Adversarial Robustness via Label-Smoothing

Morgane Goibert, Elvis Dohmatob

Adversarial Robustness via Label-Smoothing

Morgane Goibert
m.goibert@criteo.com
Criteo AI Lab

Elvis Dohmatob
e.dohmatob@criteo.com
Criteo AI Lab

Abstract

We study Label-Smoothing as a means for improving adversarial robustness of supervised deep-learning models. After establishing a thorough and unified framework, we propose several variations to this general method: adversarial, Boltzmann and second-best Label-Smoothing methods, and we explain how to construct your own one. On various datasets (MNIST, CIFAR10, SVHN) and models (linear models, MLPs, LeNet, ResNet), we show that Label-Smoothing in general improves adversarial robustness against a variety of attacks (FGSM, BIM, DeepFool, Carlini-Wagner) by better taking account of the dataset geometry. The proposed Label-Smoothing methods have two main advantages: they can be implemented as a modified cross-entropy loss, thus do not require any modifications of the network architecture nor do they lead to increased training times, and they improve both standard and adversarial accuracy.

1 INTRODUCTION

Neural Networks (NNs) have proved their efficiency in solving classification problems in areas such as computer vision [5]. Despite these successes, recent works have shown that NN are sensitive to adversarial examples (e.g [15]), which is problematic for critical applications [13] like self-driving cars and medical diagnosis. Many strategies have thus been developed to improve robustness and in an "arms race", different attacks have been proposed to evade these defenses. Broadly speaking, an adversarial attack succeeds when an image is slightly modified so that it still looks *to a human* like it belongs to a right class, but a classifier misclassifies it. Despite the number of works on it [4, 3, 16, 17], there is still no complete understanding of the adversarial phenomenon. Yet, the vulnerability of NN to adversarial attacks suggests a shortcoming in the generalization of the network. As overconfidence in predictions hinders generalization, addressing it can be a good way to tackle adversarial attacks [21]. Label-Smoothing (LS) [14, 19] is a method which creates uncertainty in the labels of a dataset used to train a NN. This uncertainty helps to tackle the over-fitting issue, and thus LS can be an efficient method to address the adversarial attack phenomenon.

Preliminary work. Under review by AISTATS 2020. Do not distribute.

1.1 Notations And Terminology

General We denote by $\mathcal{X}$ the input space of dimension d, and $\mathcal{Y} = \{1, ..., K\}$ the label space, $P_{X,Y}$ is the true (unknown) joint distribution of (X, Y) on $\mathcal{X} \times \mathcal{Y}$. $\Delta_K := \{q \in \mathbb{R}^K \mid q \geq 0,\ \sum_{k=1}^K q_k = 1\}$ is the $(K-1)$-dimensional probability simplex Δ_K, identified with the set $\mathcal{P}(\mathcal{Y})$ of probability distributions on $\mathcal{Y}$.

An iid sample drawn from $P_{X,Y}$ is written $S_n = \{(x_1, y_1), ...(x_n, y_n)\}$. To avoid any ambiguity with the label $y \in [\![K]\!]$, we use boldface $\boldsymbol{y} = (0, \dots, 1, \dots, 0) \in \Delta_K$ to denote the one-hot encoding of y. The empirical distribution of the input-label pairs (x, y) is written $\hat{P}^n_{X,Y}$. A classifier is a measurable function $h : \mathcal{X} \to \mathcal{Y}$, usually parametrized by real parameters θ; for example NN with several hidden layers, with the last always being a softmax function. The logits of the classifier (pre-softmax) are written $z(x, \theta)$, and $z^{(k)}(x, \theta)$ is its component for the kth class. The prediction vector of the classifier (post-softmax) is written $p(x; \theta)$.

Adversarial Attacks An attacker constructs an *adversarial example* based on a clean input x by adding a perturbation to it: $x^{adv} = x + \delta$. The goal of the attack is to have $h(x^{adv}) \neq h(x)$. The norm of the the perturbation vector δ measures the size of the attack. In this work, we limit ourselves to ℓ_∞-norm attacks, wherein $\|\delta\|_\infty := \max_{k=1}^p |\delta^k|$. A tolerance threshold

ε controls the size of the attack: the attacker is only allowed to inflict perturbations of size $\|\delta\|_\infty \leq \varepsilon$.

1.2 Related Works

Our work will focus on untargeted, white-box attacks, i.e. threat models that only seek to fool the NN (as opposed to tricking it into predicting a specific class), and have unlimited access to the NN parameters. State-of-the art attacks include FGSM [4] a very simple, fast and popular attack; BIM [6], an iterative attack based on FGSM; DeepFool [8] and C&W [2]. Note that the tolerance threshold ε can be explicitely tuned in FGSM and BIM, but not in DeepFool and C&W.

Many defences have been proposed to counteract adversarial attacks. The main one is adversarial training [4, 7], which consists in feeding a NN with both clean and adversarially-crafted data during training. This defense method will be used in this paper as a baseline for comparative purposes. Another important method is defensive distillation [10, 9], which is closely related to LS. This method trains a separate NN algorithm and uses its outputs as the input labels for the main NN algorithm. This was an efficient defense method until recently broken by C&W attack [2]. These defense methods are both very efficient, but time-costly and also, can reduce standard accuracy [18].

LS was first introduced as a regularization method [11, 14], but was also briefly studied as a defense method in [12]. One of the contributions of our paper is to generalize the idea of LS proposed and used in these previous works, and propose three novel variants relevant for the adversarial issue. We develop theoretical as well as empirical results about the defensive potential of LS.

For a more thorough introduction to the field, interested readers can refer to surveys like [1, 20].

1.3 Overview of main contributions

In section 2, we develop a unified framework for Label-Smoothing (LS) of which [14] is a special case. We propose a variety of new LS methods, the main one being Adversarial Label-Smoothing (ALS). We show that in general, LS methods induce some kind of logit-squeezing, which results in more robustness to adversarial attacks. In section 3.2, we give a complete mathematical treatment of the effect of LS in a simple case, with regards to robustness to adversarial attacks. Section 4 reports empirical results on real datasets. In section 5, we conclude and provide ideas for future works.

Proofs of all theorems and lemmas are provided in the Appendix (supplementary materials).

2 UNIFIED FRAMEWORK FOR LS

In standard classification datasets, each example x is hard-labeled with exactly one class y. Such overconfidence in the labels can lure a classification algorithm into over-fitting the input distribution [14]. LS [14, 19] is a resampling technique wherein one replaces the vector of probability one on the true class $\boldsymbol{y}$ (i.e the one-hot encoding) with a different vector q which is "close" to $\boldsymbol{y}$. Precisely, LS withdraws a fraction of probability mass from the "true" class label and reallocates it to other classes. As we will see, the redistribution method is quite flexible, and leads to different LS methods.

Let $\mathrm{TV}(q'\|q) := (1/2)\|q' - q\|_1$ be the Total-Variation distance between two probability vectors $q', q \in \Delta_K$. For $\alpha \in [0, 1]$, define the uncertainty set of acceptable label distributions $\mathcal{U}_\alpha(\hat{P}^n_{X,Y})$ by

$$\begin{aligned}\mathcal{U}_\alpha(\hat{P}^n_{X,Y}) &:= \{\hat{P}^n_X \hat{Q}^n_{Y|X} \mid \mathrm{TV}(\hat{Q}^n_{Y|x} \| \hat{P}^n_{Y|x}) \leq \alpha, \\ &\qquad \forall x \in \mathcal{X}\} \\ &= \left\{ \frac{1}{n}\sum_{i=1}^n \delta_{x_i} \otimes q_i \mid q_i \in \Delta_K,\ \mathrm{TV}(q_i \| \delta_{y_i}) \leq \alpha, \right. \\ &\qquad \left. \forall i \in [\![n]\!] \right\},\end{aligned}$$

made up of joint distributions $\hat{Q}^n_{X,Y} \in \mathcal{P}(\mathcal{X} \times \mathcal{Y})$ on the dataset S_n, for which the conditional label distribution $q_i := \hat{Q}^n_{Y|X=x_i} \in \Delta_K$ is within TV distance less than α of the one-hot encoding of the observed label y_i. By direct computation, one has that $\mathrm{TV}(q_i \| y_i) = (1/2)\left(\sum_{j \neq y_i} q_i^{(j)} + 1 - q_i^{(y_i)}\right) = 1 - q_i^{(y_i)}$ and so the uncertainty set can be rewritten as

$$\begin{aligned}\mathcal{U}_\alpha(\hat{P}^n_{X,Y}) = \left\{ \frac{1}{n}\sum_{i=1}^n \delta_{x_i} \otimes q_i \mid q_i \in \Delta_K, \right. \\ \left. q_i^{(y_i)} \geq 1 - \alpha\ \forall i \in [\![1, n]\!] \right\}.\end{aligned}$$

