and overall heightened the vigilance of researchers.

We can derive two significant observations from the above discussion: First, an appropriate and careful selection of the attack algorithm is vital, because, otherwise, the defender may be led astray and make false claims about the performance gains of his/her method. Second, it becomes evident that evaluating ℓ_p-bounded robustness is a troublesome procedure, seemingly far from being standardized as other Computer Vision benchmarks.

RobustBench. In an effort to standardize the assessment of ℓ_p-bounded robustness, Croce et al. [Cro+21] established the RobustBench benchmark, which performs the evaluation of ℓ_p-defenses using the AutoAttack [CH20b] method. This attack algorithm is the ensemble of four adversarial attacks: AutoPGD (with two different losses), introduced in [CH20b], which constitutes a parameter-free optimizer to counteract the negative impact of poor hyperparameter tuning, black-box SquareAttack [And+20] and FAB attack [CH20a]. The presence of the black-box component ensures that the ensemble contains enough diversity, in order to expose the models to dangers of different rationale. The establishment of AutoAttack as the default choice to evaluate robustness clearly stems from their impressive results: In the experimental analysis of [CH20b], they manage to drop the originally reported accuracies of over 50 works on ℓ_p-bounded adversarial defenses. The drop rate on some defenses is modest e.g. 0.5-1%, whereas for a small amount of cases, AutoAttack attains to completely circumvent the respective defense. Therefore, the selection of AutoAttack seems a reasonable and reliable option to evaluate adversarial defenses, at least on a primary

stage. Then, one is also encouraged to build *adaptive attacks*, akin to [Tra+20], particularly if there are signs that indicate gradient obfuscation. However, the submission of adversarial defenses to the RobustBench leaderbord is accompanied with a boolean flag which gives information about whether AutoAttack is unreliable or not.

The library of RobustBench also contains a huge collection of pre-trained robust models which are accessible with a minimal amount of effort. The very existence of such a collection offers the opportunity to researchers that do not possess enough compute to use off-the-shelf robust models either for evaluating the effectiveness of new adversarial attacks or for other downstream tasks. The author of this present dissertation expresses his gratitude to the team that maintains the project of RobustBench, since the pre-trained models allowed the thesis' experimental contribution to be considerably less computationally intensive.

4.3.3 Intriguing Properties of Robust Neural Networks

Increasing the resilience against perturbed inputs of small magnitude has been the primary goal of works in the literature of Adversarial Defenses. However, methods that increase robustness, especially the Adversarial Training framework, have sparked numerous unsolicited properties which are of great interest by their own right.

Generalization. It has been repeatedly observed in many works that Adversarial Training has a detrimental effect on the accuracy of clean inputs, establishing a trade-off between generalization and robustness. Tsipras et al. [Tsi+19] build on these empirical observations and create a synthetic data distribution where it is impossible to simultaneously achieve high standard and robust accuracies, leading them to the conclusion that there might be an inherent tension between these two goals. Stutz et al. [Stu+19] ascribe the observed tension to the distribution shift during Adversarial Training; essentially, during AT, the classifier is minimizing the loss function against adversaries that leave the data manifold and its performance is compromised when confronted with regular data. Based on that, they propose an algorithm to generate approximate on-manifold adversaries and exploiting them during AT doesn't harm generalization. Another prescribed interpretation about this trade-off stems from the work of Xie et al. [Xie+20], where they attribute this phenomenon to Batch Normalization (BN) layers. BN layers rely on the assumption that the inputs are drawn from the same data distribution. Adversarial Training, however, coerces the BN layers to learn statistics (mean and std) of a shifted distribution, hence when these statistics are used for clean data during inference, the network's performance is negatively affected because of this distribution misalignment.

Interpretability. Initially, Tsipras et al. [Tsi+19] discovered that adversarially trained models produce image gradients that align with "human perception", since they are resembling low-level image features like edges (c.f. Figure 4.5). This enticing feature also appears in models that are adversarially trained for small perturbation budgets ϵ [Agg+20]. Additionally, it was shown that the "human-aligned" gradients property also holds for certifiable robust classifiers [Kau+19], and thus it should not be exclusively attributed to the adversarial training framework. Later, Engstrom et al. [Eng+19a] demonstrated that the representations learned by robust models are invertible, meaning that for an ℓ_p-bounded adversarially trained model, given the representation vector $R(\mathbf{x})$ of the penultimate layer, we can approximately recover the original image $\mathbf{x}$ (see Figure 4.6) through simple gradient descent on the MSE loss between the represenations of the true input $\mathbf{x}$ and the optimized image $\mathbf{x}'$. An interesting implication of these findings is that maximizing the prediction

confidence of image $\mathbf{x}$ for some class y_t amounts to visually introducing salient characteristics of this particular target class (c.f. Figure 4.7). Motivated by this, Santurkar et al. [San+19] solved popular vision tasks such as Image Synthesis, Impainting, Image-to-Image translation and Super Resolution just by harnessing a single robust classifier.

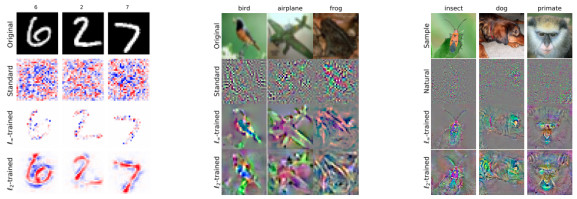


Figure 4.5. *The input gradients of standard models (2nd row) are noisy signals, whereas the input gradients of adversarially trained models resemble low-level features of the images. Figure adapted from [Tsi+19].*

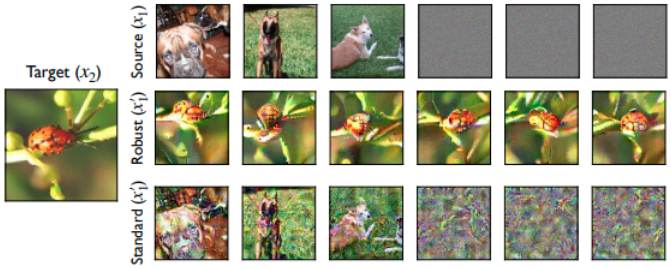


Figure 4.6. *The left-most picture represents the image* $\mathbf{x}$ *for which the penultimate layer's feature map* $R(\mathbf{x})$ *is known. In the right, the top row shows the random seed from where the gradient descent (GD) is initialized. Inverting the representation via GD on robust models produces an image* $\mathbf{x}'$ *which resembles the true input* $\mathbf{x}$ *(middle row), whereas in standard-trained networks the output is visually more similar to the random seed (bottom row). Figure adapted from [Eng+19a].*

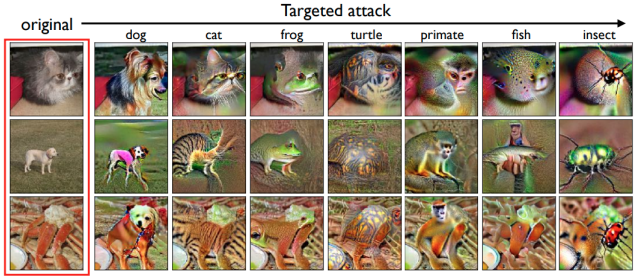


Figure 4.7. *Maximizing class scores of a robustly trained classifier. For each original image, we visualize the result of performing targeted projected gradient descent (PGD) toward different classes. The resulting images actually resemble samples of the target class. Figure adapted from [San+19].*

Transfer Learning. An additional unexpected benefit of robust classifiers is their superiority on *Transfer Learning* related applications [Sal+20; Utr+20]. In specific, those works studied the performance of robust models as feature extractors for downstream tasks: Interestingly, despite the inability of robust models to perform on par with their standard trained counterparts in terms of clean accuracy [Tsi+19], using them as feature extractors led to better performance on downstream

tasks. This result added more stable grounding in the assertion that robust models learn more meaningful representations.

Invariance to biases. The adversarial learning goal can be viewed as a form of data augmentation, wherein the network learns to be invariant against worst-case additive perturbations. Recently, Geirhos et al. [Gei+19] demonstrated that standard training results to classifiers that are texture-biased and ignore shape information which is deemed (from human subjects) more important than texture when it comes to object classification. Zhang and Zu [ZZ19] displayed that standard trained CNNs are more texture-biased than adversarially trained CNNs.

4.3.4 Other forms of Robustness in Deep Learning

Despite the emergent agitation that accompanied the discovery of adversarial examples, research community is considerably skeptic about the real-world implications of *worst-case* adversaries. A viewpoint of heightened popularity asserts that adversarial examples aren't much of a nuisance in practical scenarios, since, so far, there isn't a tangible case where a machine learning model was damaged by worst-case adversaries. Nevertheless, studying the phenomenon of adversarial attacks is interesting because they showcase serious functional blind-spots whom their study impels us to deepen our understanding of Deep Neural Networks. In the following text, we will visit other instances of instability associated with the decision making of such models.

The concerns surrounding the incompatibility of adversarial attacks with practical implications has steered the attention of researchers to study other forms of robustness. Common Corruptions, introduced by Hendrycks et al. [HD19], is a popular computer vision benchmark that, instead, evaluates the behaviour of models against a wide variety of perturbations such as noise, blurrings, tricky weather conditions etc., which are deemed as a much more realistic threat to occur in the real-world. Importantly, an inherent difference between the evaluation versus common corruptions and adversarial examples is that the assessment against common corruptions is carried out in a held-out test-set (CIFAR10-C,ImageNet-C etc.) which is fixed, regardless of which classifier is assessed, hence there is no risk of overestimating robustness. Also, there is a plethora of works which studies the relationship of adversarial robustness with the benchmark of Common Corruptions, and the effectiveness of methods from the former category to the latter setting. Interestingly, improvements of robustness against ℓ_p-bounded worst-case adversaries doesn't necessarily induce increased stability versus common corruptions. Similarly, Hendrycks et al. [Hen+21] created a pair of another challenging computer vision benchmarks, namely ImageNet-A and ImageNet-O, where the performance of machine learning models is degraded in a great extent.

Of course, another line of research work that pertains to the field of neural networks' robustness is that of Geirhos et al. [Gei+19], where the authors discovered that CNNs are biased towards texture whereas human decision making is predicated more on shape information. This study demonstrates the lack of CNNs' robustness when they are confronted with images of conflicting texture and shape cues.

Chapter 5

Alternating Objectives Generates Stronger PGD-Based Adversarial Attacks

5.1 Abstract

Designing powerful adversarial attacks is of paramount importance for the evaluation of ℓ_p-bounded adversarial defenses. Projected Gradient Descent (PGD) is one of the most effective and conceptually simple algorithms to generate such adversaries. The search space of PGD is dictated by the steepest ascent directions of an objective. Despite the plethora of objective function choices, there is no universally superior option and robustness overestimation may arise from ill-suited objective selection. Driven by this observation, we postulate that the combination of different objectives through a simple loss alternating scheme renders PGD more robust towards design choices. We experimentally verify this assertion on a synthetic-data example and by evaluating our proposed method across 15 different ℓ_∞-robust CIFAR-10 models. The performance improvement is consistent, when compared to the single loss counterparts. Additionally, our strongest adversarial attack outperforms all of the white-box components of AutoAttack (AA) ensemble [CH20b], as well as the most powerful attacks existing on the literature, achieving state-of-the-art results in the computational budget of our study ($T = 100$, no restarts).

5.2 Introduction

The advent of Deep Learning (DL) caused a paradigm shift and revolutionized the way that various interesting applications are approached. Such a wide adoption, however, demands from the research community to comprehend the scenarios where Deep Neural Networks (DNNs) malfunction. This necessity becomes even more imperative when considering the abundance of safety-critical applications that do not leave room for complacency, e.g., autonomous driving. Unfortunately, DNNs have significant failure modes and behave counterintuitively. A prominent instance of this behaviour is illustrated by Szegedy et al. [Sze+14], where they showcase that DNN-based image classifiers are vulnerable against *adversarial examples*. These examples arise from applying humanly imperceptible perturbations to clean images, which are capable of degrading the model's predictive performance. This finding triggered research interest on two fronts: *Adversarial Attacks*, which are algorithms to generate such malicious examples and *Adversarial Defenses*, which are methods of increasing the robustness of neural networks. Adversarial robustness is primarily studied through the ℓ_p-bounded threat model, where the perturbation's ℓ_p-norm is bounded by a small constant.

The robustness of Adversarial Defenses, on a given dataset, is estimated by the rate of test set's adversarial examples that the defense can properly classify. Of course, the estimated rate (also called robust accuracy) depends on the strength of the attacking algorithm that will be used for evaluation. Employing weak attacks to evaluate robustness creates a false sense of security, an issue widely known as robustness overestimation [Ath+18; Ues+18; Tra+20].

Arguably, Projected Gradient Descent (PGD) is the most popular adversarial attack used to evaluate ℓ_p-bounded robustness. PGD operates by iteratively following the steepest ascent directions of an objective function, often called the surrogate. PGD has raised in many guises in the adversarial attack literature: Goodfellow et al. [Goo+15] propose to attack networks through the Fast Gradient Sign Method (FGSM), which takes a single normalized step, i.e., applying the sign function in the case of ℓ_∞-norm, towards the steepest ascent direction. Kurakin et al. [Kur+17] demonstrate that the multi-step variants of FGSM are capable of producing significantly stronger attacks. Dong et al. [Don+18] suggest a modification of the iterative FGSM that integrates a momentum term. Madry et al. [Mad+18] link the iterative FGSM with the classical optimization algorithm of PGD.

Despite that PGD combines both simplicity (in terms of implementation) and strength, it has been shown that its performance can be hindered by ill-suited selection of hyperparameters, e.g., fixed step size [CH20b]. Another hyperparameter of consideration is the surrogate loss, for which literature has converged into 3 options: Cross-Entropy (CE) [Goo+15; Mad+18], Margin (a.k.a. CW) loss [CW17] and the Difference of Logits Ratio (DLR) loss [CH20b], with the appealing property of scale-invariance. However, empirical evidence (e.g., as in Figures 9-11 of [CH20b]) shows that there is no universally superior objective and its effectiveness depends on the architecture, weights, training dataset etc. On top of this, certain choices may be improper in special problematic cases: 1) CE yields zero gradients for inputs where the classifier assigns the entire probability mass to the ground truth class [CW17; CH20b], 2) both CE and CW are not scale-invariant hence logit rescalings may induce gradient masking [CH20b] and 3) Ma et al. [Ma+20] assert that objectives which involve multiple logit terms, i.e., all three of CE,CW and DLR, may suffer from the problem of *gradient imbalance* where logits have quite disparate magnitudes and one term alone steers the optimization trajectory towards non-optimal solutions.

