Stability for the Training of Deep Neural Networks and Other Classifiers

Leonid Berlyand, Pierre-Emmanuel Jabin, C. Alex Safsten

February 10, 2020

Abstract

We examine the stability of loss-minimizing training processes that are used for deep neural network (DNN) and other classifiers. While a classifier is optimized during training through a so-called *loss function*, the performance of classifiers is usually evaluated by some measure of accuracy, such as the overall accuracy which quantifies the proportion of objects that are well classified. This leads to the guiding question of stability: does decreasing loss through training always result in increased accuracy? We formalize the notion of stability, and provide examples of instability. Our main result is two novel conditions on the classifier which, if either is satisfied, ensure stability of training, that is we derive tight bounds on accuracy as loss decreases. These conditions are explicitly verifiable in practice on a given dataset. Our results do not depend on the algorithm used for training, as long as loss decreases with training.

1 Introduction

Our purpose in the present article is to provide rigorous justifications to the basic training method commonly used in learning algorithms. We particularly focus on classification problems and to better explain our aim, it is useful to introduce some basic notations.

We are given a set of objects $S \subset \mathbb{R}^n$ whose elements are classified in a certain number K of classes. Then we introduce a function called the *exact classifier* that maps each $s \in S$ to the index $i(s)$ of its class. However, the exact classifier is typically only known on a *finite* subset T of S called the *training set*.

In practice, objects in S are classified by an *approximate classifier*, and the mathematical problem is to identify an optimal approximate classifier among a large set of potential candidates. The optimal approximate classifier should agree (or, at least nearly agree) with the exact classifier on T. We consider here a type of approximate classifier called *soft approximate classifiers* which may again be described in a general setting as functions

$$\phi : s \in T \longrightarrow \phi(s) = (p_1(s), \dots, p_K(s)) \in [0,\ 1]^K, \text{ with } p_1(s) + \dots + p_K(s) = 1. \tag{1}$$

Such a classifier is often interpreted as giving the probabilities that s belongs to a given class: $p_i(s)$ can be described as the predicted probability that s is in class $\#i$. In this framework a perfect classifier on T is a function ϕ s.t. $p_i(s) = 1$ if and only if $i = i(s)$ (and hence $p_i(s) = 0$ if $i \neq i(s)$).

In practice of course, one cannot optimize among all possible functions ϕ and instead some sort of discretization is performed which generates *a parametrized family* $\phi(s, \alpha)$ where the parameters α can be chosen in some other large dimensional space, $\alpha \in \mathbb{R}^\mu$ with $\mu \gg 1$ for instance. Both n and μ are typically very large numbers, and the relation between them plays a crucial role in the use of classifiers for practical problems.

Deep neural networks (DNNs) are of course one of the most popular examples of method to construct such parametrize family $\phi(s, \alpha)$. In those settings $\phi(s, \alpha)$ is obtained by applying a sequence of linear and non-linear operations on the initial object s, one linear and non-linear operation per layer in the network. In this setting, the parameters α are entries of matrices used for the linear operations on each layer.

Such deep neural networks have demonstrated their effectiveness on a variety of clustering and classifying problems, such as the well-known [14] for handwriting recognition. To just give a few examples, one can refer to [12] for image classification, [11] on speech recognition, [15] for an example of applications to the life sciences, [18] on natural language understanding, or to [13] for a general presentation of deep neural networks.

The training process consists of optimizing in α the family $\phi(s, \alpha)$ to obtain the "best" choice. This naturally leads to the question of what is meant by best, which is at the heart of our investigations in this article. Here, best means the highest performing classifier. There are indeed several ways to measure the performance of classifiers: one may first consider the overall *accuracy* which corresponds to the proportion of objects that are "well-classified." We say that s is *well-classified* by $\phi(\cdot, \alpha)$ if $p_{i(s)}(s, \alpha) > \max_{1 \leq i \leq K,\, i \neq i(s)} p_i(s, \alpha)$, that is $\phi(s, \alpha)$

gives the highest probability to the class to which s actually belongs. Because this set will come up often in our analysis, we introduce a specific notation for this "good" set

$$G(\alpha) = \{s \in T \,:\, p_{i(s)}(s,\alpha) > \max_{1\leq i\leq K,\, i\neq i(s)} p_i(s,\alpha)\}, \tag{2}$$

which leads to the classical definition of overall accuracy

$$acc(\alpha) = \frac{\#G(\alpha)}{\#T}. \tag{3}$$

For simplicity in this introduction, we assign the same weight to each object in the training set T. In practice, objects in T are often assigned different weights, which will be introduced below.

Other measures of performance exist and are commonly used: average accuracy where the average is taken within each class and which gives more weights to small classes, Cohen κ's coefficient [5], etc....

However, in practice training algorithms do not optimize accuracy (whether overall accuracy or some other definition) but instead try to minimize some *loss function*. There are compelling reasons for not using the accuracy: for example, accuracy distinguishes between well-classified and not well-classified in a binary manner. That is, accuracy does not account for an object close to being well classified. Moreover, accuracy is a piecewise-constant function (taking values $0, 1/\#T, 2/\#T, \cdots, 1$) so that its gradient is zero. For these reasons, it cannot be maximized with gradient descent methods. An approximation of this piecewise constant function is computationally intensive, particularly due the high dimensionality μ of its domain.

A typical example of a loss function is the so-called cross-entropy loss:

$$\bar{L}(\alpha) = -\frac{1}{\#T}\sum_{s\in T} \log\left(p_{i(s)}(s,\alpha)\right), \tag{4}$$

which is simply the average over the training set of the functions $-\log p_{i(s)}(s,\alpha)$. Obviously each of these functions is minimized if $p_{i(s)}(s,\alpha) = 1$, *i.e.* if the classifier worked perfectly on the object s.

Minimizing (4) simply leads to finding the parameters α such that on average $p_{i(s)}$ is as large (as close to 1) as possible. Since $\bar{L}(\alpha)$ is smooth in α (at least if ϕ is), one may apply classical gradient algorithms, with stochastic gradient descent (SGD) being among the most popular due to the linear structure of $\bar{L}$ and the large size of many training sets. Indeed, since (4) consists of many terms, computing the gradient of $\bar{L}(\alpha)$ is expensive. SGD simplifies this task by randomly selecting a batch of a few terms at each iteration of the descent algorithm, and computing the gradient of only those terms. This optimization process is referred to as *training*. Though in general loss decreases through training, it need not be monotone because of e.g. effects due to stochasticity. In this work, we assume for simplicity that loss decreases monotonically, which is approximately true in most practical problems.

The main question that we aim to answer in this article is *why should decreasing the loss function improve accuracy*. We start by pointing out the following observations which explain why the answer is not straightforward.

- If the loss function $\bar{L}$ converges to 0, then the accuracy converges to 100% as in that case all predicted probability $p_{i(s)}$ converge to 1. Nevertheless the training process is necessarily stopped at some time, before $\bar{L}$ reaches exactly 0, see e.g. [2]. A first question is therefore *how close to* 100% *the accuracy is when the loss function is very small*.

- In practice, we may not be able to reach perfect accuracy (or 0 loss) on every training set. This can be due to the large dimension, of the objects s or of the space of parameters α, which makes it difficult to computationally find a perfect minimizer even if one exists, with the usual issue of local minimizers. Moreover, there may not even exist a perfect minimizer, due for instance to classification errors on the training set (some objects may have been assigned to the wrong class). As a consequence, a purely asymptotic comparison between loss and accuracy as $\bar{L} \to 0$ is not enough, and we need to ask *how the loss function $\bar{L}$ correlates with the accuracy away from $\bar{L} \approx 0$*.

- *In general, there is no reason why decreasing $\bar{L}$ would increase the accuracy*, which is illustrated by the following elementary counter-example. Consider a setting with 3 classes and an object s which belongs to the first class and such that for the initial choice of parameter α: $p_1(s,\alpha) = 0.4,\ p_2(s,\alpha) = p_3(s,\alpha) = 0.3$. We may be given a next choice of parameter α' such that: $p_1(\alpha', s) = 0.45,\ p_2(\alpha', s) = 0.5, p_3(\alpha', s) = 0.05$. Then the object s is well classified by the first choice of α and it is not well classified by the second choice α'. Yet the loss function, which is simply $-\log p_1$ here, is obviously lower for α'. Thus, while loss improves, accuracy worsens.