Any conditional label distribution q_i from the uncertainty set $\mathcal{U}_\alpha(\hat{P}^n_{X,Y})$ can be written

$$q_i = (1 - \alpha)\boldsymbol{y_i} + \alpha q'_i, \quad q'_i \in \Delta_K. \tag{1}$$

Different choices of the probability vector q'_i (which as we shall see in section 2.1, can depend on the model parameters!) lead to different Label-Smoothing methods. The training of a NN with general label smoothing then corresponds to the following optimization problem:

$$\min_\theta \frac{1}{n}\sum_{i=1}^n \mathrm{SmoothCE}(x_i, q_i; \theta), \tag{2}$$

where $\text{SmoothCE}(x, q; \theta)$ is the *smoothed* cross-entropy loss (a generalization of the standard cross-entropy loss), defined by

$$\begin{aligned}\text{SmoothCE}(x, q; \theta) &:= -q^T \log(p(x; \theta)) \\ &= -\sum_{k=1}^{K} q^{(k)} \log(p^{(k)}(x; \theta)).\end{aligned}$$

It turns out that the optimization problem (2) can be rewritten as the optimization of a usual cross-entropy loss, plus a penalty term on the gap between the components of logits (one logit per class) produced by the model on each example x_i.

Theorem 1 (General Label-Smoothing Formulation)**.** *The optimization problem* (2) *is equivalent to the logit-regularized problem*

$$\min_{\theta} L_n(\theta) + \alpha R_n(\theta),$$

where $L_n(\theta) := -\frac{1}{n}\sum_{i=1}^{n} \log(p(x_i; \theta))$ *is the standard cross-entropy loss, and*

$$R_n(\theta) := \frac{1}{n}\sum_{i=1}^{n} (\boldsymbol{y_i} - q_i')^T z_i.$$

where $z_i := z(x_i; \theta) \in \mathbb{R}^K$ *is the logits vector for* x_i*.*

In the next two subsections, we present four different LS methods that are relevant to tackle the adversarial robustness issue. A summary of these methods are presented in Table 1.

2.1 Adversarial Label-Smoothing

Adversarial Label-Smoothing (ALS) arises from the worst possible smooth label q_i for each example x_i. To this end, consider the two-player game:

$$\min_{\theta} \max_{\hat{Q}^n \in \mathcal{U}_\alpha(\hat{P}^n_{X,Y})} \frac{1}{n}\sum_{i=1}^{n} \text{SmoothCE}(x_i, q_i; \theta), \qquad (3)$$

The inner problem in (3) has an analytic solution (see Appendix 1.2) given by, $\forall\, i \in [\![1, n]\!]$:

$$q_i = q_i(\theta) = (1 - \alpha)\boldsymbol{y_i} + \alpha \boldsymbol{y_i}^{\text{worst}}, \qquad (4)$$

where $y_i^{\text{worst}} \in \text{argmin}_{k=1}^{K} z^{(k)}(x_i, \theta)$ is the index of the smallest component of the logits vector z_i for input x_i, and $\boldsymbol{y_i}^{\text{worst}}$ is the one-hot encoding thereof.

Interpretation Of ALS The scalar $\alpha \in [0, 1]$ acts as a smoothing parameter: if $\alpha = 0$, then $q_i(\theta) = \boldsymbol{y_i}$, and we recover hard labels. If $\alpha = 1$, the adversarial weights $q_i(\theta)$ live in the sub-simplex spanned by the smallest components of the predictions vector $p(x_i; \theta)$. For $0 < \alpha < 1$, $q_i(\theta)$ is a proper convex combination of the two previous cases. Applying Theorem 1, we have:

Corollary 1 (ALS enforces logit-squeezing)**.** *The logit-regularized problem equivalent of the ALS problem* (3) *is given by:* $\min_{\theta} L_n(\theta) + \alpha R_n(\theta)$*, where*

$$R_n(\theta) := \frac{1}{n}\sum_{i=1}^{n} (\boldsymbol{y_i} - \boldsymbol{y_i}^{worst})^T z_i = \frac{1}{n}\sum_{i=1}^{n} z_i^{(y_i)} - z_i^{(y_i^{worst})}.$$

For each data point x_i with true label y_i, the logit-squeezing penalty term $R_n(\theta)$ forces the model to refrain from making over-confident predictions, corresponding to large values of $z_i^{(y_i)} - z_i^{(y_i^{\text{worst}})}$ that can lead to overfitting. This means that every class label receives a positive prediction output probability: $\forall\, k \in [\![K]\!],\ p_i^{(k)} > 0$. The resulting models are less vulnerable to adversarial perturbations on the input x.

One can also see ALS as the label analog of adversarial training [4, 6]. Instead of modyfing the input data x, we modify the label data y. However, unlike adversarial training, ALS is attack-independent: it does not require to choose a specific attack method to be trained on. Moreover, in contrary to adversarial training, ALS does not entail any significant increase in training time over standard training.

ALS Implementation We noted that ALS only consists in redefining a loss, and using the smoothed cross-entropy with an adversarially-modified version of the one-hot encoding of the class labels. It is very simple to implement, and computationally as efficient as standard training.

See algorithm 1 for an easy implementation of ALS.

Algorithm 1 Adversarial Label-Smoothing (ALS) training

Input: training data $(x_1, y_1), (x_2, y_2), \ldots$; a given model; smoothing parameter $\alpha \in [0, 1]$;
for each epoch **do**
  for each mini-batch $x = x_{1:m}$, $y = y_{1:m}$ **do**
    Smooth labels $y_{ALS} \leftarrow (q_1, \ldots, q_m)$ via (4)
    Get predictions $y_{pred} \leftarrow model(x)$
    Compute loss $\leftarrow \text{SmoothCE}(y_{pred}, y_{ALS})$
    Update model parameters θ via back-prop
  end for
end for

2.2 Other Label-Smoothing Methods

Standard Label-Smoothing *Standard Label-Smoothing* (SLS) is the method developed in [14]. It corresponds to uniformly re-assigning the mass α removed from the real class over the other classes. That is, the term $q_i' \in \Delta_K$ in (1) is given by

$$q_i' = \sum_{k=1, k\neq y_i}^{K} \frac{1}{K-1}\boldsymbol{y^{(k)}} = \frac{1}{K-1}(1 - \boldsymbol{y_i}). \qquad (5)$$

Paper	Name	q'_i	Induced logit penalty $R_n(\theta)$
[14]	standard label-smoothing (SLS)	$\frac{1-\boldsymbol{y_i}}{K-1}$	$\sum_{i=1}^n \left(\frac{K\boldsymbol{y_i}-1}{K-1}\right)^T z_i$
Our paper	adversarial label-smoothing (ALS)	$\boldsymbol{y_i}^{\text{worst}}$, see (4)	$\frac{1}{n}\sum_{i=1}^n z_i^{(y_i^{\text{worst}})} - z_i^{(y_i)}$
Our paper	Boltzmann label-smoothing (BLS)	$q_i'^{\text{Boltz}}$, see (6)	$\frac{1}{n}\sum_{i=1}^n (q'_i - \boldsymbol{y_i})^T z_i$
Our paper	second-best label-smoothing (SBLS)	$\boldsymbol{y_i}^{\text{SB}}$, see (7)	$\frac{1}{n}\sum_{i=1}^n z_i^{(y_i'^{\text{SB}})} - z_i^{(y_i)}$

Table 1: **Different of LS methods.** They all derive from the general equation (1).

In this case, $R_n(\theta) = \sum_{i=1}^n \left(\frac{K\boldsymbol{y_i}-1}{K-1}\right)^T z_i$. If $K = 2$, with a perfect model (i.e. $z_i^{(k)} \geq 0$ if $y_i^{(k)} = 1$ and $z_i^{(k)} < 0$ else), we have $R_n(\theta) = \sum_{i=1}^n \|z_i\|_1$, an ℓ_1-norm penalty on the logits.