In this work, PGD is studied from the perspective of surrogate loss. In order to alleviate potentially weak PGD performance arising from poor surrogate selection, we propose to combine different objectives in the same run of PGD. Hopefully, this combination will render PGD less dependent to the surrogate hyperparameter. We identify that a simple alternation of objectives during PGD is sufficient to induce significant boost on the PGD performance over the single loss variants. Further qualitative analysis implies that the switching between different objectives helps the algorithm to expand its search space, visiting more distant intermediate points during its execution.

In this paper, we make the following key contributions:

- We propose to combine multiple objectives during PGD through alternating between them during optimization, in order to alleviate potential flaws of each objective. Our proposed strategy outperforms, in 15 out of 15 tested $\ell_\infty-$bounded robust models, both the single-loss variants as well as the three white-box components of AutoAttack [CH20b]: APGD_{CE}, APGD_{DLR} and FAB attack [CH20a]. Furthermore, in most cases our attack achieves higher Attack Success Rate (ASR) than the strongest baselines (for $T = 100, R = 1$) in the literature: GAMA-PGD [Sri+20] and MD attack [Ma+20].

- We present extensive experimentation and analysis regarding the proposed alternation scheme,

including: 1) A synthetic example which highlights how PGD with a single loss can fail, 2) Qualitative analysis indicating that switching losses promotes search diversity and 3) Ablation experiments which demonstrate that this loss combination strategy is more effective than two other combining methods.

The remainder of this paper is organized as follows: Section 5.3 provides the necessary background, covering basic aspects of the worst-case ℓ_p-bounded adversarial robustness, Section 5.4 briefly discusses research work related to PGD-based attacks, since PGD is the main topic of our study. In Section 5.6 we conduct numerous experiments to verify the effectiveness of our proposed method, whereas in Section 5.7 we discuss how our study differs from previous related work.

5.3 Background

5.3.1 Notation

Image-label pairs are denoted as $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$ where $\mathcal{X} \subseteq \mathbb{R}^D, \mathcal{Y} \subseteq \mathbb{Z}$. The classifier's logit representation will be denoted as $\mathbf{z}(\mathbf{x}) \in \mathbb{R}^C$ (or simply $\mathbf{z}$), where C: the total number of classes. Applying a softmax layer to the logit vector produces the probability vector $p(y|\mathbf{x})$. The classification decision will be denoted as $f(\mathbf{x})$, hence $f(\mathbf{x}) = \underset{i \in [C]}{\arg\max}\, \mathbf{z}(\mathbf{x})_i$, where $[C] = \{1, ..., C\}$. The surrogate loss $\mathcal{L}(\mathbf{z}(\mathbf{x}), y)$ (which will also be referred as $\mathcal{L}(\mathbf{x}, y)$, for brevity's sake), e.g., cross-entropy, measures the model's ability to assign the label y to example $\mathbf{x}$.

5.3.2 Threat Model

The constraint of visual imperceptibility is approximated through the bounded ℓ_p-norm condition. The generation of adversarial attacks should obey this restriction, returning an output that lies within the ℓ_p-ball of radius ϵ around the clean input $\mathbf{x}$. Hence, the feasible search space of adversaries for the image $\mathbf{x}$ can be expressed as:

$$\Delta(\mathbf{x}) = \{\mathbf{x}' : \|\mathbf{x} - \mathbf{x}'\|_p \leq \epsilon\}$$

Despite that the ℓ_p-bounded threat model is only a crude approximation of true visual similarity, solving the problem of ℓ_p-bounded robustness can be viewed as an important stepping stone towards confronting more realistic scenarios.

5.3.3 A taxonomy of ℓ_p-bounded adversarial attacks

Next we present a basic categorization of adversarial attacks based on their capabilities during generation and their end goal.

Adversary's Knowledge. Based on the amount of information that the adversary has at its disposal, attacks can be divided into two major categories: *white-box* and *black-box*. In the former, the attacker has access to every aspect of the model: its architecture, weights and training data. This allows the adversary to obtain the network's gradients w.r.t. the input which is particularly useful when creating attacks. In the latter category, however, the adversary can only use the model as an oracle, feeding an input point and getting access to the output vector, or sometimes just to the output class.

Despite that typical real-world scenarios are more similar to the black-box setting, white-box attacks constitute a much more stronger threat model. Therefore, the evaluation of adversarial defenses is typically performed based on white-box attacks.

Low Confidence vs Low Distortion. Attacks are also divided into minimum-confidence and minimum-norm. In the former, the attack algorithm is based on the following formulation, for the input-label pair $(\mathbf{x}, y)$:

$$\boldsymbol{\delta} : \max_{\boldsymbol{\delta}} \ \mathcal{L}_{0/1}(f(\mathbf{x}+\boldsymbol{\delta}), y) \text{ s.t. } \mathbf{x}+\boldsymbol{\delta} \in \Delta(\mathbf{x})$$

where $\mathcal{L}_{0/1}(f(\mathbf{x}), y) = \mathbb{1}[f(\mathbf{x}) \neq y]$ is the 0-1 loss, which due to its discontinuity is replaced by some surrogate loss $\mathcal{L}$ such as cross-entropy. These attacks aim to reduce the ground truth label's confidence as much as possible by spending the entire attack budget ϵ, hence they typically lie on the boundary surface of the feasible set Δ. The most prominent examples of minimum-confidence adversarial attacks is the Fast Gradient Sign Method (FGSM) [Goo+15], the Iterative-FGSM [Kur+17] and Projected Gradient Descent (PGD) [Mad+18].

Minimum-norm attacks aspire to find the smallest possible perturbation that leads to misclassification:

$$\boldsymbol{\delta} : \min_{\boldsymbol{\delta}} \|\boldsymbol{\delta}\|_p \text{ s.t. } f(\mathbf{x}+\boldsymbol{\delta}) \neq y$$

where y: the ground-truth label of $\mathbf{x}$. Such attacks usually find adversaries that are within smaller ℓ_p-distance from the clean input $\mathbf{x}$ than the perturbation bound ϵ. Popular examples of this category are: Carlini-Wagner (CW) attack [CW17], DDN attack [Ron+19], Fast Minimum Norm (FMN) [Pin+21] and Fast Adaptive Boundary (FAB) attack [CH20a] among others.

Untargeted vs Targeted. Another criterion of dividing adversarial attacks is whether the adversary desires to force a specific label to the attack. In *targeted* attacks, the attack is considered successful if the corresponding adversarial example is classified into a certain target class. In *untargeted* attacks, the goal is simply to produce an example which is incorrectly classified, with no constraint on its new label. Usually, the transition between the two categories is as simple as slightly modifying the objective function, i.e., from descending the target label's confidence to ascending the ground-truth label's confidence.

5.3.4 Empirical Adversarial Defenses

Training ℓ_p-robust neural networks, i.e., networks that are resilient against ℓ_p-bounded adversarial attacks, is a complicated problem since we aspire to simultaneously realize two goals. First, the classifier is asked to perform well on unseen examples drawn from the same distribution as the examples used during training. An additional requirement is to find networks that produce smooth predictions, assigning the same label to all data residing inside the ℓ_p-ball of such examples. The most standard way of increasing ℓ_p-bounded robustness is Adversarial Training (AT) [Goo+15; Mad+18]; in AT, the defender aims to minimize the *robust expected risk*:

$$\mathcal{R}^f_{\text{rob}}(\boldsymbol{\theta}) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\Big[\max_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_p\leq\epsilon} \mathbb{1}[f_{\boldsymbol{\theta}}(\mathbf{x}+\boldsymbol{\delta}) \neq y]\Big] \tag{5.1}$$

The inner expression coincides with the task of finding the worst-case ℓ_p-bounded adversarial example. Madry et al. [Mad+18] confront the problem through the first-order method of PGD. An important barrier of this method is the additional computational overhead. The iterative PGD process renders this method costly in terms of compute, hence a line of research aims to increase

robustness using one-step adversaries [Sha+19; Won+20; Ric+20; AF20], in order to restrain the overall training time to similar levels as with standard training. Another important work on adversarial defenses is the TRADES framework, introduced by Zhang et al. [Zha+19c]. The robust expected risk of Equation 5.1 can be decomposed as the sum of two individual terms. The first term represents the *classification error*, where the optimization searches parameters that generalize well. The other term, dubbed as *boundary error*, can be considered as exerting a regularizing effect, where it imposes decision "smoothness" between inputs inside the same ℓ_p-ball.

Schmidt et al. [Sch+18] provide evidence that adversarially training classifiers may require an increasing amount of data. Following this, many works [Car+19; Zha+19a; Ala+19] explore the use of both pseudo-labeled additional data and elaborate data augmentation techniques.

Robustness Overestimation. Evaluating the true degree of ℓ_p-bounded robustness of empirical methods is intractable, since one needs to calculate the average 0-1 risk on a held-out test set. Typically, the defender deploys a strong attacking algorithm to obtain a lower bound on the true risk. However, this trial-and-error technique can provide misleading results. Failing to select a proper attacking algorithm creates an inaccurate sense of security [Ath+18; Ues+18; TB19]. Importantly, these works propose numerous indicators that demonstrate whether the evaluation suffers from this issue and guidelines of how to properly evaluate a defense.
The introduction of RobustBench [Eng+19c], based on the AutoAttack ensemble (comprised of three white-box [CH20b; CH20a] and one black-box [And+20] methods), contributed to a consensus regarding the evaluation of ℓ_p-bounded robustness: A newly proposed defense is first "passed" through an AutoAttack evaluation, and then the defender can also perform adaptive attacks [TB19], based on potential model-specific weaknesses.
Despite the general adoption of AutoAttack as the standard way to perform first-order robustness evaluations, the community is constantly exploring faster and more powerful attack ensembles [Liu+22; Yu+21].

5.4 Related Work

Projected Gradient Descent (PGD) [Mad+18; Kur+17] is the most popular minimum-confidence attack. PGD has been the de facto standard for producing ℓ_p-bounded adversarial attacks, especially in the case of $p = \infty$. In short, PGD can be expressed as:

$$\mathbf{x}^{(t+1)} = \mathcal{P}_{\Delta(\mathbf{x})}\left[\mathbf{x}^{(t)} + \eta^{(t)}\boldsymbol{\delta}^{(t)}\right] \tag{5.2}$$

where $\mathbf{x}^{(t)}$: the iterate, $\eta^{(t)}$: step size, $\boldsymbol{\delta}^{(t)}$: update rule of t-th iteration and $\mathcal{P}_{\Delta}$: the projection operation, which maps the updated iterate into the feasible region Δ, which in our case is the ℓ_p-ball of radius ϵ around $\mathbf{x}$. Typically, this procedure is repeated multiple times from different random initializations. For a more comprehensive view of how PGD is used to generate adversarial attacks, we refer to the work of Gowal et al. [Gow+19], where they present a "holistic" pseudoalgorithm.

In the following discussion we present how one can manipulate the basic building blocks of PGD, namely the optimizer, step size, initialization strategy and surrogate loss, in order to improve its adversarial generation stregnth.

Optimizer. The optimizer determines the form of the update rule $\boldsymbol{\delta}^{(t)}$. In its simplest version, assuming the surrogate loss $\mathcal{L}(\mathbf{x}, y)$, PGD follows the steepest direction of unit ℓ_p-norm, e.g., the

sign of $\nabla_{\mathbf{x}^{(t)}}\mathcal{L}(\mathbf{x}^{(t)}, y)$ in the case of $p = \infty$, or a simple norm-rescaling when $p = 2$. In the C&W attack [CW17], the proposed objective is optimized through Adam [KB15]. The Adam optimizer has also been leveraged in PGD-based works [Gow+19; Ues+18]. Dong et al. [Don+18] suggested the incorporation of momentum [Pol64] in the PGD update rule. Subsequently, Croce and Hein [CH20b] proposed the AutoPGD (APGD) variant, wherein the update term is augmented by momentum. Yamamura et al. [Yam+22] developed the Auto Conjugate Gradient (ACG) method, which is an elaborate optimizer, adjusting the update rule based on accumulated gradient information from previous steps. ACG is experimentally shown to outperform APGD for a sizable collection of robust models.

Step Size. Another hyperparameter which affects the performance of PGD is the step size $\eta^{(t)}$. In early works, its value is held constant during the entire optimization procedure, e.g., to $\alpha = \epsilon/4$ for ℓ_∞-attacks in CIFAR-10. Croce and Hein [CH20b] conduct large-scale experiments regarding the optimal fixed value, but one immediate corollary is that it greatly depends on the model. Generally, the common trend is to perform some kind of scheduling, where the step size is gradually reduced over time: In [Gow+19], [Sri+20], the authors apply ten-fold drops at two intermediate timesteps; Ma et al. [Ma+20] propose a cosine-annealing scheme, where the step size decays from 2ϵ to 0. In their recent work, Liu et al. [Liu+22] adopt a similar decaying strategy. Another interesting way of manipulating this hyperparameter is as in the AutoPGD method [CH20b]; They initially set it to a large value $\alpha = 2\epsilon$, in order to explore the search space sufficiently well. Then, as the optimization proceeds and the iterate gets closer to some local optimum, the need of a more localized search calls for smaller step sizes. Hence, it is halved in specific checkpoints, according to the optimization progress, i.e., based on whether the objective function is reducing or not.

Initialization. Proper initialization plays also a crucial role in the final performance. Typically, the initial point $\mathbf{x}^{(0)}$ can be either set to the clean image $\mathbf{x}$, or alternatively, random noise may be added to the clean image: $\mathbf{x}^{(0)} = \mathbf{x} + \boldsymbol{\zeta}$, where $\boldsymbol{\zeta}$ is drawn from some noise distribution. The attack is then repeated multiple times, initialized from different starting points. Tashiro et al. [Tas+20] suggest that random initialization may lead to starting points with nearly identical output space representations, hence the attack generates similar results even if executed for many restarts. Output Diversified Initialization (ODI) [Tas+20] counteracts this by maximizing the similarity of starting point's logit vector with a random output direction, in the first few PGD iterations. Recently, Liu et al. [Liu+22] introduced Adaptive AutoAttack (A^3), the new state-of-the-art attack ensemble. A^3 uses an adaptive initialization strategy, where the starting points are generated by ODI, but instead of following random output space direction, the vector is selected according to prior knowledge of perturbations that led to misclassification.