- *The previous example raises the key issue of stability during training, which roughly speaking means that accuracy increases with training.* Indeed, one does not expect accuracy to increase monotonically during training, and the question becomes what conditions would guarantee that accuracy increases during training? For example, one could require that that the good set G monotonically grows during training; in mathematical terms, that would mean that $G(\alpha) \subset G(\alpha')$ if α' are parameters from a later stage of the training. However, such a condition would be too rigid and likely counterproductive by preventing the training algorithms from reaching better classifiers. At the same time, *wild fluctuations in accuracy or in the good set $G(\alpha)$ would destroy any realistic hope of a successful training process, i.e., finding a high-accuracy classifier.*

- While the *focus of this work is on the stability during training,* another crucial question is the *robustness* of the trained classifier, which is the stability of identifying classes with respect to small perturbations of objects $s \in S$, in particular $s \in T$. The issues of robustness and stability are connected. Lack of stability during the training can often lead to over-parametrization by extending the training process for too long. In turn over-parametrization typically implies poor robustness outside of the training set. This is connected to the Lipschitz norm of the classifier and we refer, for example, to [1].

- Since the marker of progress during training is the decrease of the loss function, stability is directly connected to how the loss function correlates with the accuracy. Per the known counterexamples, such correlation cannot always exist. Thus the *key question is to be able to identify which features of the dataset and of the classifier are critical to establish such correlations and therefore ensure stability.*

Our main contributions are to bring rigorous mathematical answers to this last question, in the context of simple deep learning algorithms with the very popular SGD algorithm. While part of our approach would naturally extend to other settings, it is intrinsically dependent on the approach used to construct the classifier, which is described in section 2.1. More specifically, we proceed in the following two steps.

i. We first identify conditions on the distribution of probabilities $(p_1(s,\alpha), \cdots, p_K(s,\alpha))$ defined in (1) for each $s \in T$ which guarantee that $\bar{L}$ correlates with accuracy. Specifically, we show that under these conditions, loss is controlled by accuracy (vice-versa is trivial). At this stage such conditions necessarily mix, in a non-trivial manner, the statistical properties of the dataset with the properties of the neural network (its architecture and parameters), introduced in section 2.1. Since these conditions depend on the network parameters which evolve with training, they cannot be verified before training starts, and they may depend on how the training proceeds.

ii. The second step is to disentangle the previous conditions to obtain separate conditions on the training set and on the neural network architecture and parameters. We are able to accomplish this on one of the conditions obtained in step i which is well suited to the combination of linear operations and activation function on the layer, defined in section 2.1. The main idea here is to be able to propagate backward on the neural network the required distribution of $(\phi(s,\alpha))_{s\in T}$, see Remark 3.1.

Our hope is that the present approach and results will help develop a better understanding of why learning algorithms perform so well in many cases but still fail in other settings. This is achieved by providing a framework to evaluate the suitability of training sets and of neural network construction for solving various classification problems. Rigorous analysis of neural networks has of course already started and several approaches that are different from the present one have been introduced. We mention in particular the analysis of neural nets in terms of multiscale contractions involving wavelets and scattering transforms; see for example [4, 16, 17] and [7] for scattering transforms. While there are a multitude of recent papers aimed to make neural net-based algorithms (also known as deep learning algorithms) faster, our goal is to help make such algorithms more stable.

We conclude by summarizing the practical outcomes of our work:

- First, we derive and justify an explicitly verifiable conditions on the dataset that guarantee stability. We refer in particular to subsection 2.5 for a discussion of how to check our conditions in practice.

- Our analysis characterizes how the distribution of objects in the training set and the distribution of the output of the classifier for misclassified objects affect stability of training.

- Finally, among many possible future directions of research, our results suggest that the introduction of *multiscale* loss functions could significantly improve stability.

Acknowledgments. The work of L. Berlyand and C. A. Safsten was supported by NSF DMREF DMS-1628411, and the Work of P.-E. Jabin was partially supported by NSF DMS Grant 161453, 1908739, NSF Grant RNMS (Ki-Net) 1107444, and LTS grant DO 14. The authors thank R. Creese and M. Potomkin for useful discussions.

2 Main results

2.1 Mathematical formulation of deep neural networks and stability

2.2 Classifiers

A parameterized family of soft classifiers $\phi(\cdot,\alpha):\mathbb{R}^n\to[0,1]^K$ must map objects s to a list of K probabilities. To accomplish this, a classifier is a composition $\phi(\cdot,\alpha)=\rho\circ X(\cdot,\alpha)$, where $X(\cdot,\alpha):\mathbb{R}^K\to\mathbb{R}^K$ and ρ is the so-called *softmax function* defined by

$$\rho(x)=\rho(x_1,\cdots,x_K)=\left(\frac{e^{x_1}}{\sum_{k=1}^K e^{x_k}},\cdots,\frac{e^{x_K}}{\sum_{k=1}^K e^{x_k}}\right). \tag{5}$$

Clearly, $\rho(x)\in[0,1]^K$ and $\sum_{i=1}^K\rho_i(x)=1$, so $\rho\circ X(\cdot,\alpha)$ is a soft classifier no matter what what function $X(\cdot,\alpha)$ is used (though typically $X(\cdot,\alpha)$ is differentiable almost everywhere). The form of the softmax function means that we can write the classifier as

$$\phi(s,\alpha)=\rho\circ X(s,\alpha)=\left(\frac{e^{X_1(s,\alpha)}}{\sum_{k=1}^K e^{X_k(s,\alpha)}},\cdots,\frac{e^{X_K}}{\sum_{k=1}^{n_K} e^{X_k(s,\alpha)}}\right). \tag{6}$$

As in (1), denote $\phi(s,\alpha)=(p_1(s,\alpha),\cdots,p_K(s,\alpha))$, where $p_k(s,\alpha)$ is the probability that s belongs to class k predicted by a classifier with parameters α. A key property of the softmax function is that it is order preserving in the sense that if $X_i(s,\alpha)>\max_{j\neq i}X_j(s,\alpha)$, then $p_i(s)>\max_{j\neq i}p_j(s,\alpha)$. Therefore, the predicted class of s can be determined by $X(s,\alpha)$. We define the key evaluation that determines whether an object is well-classified or not, namely

$$\delta X(s,\alpha)=X_{i(s)}(s,\alpha)-\max_{j\neq i(s)}X_j(s,\alpha). \tag{7}$$

If $\delta X(s,\alpha)>0$, then $X_{i(s)}(s,\alpha)$ is the largest component of $X(s,\alpha)$, which means that $p_{i(s)}(s,\alpha)$ is the largest probability given by $\phi(s,\alpha)$, and thus, s is classified correctly. Similarly, if $\delta X(s,\alpha)<0$, s is classified incorrectly.

As described in Section 1, a classifier learns to solve the classification problem by training on a finite set T where the correct classifications are known. Training is completed by minimizing a loss function which measures how far the classifier is from the exact classifier on T. While there are many types of loss functions, cross entropy loss introduced in (4) is very common, and it is the loss function we will consider in this work. The loss in (4) is the simple average of $-\log(p_{i(s)}(s,\alpha))$ over all $s\in T$, but there is no reason we cannot use the weighted average:

$$\bar{L}(\alpha)=-\sum_{s\in T}\nu(s)\log\left(p_{i(s)}(s,\alpha)\right), \tag{8}$$

where $0<\nu(s)\leq 1$ and $\sum_{s\in T}\nu(s)=1$. Weights could be uniform, i.e., $\nu(s)=1/\#T$ for all $s\in T$, or weights can be non uniform if e.g. some $s\in T$ are more important than others. We can also use ν to measure the size of subset of T, e.g., if $A\subset T$, $\nu(A)=\sum_{s\in A}\nu(s)$. The quantity $\delta X(s,\alpha)$ defined above facilitates some convenient estimates on loss, which are shown in section 3.1.