Boltzmann Label-Smoothing ALS puts weights on only two classes: the true class label (due to the constraint of the model) and the class label which minimizes the logit vector. It thus gives "two-hot" labels rather than "smoothed" labels. Replacing *hard-min* with a *soft-min* in (4) leads to the so-called *Boltzmann Label-Smoothing* (BLS), defined by setting the term $q'_i \in \Delta_K$ in (1) to:

$$q'_i = \sum_{k=1, k\neq y_i}^{K} \text{Boltz}_T^{(k)}(x_i, \theta), \tag{6}$$

where $\text{Boltz}_T^{(k)}(x_i, \theta) = \frac{\exp(-z^{(j)}(x_i;\theta)/T)}{\sum_{k'=1,k'\neq y_i}^K \exp(-z^{(k')}(x_i;\theta)/T)}$ is the Boltzmann distribution with energy levels $z^{(k)}(x_i;\theta)$ at temperature $T \in [0, \infty]$. It interpolates between ALS (corresponding to $T = 0$), and SLS (corresponding to $T = \infty$).

Second-Best Label-Smoothing SLS, ALS and BLS give positive prediction outputs for every label because we add weight to either every label, or the "worst" wrong label. However, in the problem we consider, it does not matter if we fool the classifier by making it predict the "worst" or the "closest" wrong class. Therefore, a completely different approach consists in concentrating our effort and add all the available mass α only on the "closest" class label. This leads to *Second Best Label-Smoothing* (SBLS) defined by:

$$q'_i = \boldsymbol{y_i}^{\text{SB}} := \underset{k=1,\, k\neq y_i}{\overset{K}{\text{argmax}}}\ p^{(k)}(x_i;\theta). \tag{7}$$

The problem can be rewritten as:

$$\min_\theta L_n(\theta) + \alpha\,(-1)\frac{1}{n}\sum_{i=1}^n \max_{k\neq y_i}(z_i^{(k)}) - z_i^{(y_i)}.$$

Note the correspondence with the opposite of the Hinge loss in the second term: this penalty tends to make the margin between the true class prediction and the closest wrong class prediction smaller.

Training with each of these Label-Smoothing methods (SLS, BLS, SBLS) can be implemented via Alg. 1, using Eqn. 5, 6 or 7 respectively, in line 5 of the the Alg. 1 instead of Eqn. 4. Note also that any other choice of q'_i can lead to different variations of LS, and can be easily implemented the same way.

Here,we finally obtain four different LS methods: ALS, BLS, SBLS and SLS. The effects of ALS in particular and LS as a general method are investigated in Section 3.1 and 3.2, and each of the four methods will be tested as defense methods in Section 4. We argue in the next sections that LS as a general method can improve robustness. The choice of the specific method should afterwards be guided by the type of data or problem one is facing, but ALS can be chosen as a default method to improve robustness in the most general cases.

3 UNDERSTANDING LABEL SMOOTHING

3.1 Fading Gaussian Example

We now explore a simple example illustrating some of the implications of using LS, with regards to standard accuracy and rebustness to adversarial attacks. Let us consider the following problem inspired by [18]: $Y \sim \mathcal{U}(\{-1, 1\})$, and $X_1, ..., X_d | Y = y \overset{\text{iid}}{\sim} \mathcal{N}(y\mu_i, \sigma_i^2)$ so that d is the number of features. The possible classifiers studied here are linear classifiers: $f_w : x \mapsto \text{sign}(w^T x)$, with parameters $w \in \mathbb{R}^d$.

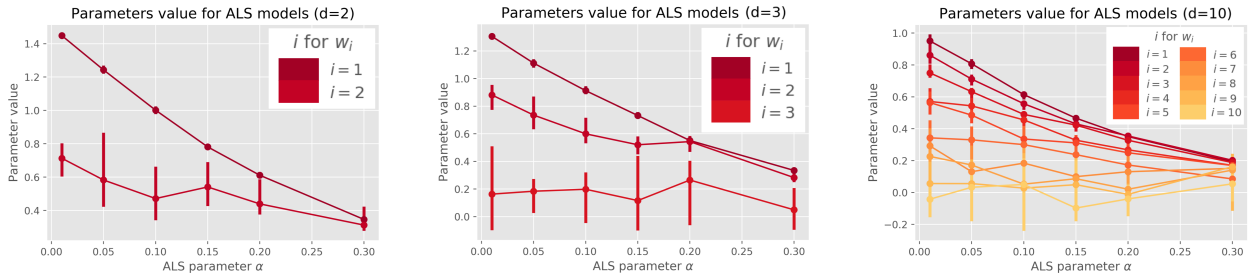
We want to compare the performances (both standard and adversarial and depending only on w) of the Bayes classifier and the ALS classifier the above problem.

Standard accuracy One computes the adversarial robustness accuracy of the linear model f_w as

$$\begin{aligned}\text{acc}(f_w) &:= \mathbb{P}(f_w(x) = y) = \mathbb{P}(yw^T x > 0) \\ &= \mathbb{P}(\mathcal{N}(w^T\mu; \sum_j w_j^2\sigma_j^2) > 0) = \Psi\left(\frac{w^T\mu}{\sqrt{\sum_j w_j^2\sigma_j^2}}\right),\end{aligned} \tag{8}$$

where Ψ is the CDF of the the standard Gaussian.

Adversarial accuracy For bounded ℓ_∞-norm attacks, one computes the adversarial robustness accu-



(a) d=2 (b) d=3 (c) d=10

Figure 1: Parameters w in the fading Gaussian experiments

racy of the linear model f_w as

$$\begin{aligned}\mathrm{acc}_\varepsilon(f_w) &:= \mathbb{P}(f_w(x') = y\ \forall x' \in \mathbb{R}^d,\ \|x' - x\|_\infty \leq \varepsilon) \\ &= \mathbb{P}\left(\min_{\|\Delta x\|_\infty \leq \varepsilon} yw^T(x + \Delta x) > 0\right) \\ &= \mathbb{P}(yw^T x - \varepsilon||w||_1 > 0) = \Psi\left(\frac{w^T\mu - \varepsilon||w||_1}{\sqrt{\sum_{j=1}^d \sigma_j^2 w_j^2}}\right),\end{aligned} \tag{9}$$

where ε is the strength of the attack. By the way, the optimal adversarial perturbation is given by $x^{adv} = x - \varepsilon y\,\mathrm{sign}(w)$, where $\mathrm{sign}(w) := (\mathrm{sign}(w_1), \ldots, \mathrm{sign}(w_d))$. When $\varepsilon = 0$, we recover the standard accuracy formula.

We have the following Lemma.

Lemma 1. *In the special case where the covariance matrix is diagonal $\Sigma = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_d^2)$, the most robust linear classifier has weights $w \in \mathbb{R}^d$ given by*

$$w_j \propto \sigma_j^{-2}\,\mathrm{sign}(\mu_j)(|\mu_j| - \varepsilon)_+, \tag{10}$$

where $(a)_+ := \max(0, a)$, and the the use of the proportionality symbol "$\propto$" indicates a hidden constant independent of the feature index j.

Moreover, the optimal adversarial accuracy amongst all linear classifiers is given by

$$\max_{w \in \mathbb{R}^d} acc_\varepsilon(f_w) = \Psi(\sqrt{\Delta(\varepsilon)}), \tag{11}$$

where $\Delta(\varepsilon) = \sum_{j=1}^d \sigma_j^{-2}\left((|\mu_j| - \varepsilon)_+\right)^2$.

Thus the best solution for w in terms of adversarial accuracy depends on the feature-wise signal-to-noise ratio (SNR) $\frac{|\mu_j|}{\sigma_j^2}$: if this ratio is high, then the jth feature is quite determinant of y because it is both far enough (high mean) from the wrong class and reliable enough (small standard deviation).