Surrogate Loss. The maximization of 0-1 loss is intractable for complex function classes as those represented by deep neural networks [Aro+97]. It is common to substitute it with a surrogate, differentiable loss which is amenable to optimization methods. A natural candidate is the cross-entropy (CE) objective, which coincides with the negative log-likelihood of the ground truth class. In their seminal work, Carlini and Wagner [CW17] tested various formulations, obtaining the best performance for the so called margin (or CW) loss. A shared defect in both of these objectives is the lack of scale-invariance, which may be translated in deteriorated performance due to gradient masking. Croce and Hein [CH20b] introduce the Difference of Logits Ration (DLR) loss, which rescales the margin loss to acquire the property of scale-invariance. Most of the literature involves

these three options, whose expressions are included below for completeness:

$$\begin{aligned} \mathrm{CE}(\mathbf{x}, y) &= -\log p(y|\mathbf{x}) = -\mathbf{z}_y + \log \sum_{j=1}^{C} \exp(\mathbf{z}_j) \\ \mathrm{CW}(\mathbf{x}, y) &= -\mathbf{z}_y + \max_{j \neq y} \mathbf{z}_j \\ \mathrm{DLR}(\mathbf{x}, y) &= -\frac{\mathbf{z}_y + \max_{j \neq y} \mathbf{z}_j}{\mathbf{z}_{\pi_1} - \mathbf{z}_{\pi_3}} \end{aligned} \tag{5.3}$$

where $\mathbf{z}_\pi$: the logit vector sorted in descending order. Gowal et al. [Gow+19] propose the Multi-Targeted PGD variant which divides the iteration budget into runs of equal size, where each run optimizes the targeted margin loss, for a different target label per run. Their experiments indicate that the MultiTargeted strategy exploits more judiciously the given computational budget. Sriramanan et al. [Sri+20] augment the standard margin loss expression with a regularization term which is set to the MSE between the logit vector of the adversary and its clean counterpart. The weighting coefficient of MSE term is gradually decayed to zero. Ma et al. [Ma+20], in an effort to ameliorate the issue of imbalanced gradients, optimize only one of the two margin loss terms for the first half of iterations before switching to the typical expression which contains both terms. In the next restart, they repeat the process by using the other term for the first stage of optimization.

5.5 Methodology

Our work is motivated by the observation that a single surrogate loss is unable to perform equally well across different robust models. Croce and Hein [CH20b] provide strong empirical evidence to back up this argument. Specifically, in their study they investigate the effectiveness of three objectives: CE, CW and DLR. These three aforementioned objectives have expressions that are distinguished by small differences, yet each option can profoundly influence the Attack Success Rate (ASR) of PGD. Of course, this phenomenon is not surprising at all: the optimization space coincides with the high-dimensional pixel space of natural images, hence even just a rescaling that links the CW with DLR loss is capable of producing non-trivial discrepancies in the respective loss landscapes. Above all, it is critical to bear in mind the surrogate loss as another hyperparameter, akin to step-size or optimizer, which has the potential of causing some degree of robustness overestimation on its own right.

The most straightforward mitigation for this behaviour is to aggregate many different formulations in the same run of PGD. The aggregation of objectives may be instantiated in a variety of ways. Our work is based on a simple *idea* for performing such an aggregation: Divide the PGD process into multiple successive stages, where the surrogate loss changes in the beginning of every stage, and the starting point of every stage coincides with the last step iterate of the previous one. This procedure, when using K stages, can be formulated as:

$$\mathcal{L}(\mathbf{x}, y) = \begin{cases} \mathcal{L}_1(\mathbf{x}, y), & \text{if } t < \frac{T}{K} \\ \mathcal{L}_2(\mathbf{x}, y), & \text{if } \frac{T}{K} \leq t < \frac{2T}{K} \\ \quad \vdots & \\ \mathcal{L}_K(\mathbf{x}, y), & \text{if } \frac{(K-1)T}{K} \leq t < T \end{cases}$$

In this paper, we will consider the cases where $K = 2, 3$, using for surrogates the most common choices: CE, CW and DLR.

Notice how this alternation strategy can be viewed as a more complicated initialization: Each PGD stage starts from the initial point $\mathbf{x}^{(0)} = \mathbf{x} + \boldsymbol{\delta}$, where $\boldsymbol{\delta}$: the accumulated perturbation of all previous stages. Of course, an immediate extension is to consider variable starting timesteps t_k for stage k, but in this work, we heuristically set equal time intervals between all stages. In the remaining discussion, our loss switching variant will be referred eas follows: $\text{PGD}_{\mathcal{L}_1\&\mathcal{L}_2\&\ldots\&\mathcal{L}_K}$, e.g., PGD_{CE} for simple PGD with CE surrogate and $\text{PGD}_{\text{CE\&CW}}$ for two-stage PGD with CE and CW.

5.6 Experiments

5.6.1 Toy Example.

We present a toy example which elucidates that using a single surrogate during PGD may deteriorate performance. Assume a 2D problem of 3-way classification (classes: y_1, y_2, y_3). Inputs are $\mathbf{x} = (x_1, x_2)^T$ and the linear classifier is $\mathbf{z} = (z_1, z_2, z_3)^T = \mathbf{W}\mathbf{x}$, with:

$$\mathbf{W} = \begin{bmatrix} 0.3 & -0.3 \\ 1 & -0.01 \\ -0.25 & 0.75 \end{bmatrix}$$

Consider an input $\mathbf{x} = (-0.45, -0.8)$, belonging to the class y_1. The linear model classifies it correctly to its ground-truth class, since $z_1 > \max(z_2, z_3)$. Suppose that our goal is to generate a perturbation δ of bounded ℓ_2-norm (say $\epsilon = 0.4$). A straightforward way to achieve this is by executing PGD, maximizing a surrogate loss, e.g., CE or CW. For the input $\mathbf{x}$ of class y_1, these losses are analytically calculated as:

$$\text{CE}(\mathbf{x}, y) = -z_1 + \log\big(\sum_{j=1}^{3} \exp(z_j)\big)$$
$$\text{CW}(\mathbf{x}, y) = -z_1 + \max(z_2, z_3)$$

Figure 5.1 illustrates the level sets of these two objectives. In the bottom left panel of Figure 5.1, we visualize the optimization trajectories of PGD for different choices of surrogates. The learning rate is held fixed to $\eta = 2\epsilon$ and PGD is executed for $T = 50$ iterations. The blue dashed circle denotes the boundary of the feasible region, whereas the circle, triangle and cross-shaped points show the intermediate points of PGD $(\mathbf{x}^{(1)}, ..., \mathbf{x}^{(50)})$. Using the CE as surrogate (red circle points) manages to successfully perturb the input $\mathbf{x}$, but CW objective (yellow triangle points) fails because the linear level sets produce gradients that gets the optimization jammed on a single point. The bottom right panel, however, demonstrates that the loss alternation method (green cross points) isn't affected from the failure mode of CW and finds an adversary. Despite being restricted, this synthetic toy example underpins the argument that using multiple surrogates in the same run of PGD renders the overall procedure more "robust" in the objective selection: Even if some individual choice is infertile for whatever reason, the other alternatives may be enough to find an adversary.

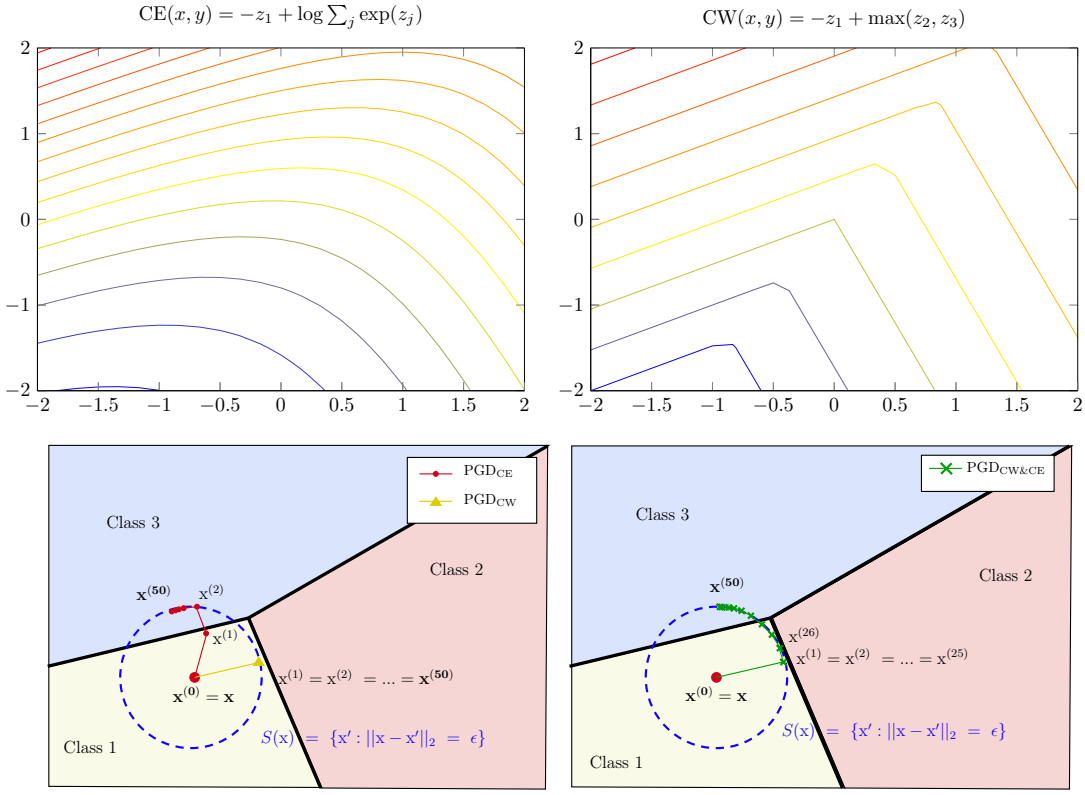


Figure 5.1. Top row*: The level sets of CE and CW losses (w.r.t class $y = 1$).* Bottom row*: (Left) Intermediate PGD points, executed with a single surrogate, where red circles indicate PGD with CE and the yellow triangle PGD with CW, (Right) Intermediate PGD points, but here the objective changes in the middle point ($T = T/2$) of the procedure (green crosses). The blue dashed circle visualizes the boundary surface, which in this case is a disk of radius $\epsilon = 0.4$ centered at $\mathbf{x}$, of the feasible PGD solutions.*

5.6.2 Models and Dataset

We will conduct our experiments in a sizable collection of 15 ℓ_∞-bounded robust models, trained on the CIFAR-10 dataset for a maximum perturbation of $\epsilon = 8/255$. The models are pre-trained and readily obtained from the ModelZoo of RobustBench [Cro+21] library. Our collection's robust models originate from various recent works: [Eng+19c; Car+19; Hen+19; Zha+19b; Zha+19c; Wu+20; Seh+22; AF20; Dai+22; Gow+21; Hua+21; Zha+21; RM21; Add+21; Seh+20]. The architectures of these models are ResNets [He+16] and Wide ResNets (WRN) [ZK16]. In Table 5.1, we exhibit our model collection: For each case (row), the classifier is matched with the respective paper/work, architecture, ModelID from RobustBench ModelZoo and the accuracy that the classifier attains on the clean CIFAR-10 test set. We also state that in the following discussion, we'll refer to the terms Attack Success Rate (ASR) and Robust Accuracy (equal to $1-$ASR) interchangeably to quantify the strength of each attack.

5.6.3 Experimental Analysis

Multi-Stage PGD versus Single-Loss

First, we compare the loss alternation strategy against the typical single loss variants of PGD. In this preliminary experimental setting, the step size is held fixed to $\eta^{(t)} = \epsilon/4$ and the optimizer

#	Paper	Model ID in RobustBench leaderboard	Architecture	Standard Acc. (%)
1	[Eng+19c]	Engstrom2019Robustness	ResNet-50	87.03
2	[Car+19]	Carmon2019Unlabeled	WideResNet-28-10	89.69
3	[Hen+19]	Hendrycks2019Using	WideResNet-28-10	87.11
4	[Zha+19b]	Zhang2019You	WideResNet-34-10	87.20
5	[Zha+19c]	Zhang2019Theoretically	WideResNet-34-10	84.92
6	[Wu+20]	Wu2020Adversarial	WideResNet-34-10	85.36
7	[Seh+22]	Sehwag2021Proxy_R18	ResNet-18	84.59
8	[AF20]	Andriushchenko2020Understanding	PreActResNet-18	79.84
9	[Dai+22]	Dai2021Parameterizing	WideResNet-28-10	87.02
10	[Gow+21]	Gowal2021Improving_28_10_ddpm_100m	WideResNet-28-10	87.50
11	[Hua+21]	Huang2021Exploring_ema	WideResNet-34-R	91.23
12	[Zha+21]	Zhang2020Geometry	WideResNet-28-10	89.36
13	[RM21]	Rade2021Helper_R18_extra	PreActResNet-18	89.02
14	[Add+21]	Addepalli2021Towards_RN18	ResNet-18	80.24
15	[Seh+20]	Sehwag2020Hydra	WideResNet-28-10	88.98

Table 5.1. *Our model collection, consisting of 15 ℓ_∞-bounded classifiers obtained from the ModelZoo of RobustBench.*

	K=1			*K=2*			*K=3*
Model	PGD_{CE}	PGD_{CW}	PGD_{DLR}	$\text{PGD}_{\text{CE\&CW}}$	$\text{PGD}_{\text{CE\&DLR}}$	$\text{PGD}_{\text{CW\&DLR}}$	$\text{PGD}_{\text{CE\&CW\&DLR}}$
[Eng+19c]	52.24	52.59	53.55	50.29	**50.22**	52.63	50.27
[Car+19]	62.09	60.86	61.16	60.00	60.00	60.88	**59.97**
[Hen+19]	57.38	56.61	57.47	55.41	55.37	56.55	**55.35**
[Zha+19b]	46.28	47.44	47.97	45.33	**45.32**	47.42	**45.32**
[Zha+19c] †	55.47	54.21	54.39	53.45	53.43	54.23	**53.41**
[Wu+20]	59.05	56.93	57.02	56.47	56.44	56.94	**56.42**
[Seh+22]	58.68	57.22	57.89	56.06	**56.05**	57.21	56.06
[AF20]	47.14	46.62	47.62	44.56	44.53	46.62	**44.50**
[Dai+22]	63.98	63.23	63.83	61.80	**61.76**	63.23	61.77
[Gow+21]	65.79	65.20	65.76	63.86	63.85	65.20	**63.84**
[Hua+21]	64.95	64.15	64.64	63.09	**63.03**	64.12	63.06
[Zha+21]	66.67	60.40	60.59	59.78	**59.69**	60.37	**59.69**
[RM21]	61.48	58.51	58.56	57.77	**57.74**	58.51	**57.74**
[Add+21]	56.00	51.88	51.97	51.45	51.43	51.86	**51.41**
[Seh+20]	59.86	58.41	58.57	57.66	**57.61**	58.40	**57.61**

Table 5.2. *Comparing single-loss PGD with the multi-stage variant of PGD (with $K = 2, 3$). PGD starts from the clean point (no added noise). The experiments are executed for $T = 100$ with no restarts. Each entry reports the robust accuracy of each classifier for the given method. (†): Attacked with $\epsilon = 0.031$.*

is set to standard gradient with the sign operation. Our computational budget is $T = 100$ with no restarts. Since no restarts are used, we choose to initiate PGD from the clean points (no initial perturbation) in order to eliminate any source of randomness in the results. Table 5.2 presents the robust accuracy obtained of PGD with different choices of surrogates, for every classifier in our collection.