2.2.1 Deep neural network structure

Deep neural networks (DNNs) are a diverse set of algorithms with the classification problem being just one of many of their applications. In this article, however, we will restrict our attention to DNN classifiers. DNNs provide a useful parameterized family $X(\cdot,\alpha):\mathbb{R}^n\to\mathbb{R}^K$ which can be composed with the softmax function to form a classifier. The function $X(\cdot,\alpha)$ is a composition of several simpler functions:

$$X(\cdot,\alpha)=f_M(\cdot,\alpha_M)\circ f_{M-1}(\cdot,\alpha_{M-1})\circ\cdots\circ f_1(\cdot,\alpha_1).$$

Each f_k for $1\leq k\leq M$ is a composition of an affine transformation and a nonlinear function. The nonlinear function is called an *activation function*. A typical example is the so-called *rectified linear unit* (ReLU), which is defined for any integer $N\geq 0$ by $\mathrm{ReLU}(x_1,\cdots,x_N)=(\max\{0,x_1\},\cdots,\max\{0,x_N\})$. Another example is the componentwise absolute value, $\mathrm{abs}(x_1,\cdots,x_N)=(|x_1|,\cdots,|x_N|)$. The affine transformation depends on many parameters (e.g., matrix elements) which are denoted together as α_k. The collection of all DNN parameters is denoted $\alpha=(\alpha_1,\cdots,\alpha_{M-1})$.

Though we use DNN classifiers as a guiding example for this article, most results apply to classifiers of the form $\phi(\cdot,\alpha)=\rho\circ X(\cdot,\alpha)$ where ρ is the softmax function, and X is any family of functions parameterized by α. In this article, we will use the term *classifier* to refer to any composition $f_M\circ X(\circ,\alpha)$, while *DNN classifier* refers to a classifier where X has the structure of a DNN.

2.2.2 Training, Accuracy, and Stability

Training a DNN is the process of minimizing loss. In practice, one randomly selects a starting parameter $\alpha(0)$, and then uses an iterative minimization algorithm such as gradient descent or stochastic gradient descent to find a minimizing α. Whatever algorithm is used, the n^{th} iteration calculates $\alpha(n)$ using $\alpha(t)$ for $0 \leq t \leq n-1$. Our results do not depend on which algorithm is used for training, but will make the essential assumption that loss decreases with training, $\bar{L}(\alpha(t_2)) \leq \bar{L}(\alpha(t_2))$ for $t_2 > t_1$. Throughout this article, we will abuse notation slightly by writing $\bar{L}(t) := \bar{L}(\alpha(t))$.

Accuracy is simply the proportion of well-classified elements of the training set. Using δX and the weights $\nu(s)$, we can define a function that measures accuracy for all times t during training:

$$\text{acc}(t) = \nu\left(\{s \in T : \delta X(s, \alpha(t)) > 0\}\right). \tag{9}$$

We will find it useful to generalize the notion of accuracy. For instance, we may want to know how many $s \in T$ are not only well-classified, but are well-classified by some margin $\eta \geq 0$. We therefore define the *good set of margin* η as

$$G_\eta(t) = \{s \in T : \delta X(s, \alpha(t)) > \eta\}. \tag{10}$$

Observe that $\nu(G_0(t)) = \text{acc}(t)$. For large η, the good set comprises those elements of T that are exceptionally well-classified by the DNN with parameter values $\alpha(t)$. We will also consider the *bad set of margin* η

$$B_{-\eta}(t) = \{s \in T : \delta X(s, \alpha(t)) \leq -\eta\} \tag{11}$$

which are the elements that are misclassified with a margin of η by the DNN with parameters $\alpha(t)$.

Stability is the idea that when, during training, accuracy becomes high enough, it remains high for all later times. Specifically, we will prove that under certain conditions, for all ϵ, there exists δ and η so that if at some time t_0, $\nu(G_\eta(t_0)) > 1-\delta$, then at all later times, $\text{acc}(t) > 1-\epsilon$.

2.3 Preliminary remarks and examples

2.3.1 Relationship between accuracy and loss

Intuitively, accuracy and loss should be connected, i.e., as loss decreases, accuracy increases and vice versa. However, as we will see in examples below, this is not necessarily the case. Nevertheless, we can derive some elementary relations between the two. For instance, from equation (30), we may easily derive a bound on the good set $G_\eta(t)$ via $\bar{L}(t_0)$ for some η for all times $t \geq t_0$:

$$\nu(G_\eta(t)) \geq 1 - \frac{\bar{L}(t_0)}{\log\left(1+e^{-\eta}\right)} \geq 1 - 2e^{\eta}\bar{L}(t_0), \tag{12}$$

and in particular,

$$\nu(\text{acc}(t)) = \nu(G_0(t)) \geq 1 - \frac{\bar{L}(t_0)}{\log 2}. \tag{13}$$

This shows if loss is sufficiently small at time t_0, then accuracy will be high for all later times. But this is not the same as stability; stability means that if *accuracy* is sufficiently high at time t_0, then it will remain high for all $t > t_0$. To obtain stability from (12), we somehow *need to guarantee that high accuracy at time* t_0 *implies low loss at time* t_0.

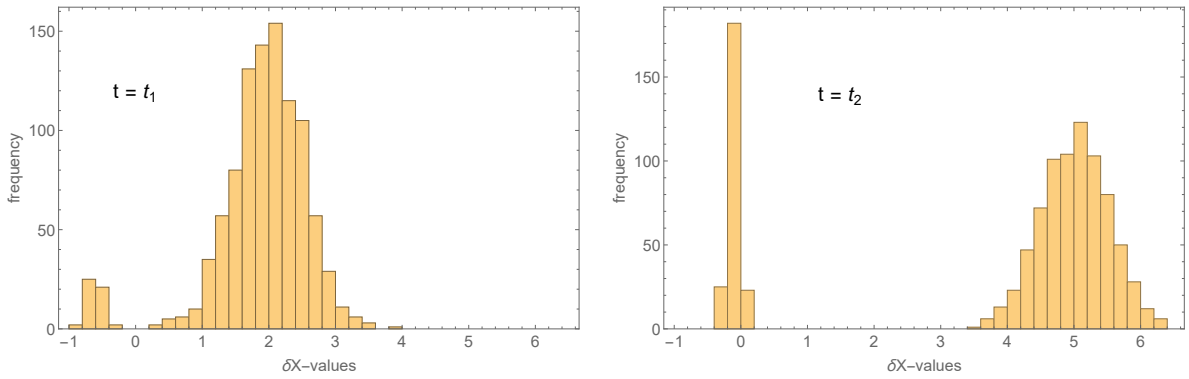
Example 2.1. This example will demonstrate instability in a soft classifier resulting from a small number of elements of the training set that are misclassified. Let T be a training set with 1000 elements with uniform weights, each classified into one of two classes. Suppose that at some time t_0, after some training, the parameters $\alpha(t_0)$ are such that most of the $\delta X(s, \alpha(t_0))$ values are positive, but a few $\delta X(s, \alpha(t_0))$ are clustered near -0.6. An example histogram of these $\delta X(s, \alpha(t_0))$ values is shown in Figure 1a. The loss $\bar{L}(t_0)$ accuracy can be calculated using (8) and (9) respectively. For the $\delta X(s, \alpha(t_0))$ values in Figure 1a, the loss and accuracy are is

$$\bar{L}(t_0) = 0.1845 \quad \text{acc}(t_0) = 0.95.$$

Suppose that at some later time $t = t_0$, the $\delta X(s, \alpha(t_0))$ values are those shown in Figure 1b. Most δX values have improved from $t = t_0$ to $t = t_1$, but a few have worsened. We can again calculate loss and accuracy:

$$\bar{L}(t_1) = 0.1772 \quad \text{acc}(t_1) = 0.798.$$

Since $\bar{L}(t_1) < \bar{L}(t_0)$, this example satisfies the condition that loss must decrease during training. However, accuracy has fallen considerably. This indicates an unstable classifier. The instability arises because enough objects have sufficiently poor classifications that by improving their classification (increasing $\delta X(s, \alpha)$), training can still decrease loss if a few correctly classified objects become misclassified, decreasing accuracy.