Bayes classifier One computes the posterior probability of the positive class label given a sample x as $\mathbb{P}(Y = 1|x) = (1 + \mathrm{e}^{-2\sum \frac{\mu_j}{\sigma_j^2} x_j})^{-1}$, and so $\mathbb{P}(Y = 1|x) > 1/2 \Leftrightarrow \sum_{j=1}^d \frac{\mu_j}{\sigma_j^2} x_j > 0$. Thus the Bayes-optimal classifier is linear with parameters given by $w_j = \mu_j/\sigma_j^2,\ \forall j$. Thus in particular, if $\mu_j/\sigma_j^2 =$ constant $\forall j$ (as in [18]), then the Bayes optimal classifier corresponds to the linear model with parameters $w = (1, \ldots, 1)$.

ALS classifier The ALS problem reduced to finding w minimizing the expected value of the smooth cross-entropy loss defined in 2. Let us note p_1 and p_{-1} the predicted probabilities of each class, so $p_1 = 1/(1 + \mathrm{e}^{-w^T x})$ and $p_{-1} = 1 - p_1$. Depending on the values of x and y, the loss can take different forms:

- $l_1(x; w) = \alpha \ln(p_{-1}) + (1 - \alpha)\ln(p_1)$ if $y = 1$ and $w^T x > 0$
- $l_2(x; w) = \ln(p_1)$ if $y = 1$ and $w^T x < 0$
- $l_3(x; w) = \ln(p_{-1})$ if $y = -1$ and $w^T x > 0$
- $l_4(x; w) = (1 - \alpha)\ln(p_{-1}) + \alpha \ln(p_1)$ if $y = -1$ and $w^T x < 0$, thus

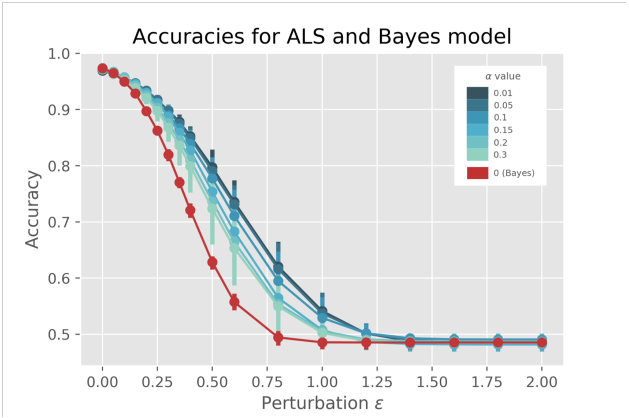
$$\begin{aligned}\mathbb{E}_{X,Y}(loss_w(X, Y)) = \frac{1}{2}\Bigg[&\int_{w^T x > 0} l_1(x; w) f_{Y=1}(x)\,\mathrm{d}x + \\ &\int_{w^T x < 0} l_2(x; w) f_{Y=1}(x)\,\mathrm{d}x + \int_{w^T x > 0} l_3(x; w) f_{Y=-1}(x)\,\mathrm{d}x \\ &+ \int_{w^T x < 0} l_4(x; w) f_{Y=-1}(x)\,\mathrm{d}x\Bigg]\end{aligned}$$

where $f_{Y=1}$ (resp. $f_{Y=-1}$) is the density of $X|Y = 1$ (resp. $X|Y = -1$)

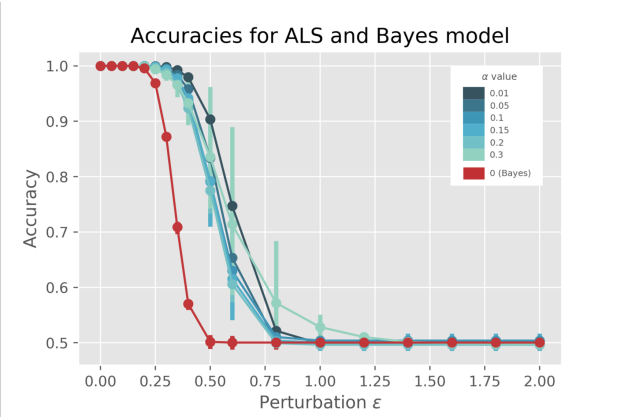
The derivative of equation 3.1 is difficult to express in close form. The values for w obtained when running the experiments (see below) can be derived from Figure 1. We easily check that the features are "correctly ordered" in terms of parameter w : the more important the feature is, higher the w is, which leads to better adversarial accuracy.

Empirical illustration We ran some experiments choosing $\forall\, i \in [\![1, d]\!], \sigma_i = 1 - (i - 1)/d$ and so $\mu_i = \sigma_i^2$.

Figure 2 shows the standard and adversarial accuracies for the Bayes classifier as well as ALS classifiers for



(a) **Results:** Fading gaussian experiment for d=10 features. ALS accuracy is better especially for small α



(b) **Results:** Fading gaussian exp. for d=100 features. ALS accuracy is also better especially for small α

Figure 2: Fading gaussian experiment

different values of α, as a function of the perturbation ε. We see that even for d as small as $d = 10$, the ALS classifiers do better (especially for small values of α) than the Bayes classifier under the adversarial regime, which is consistent with the w values obtained in Fig. 1 and Lemma 1. Importantly, the standard accuracy ($\varepsilon = 0$) is not so different between Bayes and ALS classifiers, suggesting that ALS enables better adversarial generalization without damaging the "natural" one.

Figure 3 shows the kernel density plot (KDE) of the dataset when $d = 2$. The black line is the main direction of the density, while the two red lines are different decision boundaries for two different choices of w. The plain red line is the decision boundary corresponding to the Bayes case, so $w = (1, 1)$, while the dashed red line corresponds here to the case $w = (4, 1)$ and is orthogonal to the black line.

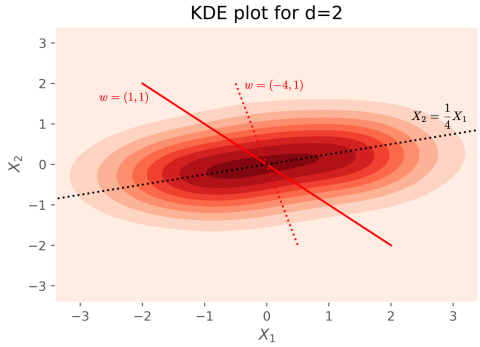


Figure 3: KDE plot: adversarial acc. dashed decision boundary is higher than plain's one.

It is interesting to realize that potential adversarial examples are points that lay around the decision boundary (of course, at a distance depending on the choice of the perturbation ε). Thus, it is more efficient, under the adversarial regime, to have a decision boundary that crosses regions of low density, or spends the shortest possible time in regions of high density. This is exactly what does the dashed red decision boundary, compared to the plain red/Bayes decision boundary that does not take into account this phenomenon. This is equivalent to say that in order to gain better robustness, a classifier must use the information provided by the variability of the features, a fact which was already apparent in the formula for w_j derived in the adversarial accuracy paragraph.

3.2 A Closer Look At Label-Smoothing

3.2.1 Logit-squeezing and gradient-based methods

Applying label-smoothing (LS) generates a logit-squeezing effect (see Theorem 1) which tends to prevent the model from being over-confident in its predictions. This effect was investigated in [11] and is illustrated in Fig. 4a, where we plot the prediction values for different MLP classifiers trained on the moon dataset.

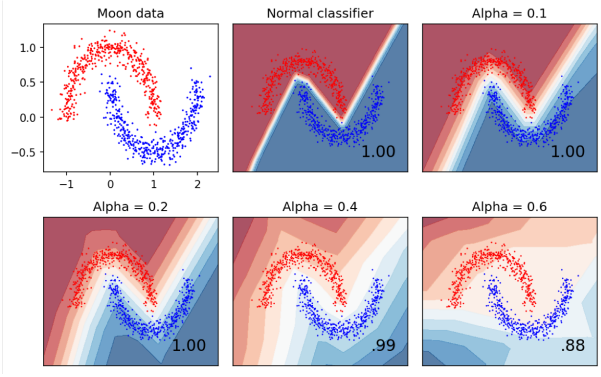
In addition to this impact on the logits and predictions, LS also have an effect on the logits' gradients (with respect to x, see Fig. 4b where we plot these gradients for one-layer linear classifiers -ALS and natural- trained on MNIST) which can help explain why ALS trained models are more robust to adversarial attacks. As described in [12], using a linear approximation, an attack is successful if

$$p_y(x,\theta) + \delta^T \nabla_x p_y(x,\theta) \leq p_k(x,\theta) + \delta^T \nabla_x p_k(x,\theta)$$

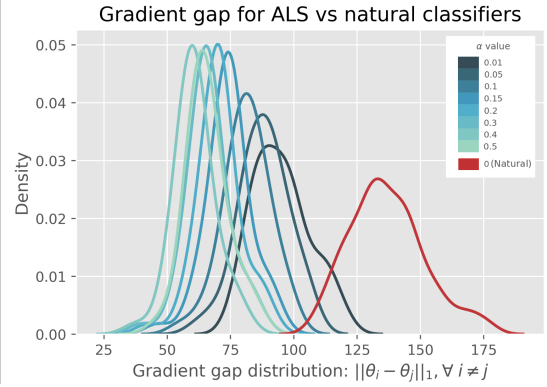
for any $k \neq y$, where δ is the attack perturbation. With an FGSM-like attack of strength ϵ, it thus works if

$$\epsilon \geq \frac{1}{||\nabla_x z_y(x,\theta) - \nabla_x z_j(x,\theta)||_1}.$$

By reducing this gradient gap, LS provides more robust models at least against gradient-based attack methods.