Overall, there are several noteworthy remarks: First, the single-loss columns ($K = 1$) demonstrate that the surrogate loss can greatly affect the ASR of PGD, confirming the findings of previous studies, as that of Croce and Hein [CH20b]. On average, margin loss is the most reliable option but there are cases where it performs worse than CE. There are instances where CE lags behind the other two options by a large margin, e.g., as in the model from [Add+21] (`Addepalli2021Towards_RN18`), where the gap is greater than 4%. This indicates that it is impossible to select *a priori* the best possible objective for a given model. This observation consitutes strong evidence that no surrogate loss is reliable enough on its own.

Next, the results highlight the advantage of using multiple losses: When combining CE with CW

Model	APGD_{CE}	APGD_{DLR}	FAB	$\text{PGD}_{\text{CE\&CW\&DLR}}$	Δ
[Eng+19c]	51.72	52.67	50.67	**50.27**	-0.40
[Car+19]	61.74	60.67	60.88	**59.97**	-0.70
[Hen+19]	57.23	57.03	55.55	**55.35**	-0.20
[Zha+19b]	46.15	47.39	45.83	**45.32**	-0.51
[Zha+19c] †	55.28	53.52	53.92	**53.41**	-0.11
[Wu+20]	58.90	56.68	56.82	**56.42**	-0.26
[Seh+22]	58.38	57.37	56.27	**56.06**	-0.21
[AF20]	46.93	47.08	44.72	**44.50**	-0.22
[Dai+22]	63.93	63.44	62.27	**61.77**	-0.50
[Gow+21]	65.63	65.14	64.14	**63.84**	-0.30
[Hua+21]	64.55	64.14	64.45	**63.06**	-1.08
[Zha+21]	66.37	60.19	59.97	**59.69**	-0.28
[RM21]	61.40	58.41	58.42	**57.74**	-0.67
[Add+21]	55.80	51.56	51.93	**51.41**	-0.15
[Seh+20]	59.60	58.29	58.29	**57.61**	-0.68

Table 5.3. *Comparing $PGD_{CE\&CW\&DLR}$ with the untargeted version of every single white-box component from the AutoAttack ensemble. Each entry reports the robust accuracy of each classifier for the given method. Δ column report the robust accuracy gap between $PGD_{CE\&CW\&DLR}$ and the best among the AutoAttack components. The experiments are executed for $T = 100$ with no restarts. (†): Attacked with $\epsilon = 0.031$.*

or DLR ($\text{PGD}_{\text{CE\&CW}}$ and $\text{PGD}_{\text{CE\&DLR}}$ columns), or both ($\text{PGD}_{\text{CE\&CW\&DLR}}$ column) the attack is always stronger (lower rob. acc.) than the respective single-loss PGD. On average, $\text{PGD}_{\text{CE\&CW}}$ and $\text{PGD}_{\text{CE\&DLR}}$ decrease robust accuracy by 1.05% and 1.39% (absolute) respectively over their corresponding single-loss variants. In the case of $\text{PGD}_{\text{CW\&DLR}}$, the obtained ASR is nearly identical with PGD_{CW}, implying that the alternation step in this case may be futile because of the similarity between the expressions of CW and DLR losses.

Finally, it is illustrated that on average $\text{PGD}_{\text{CE\&CW\&DLR}}$ is better than $\text{PGD}_{\text{CE\&CW}}$ and $\text{PGD}_{\text{CE\&DLR}}$, yet the differences are small. In some cases, using the alternation scheme with two stages is better than $\text{PGD}_{\text{CE\&CW\&DLR}}$. This informs us that it is not always better to add another stage/objective in the alternation process. In a fixed iteration budget, adding another loss reduces the overall time allotted to each stage. We assume that this hurts performance because the reduced number of iterations is not enough to reach the stagnating region of each loss.

Multi-Stage PGD versus AutoAttack Components

Next, we compare our best method (on average, that is $\text{PGD}_{\text{CE\&CW\&DLR}}$) with every white-box component from AutoAttack [CH20b], i.e., APGD_{CE}, APGD_{DLR} and FAB attack [CH20a]. In the original AutoAttack evaluation, the last two components are run for $T = 100$ iterations and $R = 9$ restarts, using the targeted version of each attack. However, we adapt these attacks to our computational budget, evaluating the performance of their untargeted versions for $T = 100$. In our experiments, we execute the official code[1] of AutoAttack for every single model. We clarify that the official code does not provide a way to turn off random initialization when evaluating the AA components, but the fluctuations are expected to be small enough.

As it is clearly illustrated in Table 5.3, our proposed method, $\text{PGD}_{\text{CE\&CW\&DLR}}$ consistently outperforms the white-box components of AutoAttack. It becomes evident that the advantage of using the loss switching strategy is significant, since in this setting we run our attack for fixed step size equal to $\epsilon/4$ and the simplest optimizer possible (sign operation with no momentum).

[1] https://github.com/fra31/auto-attack

APGD_{CE} and APGD_{DLR} are both based in the evidently better APGD optimizer and step size is decayed according to some schedule, yet they lag behind $\text{PGD}_{\text{CE\&CW\&DLR}}$ by a large margin. Particularly, $\text{PGD}_{\text{CE\&CW\&DLR}}$ achieves (on average) 0.418% lower robust accuracy than the strongest component.

Multi-Stage PGD versus the strongest baselines

We extend the assessment of our method's effectiveness by comparing our results with the strongest ℓ_∞-bounded attacks, for $T = 100$ and no restarts. We consider the two best baselines found in literature (in our computational budget): GAMA-PGD [Sri+20] and MD attack [Ma+20]. Both of these methods suggest improving PGD through modifications on the surrogate loss and step size schedule. In Subsection 5.7.1, we delve into the exact similarities between the examined methods and our work.

We execute these attacks through the official codebases[2] [3]. When comparing $\text{PGD}_{\text{CE\&CW\&DLR}}$ with each baseline, we adapt the learning rate schedule according to each work (See Appendix for details). The results of these comparisons are summarized in Table 5.4. In the parentheses of $\text{PGD}_{\text{CE\&CW\&DLR}}$ columns, we display which learning rate schedule is used. These results indicate the effectiveness of our attack, achieving state-of-the-art performance (in the $T = 100, R = 1$ budget), for the majority of evaluated models.

Specifically, $\text{PGD}_{\text{CE\&CW\&DLR}}$ outperforms GAMA-PGD [Sri+20] in 11 out of 15 ℓ_∞-bounded robust models, whereas in 2 models they achieve the exact same ASR. In the 2 networks that $\text{PGD}_{\text{CE\&CW\&DLR}}$ returns higher robust accuracy, the differences are quite small, i.e., 0.02% and 0.04%. An extreme case is the model of [Zha+21], since GAMA-PGD lags behind our method for 1.10%. These observations indicate that, in general, $\text{PGD}_{\text{CE\&CW\&DLR}}$ suffers less from robustness overestimation.

In the case of MD attack [Ma+20], our method achieves lower robust accuracy in 13 out of 15 tested models, with an average improvement of 0.15%. In two models [Hen+19; Seh+20], however, the estimated robust accuracy is 0.05% and 0.14% higher than that of MD attack. Overall, this comparison, similarly to the previous one, highlights that $\text{PGD}_{\text{CE\&CW\&DLR}}$ provides the most reliable ℓ_∞-bounded robustness evaluations.

5.6.4 Qualitative Analysis

Here, we conduct a qualitative analysis to better grasp the impact of changing surrogate losses during optimization. Our experiments are inspired by the work of Yamamura et al. [Yam+22], where they visualize the ℓ_2-distance between successive PGD steps: $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|_2$ in order to empirically show that their proposed optimizer explores the input space more extensively. In a similar vein, we replicate their method for $\text{PGD}_{\text{CE}}, \text{PGD}_{\text{CW}}, \text{PGD}_{\text{CE\&CW}}, \text{PGD}_{\text{CE\&CW\&DLR}}$ in Figure 5.2, inspecting four different classifiers. To generate smoother curves, the y-axis quantity is averaged on a batch of 100 examples.

Altogether, it appears that in the single loss variants, the search of PGD becomes quite localized and after some time the successive steps are within small distances. In the cases where multiple surrogates are used, the curve presents a sudden rise in the alternation timestep, indicating that the objective alternation helps the algorithm to diversify its search.

[2] https://github.com/val-iisc/GAMA-GAT
[3] https://github.com/Jack-lx-jiang/MD_attacks

Model	GAMA-PGD [Sri+20]	$\text{PGD}_{\text{CE\&CW\&DLR}}$ (GAMA-PGD sch.)	Δ	MD Attack [Ma+20]	$\text{PGD}_{\text{CE\&CW\&DLR}}$ (MD sched.)	Δ
[Eng+19c]	50.05	**49.88**	-0.17	50.34	**49.87**	-0.47
[Car+19]	59.84	**59.78**	-0.06	59.83	**59.72**	-0.11
[Hen+19]	**55.22**	55.26	+0.04	**55.15**	55.20	+0.05
[Zha+19b]	45.32	**45.20**	-0.12	45.49	**45.17**	-0.32
[Zha+19c]†	53.29	53.29	0	53.36	**53.26**	-0.10
[Wu+20]	56.30	56.30	0	56.28	**56.26**	-0.02
[Seh+22]	56.01	**55.95**	-0.06	55.92	**55.89**	-0.03
[AF20]	44.42	**44.41**	-0.01	44.57	**44.44**	-0.13
[Dai+22]	61.94	**61.74**	-0.20	61.99	**61.72**	-0.27
[Gow+21]	63.78	**63.72**	-0.06	63.94	**63.73**	-0.21
[Hua+21]	**62.87**	62.89	+0.02	62.93	**62.86**	-0.07
[Zha+21]	60.72	**59.62**	-1.10	59.73	**59.58**	-0.15
[RM21]	57.78	**57.73**	-0.05	57.74	**57.72**	-0.02
[Add+21]	51.43	**51.26**	-0.17	51.30	**51.25**	-0.05
[Seh+20]	57.49	**57.43**	-0.06	**57.31**	57.45	+0.14

Table 5.4. *Comparing $PGD_{CE\&CW\&DLR}$ with the strongest attacks of our computational budget ($T = 100$ with no restarts). Each entry reports the robust accuracy of each classifier for the given method. Δ columns report the robust accuracy gap between the compared methods. (†): Attacked with $\epsilon = 0.031$.*

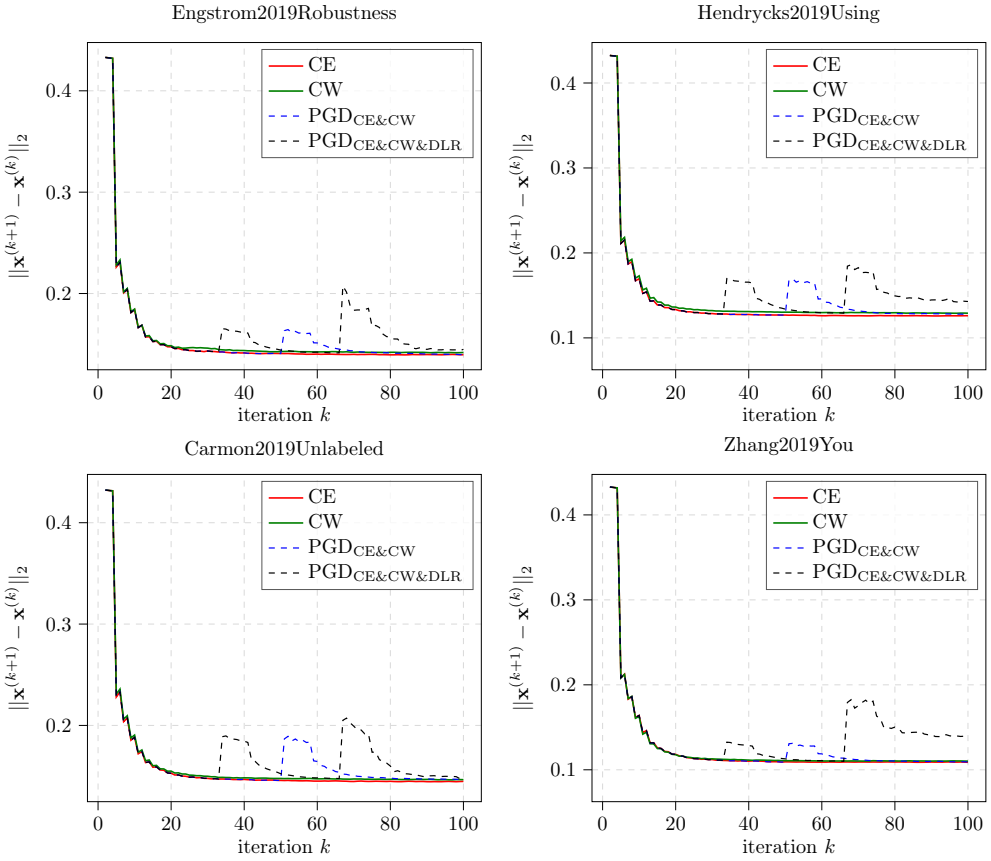


Figure 5.2. *Plotting the ℓ_2-norm between successive PGD steps, for various surrogate losses. Each panel represents this quantity over iterations, for a different classifier (ModelID is on top of each panel).*

5.6.5 Ablation: Surrogate Loss Order in Multi-Stage PGD

A research question regarding the multi-stage variant of PGD is whether the objective ordering affects the results. Specifically, we are interested in understanding whether any change occurs if we

optimize the objectives with reverse ordering. To address this question, we execute the two-stage PGD, with $T = 100$ and no restarts, for every possible pair (order matters) of CE, CW and DLR.