(a) Histogram of $\delta X(s, \alpha(t_0))$ values

(b) Histogram of $\delta X(s, \alpha(t_1))$ values

Figure 1: These two histograms of $\delta X(s, \alpha(t))$-values show that when $t = t_0$ (a), 95% of δX-values are positive, but when $t = t_1$, only 79.8% of δX values are positive, indicating a decrease in accuracy and therefore, this DNN is unstable.

2.4 Main Results

As explained above, to establish stability, we must bound $\bar{L}(t_0)$ in terms of the accuracy at t_0. Assuming that accuracy at t_0 is high, we may separate the training set into a large good set $G_\eta(t_0)$ for some $\eta > 0$ its small complement $G_\eta^C(t_0)$. Using (30), we may make the following estimate, details for which are found in Section 4.

$$\bar{L}(t_0) \leq (K-1)e^{-\eta} + \sum_{s \in G_\eta^C(t_0)} \nu(s) \log\left(1 + (K-1)e^{-\delta X(s,\alpha(t_0))}\right). \tag{14}$$

The first term in (14) is controlled by η. If η is even moderately large, then the second term dominates (14), with most of the loss coming from a few $s \in G_\eta^C(t_0)$. It is therefore sufficient to control the distribution of $\{\delta X(s, \alpha(t_0)) : s \in G_\eta^C(t_0)\}$. There are two primary reasons why this distribution may lead to large $\bar{L}(t_0)$, both relating to $\beta := \min_{s \in T} \delta X(s, \alpha(t_0))$:

1. There is a single $\delta X(s, \alpha(t_0))$ that is very large and negative, that is, $\beta \ll 0$.
2. β is not far from zero, but there are many $\delta X(s, \alpha(t_0))$ near β.

To obtain a good bound on $\bar{L}(t_0)$, we must address both issues:

1. Assume that $\delta X(s, \alpha(t_0)) > -1$ for all $s \in T$, i.e., the bad set $B_{-1}(t_0)$ is empty.
2. Impose a condition that prevents $\{\delta X(s, \alpha(t)) : s \in T\}$ form concentrating near β. Such conditions are called *small mass conditions* since they only concern $\delta X(s, \alpha(t_0))$ for s in the small mass of objects in $G_\eta^C(t_0)$. Example 2.1 illustrates this issue.

In the following subsections, we introduce two small mass conditions that lead to stability, and discuss the advantages of each. We will show that each condition leads to small loss $\bar{L}(t_0)$ and ultimately to stability.

2.4.1 First result

The first small mass condition we will consider ensures that the distribution of $\delta X(s, \alpha(t_0))$ values decays very quickly near β, the minimum $\delta X(s, \alpha(t_0))$ value, so there cannot be a concentration of δX near β, resulting in high accuracy at t_0 implying low loss at t_0. Additionally, by applying this condition at all times, not just at $t = t_0$, we can improve the estimate (12). How precisely this condition accomplishes these dual purposes is presented in Section 4.1.

Definition 2.1. The set $\{\delta X(s, \alpha(t)) : s \in T\}$ satisfies *condition A* at time t if there exist constants $\Lambda \geq 1$, $m_0 > 0$, $\psi > 0$, and $0 < \phi < 1$ so that for $x_1 = 0, \beta$ and all $x_2 > x_1$,

$$\begin{aligned} \nu\left(\left\{s \in T : \delta X(s, \alpha(t_0)) < \frac{x_1 + x_2}{2}\right\}\right) - m_0 \leq \Lambda \nu\left(\{s \in T : \delta X(s, \alpha(t_0)) < x_1\}\right)^\phi \\ + \Lambda \nu\left(\{s \in T : \delta X(s, \alpha(t_0)) < x_2\}\right)^{\psi+1} \end{aligned} \tag{15}$$

With condition A in hand, we can state the first stability result. Proofs and supporting lemmas are left for Section 3.

Theorem 2.1. *Suppose that $T \subset \mathbb{R}^n$ with weights $\nu(s)$ is a training set for a classifier such that $\{\delta X(s, \alpha(t)) : s \in T\}$ which satisfies condition A for some constants $\Lambda \geq 1$, $\psi = 1$, $0 < \phi < 1$,and $m_0 = 0$. for all $t \geq t_0$. Then for every $\varepsilon > 0$ there exist $\delta(\Lambda, \varepsilon), \eta(\Lambda, \varepsilon) > 0$ such that if good and bad sets at $t = t_0$ satisfy*

$$\nu(G_\eta(t_0)) > 1 - \delta \quad \textit{and} \quad B_{-1}(t_0) = \emptyset, \tag{16}$$

then for all $t \geq t_0$,

$$acc(t) = \nu(G_0(t)) \geq 1 - \varepsilon. \tag{17}$$

and

$$\nu(G_{\eta^*}(t)) > 1 - \varepsilon - (3/4)^{-\frac{\log(3\Lambda \bar{L}(t_0))}{2\eta^*}} \tag{18}$$

for all η^*: $0 < \eta^* < -\log(3\Lambda \bar{L}(t_0))$.

Remark 2.1. The conclusion of theorem 2.1 depends on the hypothesis that condition A holds. Short of brute force calculation on a case-by-case basis, there is at present no way to determine whether condition A holds for a given training set T and parameter values $\alpha(t_0)$, and for which constants Λ, ψ, ϕ, and m_0 it might hold. Furthermore, Theorem 2.1 does not control the dynamics of training with sufficient precision to guarantee that if condition A holds at t_0 then it will also hold for all $t > t_0$. Therefore, we have to further strengthen the hypotheses by insisting that condition A holds for all $t > t_0$.

Remark 2.2. Though a more general version of Theorem 2.1 can be proved for $\psi > 0$ and $m_0 \geq 0$, both the statement and proof are much more tractable with $\psi = 1$ and $m_0 = 0$.

Remark 2.3. For Theorem 2.1, it is sufficient to choose

$$\delta = \frac{1}{2\Lambda} \quad \text{and} \quad \eta = \max\left\{1, \left(\frac{1}{\phi}\log\left(\left(\frac{10(K-1)}{\log 2}\right)^{\phi}\frac{3\Lambda}{\varepsilon}\right)\right)^2, \log\left(\frac{30\Lambda(K-1)}{\log 2}\right)^2\right\}. \tag{19}$$

2.4.2 Unconditional result

Our second condition will also limit the number of δX values that may cluster near β. It does this by ensuring that if a few $\delta X(t_0)$ are clustered near β, then there must be more δX values that are larger than the δX values in the cluster. This means that not all of the $s \in G_\eta^C(t_0)$ can have $\delta X(s, \alpha(t_0))$ near β.

Definition 2.2. The set $\{\delta X(s, \alpha(t_0)) : s \in T\}$ satisfies *condition B* at time t_0 if there exist a mass m_0 and constants $\kappa > 1$, and $\delta, \sigma > 0$ such that for all intervals $I \subset \mathbb{R}$,

$$\nu\left(\{s \in T : \delta X(s, \alpha(t_0)) \in \kappa I\}\right) \geq \min\Big\{\delta,\ \max\big\{m_0,\ (1+\sigma)\,\nu\left(\{s \in T : \delta X(s, \alpha(t_0)) \in I\}\right)\big\}\Big\}, \tag{20}$$

where κI is the interval with the same center as I but whose width is multiplied by κ.

Condition B comes with a notable advantage: it is a consequence of a similar condition on the training set T called the *no small data clusters condition*:

Definition 2.3. Let $\bar{\nu}$ be the extension of the measure ν from T to $\mathbb{R}^n$ by $\bar{\nu}(A) = \nu(A \cap T)$ for all $A \subset \mathbb{R}^n$. The *no small isolated data clusters* condition holds if there exists $m_0' > 0$, $\kappa' > 1$ and $\delta, \sigma' > 0$ so that for each slab Sl in $\mathbb{R}^n$,

$$\bar{\nu}(\kappa' Sl) \geq \min\Big\{\delta,\ \max\big\{m_0',\ (1+\sigma')\,\bar{\nu}(Sl)\big\}\Big\}. \tag{21}$$

where $\kappa' Sl$ is the slab with the same center as Sl, but whose width is multiplied by κ'.