(a) **Regularization effect:** logit squeezing using ALS (different α) and a MLP classifier. Darker is more confidence.



(b) **Gradient gap reduction** for a one-layer linear classifier on MNIST.

Figure 4: Effects of LS

3.2.2 Why does LS help adversarial robustness?

Pointwise, the SmoothCE loss induces different costs compared to the traditional CE loss. As discussed in Sec. 3.2.1, over-confidently classified points are more penalized. Likewise, very badly classified points are also more penalized. The model is thus forced to put the decision boundary in a region with few data points (see Section 3.1). If not, either the penalty term $R_n(\theta)$ or the general term $L_n(\theta)$, defined in Sec. 2.1, will be too high. The underlying geometry of the dataset is thus better addressed compared to a traditional training: boundaries are closer to "the middle", i.e the margin between two classes is bigger (similar to how SVM operates), leading to increased robustness. Traditional CE loss, however, induces a direct power relationship: the boundary between two classes is pushed close to the smallest one.

4 EXPERIMENTS

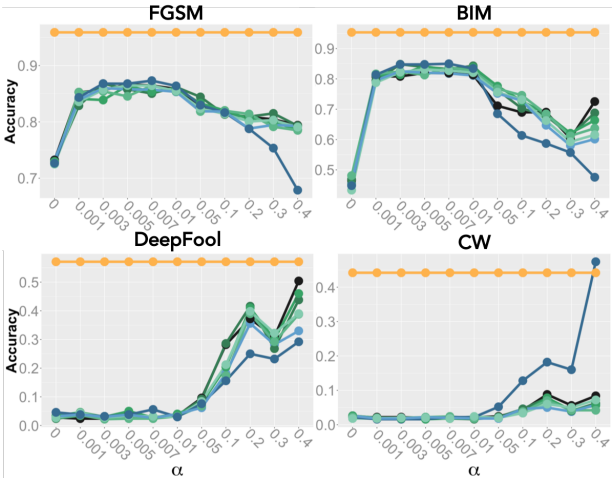
Set-up We run the four different attacks[1] on different set ups (datasets MNIST, CIFAR10, SVHN where the pixel range is 1, and models MLP, LeNet, ResNet18). For comparison purposes, we also run the attacks on reference models: "natural" classifier, i.e. the same models used in the experiments but without any LS or regularization (see red lines in Fig. 5d and $\alpha = 0$ values for Figs. 5a, 5b, 5c); and models trained using adversarial training against PGD as defined in [7] with 3 iterations (see yellow lines in Fig. **??**). The parameters used here for the PGD-training are: for MNIST (Linear and LeNet), $\epsilon_{PGD} = 0.25$ and $\alpha_{PGD} = 0.1$ (same notations as in [7]), and for every other set-up, $\epsilon_{PGD} = 0.05$ and $\alpha_{PGD} = 0.02$. PGD-training is a SOTA defense, and we chose to use 3 iterations here once again for comparative purposes, but this time with respect to training time: our method is as fast as a natural training, but PGD takes way longer to train (in this case with 3 iterations, the training time was at least 3 time longer on every set-up than for ALS training, sometimes even 15 times longer). PGD-training is not run on the CIFAR10 ResNet18 set-up because of this long training time.

Results Some results are shown in Fig. **??**, and see Tables 2 to 6 (Appendix 1.4) for more extensive results.

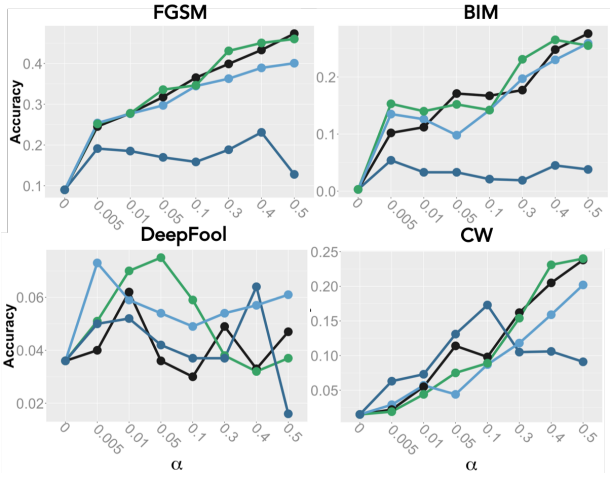
One can see that Label-Smoothing performs always better in terms of robustness than the natural classifiers, which indicates that LS indeed provides some kind of robustness. Moreover, our LS methods are competitive with PGD-training. On MNIST LeNet (see fig. 5a), PGD training yields better results against FGSM, BIM and DeepFool, but in these three cases, our LS methods are not far from PGD-training and SBLS is better than PGD-training against C&W attack. On the other set-ups, LS is better against every attack except FGSM or BIM only when the strenght of the attack ε is very small (but note that for SVHN and CIFAR10, the standard accuracy with PGD-training is weakened). On the whole, ALS and BLS give better results than SLS and SBLS. They thus should be preferred as default methods when implementing LS. However, overall, there is no major differences in the results obtained with the different LS methods, and the temperature hyperparameter for BLS method does not seem to have a great impact on the results. ($T = 0.001$ for example is a good default value).

Altogether, we see that LS is a good candidate for im-

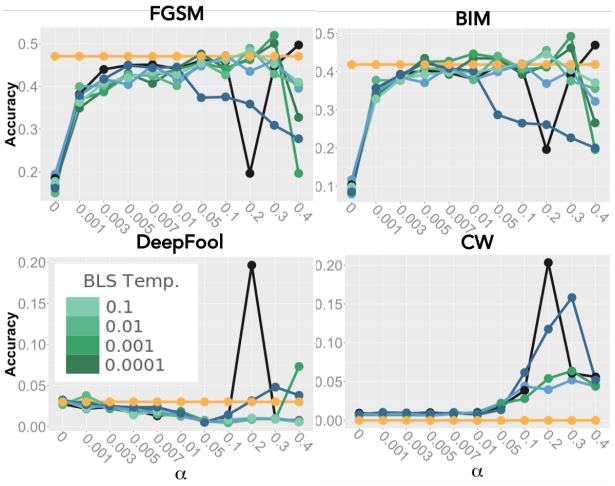
[1] In the CIFAR10 ResNet and SVHN LeNet set-ups, the number of iteration for C&W attack is sub-optimal. More iterations would have broken the models, however, C&W is hardly scalable because it requires a long time to train, especially for sophisticated models like ResNet.