The results of Table 5.5 demonstrate that the order plays an essential role. Particularly, it is clearly illustrated that it is better to start the optimization procedure with the CE loss, then finishing off with CW or DLR. However, we observe that regardless of the objective ordering, every multi-stage PGD variant which alternates between CE and one of CW, DLR ($\text{PGD}_{\text{CE\&CW}}$, $\text{PGD}_{\text{CW\&CE}}$, $\text{PGD}_{\text{CE\&DLR}}$, $\text{PGD}_{\text{DLR\&CE}}$ columns) performs better than single-loss PGD.

Model	$\text{PGD}_{\text{CE\&CW}}$	$\text{PGD}_{\text{CW\&CE}}$	$\text{PGD}_{\text{CE\&DLR}}$	$\text{PGD}_{\text{DLR\&CE}}$	$\text{PGD}_{\text{CW\&DLR}}$	$\text{PGD}_{\text{DLR\&CW}}$
[Eng+19c]	**50.29**	50.95	**50.22**	51.13	**52.63**	52.97
[Car+19]	**60.00**	60.27	**60.00**	60.39	**60.88**	60.94
[Hen+19]	**55.41**	55.62	**55.37**	55.72	**56.55**	56.84
[Zha+19b]	**45.33**	45.85	**45.32**	45.86	**47.42**	47.58
[Zha+19c]	**53.45**	53.76	**53.43**	53.86	**54.23**	54.31
[Wu+20]	**56.47**	56.68	**56.44**	56.72	**56.94**	56.98
[Seh+22]	**56.06**	56.42	**56.05**	56.52	**57.21**	57.49
[AF20]	**44.56**	44.94	**44.53**	45.03	**46.62**	46.77
[Dai+22]	**61.80**	62.18	**61.76**	62.33	**63.23**	63.42
[Gow+21]	**63.86**	64.80	**63.85**	64.30	**65.20**	65.31
[Hua+21]	**63.09**	63.52	**63.03**	63.64	**64.12**	64.34
[Zha+21]	**59.78**	60.16	**59.69**	60.31	**60.37**	60.51
[RM21]	**57.77**	58.17	**57.74**	58.18	**58.51**	58.54
[Add+21]	**51.45**	51.79	**51.43**	51.84	**51.86**	51.93
[Seh+20]	**57.66**	57.92	**57.61**	57.93	**58.40**	58.47

Table 5.5. *Ablation Study. Exploring the importance of the surrogates' order. The experiments are executed for $T = 100$ with no restarts. Each entry reports the robust accuracy of each classifier for the given method. (†): Attacked with $\epsilon = 0.031$.*

5.6.6 Ablation: Additional Techniques of Combining Surrogates

Another interesting research question is to explore whether there exist other ways of combining surrogates. To settle this, we compare the alternation method with two additional combining techniques. First, one can combine different surrogates through a convex combination, i.e., setting the surrogate according to the following expression:

$$\mathcal{L}(\mathbf{x}, y) = \gamma \cdot \mathcal{L}_1(\mathbf{x}, y) + (1 - \gamma) \cdot \mathcal{L}_2(\mathbf{x}, y)$$

Another way is to combine different surrogates in an ensemble-like manner, i.e., split the entire iteration budget into K equally sized intervals, execute PGD using the k-th surrogate $\mathcal{L}_k$, starting from the clean point (not from where the previous stage ended), and then aggregate the output decisions. This method is inspired by the MultiTargeted surrogate, introduced by Gowal et al. [Gow+19]. For the CE and CW losses, we denote the latter combining strategy as $\text{PGD}_{\text{CE}} \vee \text{PGD}_{\text{CW}}$, because the output decisions of each surrogate are aggregated through binary OR, i.e., the input is deemed misclassified if at least one of PGD_{CE}, PGD_{CW} generate a successful perturbation.

We conduct an ablation study, using the CE and CW objectives, to explore the effectiveness of these methods. The results are illustrated in Table 5.6, where we also report the robust accuracy of $\text{PGD}_{\text{CE}}, \text{PGD}_{\text{CW}}, \text{PGD}_{\text{CE\&CW}}$ for direct comparison (we also include the iteration budget on the superscript to draw a distinction with the ensemble method). As expected, the robust accuracy of convex combination is susceptible to the choice of γ, with its performance depending on whether the best-performing objective has a larger weight. The ensemble method, on the other hand, consistently outperforms the single-loss PGD, and much like $\text{PGD}_{\text{CE\&CW}}$, is more "robust"

Model	$\text{PGD}^{100}_{\text{CE}}$	$\text{PGD}^{100}_{\text{CW}}$	$\text{PGD}^{50}_{\text{CE}} \vee \text{PGD}^{50}_{\text{CW}}$	Convex $\gamma = 0.25$	Convex $\gamma = 0.75$	$\text{PGD}_{\text{CE\&CW}}$
[Eng+19c]	52.24	52.59	50.75	51.59	52.30	**50.29**
[Car+19]	62.09	60.86	60.18	60.97	60.85	**60.00**
[Hen+19]	57.38	56.61	55.52	56.10	56.36	**55.41**
[Zha+19b]	46.28	47.44	45.56	46.37	47.19	**45.33**
[Zha+19c] †	55.47	54.21	53.68	54.34	54.18	**53.45**
[Wu+20]	59.05	56.93	56.66	57.46	57.00	**56.47**
[Seh+22]	58.68	57.22	56.37	57.22	57.10	**56.06**
[AF20]	47.14	46.62	44.81	45.78	46.17	**44.56**
[Dai+22]	63.98	63.23	62.17	62.89	63.15	**61.80**
[Gow+21]	65.79	65.20	64.20	64.72	65.00	**63.86**
[Hua+21]	64.95	64.15	63.35	64.01	64.09	**63.09**
[Zha+21]	66.67	60.40	60.14	63.88	60.86	**59.78**
[RM21]	61.48	58.51	58.14	59.14	58.49	**57.77**
[Add+21]	56.00	51.88	51.78	53.23	51.96	**51.45**
[Seh+20]	59.86	58.41	57.85	58.59	58.41	**57.66**

Table 5.6. *Ablation Study. In the convex columns, γ $(1-\gamma)$ corresponds to CE (CW). The experiments are executed for $T = 100$ with no restarts. Each entry reports the robust accuracy of each classifier for the given method. (†): Attacked with $\epsilon = 0.031$.*

against issues arising from individual use of objectives. However, the loss alternation strategy, $\text{PGD}_{\text{CE\&CW}}$, performs better than the ensemble-like combination. We advocate that this occurs because $\text{PGD}_{\text{CE\&CW}}$ utilizes the progress made in previous stages to perform better initialization for the next stage. The ensemble-like method, however, discards the perturbation found by previous objectives, and starts optimization all over again.

5.7 Discussion

5.7.1 Similarity with Previous Works

Next, we discuss previous works that also employ a loss alternating strategy. First, the most similar work is that of Ma et al. [Ma+20], where they employ an identical alternation step to evade the issue of imbalanced gradients. The first PGD stage optimize only one of the two logit terms, whereas in the final stage, the typical margin loss is optimized. Notice a striking difference: Our work involves the CE,CW and DLR losses, all containing more than one logit terms, hence potentially suffering from gradient imbalance that should translate to reduced ASR. Our method outperforms MD attack. Therefore, our study provides evidence that the success of MD attack [Ma+20] is more likely the outcome of switching surrogates, rather than deterring the magnitudes of logit terms' gradients from becoming highly disparate.

The second method is GAMA-PGD, introduced by Sriramanan et al. [Sri+20]. The authors propose to regularize the margin loss with a MSE term, weighted by a decaying coefficient. In their implementation, the initial rate of weights between the MSE and CW losses is 50:1, hence for the first few iterations the contribution of CW loss is negligible. The weight of MSE is linearly decayed to 0 for $T/4$ iterations, and after that point the surrogate is set to the standard margin loss. Essentially, their attack alternates the surrogate loss used by PGD as many times as the duration of the interval during which MSE decays, i.e., $T/4$ out of T iterations. Their analysis conveys the intuition that the improvement originates solely from the regularizing effect that MSE exerts on the margin loss. Our work demonstrates that the benefits of GAMA-PGD may arise from the loss alternation, still further experimentation is required.

Another method loosely connected with ours is the Composite Adversarial Attack (CAA)

[Mao+21]. Mao et al. propose to generate adversaries by searching for the best composition of individual base attacks. Our method can be seen as a more special study of CAA, since it composes PGD attacks for two (or three) different objectives. Our work indicates much more markedly the value of using multiple losses. The effectiveness of CAA appears more like the result of a brute-force-like search.

Overall, our paper differs from the aforementioned works in that it manages to showcase the true efficacy of the alternation step, stripped down from other redundant components. The experiments provide direct evidence that using multiple objectives is sufficient to induce large performance gains. Additionally, our work is an extension of these methods since we evaluate the combination of all possible pairs of CE,CW and DLR losses, rather than using only CW with its individual terms [Ma+20] or CW and MSE [Sri+20].

5.7.2 Future Work

There are several questions arising from the proposed work than require further investigation and could be of value to the community. Notably, it is critical to address whether there is a trade-off between the number of surrogates used and PGD performance, for a fixed number of iterations. We assumed that adding more stages for fixed budget may hinder performance due to the decreased duration allotted to each stage. However, our intuition is that adding more objectives shouldn't drop the Attack Success Rate (ASR), given that PGD spends a sufficient time in each stage. This can be easily verified by increasing the computational budget, e.g., from $T = 100$ to $T = 1000$, and observing that more surrogates leads to higher ASR.

Another interesting observation to explore is how the alternation step depends on the choice of objectives and their respective formulations. Particularly, we observed that $\text{PGD}_{\text{CW\&DLR}}$ performs at a par (or even worse) than the respective single-loss variants, PGD_{CW} and PGD_{DLR}, which was credited to the similarity of CW and DLR. This indicates that the loss alternation technique is an improvement only if the expressions generate landscapes which are diverse enough. In this vein, it would be valuable to encompass other expressions which deviate from the objective functions of our study, i.e., CE, CW and DLR.

Since we experimentally demonstrate that our PGD variant is the strongest adversarial attack in the computational budget of 100 iterations, another direct extension is to integrate our attack into powerful ensembles. Specifically, in the case of AutoAttack [CH20b], $\text{PGD}_{\text{CE\&CW\&DLR}}$ is outperforming every white-box component (Table 5.3), hence we assume that replacing e.g. APGD_{DLR} with $\text{PGD}_{\text{CE\&CW\&DLR}}$ would produce more reliable robustness evaluations.

5.8 Conclusion

In this work, we propose a method of alternating objectives for improving the strength of PGD-based attacks. The proposed method performs better than single loss variants, as well as strong baselines which are used for evaluating the ℓ_p-bounded robustness of neural networks: AutoPGD [CH20b], FAB [CH20a], GAMA-PGD [Sri+20] and MD Attack [Ma+20]. Our experiments show that alternating objectives is a very effective way of combining different objectives compared, e.g., to convex combination and ensemble-like methods. It is also experimentally shown that the proposed method offers significant robustness towards overcoming loss-specific weaknesses. Furthermore, our qualitative analysis offers intuition on the reasons behind our method's strength that may be related to the algorithm's search space diversification induced by the alternation

step. Finally, we offer a new perspective on how the success of other state-of-the-art attacks, i.e., GAMA-PGD and MD Attack, can be ascribed to loss alternation.

Appendix A

Implementation Details

For our experiments, we implement code on the PyTorch framework. The PGD implementation is based on the TRADES [Zha+19c] repository[4]. All attacks are executed with a ℓ_∞-norm bound of $\epsilon = 8/255$ and for $T = 100$ iterations, with no restarts. Our code returns the best intermediate PGD point instead of the last. The robust models of our study are obtained from the ModelZoo of RobustBench [Cro+21]. Our experiments are run in a NVIDIA GeForce GTX 1080 Ti GPU with 12GB VRAM.

Step Size Schedules

Here, we discuss the step size schedules used when comparing our method with the GAMA-PGD [Sri+20] and MD Attack [Ma+20] baselines. In GAMA-PGD, the step size schedule incurs tenfold drops at $T = 60$ and $T = 85$, starting from $\eta^{(0)} = 2\epsilon$.

In [Ma+20], step size is regulated according to a cosine-annealing scheme. In particular, the step size in $t - th$ iteration equals:

$$\eta^{(t)} = \begin{cases} \epsilon \cdot (1 + \cos(\frac{t-1}{T'})\pi) & , t < T' \\ \epsilon \cdot (1 + \cos(\frac{t-T'}{T-T'}\pi)) & , T' \leq t < T \end{cases}$$

where $T = 100$, $T' = T/2$. Therefore, step size is decayed from 2ϵ to 0 in each stage. We extend this scheme to our three-stage variant as follows:

$$\eta^{(t)} = \begin{cases} \epsilon \cdot (1 + \cos(\frac{t-1}{T/3})\pi) & , t < T/3 \\ \epsilon \cdot (1 + \cos(\frac{t-T/3}{T/3}\pi)) & , T/3 \leq t < 2T/3 \\ \epsilon \cdot (1 + \cos(\frac{t-2T/3}{T/3}\pi)) & , 2T/3 \leq t < T \end{cases}$$

Images of Adversarial Examples

In this appendix, we visualize adversaries, generated to fool the ResNet50 robust classifier from [Eng+19c] (Engstrom2019Robustness) that our obtained through executing $\text{PGD}_{\text{CE\&CW}}$ and $\text{PGD}_{\text{CE\&CW\&DLR}}$ methods, alongside their clean counterparts. Obviously, there are no visible differences between the clean and perturbed images.