We can now present our second stability theorem:

Theorem 2.2. *Assume that the set $\{\delta X(s, \alpha(t_0)) : s \in T\}$ satisfies condition B as given by definition (2.2) at some time t_0. Then for every $\varepsilon > 0$ there exists a constant $C = C(K, \kappa, \sigma)$ such that if*

$$m_0 \leq \frac{\varepsilon}{C}, \quad \eta \geq \log\frac{1}{\varepsilon} + C, \quad \delta_0 \leq C\,\varepsilon\,\eta^{\log(1+\sigma)/\log\kappa}, \tag{22}$$

and if good and bad sets at $t = t_0$ satisfy

$$\nu(G_\eta(t_0)) > 1 - \delta_0 \quad \textit{and} \quad B_{-1}(t_0) = \emptyset, \tag{23}$$

then for all $t \geq t_0$,

$$acc(t) = \nu(G_0(t)) \geq 1 - \varepsilon. \tag{24}$$

and

$$\nu(G_{\eta^*}(t)) \geq 1 - 2\log 2\,\varepsilon\, e^{\eta^*}. \tag{25}$$

For all $\eta^* > 0$.

The following theorem guarantees that if the training set T satisfies the no small data clusters condition, then the set $\{\delta X(s, \alpha(t_0)) : s \in T\}$ satisfies condition B, no matter what what the values of the parameters $\alpha(t_0)$ are. Theorem 2.3 is useful because it allows us to establish stability based solely on training set T, without needing to begin training. Unlike Theorem 2.1 and 2.2, Theorem 2.3 only applies to classifiers consisting of a DNN with softmax as its last layer.

Theorem 2.3. *Let* $T \subset \mathbb{R}^n$ *be a training set for a DNN with weights* $\nu(s)$ *and whose activation functions are the absolute value function. For all* $\kappa > 1$, $\delta > 0$ *and* $\sigma > 0$, *there exists* $\kappa' > 1$, δ, *and* $\sigma' > 0$ *so that if the no small isolated data clusters condition holds on* T *with constants* κ', δ, *and* σ', *then condition B holds on* $\{\delta X(s, \alpha(t_0)) : s \in T\}$ *with constants* κ, δ, *and* σ. *for any* $\alpha(t_0)$.

2.4.3 Examples of applications of Condition A and B

Revisiting example 2.1. 1Conditions A and B and their associated Theorems 2.1 and 2.2 guarantee stability, so why does stability fail in Example 2.1? The answer lies in the constants found in conditions A and B. Condition A is satisfied relative to constants m_0, Λ, ϕ, and ψ, while condition B is satisfied relative to constants κ, σ, δ, and m_0. We will determine for which constants conditions A and B are satisfied. For both conditions, we will consider a typical m_0 value of $m_0 = 0.003$.

Theorem 2.1 requires $\psi = 1$. Taking $x_1 = \beta(t_0)$, we find via brute force calculation that for the δX values given in example Λ must exceed 18.0. In the proof of Theorem 2.1, we will see that δ and η must be chosen so that $3\Lambda\bar{L}(t_0) < 1$ However, in Example 2.1,

$$3\Lambda\bar{L}(t_0) \geq 3 \cdot 18 \cdot 0.1845 > 1.$$

Therefore, though Theorem 2.1 guarantees the existence of δ small enough and η large enough to get stability, such δ and η for Example 2.1 will not satisfy the primary hypothesis of the theorem, that is $\nu(G_\eta(t_0)) > 1 - \delta$.

Theorem 2.2 requires $\kappa = 2$. If $I = [-1, -0.2]$, then $\nu(\{s \in T : \delta X(s, \alpha(t_0)) \in I\}) = \nu(\{s \in T : \delta X(s, \alpha(t_0)) \in \kappa I\}) = 0.047$, so either $\sigma = 0$, which is not allowed in condition B, or $\delta < .047$. But if $\delta < 0.047$, then since $\nu(G_0(t_0)) = 0.95$, it is impossible to obtain $\nu(G_\eta(t_0)) > 1 - \delta$ for any $\eta > 0$.

Since Example 2.1 can only satisfy conditions A and B with constants that are either too large or too small, we cannot apply the stability theorems to it.

To see how Theorems 2.1 and 2.2 can be applied, consider the following example.

Example 2.2. Suppose a classifier properly classifies all elements in its training set at t_0. In fact, for some large η, $\nu(G_\eta(t_0) = 1$. How large high will accuracy be at later times?

First suppose that T is a two-class training set for a classifier which satisfies the condition A for all $t \geq t_0$ for some constants $\Lambda \geq 1$, $\psi = 1$, $0 < \phi < 1$ and $m_0 = 0$. Theorem 2.1 tells us that given $\varepsilon > 0$, if η is large enough, then

$$\text{acc}(t) > 1 - \varepsilon$$

for all $t > t_0$. But how small can ε be? Remark 2.3 gives a relationship between η and ϵ, and in particular, provided η is sufficiently large, we may choose

$$\varepsilon = \left(\frac{10}{\log 2}\right)^{\phi} \frac{3\Lambda}{e^{\phi\sqrt{\eta}}}.$$

Since η is large, ε is small. Therefore, at all later times, we are guaranteed high accuracy. Additionally, good sets also remain large. For example, by choosing

$$\eta^* = \frac{\log(3/4)}{2\log\epsilon}(\eta - \log(3\Lambda))$$

A simple computation with (18) using $\bar{L}(t_0) \leq e^{-\eta}$ shows that

$$\nu(G_{\eta^*}(t)) > 1 - 2\varepsilon.$$

for all $t > t_0$. Therefore, $G_{\eta^*}(t)$ set remains large for all t, the price paid being that $\eta^* < \eta$. If η is very large, we can in fact note that $\eta^* \sim \sqrt{\eta}$.

Alternatively, we may choose

$$\eta^* = \frac{\log 3/4}{2\log 2}(\eta - \log(3\Lambda))$$

gives

$$\nu(G_{\eta^*}(t)) \geq \frac{1}{2} - \varepsilon,$$

meaning the median of the distribution of δX values is greater than η^* with now $\eta^* \sim \eta$.

Similarly, if condition B is satisfied at $t = t_0$ for some constants δ, κ, σ and m_0, instead of condition A, then we can apply theorem 2.2. Using (22), we conclude that letting

$$\varepsilon = \max\{m_0 C, e^{C-\eta}\}$$

we have

$$\text{acc}(t) > 1 - \varepsilon.$$

Since m_0 is small and η is large, ε is also small, so accuracy remains high. By letting $\eta^* = \log(2/\log 2)$, (25), we have

$$\nu(G_{\eta^*}(t)) > 1 - 2\varepsilon,$$

so this good set also remains large but with a significantly worse η^* than for condition A.

Finally, if $\eta^* = -\log(4\varepsilon \log 2)$, then

$$\nu(G_{\eta^*}(t)) > 1/2,$$

so again the median of the δX distribution is greater than η^*. Nevertheless, here we still have that $\eta^* \sim \eta$.

2.5 How to verify conditions A and B for a given dataset

In this section, we discuss how to verify conditions A and B and the no-small-isolated data clusters condition to ensure stability of training algorithms in a real-world setting. For completeness, we review notations:

- We consider classifiers of the form $\phi(\cdot, \alpha(t)) = \rho \circ X(\cdot, \alpha(t))$ where ρ is the softmax function and $X(\cdot, \alpha(t)) : \mathbb{R}^n \to \mathbb{R}^K$ depends on parameters $\alpha(t)$ where t is the present iteration of the training process.
- T is a training set for the classifier containing objects s. For each s, $i(s)$ is the index of the correct class of s.
- Each object s has a positive weight $\nu(s)$ with $\sum_{s\in T} \nu(s) = 1$.
- $\delta X(s, \alpha(t)) = X_{i(s)}(s, \alpha(t)) - \max_{j\neq i(s)} X_j(s, \alpha(t))$.

Now we will explain how to verify each of the conditions, and how to use them to guarantee stability.