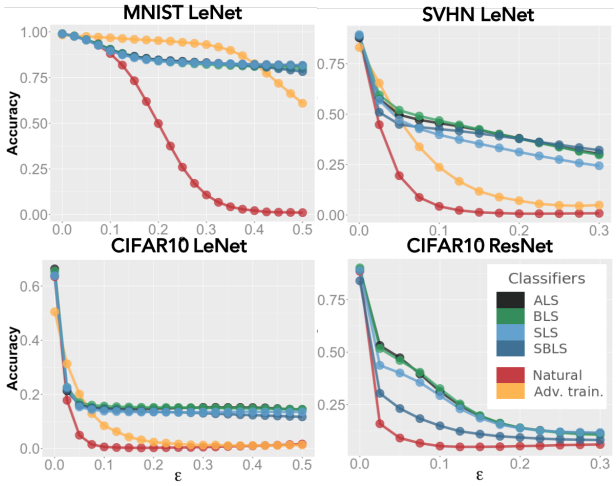
(a) **MNIST LeNet.** $\epsilon = 0.15$ for FGSM and BIM



(b) **CIFAR10 ResNet.** $\epsilon = 0.05$ for FGSM and BIM



(c) **SVHN LeNet.** $\epsilon = 0.05$ for FGSM and BIM



(d) **FGSM:** all set-ups

Figure 5: **Experiments:** Figs 5a, 5b, 5c show the evolution of adversarial accuracy as a function of α for different models against all attacks. Fig 5d shows it as a function of ϵ for all models against FGSM.

proving the adversarial robustness of NNs, eventhough it will not deliver perfectly robust model. Still, the absence of computation cost makes it a very useful robustification method for lots of real life applications where NNs are required to be trained quickly.

5 CONCLUSION

We have proposed a general framework for Label-Smoothing (LS) as well as a new variety of LS methods (Section 2) as a way to alleviate the vulnerability of Deep learning image classification algorithms. We developed a theoretical understanding of LS (Theorem 1 and Section 3.2) and our results have been demonstrated empirically via experiments on real datasets (CIFAR10, MNIST, SVHN), neural-network models (MLP, LeNet, ResNet), and SOTA attack models (FGSM, BIM, DeepFool, C&W).

LS improves the adversarial accuracy of neural networks, and can also boost standard accuracy, suggesting a connection between adversarial robustness and generalization. Even though our results (see Section 3.2) provide evidence that LS classifiers are more robust because they take the dataset geometry into better consideration, better understanding of the adversarial phenomenon and the representations learned by NNs would be desirable.

Moreover, compared to other defense methods (e.g adversarial training), the ease of implementation of LS is very appealing: it is simple, fast, with one interpretable hyperparameter ($\alpha \in [0, 1]$). Being costless is one of the major benefits of implementing LS. Experimental results (section 4) could be completed with various NNs and datasets, which is also left for future works.

References

[1] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

[2] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.

[3] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Robustness of classifiers: from adversarial to random noise. In *Advances in Neural Information Processing Systems*, pages 1632–1640, 2016.

[4] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[6] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[7] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[8] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

[9] Nicolas Papernot and Patrick McDaniel. On the effectiveness of defensive distillation. *arXiv preprint arXiv:1607.05113*, 2016.

[10] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597. IEEE, 2016.

[11] Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

[12] Ali Shafahi, Amin Ghiasi, Furong Huang, and Tom Goldstein. Label smoothing and logit squeezing: A replacement for adversarial training? 2018.

[13] Chawin Sitawarin, Arjun Nitin Bhagoji, Arsalan Mosenia, Mung Chiang, and Prateek Mittal. Darts: Deceiving autonomous cars with toxic signs. *arXiv preprint arXiv:1802.06430*, 2018.

[14] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, June 2016.

[15] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[16] Thomas Tanay and Lewis Griffin. A boundary tilting persepective on the phenomenon of adversarial examples. *arXiv preprint arXiv:1608.07690*, 2016.

[17] Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*, 2017.

[18] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *stat*, 1050:11, 2018.

[19] David Warde-Farley. Adversarial perturbations of deep neural networks. 2016.

[20] Jiliang Zhang and Xiaoxiong Jiang. Adversarial examples: Opportunities and challenges. *arXiv preprint arXiv:1809.04790*, 2018.

[21] Qinghe Zheng, Mingqiang Yang, Jiajie Yang, Qingrui Zhang, and Xinxin Zhang. Improvement of generalization ability of deep cnn via implicit regularization in two-stage training process. *IEEE Access*, 6:15844–15869, 2018.

1 Appendix

1.1 LS optimization program

Proof of Theorem 1. Recall that $\boldsymbol{y_i} \in \Delta_K$ is the one-hot encoding of the example x_i with label $y_i \in [\![1, K]\!]$. By direct computation, one has

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}\text{SmoothCE}(x_i, q_i; \theta) &= \frac{1}{n}\sum_{i=1}^{n} q_i^T \ln(p(x_i;\theta)) \\ &= -\frac{1}{n}\sum_{i=1}^{n}((1-\alpha)\boldsymbol{y_i} + \alpha q_i')^T \ln(p(x_i;\theta)) \\ &= -\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{y_i}^T \ln(p(x_i;\theta)) + \\ &\qquad \alpha\frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{y_i} - q_i')^T \ln(p(x_i;\theta)) \\ &= L_n(\theta) + \alpha\frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{y_i} - q_i')^T z_i,\end{aligned}$$

where $z_i \in \mathbb{R}^K$ is the vector logits for example x_i. □

1.2 Analytic solution for ALS formula

Lemma 2. *Let $\alpha \in [0, 1]$, $t \in [\![k]\!]$, and $g \in \mathbb{R}^k$. The general solution of the problem*

$$\underset{q\in\Delta_k,\, q^{(t)}\geq 1-\alpha}{argmax} \quad q^T g \tag{12}$$

is $q^* = (1-\alpha)\delta_t + \alpha\bar{q}$, where $\bar{q}$ is any solution to the problem with $1-\alpha = 0$, namely $\bar{q} \in argmax_{q\in\Delta_k} q^T g$

Proof. Consider the invertible change of variable $q = h(\bar{q}) := (1-\alpha)\delta_1 + \alpha\bar{q}$ which maps the simplex Δ_k unto itself, with inverse $\bar{q} = h^{-1}(x) = \alpha^{-1}(x - (1-\alpha)\delta_1)$.

It follows, that

$$\begin{aligned}\min_{q\in\Delta_k | q_1\geq 1-\alpha} q^T b &= \min_{\bar{q}\in\Delta_k | \bar{q}_1\geq 0} ((1-\alpha)\delta_1 + \alpha\bar{q})^T b \\ &= \min_{\bar{q}\in\Delta_k} ((1-\alpha)\delta_1 + \alpha\bar{q})^T b\end{aligned}$$

which is attained by

$$\bar{q}^* \in \text{argmin}_{\bar{q}\in\Delta_k}\bar{q}^T b = \text{ConvHull}(\text{argmin}_{j=1}^k b_j),$$

yielding $q^* = (1-\alpha)\delta_1 + \alpha\bar{q}^*$. □

1.3 Proof of Lemma 1

For a general covariance matrix in the fading Gaussian model, the best possible adversarial robustness accuracy is

$$\max_{w\in\mathbb{R}^d} \text{acc}_\varepsilon(f_w) = \Psi\left(\frac{w^T\mu - \varepsilon||w||_1}{\|w\|_\Sigma}\right),$$

where $\|w\|_\Sigma := \sqrt{w^T\Sigma w}$. Since the Gaussian CDF Φ is an increasing function, and the objective function in the above problem is 1-homogeneous in w, we are led to consider problems of the form

$$\alpha \quad := \min_{\|w\|_\Sigma\leq 1} \varepsilon\|w\|_1 - w^T a, \tag{13}$$

where $a \in \mathbb{R}^d$ and Σ be a positive definite matrix of size n. Of course, the solution value might not be analytically expressible in general, but there is some hope, when the matrix Σ is diagonal. That notwithstanding, using the dual representation of the ℓ_1-norm, one has

$$\begin{aligned}\alpha &= \min_{\|w\|_\Sigma\leq 1}\max_{\|z\|_\infty\leq\varepsilon} z^T w - w^T a \\ &= \max_{\|z\|_\infty\leq\varepsilon}\min_{\|w\|_\Sigma\leq 1} w^T(z-a) \\ &= \max_{\|z\|_\infty\leq\varepsilon} -\left(\max_{\|w\|_\Sigma\leq 1} -w^T(z-a)\right) \\ &= \max_{\|z\|_\infty\leq\varepsilon} -\left(\max_{\|\tilde{w}\|_2\leq 1} -\tilde{w}^T\Sigma^{-1}(z-a)\right) \\ &= \max_{\|z\|_\infty\leq\varepsilon} -\|z-a\|_{\Sigma^{-1}} = -\min_{\|z\|_\infty\leq\varepsilon}\|z-a\|_{\Sigma^{-1}},\end{aligned} \tag{14}$$

where we have used *Sion's minimax theorem* to interchange min and max in the first line, and we have introduced the auxiliary variable $\tilde{w} := \Sigma^{-1/2}w$ in the fourth line. We note that given a value for the dual variable z, the optimal value of the primal variable w is

$$w \propto \frac{\Sigma^{-1}(a-z)}{\|\Sigma^{-1}(a-z)\|_2} \tag{15}$$

The above expression (14) for the optimal objective value α is unlikely to be computable analytically in general, due to the non-separability of the objective (even though the constraint is perfectly separable as a product of 1D constraints). In any case, it follows from the above display that $\alpha \leq 0$, with equality iff $\|a\|_\infty \leq \varepsilon$. □