[4] https://github.com/yaodongyu/TRADES

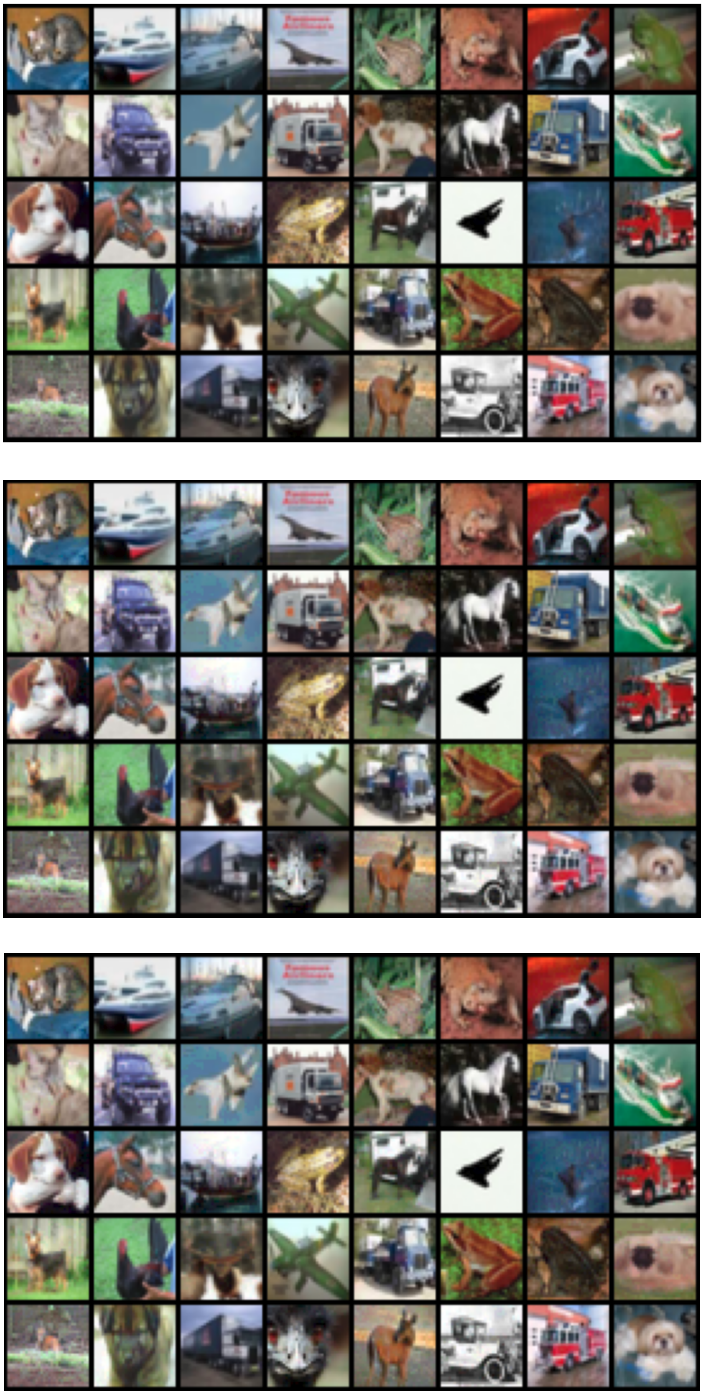


Figure 5.3. *From top to bottom: First 5 rows correspond to clean CIFAR-10 test images, next 5 rows correspond to* $PGD_{CE\&CW}$ *adversaries and last 5 correspond to adversaries generated by* $PGD_{CE\&CW\&DLR}$.

Chapter 6

Conclusions

6.1 Discussion

In this thesis, we conducted a pedantic study on the influence that the surrogate loss exerts in performance of Projected Gradient Descent. Previous studies, as this of Croce and Hein [CH20b], alluded that the performance of each surrogate is dependant on multiple factors, namely the architecture, model's weights, training dataset etc. This observation is the crux of our experimental work, where we postulate that by combining different objectives in the same run of PGD we will manage to alleviate individual flaws of each surrogate, rendering the optimization procedure more robust to the selection of surrogate loss. We underpin this assertion initially by considering a synthetic toy example, where one of the two losses (in this case, the margin loss) is blatantly weak due to the morphology of its level sets. In this example, we show how our combining method successfully perturbs the clean input, despite having used the problematic loss function for the first half of iterations. Next, to extend the credibility of our assertion in more realistic scenarios, we compare our alternating method with the single loss variants of PGD, in a large array of $\ell_\infty-$robust models, where it is vividly demonstrated that our proposed alternating strategy (when involving CE and at least one of CW and DLR) consistently outperforms PGD when using a single objective. These findings consist a satisfactory sample of empirical evidence to justify our initial intuition. In addition to that, we also conduct a qualitative experiment to better grasp the reasons behind our method's superior performance: In this setting, we visualize the ℓ_2-distance between successive PGD steps, both for the single-loss variants of PGD and for the alternation method. From the obtained figures we derive the following conclusion: Alternating between different losses during PGD is a tool to promote search diversity, expanding the reachable set of points/regions (this is implied by the abrupt leaps of ℓ_2-distance in the timesteps where the loss alternation occurs). Finally, we compare our best method against multiple baselines: First, with every white-box component from AutoAttack, i.e. APGD_{CE},APGD_{DLR} and FAB, adapted to our computational budget ($T = 100, R = 1$), where it is highlighted that the multi-stage PGD which alternates objectives manages to reach lower robust accuracies for every single model. Next, our attack is compared with the strongest adversarial attacks (for the budget $T = 100, R = 1$) found in the literature: GAMA-PGD [Sri+20] and Margin Decomposition (MD) Attack [Ma+20]. Our proposed attack attains to outperform both of these methods (for the majority of the examined networks), achieving state-of-the-art results.

6.2 Future Work

There are plenty of future research steps emerging from the findings of our work. In this paragraph, we will attempt to present various ideas that someone can implement in order to expand the scope of our study:

- An immediate extension is to apply these experiments in a larger scale, both in the number of examined models and in the datasets used. Notably, Croce and Hein [Cro+21] use more than 50 models, whereas Yamamura et al. [Yam+22] experiment on 64 models. Both of these studies evaluate their proposed methods on three datasets: CIFAR-10, CIFAR-100 and ImageNet. In this thesis, we chose not to conduct our study in such a large scale mainly due to compute limitations. Extending our work can also be done through the study of different threat models: We only explored the combination of different losses for the ℓ_∞-bounded threat model, but that can be done also for ℓ_0,ℓ_1 or ℓ_2-bounded attacks.

- Also, our method can be exploited in ensemble-like attacks, like AutoAttack [CH20b] or A^3 [Liu+22]. Specifically, since our method was shown to outperform all of the white-box components of AA, it would be safe to assume that replacing them with our attack (at least the first, i.e. APGD_{CE},APGD_{DLR} which are PGD-based) would lead to higher Attack Success Rate (ASR), enabling more reliable robustness evaluations.

- Notice that our work involved the combination of three losses: CE, CW and DLR. A straightforward way to broaden the caliber of our work is to encompass numerous other expressions, e.g. the Mean Square Error (MSE) or some objective from the ones proposed in the study of Carlini and Wagner [CW17]. Dropping the budget restriction, it is interesting to explore how the combination of many losses affects the ASR e.g. cascading 10 different objectives, devoting 100 iterations to each one of them. This may push the limits of PGD even more in terms of Attack Success Rate (ASR). As regarding additional objectives, another interesting line of research is to explore the derivation of new formulations, which induce largely different geometries than the landscapes of the typical losses used in our study. In that regard, it is valuable to conduct additional qualitative experiments to understand when the combination of two or more different objectives is advantageous or not, E.g. an immediate question arising from our results is why the combination of the CW and DLR losses didn't provide gains over their single-loss variants, which we attributed to the highly similar expressions of the objectives but without verifying it in a more convincing way.

- Next, another way of improving PGD is to combine our loss alternation strategy with advances made on other hyperparameters/components of the algorithm, e.g. combine the surrogate loss alternation with more elaborate optimizers (APGD [CH20b] or ACG [Yam+22]) and initialization techniques (e.g. ODI [Tas+20] or adaptive initialization from [Liu+22]).

- Finally, the ultimate question emerging from the work of this dissertation is whether the conclusion about the advantage of switching objectives during PGD can be also generalized in the adversarial attack generation process in other domains, for example in applications of textual or acoustic data.

Chapter 7

Appendix: The log-likelihood attack

In a completely different note, we devote the content of this Appendix to present a couple of other experiments that were conducted during the course of this thesis. This section is discussing our ideas in a slightly informal manner, with the reason for this being that, overall, the final results were inconclusive and hence we include them in case any acquainted reader find them interesting.

7.1 Introduction

Deep Neural Networks (DNNs), despite their exceptional performance on an abundance of previously hard-to-solve tasks, have functional blindspots. The most prominent instance of such problematic functionality is their sensitivity against *adversarial examples*. Adversarial examples arise by adding carefully designed perturbations into images which are correctly classified by the system, resulting to a new image which dramatically alters the output of the classifier. Interestingly, these perturbations are visually so subtle that even humans can't recognize them.

The research community primarily studies the problem of adversarial examples through the lens of ℓ_p-bounded threat model. In this setting, the adversary can only change the input in a way that the produced example has a small enough ℓ_p-distance from its clean counterpart. Despite that there is a significant progress towards improving robustness against ℓ_p-bounded attacks, the proposed defenses can provide some degree of robustness only against the adversarial threat model that was used during training [SC18]. The issue of ℓ_p-bounded adversarial defenses being unable to defend against other types of adversaries is omnipresent. Engstrom et al. [Eng+19a] fool adversarially trained models by applying small spatial transformations to images. Other works fool defended classifiers by recoloring the image [HP18; LF19] or applying spatial transformations like pixel displacements [Xia+18]. These attacks violate the constraint of ℓ_p-bounded norm, but they remain imperceptible to the human eye, implying that the ℓ_p-bounded threat model is extremely flawed and restricted. We will refer to such attacks as *Unrestricted Adversarial Attacks*.

Song et al. [Son+18a] provide empirical evidence that typical adversarial attacks generate examples which have low probability density. Based on this observation, we aspire to settle the question of whether it is possible to generate adversaries only by minimizing the data log-likelihood, which is an unsupervised objective and has no relation at all with the classification problem. The minimization can be done through known adversarial attacks like PGD. In our work, we confront this problem by exploiting Normalizing Flows (NFs) [TT13; Din+15; Din+17] as the density estimator. Since NFs map data points from the input space to a latent code of equal dimensionality, we explore the idea of log-likelihood minimization both in the pixel and the latent space. We assume that the latter method, i.e. producing an ℓ_p-bounded adversarial perturbation in the latent space, is meaningful since small tweaks on the latent code should retain visual similarity. In this case,

the produced adversaries are unrestricted since mapping the adversarial latent perturbation to the pixel space doesn't bound the image ℓ_p-norm.

In general, we draw two conclusions from our study: First, log-likelihood minimization in the pixel space leads to weak attacks, that are unable to harm ℓ_p-bounded adversarial defenses. In the case of latent code optimization, the attack is capable of degrading performance of ℓ_p-bounded defenses, which is expected because the perturbations are unrestricted in terms of ℓ_p-norm. Performance degradation is monotonically related with the perturbation bound in the latent space. Increasing this bound leads to adversaries which have visible differences with their clean counterparts.

Notation. Our experiments involve NF-based models, hence we denote a NF model as $g_\phi : \mathcal{X} \to \mathcal{Z}$, mapping data points from $\mathcal{X}$ to a latent space $\mathcal{Z}$. Of course, in order to keep g_ϕ, the input (also called pixel) space $\mathcal{X}$ and the latent space $\mathcal{Z}$ have the same dimensionality. Whenever we refer to log-likelihood (in the NF-based models), we suppose it is calculated by the density function $p_\phi : \mathcal{X} \to [0, 1]$ learned by the model (unfamiliar readers are referred to our introduction in NFs, in Subsection 3.6.3). Also, we state that in this section, the bold lowercase $\mathbf{z}$ refers to latent codes, instead of the logit vector of some classifier (this quantity is not of interest in this section).

7.2 Motivation

Song et al. [Son+18b] conduct the following experiment: Using a CIFAR-10 pretrained PixelCNN [Oor+16] density estimator, they observe that adversarial attacks lie in regions of lower probability density than their clean counterparts. This is also depicted in Figure 7.1, where they visualize the histograms of bits per dimension (bpd) which is proportional to the negative log-likelihood. This empirical finding motivates them to derive a new way of defending against adversaries, called purification. The reasoning behind this term is that their method receives an input $\mathbf{x}$ at test time, possibly adversarial, and transforms it to a new input $\mathbf{x}'$, which is within small ℓ_p-distance and has higher log-likelihood. Hence, their method attempts to purify the image, hopefully subtracting the adversarial noise through this procedure.

However, there are some important caveats related to this work: First, the purifcation defense from [Son+18b] has been shown to be ineffective, since later works evaded it [Ath+18]. Since this defense has been fooled, this informs us that it is possible to find adversarial examples that lie in high density regions, as their clean counterparts. Indeed, Stutz et al. [Stu+19] demonstrate that such adversaries are easy to find.

The finding of Song et al. [Son+18b] about the probability densities of adversarial attacks motivates to explore the task of generating adversarial examples completely through the lens of log-likelihood. Particularly, we put forth the following question as the driving force of our study:

"*Can we generate adversarial examples simply through minimizing the probability density of the input?*"

More intuitively, we are interested in finding out whether a procedure which distracts the input from the learned data distribution is able to create a new input which is hard to classify. Notice that producing adversaries through this process doesn't require knowledge about the classifier, since the only requirement is the availability of some generative model, trained on the same data on which the classifier is trained. Hence, such an adversarial attack belongs to the category of black-box methods.