- **Condition A.** Train the classifier until a time t_0 when a reasonable degree of accuracy is achieved. Calculate the values of $\delta X(s, \alpha(t_0))$ for each $s \in T$. One way to do this is to choose the typical values $\psi = 1$, $\phi = 1/2$, and $m_0 \approx 0.001$ so that the only constant to solve for in (15) is Λ. To this end, make the observation:

$$\begin{matrix}\text{minimal } \Lambda \text{ such that} \\ (15) \text{ is satisfied}\end{matrix} = \max_{x_1\in\{\beta,0\}, x_2 > x_1} \frac{\nu\left(\left\{s \in T : \delta X(s, \alpha(t_0)) < \frac{x_1+x_2}{2}\right\}\right) - m_0}{\nu\left(\{s \in T : \delta X(s, \alpha(t_0)) < x_1\}\right)^{\phi} + \nu\left(\{s \in T : \delta X(s, \alpha(t_0)) < x_2\}\right)^{\psi}}. \tag{26}$$

  Therefore, finding the optimal Λ is a matter of solving a maximization problem. For fixed x_1, the right hand side of (26) is piecewise constant with discontinuities at $\delta X(s, \alpha(t_0))$ for $s \in T$. Thus, the maximization problem is solved by sampling the right hand side of (26) at $x_1 = 0$ and $x_1 = \beta$, and $x_2 = \delta X(s, \alpha(t_0))$ for all $s \in T$ and then finding the maximum of the resulting list of samples.

  With condition A satisfied for some known constants, one may apply Theorem 2.1. However, as seen with Example 2.1, if Λ is too large, Theorem 2.1 may still require $G_\eta(t_0)$ larger than it actually is. It may be possible to decrease Λ by repeating the maximization process with smaller ϕ or larger m_0. If an acceptable Λ is found, 2.1 guarantees stability. If not, one may need to train longer to find an acceptable Λ.

- **No-small-isolated data clusters condition.** Assume that the classifier is a deep neural network with the absolute value function as its activation function. By verifying the no-small-isolated data clusters condition, we are guaranteed that condition B holds for all time. Therefore, we need only verify once that the no-small-isolated data clusters condition holds, which is its principal advantage. To verify this condition, choose constants $\delta = 0.01$, $\kappa' = 2$, and $m_0 \approx 0.001$. Let P be the set of all slabs Sl in $\mathbb{R}^n$ such that $\nu(\kappa\, Sl \cap T) \leq \delta$ and $\nu(Sl \cap T) > m_0$. We are left with finding the constant σ' which satisfies (21) for all slabs in P:

$$\text{maximal } \sigma' \text{ such that (21) is satisfied} = \min_{Sl \in P} \frac{\nu(\kappa Sl \cap T)}{\nu(Sl \cap T)} - 1. \tag{27}$$

As in verifying condition A, we will solve this minimization problem by discretizing the domain of minimization, P, and sampling the objective function only on that discretization. It should be noted that the dimension of P is $n+1$. Since n (the dimension of the space containing T) is typically high, this means that a sufficiently fine discretization is necessarily quite large.

After finding σ', we may be sure that condition B is satisfied at every iteration of the training algorithm for known constants. Therefore, we may apply Theorem 2.2 to guarantee stability.

3 Proof of Theorems 2.1, 2.2 and 2.3

3.1 Elementary estimates

Here, we will show the details and derivations of many several simple equations, inequalities, and some technical lemmas.

Estimates for loss. As mentioned in Section 2.3.1, the quantity $\delta X(s,\alpha)$ facilitates convenient estimates for loss. To start, (6) gives

$$p_{i(s)}(s,\alpha) = \frac{e^{X_{i(s)}(s,\alpha)}}{\sum_{k=1}^K e^{X_k(s,\alpha)}} = \frac{1}{\sum_{k=1}^K e^{X_k(s,\alpha) - X_{i(s)}(s,\alpha)}} = \frac{1}{1 + \sum_{k \neq i(s)} e^{X_k(s,\alpha) - X_{i(s)}(s,\alpha)}}. \tag{28}$$

For each $k \neq i(s)$, $X_k(s,\alpha) - X_{i(s)}(s,\alpha) \leq \max_{k \neq i(s)} X_k(s,\alpha) - X_{i(s)}(s,\alpha) = -\delta X(s,\alpha)$. Using (7) and (28), we obtain estimates on $p_{i(s)}(s,\alpha)$:

$$\frac{1}{1 + (K-1)e^{-\delta X(s,\alpha)}} \leq p_{i(s)}(s,\alpha) \leq \frac{1}{1 + e^{-\delta X(s,\alpha)}}. \tag{29}$$

Finally, (29) gives estimates on loss:

$$\sum_{s \in T} \nu(s) \log\left(1 + e^{-\delta X(s,\alpha)}\right) \leq \bar{L}(\alpha) \leq \sum_{s \in T} \nu(s) \log\left(1 + (K-1)e^{-\delta X(s,\alpha)}\right). \tag{30}$$

In particular, if there are only two classes, the inequalities in (29) and (30) are equalities.

Derivations of (12) **and** (13)**.** Equations (12) and (13) show how low loss leads to high accuracy. Starting from the lower bound for loss in (30), we make the following estimates for any t:

$$\begin{aligned}
\bar{L}(t) &\geq \sum_{s \in T} \nu(s) \log(1 + e^{-\delta X(s,\alpha(t))}) \\
&\geq \sum_{s \in G_\eta^C(t)} \nu(s) \log(1 + e^{-\delta X(s,\alpha(t))}) \\
&\geq \log(1 + e^{-\eta}) \sum_{s \in G_\eta^C(t)} \nu(s) \\
&= \nu(G_\eta^C(t)) \log(1 + e^{-\eta}) \\
&= (1 - \nu(G_\eta(t))) \log(1 + e^{-\eta}).
\end{aligned} \tag{31}$$

Observe that $\log(1+x) \geq x/2$ for $0 \leq x \leq 1$. It follows that

$$\nu(G_\eta(t)) \geq 1 - 2e^\eta \bar{L}(t).$$

With the assumption that loss is decreasing, and $t \geq t_0$, we conclude that

$$\nu(G_\eta(t)) \geq 1 - 2e^\eta \bar{L}(t) \geq 1 - 2e^\eta \bar{L}(t_0).$$

On the other hand, we can obtain an improved estimate for $\nu(\mathrm{acc}(t))$ by applying (31) with $\eta = 0$:

$$\nu(\mathrm{acc}(t)) \geq 1 - \frac{\bar{L}(t)}{\log 2} \geq 1 - \frac{\bar{L}(t_0)}{\log 2}.$$

Derivation of (14)**.** Equation (14) shows that when $G_\eta(t)$ is large, the sum (8) is dominated by a few terms that correspond to poorly classified objects. To derive (14), start from the upper bound in (30), and then make the following series of estimates:

$$\begin{aligned}
\bar{L}(t) &= \sum_{s\in T} \nu(s) \log\left(1 + (K-1)e^{-\delta X(s,\alpha(t))}\right) \\
&= \sum_{s\in G_\eta(t)} \nu(s) \log\left(1 + (K-1)e^{-\delta X(s,\alpha(t))}\right) + \sum_{s\in G_\eta^C(t)} \nu(s) \log\left(1 + (K-1)e^{-}\delta X(s,\alpha(t))\right) \\
&\leq \log\left(1 + (K-1)e^{-\eta}\right) \sum_{s\in G_\eta(t)} \nu(s) + \sum_{s\in G_\eta^C(t)} \nu(s) \log\left(1 + (K-1)e^{-\delta X(s,\alpha(t))}\right) \\
&\leq (K-1)e^{-\eta}\nu(G_\eta(t)) + \sum_{s\in G_\eta^C(t)} \nu(s) \log\left(1 + (K-1)e^{-\delta X(s,\alpha(t))}\right) \\
&\leq (K-1)e^{-\eta} + \sum_{s\in G_\eta^C(t)} \nu(s) \log\left(1 + (K-1)e^{-\delta X(s,\alpha(t))}\right).
\end{aligned}$$

The following two technical lemmas will be used in later proofs.