Exact formula for diagonal Σ. In the special case where $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_2)$, the square of the optimal objective value α^2 can be separated as

$$\begin{aligned}\alpha \leq 0,\ \alpha^2 &= \sum_{i=1}^{d}\min_{|z_i|\leq\varepsilon}\sigma_i^{-2}(z_i - a_i)^2 \\ &= \sum_{i=1}^{d}\sigma_i^{-2}\begin{cases}(a_i+\varepsilon)^2, & \text{if } a_i \leq -\varepsilon, \\ 0, & \text{if } -\varepsilon < a_i \leq \varepsilon, \\ (a_i-\varepsilon)^2, & \text{if } a_i > \varepsilon,\end{cases} \\ &= \sum_{j=1}^{d}((|a_j| - \varepsilon)_+)^2,\end{aligned}$$

which is indeed an analytical formula, albeit a very "hairy" one.

By the ways, the optimium is attained at

$$z_i = \begin{cases} -\varepsilon, & \text{if } a_i \leq -\varepsilon, \\ a_i, & \text{if } -\varepsilon < a_i \leq \varepsilon, \\ \varepsilon, & \text{if } a_i > \varepsilon, \end{cases} \tag{16}$$
$$= a_i - \text{sign}(a_i)(|a_i| - \varepsilon)_+$$

Plugging this into (15) yields the optimal weights

$$w_i \propto \sigma_j^{-2} \text{sign}(a_j)(|a_j| - \varepsilon)_+. \tag{17}$$

Upper and lower bounds for general Σ. Let $\sigma_d \geq \ldots \geq \sigma_2 \geq \sigma_1 > 0$ be the eigenvalues of Σ. Then, one has

$$\sigma_d^{-1}\|w - a\|_2 \leq \|w - a\|_{\Sigma^{-1}} \leq \sigma_1^{-1}\|w - a\|_2.$$

Thus one has the bounds $-\sqrt{\gamma/\sigma_1} \leq \alpha \leq -\sqrt{\gamma/\sigma_d}$, where $\gamma := \sum_{i=1}^d \begin{cases} (a_i + \varepsilon)^2, & \text{if } a_i \leq -\varepsilon, \\ 0, & \text{if } -\varepsilon < a_i \leq \varepsilon, \\ (a_i - \varepsilon)^2, & \text{if } a_i > \varepsilon. \end{cases}$

Moreover, if $\|a\|_\infty > \varepsilon$, then it isn't hard to see that $\gamma \leq d(\|a\|_\infty - \varepsilon)$ (see details below), from where $\alpha \geq -\sqrt{\gamma/\sigma_1} \geq -\sqrt{d/\sigma_1}(\|a\|_\infty - 1)$, which corresponds to a bound which has been observed by someone in the comments. However, by construction, this bound is potentially very loose.

Note that

$$a_i \leq -\varepsilon \implies (a_i + \varepsilon)^2 \leq (\|a\|_\infty - \varepsilon)^2,$$

and similarly

$$a_1 > 1 \implies 0 \leq (a_i - \varepsilon)^2 \leq (\|a\|_\infty - \varepsilon)^2.$$

Thus $\gamma \leq d(\|a\|_\infty - \varepsilon)$. □

1.4 Experiments: numerical results

The following tables show the adversarial accuracy for different model set-ups, defenses and attacks. In each table, we have the adversarial accuracies for one attack (or the standard accuracies in Table 6). Accuracies for LS-regularized models are presented for three different choices of $\alpha : 0.005, 0.1$ and 0.4. For FGSM and BIM (Tables 2 and 3), we chose 3 different values of the attack strength $\epsilon : 0.05, 0.2$ and 0.4. For example, the adversarial accuracy against FGSM attack with $\epsilon = 0.2$ for the BLS-regularized model with $\alpha = 0.005$ using MNIST LeNet set-up is shown in Table 2 and is equal to 0.838.

Moreover, we highlighted in color the best accuracy for a set-up and a particular attack (or attack and strength in the case of FGSM and BIM). Each set-up corresponds to one color (e.g. light yellow for MNIST Linear and red for SVHN LeNet). If the best accuracy is less than the accuracy obtained with random predictions (i.e. 0.1 in all our set-ups), it is not highlighted. For example, the best accuracy against FGSM of strenght $\epsilon = 0.05$ in the CIFAR LeNet set-up is equal to 0.160 and is obtained by both a ALS and BLS-regularized NN with $\alpha = 0.1$. Overall, adversarial training is better on FGSM (more colors on the adversarial training lines in Table 2 compared to other defenses), but ALS and BLS are better on other attacks. SBLS is better only against C&W. In Table 6, we see that ALS, BLS and SLS NNs are always better or equivalent to a normal classifier (no regularization, no defense method) in terms of standard accuracy.

Table 2: FGSM

		$\epsilon = 0.05$			$\epsilon = 0.2$			$\epsilon = 0.4$		
	α val.	0.005	0.1	0.4	0.005	0.1	0.4	0.005	0.1	0.4
ALS	MNIST Linear	0.832	0.870	0.857	0.507	0.526	0.489	0.450	0.465	0.432
	MNIST LeNet	0.957	0.959	0.954	0.847	0.732	0.639	0.822	0.561	0.130
	CIFAR LeNet	0.098	0.160	0.002	0.069	**0.147**	0.002	0.067	**0.151**	0.001
	CIFAR ResNet	0.245	0.365	0.433	0.099	0.125	0.123	/	/	/
	SVHN LeNet	0.450	0.458	**0.497**	0.370	0.302	**0.381**	/	/	/
SLS	MNIST Linear	0.818	0.848	0.840	0.532	0.470	0.412	**0.506**	0.437	0.384
	MNIST LeNet	0.956	0.956	0.954	0.840	0.764	0.656	**0.823**	0.699	0.229
	CIFAR LeNet	0.119	0.153	0.127	0.097	0.134	0.101	0.092	0.136	0.093
	CIFAR ResNet	0.254	0.345	0.389	0.098	0.115	0.116	/	/	/
	SVHN LeNet	0.404	0.472	0.395	0.353	0.312	0.195	/	/	/
BLS	MNIST Linear	0.821	0.868	0.858	0.494	0.525	0.494	0.442	0.457	0.441
	MNIST LeNet	0.956	0.958	0.954	0.838	0.741	0.642	0.812	0.616	0.131
	CIFAR LeNet	0.085	0.160	0.122	0.055	0.141	0.107	0.055	0.130	0.114
	CIFAR ResNet	0.252	0.346	**0.450**	0.096	0.129	**0.138**	/	/	/
	SVHN LeNet	0.430	0.442	0.327	0.352	0.293	0.153	/	/	/
SBLS	MNIST Linear	0.763	0.804	0.639	0.530	0.353	0.327	0.431	0.212	0.186
	MNIST LeNet	0.955	0.956	0.929	0.855	0.695	0.491	**0.836**	0.279	0.141
	CIFAR LeNet	0.103	0.143	0.136	0.061	0.098	0.068	0.058	0.085	0.053
	CIFAR ResNet	0.191	0.159	0.231	0.077	0.079	0.093	/	/	/
	SVHN LeNet	0.449	0.375	0.277	0.378	0.157	0.149	/	/	/
Normal classifier	MNIST Linear	0.791			0.003			0.000		
	MNIST LeNet	0.956			0.498			0.022		
	CIFAR LeNet	0.049			0.002			0.009		
	CIFAR ResNet	0.090			0.052			/		
	SVHN LeNet	0.195			0.006			/		
Adv. training	MNIST Linear	**0.970**			**0.915**			0.286		
	MNIST LeNet	**0.975**			**0.952**			**0.823**		
	CIFAR LeNet	**0.201**			0.023			0.011		
	CIFAR ResNet	/			/			/		
	SVHN LeNet	0.475			0.071			0.075		