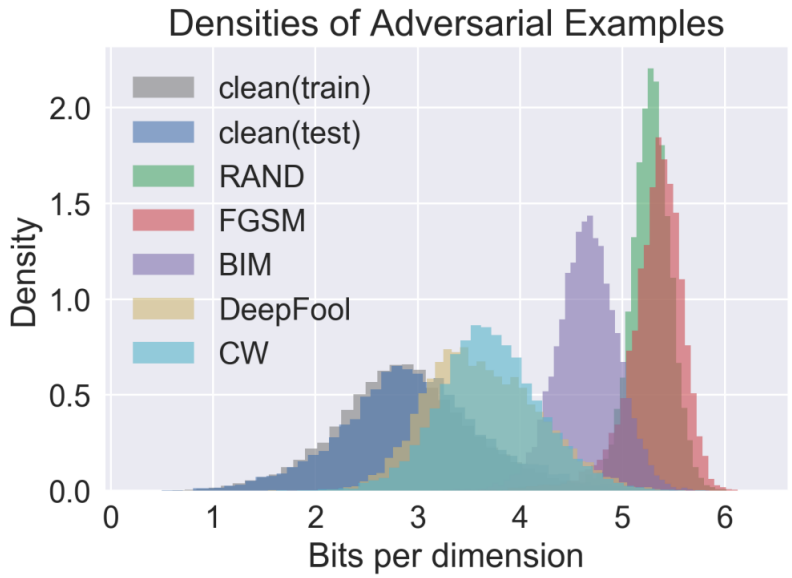


Figure 7.1. *Adversarial examples (on CIFAR-10) are concentrated in regions of higher bits-per-dimension (thus lower log-likelihood) than clean inputs. Figure adapted by Song et al. [Son+18b].*

7.3 Methodology

The most popular method of generating adversarial examples is Projected Gradient Descent (PGD) [Mad+18; Kur+17], where one updates the current adversary by following the normalized gradient direction and then projects the result into the ℓ_p-ball around the clean input $\mathbf{x}$. Assuming that we have access to an NF-based model, the input $\mathbf{x}$ is differentiably mapped through its associated log-likelihood $\log p(\mathbf{x})$. Hence, minimizing log-likelihood while remaining inside the ℓ_p-ball of radius ϵ around the clean input can be done through PGD.

However, NF-based models map the input $\mathbf{x}$ to a corresponding latent code $\mathbf{z}$. The latent code can also be differentiably mapped to the data log-likelihood, since the flow model learns an invertible transformation between the two spaces. The existence of such a latent space gives rise to the prospect of performing optimization on this exact space instead on the typical image space. Such a possibility has been examined in various works, either using the latent space learned by NFs [Yuk+21] or GANs [Son+18a; Zha+18].

A straightforward way to explore the latent space is to perform optimization on the latent code, yielding an adversarial latent code $\mathbf{z}'$ for a clean input $\mathbf{x}$. Finally, decoding the latent code $\mathbf{z}'$ through the learned inverse NF transformation will result in the pixel space adversary $\mathbf{x}'$. This procedure mentioned above is tailored to generative models that have the capabality of performing inference and decoding the latent code. Natural candidates to realize this prospect are VAEs (though they only perform approximate inference) and Normalizing Flows. Bounding the distance $||\mathbf{z} - \mathbf{z}'||_p$ on the latent space and decoding $\mathbf{z}'$ doesn't guarantee that this bound will still hold on the image space. Hence, such an approach of exploiting the latent space to generate adversaries should be classified in the *Unrestricted Adversarial Attacks* category. In this work, we will also explore the capability of latent space PGD using the log-likelihood as objective. Yuksel et al. [Yuk+21] perform exactly this, but only for optimizing the classification loss.

In Table 7.1, we represent abbreviations for the various methods of generating adversarial examples. The first row, i.e. pixels-space PGD using the classification loss, is the most typical

white-box attack, whereas the second row has been studied by Yuskel et al. [Yuk+21]. Our work will evaluate the effectivness of two last rows, minimizing the data log-likelihood learned by a NF-based model.

Abbr. of Method	Update rule	Comments
$\text{PGD}^{x,\text{CE},p}$	$\mathbf{x}'_{t+1} = \Pi^p_{\mathbf{x},\epsilon}\Big(\mathbf{x}'_t + \alpha \cdot \text{norm}\big(\nabla_{\mathbf{x}'_t}\mathcal{L}(\mathbf{x}'_t, y)\big)\Big)$	White-Box, Common PGD
$\text{PGD}^{z,\text{CE},p}$	$\mathbf{z}'_{t+1} = \Pi^p_{\mathbf{z},\epsilon}\Big(\mathbf{z}'_t + \alpha \cdot \text{norm}\big(\nabla_{\mathbf{z}'_t}\mathcal{L}(g_\phi^{-1}(\mathbf{z}'_t), y)\big)\Big)$	Similar to this in [Yuk+21]
$\text{PGD}^{\mathbf{x},\text{NLL},p}$	$\mathbf{x}'_{t+1} = \Pi^p_{x,\epsilon}\Big(\mathbf{x}'_t - \alpha \cdot \text{norm}\big(\nabla_{\mathbf{x}'_t}\log p(\mathbf{x}'_t)\big)\Big)$	Black-box
$\text{PGD}^{\mathbf{z},\text{NLL},p}$	$\mathbf{z}'_{t+1} = \Pi^p_{z,\epsilon}\Big(\mathbf{z}'_t - \alpha \cdot \text{norm}\big(\nabla_{\mathbf{z}'_t}\log p(g_\phi^{-1}(\mathbf{z}'_t))\big)\Big)$	Black Box

Table 7.1. *Inventory of methods. CE (Cross-entropy) is the classification loss and NLL (Negative Log-likelihood) is the density estimation function learned by Glow.*

7.4 Experimental Work

7.4.1 Preliminary Experiment

Based on the previous discussions, our work aims to generate adversarial examples simply through minimizing the log-likelihood of the data. As for the density estimator, we will utilize a Normalizing Flow model, and specifically Glow [KD18]. The Glow model lends us flexibility, since we can also produce adversaries by optimizing on the latent space, as discussed in the previous paragraph.

First of all, we conduct an experiment to confirm that Glow has the same behaviour as PixelCNN, when confronted with adversarial examples. In our code, we used the pre-trained Glow model of this GitHub repository. This re-implementation of Glow has enough credibility since it achieves 3.39 bits-per-dimension (bpd) on the CIFAR-10 test data, as compared with the 3.35 bpd reported to the original paper. For this initial experiment, we generate adversaries against the ResNet50 classifier which is standardly trained on CIFAR-10, available in the Robustness Library [Eng+19c] repository. Next, we generate adversarial examples on the test data, using PGD (both ℓ_∞ and ℓ_2 bounded) either for multiple steps or just for a single step (FGSM and FGM) [Goo+15]. Feeding these adversarial examples to the density estimator learned by Glow, we observe the same pattern as in the work of Song et al. [Son+18b]: Adversaries reside in regions of low-likelihood, as Figure 7.2 demonstrates. This experiment alludes that minimizing the density fucntion learned by Glow may be helpful towards generating adversarial examples.

7.4.2 Evaluating the strength of our proposed attacks

Pixel-Space optimization. First, we evaluate the pixel-space PGD variant which minimizes the data lol-likelihood. The adversaries are generated against the ℓ_p-adversarially trained ResNet50 from Robustness library [Eng+19c], which achieves 53.49% against $\ell_\infty-$bounded adversaries (for $\epsilon = 8/255$). Surprisingly, the Attack Success Rate (ASR) of this method is 0%, meaning that minimizing the log-likelihood is unable to find ℓ_p-bounded adversarial examples against this robust classifier.

Latent-Space optimization. Here, our aspiration is to degrade the performance of ℓ_p-robust classifiers through the black-box, unrestricted attack method $\text{PGD}^{\mathbf{z},\text{NLL},p}$. In general, this attack

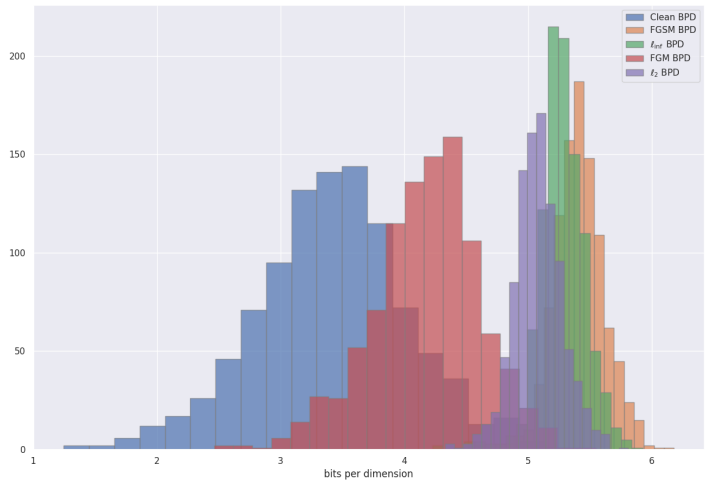


Figure 7.2. *The density estimator learned by Glow assigns low-likelihood (high bpd) to adversarial examples of ResNet50 classifier from the Robustness library.*

algorithm can be considered similar to the work of [Eng+19b], where they manage to harm the classification accuracy of robust models just by applying spatial transformation on images (hence requiring no knowledge of the models' weights). As for the constant ϵ which determines the radius of ℓ_p-ball which restricts the search of adversarial latent codes $\mathbf{z}'$, we are manually checking three different values : $\{0.1, 0.15, 0.2\}$. In these experiements, we use the exact same robust classifier as before, i.e. the ℓ_∞-bounded adversarially trained ResNet50 from [Eng+19c]. In Figure Figure 7.3, we demonstrate the visual outcome of generating adversaries through $\text{PGD}^{\mathbf{z},\text{NLL},p}$, for these three different ϵ values, as well as for different choices of ℓ_p-norms, i.e. $p = \infty, 2$.

Figure 7.3. *Adversaries produced by $PGD^{\mathbf{z},NLL,p}$ for $p = \infty$ (Left) and $p = 2$ (Right). From top to bottom, we demonstrate clean images ($\epsilon = 0$) and perturbed images for ever-increasing ϵ (0.10, 0.15 and 0.2).*

The visual results in the case of $p = \infty$ are dissatisfying: small changes in the ϵ bound introduce undesired artifacts in the image, largely deteriorating its visual quality. In the case of ℓ_2-norm, the visual fidelity of produced images is better, but overall the process seems extremely unstable. Next, we evaluate the degree of performance drop that these adversaries induce to the robust classifier under examination. Table 7.2 exhibits that even in the case of $\epsilon = 0.2$, where the generated attacks are too artificially-looking, this attack can't drop the classifier's performance (as in the spatial attacks [Eng+19b]).

Method	Size of Perturbation ϵ	Accuracy (%)
$\text{PGD}^{x,\text{CE}}$	0	87.03
	8/255	53.49
$\text{PGD}^{z,\text{NLL},\infty}$	0.10	80.47
	0.15	69.33
	0.20	56.08

Table 7.2. *(CIFAR-10) Test Set Accuracy*

7.5 Conclusion

We attempt to generate attacks only by minimizing the log-likelihood of the image, as this is estimated by Glow [KD18]. We considered to apply the PGD algorithm on the latent space, as small deviations on the latent codes could preserve visual similarity w.r.t. original image, while endowing greater capabilities on the adversary than in the case of bounded pixel-space perturbations (since now the adversarial search space is not confined in terms of pixel space proximity). The generated adversaries, even for small ϵ bounds, were not visually satisfying and besides that, they did not manage to greatly affect the performance of the robust classifier. In retrospective, our conclusion is that the assumption to generate adversaries without explicitly driving the process to minimize the classifier's confidence on the ground truth class was bold and practically, in the way we approached the problem, this attempt was fruitless.

Bibliography

[Add+21] Sravanti Addepalli et al. "Towards Achieving Adversarial Robustness Beyond Perceptual Limits". In: *ICML 2021 Workshop on Adversarial Machine Learning*. 2021.

[AF20] Maksym Andriushchenko and Nicolas Flammarion. "Understanding and improving fast adversarial training". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 16048–16059.

[Agg+20] Gunjan Aggarwal et al. "On the Benefits of Models with Perceptually-Aligned Gradients". In: *ArXiv* abs/2005.01499 (2020).

[Ala+19] Jean-Baptiste Alayrac et al. "Are Labels Required for Improving Adversarial Robustness?" In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. 2019, pp. 12192–12202.

[And+20] Maksym Andriushchenko et al. "Square attack: a query-efficient black-box adversarial attack via random search". In: *European Conference on Computer Vision*. Springer. 2020, pp. 484–501.

[Arj+17] Martín Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein GAN". In: *ArXiv* abs/1701.07875 (2017).

[Aro+97] Sanjeev Arora et al. "The Hardness of Approximate Optima in Lattices, Codes, and Systems of Linear Equations". In: *Journal of Computer and System Sciences* 54.2 (1997), pp. 317–331. ISSN: 0022-0000.

[Ath+18] Anish Athalye, Nicholas Carlini, and David A. Wagner. "Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 274–283.

[Bre+18] Wieland Brendel, Jonas Rauber, and Matthias Bethge. *Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models*. 2018. arXiv: 1712.04248 [stat.ML].

[Car+19] Yair Carmon et al. "Unlabeled data improves adversarial robustness". In: *Advances in Neural Information Processing Systems* 32 (2019).

[CH20a] Francesco Croce and Matthias Hein. "Minimally distorted Adversarial Examples with a Fast Adaptive Boundary Attack". In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 2196–2205.

[CH20b] Francesco Croce and Matthias Hein. "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks". In: *International conference on machine learning*. PMLR. 2020, pp. 2206–2216.

[Che+17] Pin-Yu Chen et al. "ZOO: Zeroth Order Optimization Based Black-box Attacks to Deep Neural Networks without Training Substitute Models". In: *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security* (2017).