Lemma 3.1. *Suppose* $0 < \Lambda\delta < 1/2$*. The inequality*

$$\sum_{k=0}^{\infty} e^{2^k \log(\Lambda\delta) - \frac{1+\eta}{2^{k+1}}} \leq C\, e^{-\sqrt{-2\log(\Lambda\delta)(1+\eta)}}$$

can always be satisfied for some $C = C(\Lambda, \varepsilon, \eta) \leq 13/4$

The proof of Lemma 3.1 is essentially a long series of elementary estimates which are not very enlightening. Consequently, it is relegated to the appendix.

Lemma 3.2. *For any* $p > 0$ *and any* $\kappa \geq 2$*,*

$$\sum_{0}^{p-1} \kappa^i\, e^{-\kappa^i} \leq 2. \tag{32}$$

Proof. Observe that for $x \geq 2$, one trivially has that

$$x^2\, e^{-x} \leq 1,$$

so that

$$\sum_{1}^{p-1} \kappa^i\, e^{-\kappa^i} \leq \sum_{1}^{p-1} \kappa^{-i} \leq \sum_{1}^{p-1} 2^{-i} \leq 1.$$

□

3.2 Upper bound on the loss function for condition A

A key part of the proof of Theorem 2.1 is to obtain an upper bound on the Loss function at the initial time t_0, as given by

Lemma 3.3. *Suppose that* $T \subset \mathbb{R}^n$ *with measure* $\nu(s)$ *is a training set for a softmax DNN which satisfies condition A for some constants* $\Lambda \geq 1$*,* $\psi = 1$ *and* $0 < \phi < 1$ *and for* $t = t_0$*. If for some* $\eta > 0$*,*

$$\nu(G_\eta(t)) > 1 - \delta \tag{33}$$

for $\delta < 1/2\Lambda$ *and*

$$B_{-1}(t) = \emptyset, \tag{34}$$

then the cross-entropy loss is bounded by:

$$\bar{L}(t) \leq e^{-\eta} + Ce^{-\sqrt{-2\log(\Lambda\delta)(1+\eta)}}, \tag{35}$$

where C *is a constant less than* $13e/4$*.*

Proof. For each $k = 0, 1, 2, \cdots ...$, let

$$\eta_k = \beta + \frac{-\beta + \eta}{2^k} \text{ and } I_k = \{s \in T : \delta X(s, \alpha(t_0)) < \eta_k\},$$

where $\beta := \min_{s \in T} \delta X(s, \alpha(t_0))$. Observe the relation $(\eta_k + \beta)/2 = \eta_{k+1}$. For fixed k, apply condition A for $x_1 = \beta$ and $x_2 = \eta_k$:

$$\begin{aligned}
\nu(I_{k+1}) &= \nu\left(\{s \in T : \delta X(s, \alpha(t_0)) < \eta_{k+1}\}\right) \\
&= \nu\left(\left\{s \in T : \delta X(s, \alpha(t_0)) < \frac{\eta_k + \beta}{2}\right\}\right) \\
&= \nu\left(\left\{s \in T : \delta X(s, \alpha(t_0)) < \frac{x_1 + x_2}{2}\right\}\right).
\end{aligned}$$

This implies that

$$\begin{aligned}
\nu(I_{k+1}) &= \Lambda\nu\left(\{s \in T : \delta X(s, \alpha(t_0)) < x_1\}\right)^{\phi} + \Lambda\nu\left(\{s \in T : \delta X(s, \alpha(t_0)) < x_2\}\right)^{\psi+1} \\
&= \Lambda\nu\left(\{s \in T : \delta X(s, \alpha(t_0)) < \beta\}\right)^{\phi} + \Lambda\nu\left(\{s \in T : \delta X(s, \alpha(t_0)) < \eta_k\}\right)^{\psi+1} \\
&= 0 + \Lambda\nu(I_k)^2 \quad \text{by the definition of } \beta \\
&= \Lambda\nu(I_k)^2
\end{aligned}$$

Since $\eta_0 = \eta$, (33) gives

$$\nu(I_0) = \nu\left(\{s \in T : \delta X(s, \alpha(t)) < \eta\}\right) = 1 - \nu(G_\eta) \leq \delta.$$

Therefore, by induction

$$\nu(I_k) \leq \Lambda^{2^k - 1}\delta^{2^k}.$$

We may now simply bound loss from above. Recall that the cross-entropy loss may be bounded by

$$\bar{L}(t) \leq \sum_{s \in T} \nu(s) \log\left(1 + (K-1)e^{-\delta X(s, \alpha(t))}\right).$$

Since β is the minimum δX value, $\delta X(s, \alpha(t_0)) > \beta$ for all $s \in T$, so either $s \in G_\eta(t)$ or $s \in I_k \setminus I_{k+1}$ for some k. Additionally, $\log(1 + (K-1)e^{-x})$ is decreasing in x, so if $\delta X(s, \alpha(t_0)) \in I_k \setminus I_{k+1}$, then $\delta X(s, \alpha(t_0)) > \eta_{k+1}$. Therefore,

$$\log\left(1 + (K-1)e^{-\delta X(s, \alpha(t))}\right) \leq \log\left(1 + (K-1)e^{-\eta_{k+1}}\right).$$

Using these facts, we make the following estimate

$$\begin{aligned}
\bar{L}(t_0) &\leq \sum_{s \in G_\eta(t)} \nu(s) \log\left(1 + (K-1)e^{-\delta X(s, \alpha(t_0))}\right) + \sum_{k=0}^{\infty} \sum_{s \in I_k \setminus I_{k+1}} \nu(s) \log\left(1 + (K-1)e^{-\delta X(s, \alpha(t_0))}\right) \\
&\leq \sum_{s \in G_\eta(t)} \nu(s) \log\left(1 + (K-1)e^{-\eta}\right) + \sum_{k=0}^{\infty} \sum_{s \in I_k \setminus I_{k+1}} \nu(s) \log\left(1 + (K-1)e^{-\eta_{k+1}}\right) \\
&= \log\left(1 + (K-1)e^{-\eta}\right) \sum_{s \in G_\eta(t)} \nu(s) + \sum_{k=0}^{\infty} \log\left(1 + (K-1)e^{-\eta_{k+1}}\right) \sum_{s \in I_k \setminus I_{k+1}} \nu(s).
\end{aligned}$$

As a consequence, we have that

$$\begin{aligned}
\bar{L}(t_0) &\leq \nu(G_\eta(t))(K-1)e^{-\eta} + \sum_{k=0}^{\infty} \nu(I_k \setminus I_{k+1})(K-1)e^{-\eta_{k+1}} \\
&\leq (1-\delta)(K-1)e^{-\eta} + \sum_{k=0}^{\infty} \nu(I_k)(K-1)e^{-\eta_{k+1}} \\
&\leq (K-1)e^{-\eta} + \sum_{k=0}^{\infty} (K-1)\Lambda^{2^k - 1}\delta^{2^k}e^{-\eta_{k+1}},
\end{aligned}$$

and

$$\bar{L}(t_0) = (K-1)e^{-\eta} + (K-1)\sum_{k=0}^{\infty} \Lambda^{2^k-1}\delta^{2^k} e^{-\beta-\frac{\eta-\beta}{2^{k+1}}}$$
$$\leq (K-1)e^{-\eta} + (K-1)e^{-\beta}\sum_{k=0}^{\infty} e^{2^k\log(\Lambda\delta)-\frac{\eta-\beta}{2^{k+1}}}.$$

Since $\beta > -1$, we have

$$\bar{L}(t_0) \leq (K-1)e^{-\eta} + (K-1)e\sum_{k=0}^{\infty} e^{2^k\log(\Lambda\delta)-\frac{\eta+1}{2^{k+1}}}.$$

By Lemma 3.1, we can find a constant C less than $13e/4$ such that

$$e\sum_{k=0}^{\infty} e^{2^k\log(\Lambda\delta)-\frac{\eta+1}{2^{k+1}}} \leq Ce^{-\sqrt{-2\log(\Lambda\delta)(1+\eta)}}.$$

Thus,

$$\bar{L}(t) \leq (K-1)\left(e^{-\eta} + Ce^{-\sqrt{-2\log(\Lambda\delta)(1+\eta)}}\right).$$

□

3.3 Proof of Theorem 2.1

With Lemma 3.3, we are now ready to prove Theorem 2.1. Fix $\varepsilon > 0$. Let $\delta_0 = 1/2\Lambda$, and choose $\delta < \delta_0$. We may apply Lemma 3.3 to see that

$$\bar{L}(t_0) \leq P(\eta,\delta,\Lambda) := (K-1)\left(e^{-\eta} + Ce^{-\sqrt{-2\log(\Lambda\delta)(1+\eta)}}\right).$$

Observe that $\lim_{\eta\to\infty} P(\eta,\delta,\Lambda) = 0$, so by choosing η_0 sufficiently large and $\eta > \eta_0$, we can make loss arbitrarily small (see Remark 2.3 for an explicit estimate on η_0).