Table 3: BIM

		$\epsilon = 0.05$			$\epsilon = 0.2$			$\epsilon = 0.4$		
	α val.	0.005	0.1	0.4	0.005	0.1	0.4	0.005	0.1	0.4
ALS	MNIST Linear	0.816	0.854	0.845	0.429	0.485	0.456	0.420	0.472	0.435
	MNIST LeNet	0.947	0.946	0.945	0.813	0.614	0.606	0.811	0.560	0.484
	CIFAR LeNet	0.064	0.140	0.000	0.055	**0.137**	0.000	0.054	**0.136**	0.000
	CIFAR ResNet	0.102	0.167	0.248	0.046	0.076	0.074	/	/	/
	SVHN LeNet	0.403	0.408	**0.470**	0.386	0.324	**0.435**	/	/	/
SLS	MNIST Linear	0.801	0.829	0.826	0.495	0.447	0.398	0.492	0.439	0.391
	MNIST LeNet	0.946	0.942	0.942	0.817	0.700	0.398	0.815	0.689	0.165
	CIFAR LeNet	0.093	0.131	0.100	0.087	0.128	0.096	0.087	0.128	0.094
	CIFAR ResNet	0.135	0.142	0.230	0.051	0.035	0.046	/	/	/
	SVHN LeNet	0.371	0.421	0.322	0.362	0.345	0.193	/	/	/
BLS	MNIST Linear	0.803	0.853	0.841	0.534	0.481	0.458	**0.528**	0.470	0.443
	MNIST LeNet	0.946	0.944	0.947	0.835	0.645	0.532	**0.834**	0.603	0.389
	CIFAR LeNet	0.062	0.140	0.103	0.056	0.136	0.097	0.055	0.135	0.097
	CIFAR ResNet	0.153	0.142	**0.265**	0.060	0.043	0.075	/	/	/
	SVHN LeNet	0.435	0.393	0.266	0.424	0.322	0.142	/	/	/
SBLS	MNIST Linear	0.736	0.772	0.597	0.470	0.227	0.299	0.370	0.091	0.252
	MNIST LeNet	0.945	0.945	0.914	0.841	0.359	0.239	0.808	0.126	0.072
	CIFAR LeNet	0.074	0.114	0.090	0.057	0.077	0.036	0.055	0.068	0.028
	CIFAR ResNet	0.054	0.021	0.045	0.010	0.008	0.018	/	/	/
	SVHN LeNet	0.423	0.265	0.200	0.383	0.073	0.116	/	/	/
Normal classifier	MNIST Linear	0.776			0.001			0.000		
	MNIST LeNet	0.946			0.114			0.000		
	CIFAR LeNet	0.015			0.000			0.000		
	CIFAR ResNet	0.003			0.000			/		
	SVHN LeNet	0.117			0.000			/		
Adv. training	MNIST Linear	**0.970**			**0.899**			0.288		
	MNIST LeNet	**0.974**			**0.940**			0.498		
	CIFAR LeNet	**0.168**			0.003			0.000		
	CIFAR ResNet	/			/			/		
	SVHN LeNet	0.419			0.014			0.000		

Table 4: DeepFool

		Adv. acc.		
	α val.	0.005	0.1	0.4
ALS	MNIST Linear	0.037	0.092	0.341
	MNIST LeNet	0.047	0.299	0.529
	CIFAR LeNet	0.005	0.006	0.001
	CIFAR ResNet	0.040	0.030	0.33
	SVHN LeNet	0.020	0.010	0.008
SLS	MNIST Linear	0.036	0.080	**0.683**
	MNIST LeNet	0.035	0.183	0.380
	CIFAR LeNet	0.006	0.007	0.008
	CIFAR ResNet	0.073	0.049	0.057
	SVHN LeNet	0.020	0.007	0.007
BLS	MNIST Linear	0.033	0.088	0.414
	MNIST LeNet	0.031	0.314	0.500
	CIFAR LeNet	0.007	0.008	0.008
	CIFAR ResNet	0.051	0.059	0.032
	SVHN LeNet	0.024	0.006	0.006
SBLS	MNIST Linear	0.035	0.152	0.516
	MNIST LeNet	0.028	0.142	0.305
	CIFAR LeNet	0.005	0.006	0.006
	CIFAR ResNet	0.050	0.037	0.064
	SVHN LeNet	0.024	0.014	0.038
Normal classifier	MNIST Linear	0.025		
	MNIST LeNet	0.050		
	CIFAR LeNet	0.009		
	CIFAR ResNet	0.036		
	SVHN LeNet	0.031		
Adv. training	MNIST Linear	0.07		
	MNIST LeNet	**0.571**		
	CIFAR LeNet	0.008		
	CIFAR ResNet	/		
	SVHN LeNet	0.030		

Table 5: CW

		Adv. acc.		
	α val.	0.005	0.1	0.4
ALS	MNIST Linear	0.014	0.056	0.073
	MNIST LeNet	0.020	0.042	0.084
	CIFAR LeNet	0.000	0.002	0.000
	CIFAR ResNet	0.022	0.098	0.205
	SVHN LeNet	0.010	0.039	0.056
SLS	MNIST Linear	0.013	0.025	0.031
	MNIST LeNet	0.022	0.046	0.058
	CIFAR LeNet	0.000	0.004	0.000
	CIFAR ResNet	0.029	0.087	0.159
	SVHN LeNet	0.006	0.044	0.044
BLS	MNIST Linear	0.014	0.035	0.059
	MNIST LeNet	0.018	0.046	0.062
	CIFAR LeNet	0.000	0.004	0.000
	CIFAR ResNet	0.019	0.089	0.231
	SVHN LeNet	0.006	0.028	0.045
SBLS	MNIST Linear	0.017	0.071	0.007
	MNIST LeNet	0.016	0.128	**0.474**
	CIFAR LeNet	0.002	0.000	0.014
	CIFAR ResNet	0.063	**0.173**	0.106
	SVHN LeNet	0.009	0.062	0.052
Normal classifier	MNIST Linear	0.015		
	MNIST LeNet	0.026		
	CIFAR LeNet	0.000		
	CIFAR ResNet	0.015		
	SVHN LeNet	0.031		
Adv. training	MNIST Linear	0.108		
	MNIST LeNet	0.442		
	CIFAR LeNet	0.000		
	CIFAR ResNet	/		
	SVHN LeNet	0.000		

Table 6: Std. accuracies

		Std. acc.		
	α val.	0.005	0.1	0.4
ALS	MNIST Linear	0.979	**0.981**	0.976
	MNIST LeNet	**0.990**	**0.990**	0.989
	CIFAR LeNet	0.623	**0.664**	0.148
	CIFAR ResNet	0.887	0.890	0.889
	SVHN LeNet	0.890	**0.894**	0.879
SLS	MNIST Linear	0.978	**0.981**	0.975
	MNIST LeNet	0.989	**0.990**	0.986
	CIFAR LeNet	0.628	0.638	0.643
	CIFAR ResNet	0.885	0.894	0.895
	SVHN LeNet	0.892	**0.894**	0.889
BLS	MNIST Linear	0.978	**0.981**	0.976
	MNIST LeNet	0.989	**0.990**	0.989
	CIFAR LeNet	0.639	0.651	0.622
	CIFAR ResNet	0.888	0.889	**0.897**
	SVHN LeNet	0.891	0.890	0.841
SBLS	MNIST Linear	0.977	0.977	0.949
	MNIST LeNet	0.989	0.988	0.975
	CIFAR LeNet	0.628	0.619	0.572
	CIFAR ResNet	0.883	0.881	0.840
	SVHN LeNet	0.886	0.883	0.816
Normal classifier	MNIST Linear	0.980		
	MNIST LeNet	**0.990**		
	CIFAR LeNet	0.635		
	CIFAR ResNet	0.886		
	SVHN LeNet	0.884		
Adv. training	MNIST Linear	**0.981**		
	MNIST LeNet	0.982		
	CIFAR LeNet	0.505		
	CIFAR ResNet	/		
	SVHN LeNet	0.831		