[Coh+19] Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. "Certified Adversarial Robustness via Randomized Smoothing". In: *ICML*. 2019.

[Cro+21] Francesco Croce et al. "RobustBench: a standardized adversarial robustness benchmark". In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. 2021.

[CW17] Nicholas Carlini and David Wagner. "Towards evaluating the robustness of neural networks". In: *2017 ieee symposium on security and privacy (sp)*. IEEE. 2017, pp. 39–57.

[CW18] Nicholas Carlini and David Wagner. "Audio adversarial examples: Targeted attacks on speech-to-text". In: *2018 IEEE security and privacy workshops (SPW)*. IEEE. 2018, pp. 1–7.

[Dai+22] Sihui Dai, Saeed Mahloujifar, and Prateek Mittal. "Parameterizing Activation Functions for Adversarial Robustness". In: *43rd IEEE Security and Privacy, SP Workshops 2022, San Francisco, CA, USA, May 22-26, 2022*. IEEE, 2022.

[Dan66] John M. Danskin. "The Theory of Max-Min, with Applications". In: *SIAM Journal on Applied Mathematics* 14.4 (1966), pp. 641–664. ISSN: 00361399. (Visited on 04/13/2022).

[Dev+19] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *ArXiv* abs/1810.04805 (2019).

[Din+15] Laurent Dinh, David Krueger, and Yoshua Bengio. "NICE: Non-linear Independent Components Estimation". In: *CoRR* abs/1410.8516 (2015).

[Din+17] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using Real NVP". In: *ArXiv* abs/1605.08803 (2017).

[Don+18] Yinpeng Dong et al. "Boosting adversarial attacks with momentum". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 9185–9193.

[Dos+20] Alexey Dosovitskiy et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[Eng+19a] Logan Engstrom et al. *Adversarial Robustness as a Prior for Learned Representations*. 2019. arXiv: 1906.00945 [stat.ML].

[Eng+19b] Logan Engstrom et al. *Exploring the Landscape of Spatial Robustness*. 2019. arXiv: 1712.02779 [cs.LG].

[Eng+19c] Logan Engstrom et al. *Robustness (Python Library)*. 2019. URL: https://github.com/MadryLab/robustness.

[Fuk79] Kunihiko Fukushima. "Self-Organization of a Neural Network which Gives Position-Invariant Response". In: *Proceedings of the Sixth International Joint Conference on Artificial Intelligence, IJCAI 79, Tokyo, Japan, August 20-23, 1979, 2 Volumes*. Ed. by Bruce G. Buchanan. William Kaufmann, 1979, pp. 291–293.

[Gei+19] Robert Geirhos et al. "ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness". In: *ArXiv* abs/1811.12231 (2019).

[Goo+14] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems* 27 (2014).

[Goo+15] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015.

[Goo+16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[Gow+19] Sven Gowal et al. "An Alternative Surrogate Loss for PGD-based Adversarial Testing". In: *ArXiv* abs/1910.09338 (2019).

[Gow+21] Sven Gowal et al. "Improving robustness using generated data". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 4218–4233.

[Gul+17] Ishaan Gulrajani et al. "Improved Training of Wasserstein GANs". In: *NIPS*. 2017.

[HD19] Dan Hendrycks and Thomas G. Dietterich. "Benchmarking Neural Network Robustness to Common Corruptions and Perturbations". In: *ArXiv* abs/1903.12261 (2019).

[He+16] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[Hen+19] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. "Using pre-training can improve model robustness and uncertainty". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2712–2721.

[Hen+21] Dan Hendrycks et al. "Natural adversarial examples". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 15262–15271.

[HP18] Hossein Hosseini and Radha Poovendran. *Semantic Adversarial Examples*. 2018. arXiv: 1804.00499 [cs.CV].

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[Hua+21] Hanxun Huang et al. "Exploring architectural ingredients of adversarially robust deep neural networks". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 5545–5559.

[Ily+19] Andrew Ilyas et al. "Adversarial examples are not bugs, they are features". In: *Advances in neural information processing systems* 32 (2019).

[IS15] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *ArXiv* abs/1502.03167 (2015).

[Kau+19] Simran Kaur, Jeremy M. Cohen, and Zachary Chase Lipton. "Are Perceptually-Aligned Gradients a General Property of Robust Classifiers?" In: *ArXiv* abs/1910.08640 (2019).

[KB15] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015.

[KD18] Diederik P. Kingma and Prafulla Dhariwal. "Glow: Generative Flow with Invertible 1x1 Convolutions". In: *NeurIPS*. 2018.

[Kri+12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012.

[Kur+17] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. "Adversarial examples in the physical world". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. 2017.

[KW14] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *CoRR* abs/1312.6114 (2014).

[Lec89] Yann Lecun. "Generalization and network design strategies". English (US). In: *Connectionism in perspective*. Ed. by R. Pfeifer et al. Elsevier, 1989.

[LF19] Cassidy Laidlaw and Soheil Feizi. *Functional Adversarial Attacks*. 2019. arXiv: 1906.00001 [cs.LG].

[Liu+22] Ye Liu et al. "Practical evaluation of adversarial robustness via adaptive auto attack". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 15105–15114.

[Ma+20] Xingjun Ma et al. "Imbalanced Gradients: A Subtle Cause of Overestimated Adversarial Robustness". In: *arXiv preprint arXiv:2006.13726* (2020).

[Mad+18] Aleksander Madry et al. "Towards Deep Learning Models Resistant to Adversarial Attacks". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. 2018.

[Mao+21] Xiaofeng Mao et al. "Composite adversarial attacks". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 10. 2021, pp. 8884–8892.

[Mit97] T.M. Mitchell. *Machine Learning*. McGraw-Hill International Editions. McGraw-Hill, 1997. ISBN: 9780071154673.

[Moo+16] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. "DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 2574–2582.

[Oor+16] Aaron Van Den Oord et al. *Conditional Image Generation with PixelCNN Decoders*. 2016. arXiv: 1606.05328 [cs.CV].

[Pap+15] Nicolas Papernot et al. *The Limitations of Deep Learning in Adversarial Settings*. 2015. arXiv: 1511.07528 [cs.CR].

[Pap+16] Nicolas Papernot, Patrick Mcdaniel, and Ian J. Goodfellow. "Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples". In: *ArXiv* abs/1605.07277 (2016).

[Pap+21] George Papamakarios et al. "Normalizing Flows for Probabilistic Modeling and Inference". In: *J. Mach. Learn. Res.* 22 (2021), 57:1–57:64.

[Pin+21] Maura Pintor et al. "Fast minimum-norm adversarial attacks through adaptive norm constraints". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 20052–20062.

[Pol64] B.T. Polyak. "Some methods of speeding up the convergence of iteration methods". In: *USSR Computational Mathematics and Mathematical Physics* 4.5 (1964), pp. 1–17. ISSN: 0041-5553.

[Ric+20] Leslie Rice, Eric Wong, and J. Zico Kolter. "Overfitting in adversarially robust deep learning". In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 8093–8104.

[RM21] Rahul Rade and Seyed-Mohsen Moosavi-Dezfooli. "Helper-based Adversarial Training: Reducing Excessive Margin to Achieve a Better Accuracy vs. Robustness Trade-off". In: *ICML 2021 Workshop on Adversarial Machine Learning*. 2021.

[Ron+19] Jérôme Rony et al. "Decoupling Direction and Norm for Efficient Gradient-Based L2 Adversarial Attacks and Defenses". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 4317–4325.

[Ros58] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65 6 (1958), pp. 386–408.

[Rum+86] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: 323.6088 (Oct. 1986), pp. 533–536. DOI: 10.1038/323533a0.

[Sal+19] Hadi Salman et al. "Provably Robust Deep Learning via Adversarially Trained Smoothed Classifiers". In: *NeurIPS*. 2019.

[Sal+20] Hadi Salman et al. "Do Adversarially Robust ImageNet Models Transfer Better?" In: *ArXiv* abs/2007.08489 (2020).

[Sam+18] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. *Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models*. 2018. arXiv: 1805.06605 [cs.CV].

[San+19] Shibani Santurkar et al. "Computer Vision with a Single (Robust) Classifier". In: *ArXiv* abs/1906.09453 (2019).

[SC18] Yash Sharma and Pin-Yu Chen. "Attacking the Madry Defense Model with L1-based Adversarial Examples". In: *ArXiv* abs/1710.10733 (2018).

[Sch+18] Ludwig Schmidt et al. "Adversarially Robust Generalization Requires More Data". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. 2018, pp. 5019–5031.

[Seh+20] Vikash Sehwag et al. "Hydra: Pruning adversarially robust neural networks". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 19655–19666.

[Seh+22] Vikash Sehwag et al. "Robust Learning Meets Generative Models: Can Proxy Distributions Improve Adversarial Robustness?" In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. 2022.

[Sha+18] Mahmood Sharif, Lujo Bauer, and Michael K. Reiter. "On the Suitability of Lp-Norms for Creating and Preventing Adversarial Examples". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2018), pp. 1686–16868.

[Sha+19] Ali Shafahi et al. "Adversarial training for free!" In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. 2019, pp. 3353–3364.

[Son+18a] Yang Song et al. *Constructing Unrestricted Adversarial Examples with Generative Models*. 2018. arXiv: 1805.07894 [cs.LG].

[Son+18b] Yang Song et al. *PixelDefend: Leveraging Generative Models to Understand and Defend against Adversarial Examples*. 2018. arXiv: 1710.10766 [cs.LG].

[SP97] Schuster and K.K. Paliwal. "Bidirectional recurrent neural networks". In: *IEEE Transactions on Signal Processing* 45.11 (1997), pp. 2673–2681. DOI: 10.1109/78.650093.

[Spa92] J.C. Spall. "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation". In: *IEEE Transactions on Automatic Control* 37.3 (1992), pp. 332–341. DOI: 10.1109/9.119632.

[Sri+14] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958.

[Sri+20] Gaurang Sriramanan, Sravanti Addepalli, Arya Baburaj, et al. "Guided adversarial attack for evaluating and enhancing adversarial defenses". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 20297–20308.

[Stu+19] David Stutz, Matthias Hein, and Bernt Schiele. "Disentangling adversarial robustness and generalization". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 6976–6987.

[Sze+14] Christian Szegedy et al. "Intriguing properties of neural networks". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. 2014.

[Tas+20] Yusuke Tashiro, Yang Song, and Stefano Ermon. "Diversity can be transferred: Output diversification for white-and black-box attacks". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 4536–4548.

[TB19] Florian Tramèr and Dan Boneh. "Adversarial Training and Robustness for Multiple Perturbations". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. 2019, pp. 5858–5868.

[Tra+18] Florian Tramèr et al. "Ensemble Adversarial Training: Attacks and Defenses". In: *ArXiv* abs/1705.07204 (2018).

[Tra+20] Florian Tramèr et al. "On Adaptive Attacks to Adversarial Example Defenses". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. 2020.

[Tsi+19] Dimitris Tsipras et al. *Robustness May Be at Odds with Accuracy*. 2019. arXiv: 1805.12152 [stat.ML].

[TT13] E. Tabak and C. Turner. "A Family of Nonparametric Density Estimation Algorithms". In: *Communications on Pure and Applied Mathematics* 66 (Feb. 2013). DOI: 10.1002/cpa.21423.

[Ues+18] Jonathan Uesato et al. "Adversarial risk and the dangers of evaluating against weak attacks". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 5025–5034.

[Utr+20] Francisco Utrera et al. "Adversarially-Trained Deep Nets Transfer Better". In: *ArXiv* abs/2007.05869 (2020).

[Vas+17] Ashish Vaswani et al. "Attention is All you Need". In: *ArXiv* abs/1706.03762 (2017).

[WB09] Zhou Wang and Alan C Bovik. "Mean squared error: Love it or leave it? A new look at signal fidelity measures". In: *IEEE signal processing magazine* 26.1 (2009), pp. 98–117.

[Won+20] Eric Wong, Leslie Rice, and J. Zico Kolter. "Fast is better than free: Revisiting adversarial training". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. 2020.

[Wu+20] Dongxian Wu, Shu-Tao Xia, and Yisen Wang. "Adversarial weight perturbation helps robust generalization". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 2958–2969.

[Xia+18] Chaowei Xiao et al. *Spatially Transformed Adversarial Examples*. 2018. arXiv: 1801.02612 [cs.CR].

[Xie+20] Cihang Xie et al. *Adversarial Examples Improve Image Recognition*. 2020. arXiv: 1911.09665 [cs.CV].

[Yam+22] Keiichiro Yamamura et al. "Diversified Adversarial Attacks based on Conjugate Gradient Method". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 24872–24894.

[Yu+21] Yunrui Yu, Xitong Gao, and Cheng-Zhong Xu. "LAFEAT: piercing through adversarial defenses with latent features". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 5735–5745.

[Yuk+21] Oguz Kaan Yuksel et al. *Semantic Perturbations with Normalizing Flows for Improved Generalization*. 2021. arXiv: 2108.07958 [stat.ML].

[Zha+18] Zhengli Zhao, Dheeru Dua, and Sameer Singh. *Generating Natural Adversarial Examples*. 2018. arXiv: 1710.11342 [cs.LG].

[Zha+19a] Runtian Zhai et al. "Adversarially Robust Generalization Just Requires More Unlabeled Data". In: *ArXiv* abs/1906.00555 (2019).

[Zha+19b] Dinghuai Zhang et al. "You only propagate once: Accelerating adversarial training via maximal principle". In: *Advances in Neural Information Processing Systems* 32 (2019).

[Zha+19c] Hongyang Zhang et al. "Theoretically principled trade-off between robustness and accuracy". In: *International conference on machine learning*. PMLR. 2019, pp. 7472–7482.

[Zha+20] Wei Emma Zhang et al. "Adversarial attacks on deep-learning models in natural language processing: A survey". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 11.3 (2020), pp. 1–41.

[Zha+21] Jingfeng Zhang et al. "Geometry-aware Instance-reweighted Adversarial Training". In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. 2021.

[ZK16] Sergey Zagoruyko and Nikos Komodakis. "Wide Residual Networks". In: *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*. BMVA Press, 2016.

[ZZ19] Tianyuan Zhang and Zhanxing Zhu. "Interpreting Adversarially Trained Convolutional Neural Networks". In: *ICML*. 2019.