Since loss is decreasing in time, if $\bar{L}(t_0)$ is small, then $\bar{L}(t)$ is also small for all $t > t_0$. By making $\bar{L}(t)$ small, we will be able bound the size of good sets from below. To start, we will show that if $\bar{L}(t_0) < 1/3\Lambda$, then

$$G_{-\log(3\Lambda\bar{L}(t_0))}(t) > 1 - \frac{1}{2\Lambda} \tag{36}$$

for all t. Suppose, to the contrary, that $G_{-\log(3\Lambda\bar{L}(t_0))}(t) \leq 1 - \frac{1}{2\Lambda}$ for some t. Then the size of the complement of the good set is bounded below:

$$G^C_{-\log(3\Lambda\bar{L}(t_0))}(t) > \frac{1}{2\Lambda}$$

Therefore,

$$\begin{aligned}\bar{L}(t_0) &\geq \bar{L}(t)\\ &\geq \sum_{s\in T}\nu(s)\log(1+e^{-\delta X(s,\alpha(t))})\\ &\geq \sum_{s\in G^C_{-\log(3\Lambda\bar{L}(t_0))}}\nu(s)\log(1+e^{-\delta X(s,\alpha(t))}),\end{aligned}$$

which gives

$$\begin{aligned}\bar{L}(t_0) &\geq \nu(G^C_{-\log(3\Lambda\bar{L}(t_0))}(t))\log(1+e^{\log(3\Lambda\bar{L}(t_0))})\\ &> \frac{1}{2\Lambda}\log(1+3\Lambda\bar{L}(t_0)).\end{aligned}$$

Since log is concave down, $\log(1+x) < x\log(2)$ for $0 < x < 1$. Thus,

$$\bar{L}(t_0) > \frac{1}{2\Lambda}3\Lambda\bar{L}(t_0)\log(2) = \frac{3}{2}\log(2)\bar{L}(t_0) > \bar{L}(t_0).$$

This is a contradiction, so $G_{-\log(3\Lambda\bar{L}(t_0))}(t) > 1 - \frac{1}{2\Lambda}$.

We will now use (36) to bound the size of all good sets from below. Suppose that $\bar{L}(t_0) < 1/3\Lambda$. Let $\eta_k = -2^{-k}\log(3\Lambda\bar{L}(t_0))$ for all $k = 0, 1, 2, \cdots$. Clearly, there is a recurrence relation: $\eta_{k+1} = \eta_k/2$. For fixed k, let $x_1 = 0$ and $x_2 = \eta_k$. Then by condition A we can estimate the size of the complement of $G_{\eta_k}(t)$ for all t. First remark that

$$\begin{aligned}\nu\left(G^C_{\eta_{k+1}}(t)\right) &= \nu\left(\{s \in T : \delta X(s, \alpha(t)) < \eta_{k+1}\}\right) \\ &= \nu\left(\left\{s \in T : \delta X(s, \alpha(t)) < \frac{0 + \eta_k}{2}\right\}\right) \\ &= \nu\left(\left\{s \in T : \delta X(s, \alpha(t)) < \frac{x_1 + x_2}{2}\right\}\right).\end{aligned}$$

Applying now condition A, we obtain

$$\begin{aligned}\nu\left(G^C_{\eta_{k+1}}(t)\right) &\le \Lambda\left(\nu\left(\{s \in T : \delta X(s, \alpha(t)) < x_1\}\right)^\phi + \nu\left(\{s \in T : \delta X(s, \alpha(t)) < x_2\}\right)^2\right) \\ &\le \Lambda\left(\nu\left(\{s \in T : \delta X(s, \alpha(t)) < 0\}\right)^\phi + \nu\left(\{s \in T : \delta X(s, \alpha(t)) < \eta_k\}\right)^2\right) \\ &\le \Lambda\left(\nu\left(G^C_0(t)\right)^\phi + \nu\left(G^C_{\eta_k}(t)\right)^2\right)\end{aligned}$$

From (13),

$$\nu(G^C_0(t)) = 1 - \mathrm{acc}(t) \le \frac{\bar{L}(t_0)}{\log 2}.$$

Therefore,

$$\nu\left(G^C_{\eta_{k+1}}(t)\right) \le \Lambda\left(\left(\frac{\bar{L}(t_0)}{\log 2}\right)^\phi + \nu\left(G^C_{\eta_k}(t)\right)^2\right) \tag{37}$$

Since η_k is decreasing with k, $\nu(G^C_{\eta_k}(t))$ is also decreasing in k. This means that if for some k_0,

$$\nu\left(G^C_{\eta_{k_0}}(t)\right) \le \sqrt{2}\left(\frac{\bar{L}(t_0)}{\log 2}\right)^{\phi/2}, \tag{38}$$

then (38) also holds with η_{k_0} replaced by η_k for all $k > k_0$. Therefore, for all $k > k_0$, we may use (37) to estimate:

$$\nu\left(G^C_{\eta_k}(t)\right) \le \Lambda\left(\left(\frac{\bar{L}(t_0)}{\log 2}\right)^\phi + \nu\left(G^C_{\eta_{k-1}}(t)\right)^2\right) \le \Lambda\left(\left(\frac{\bar{L}(t_0)}{\log 2}\right)^\phi + 2\left(\frac{\bar{L}(t_0)}{\log 2}\right)^\phi\right) = 3\Lambda\left(\frac{\bar{L}(t_0)}{\log 2}\right)^\phi. \tag{39}$$

On the other hand, if $k \le k_0$,

$$\nu\left(G^C_{\eta_k}(t)\right) > \sqrt{2}\left(\frac{\bar{L}(t_0)}{\log 2}\right)^{\phi/2},$$

or equivalently,

$$\left(\frac{\bar{L}(t_0)}{\log 2}\right)^\phi < \frac{1}{2}\nu\left(G^C_{\eta_k}(t)\right)^2.$$

Thus, for $k < k_0$, we can use (37) to obtain

$$\nu(G^C_{\eta_k}(t)) \le \Lambda\left(\left(\frac{\bar{L}(t_0)}{\log 2}\right)^\phi + \nu\left(G^C_{\eta_{k-1}}(t)\right)^2\right) \le \Lambda\left(\frac{1}{2}\nu\left(G^C_{\eta_{k-1}}(t)\right)^2 + \nu\left(G^C_{\eta_{k-1}}(t)\right)^2\right) = \frac{3}{2}\nu\left(G^C_{\eta_{k-1}}(t)\right)^2.$$

By induction,

$$\nu(G^C_{\eta_k}(t)) \le \left(\frac{3}{2}\Lambda\right)^{2^k-1}\nu\left(G_{\eta_0}(t)\right)^{2^k}.$$

From (36), $\nu\left(G_{\eta_0}(t)\right) \le 1/2\Lambda$, so

$$\nu\left(G^C_{\eta_k}(t)\right) \le \left(\frac{3}{2}\Lambda\right)^{2^k-1}\left(\frac{1}{2\Lambda}\right)^{2^k} \le \left(\frac{3}{4}\right)^{2^k} = \left(\frac{3}{4}\right)^{-\frac{\log(3\Lambda\bar{L}(t_0))}{\eta_k}}. \tag{40}$$

For any k, either (39) and (40) holds, so for all $k \geq 0$,

$$\nu\left(G^C_{\eta_k}(t)\right) \leq \left(\frac{3}{4}\right)^{-\frac{\log(3\Lambda\bar{L}(t_0))}{\eta_k}} + 3\Lambda\left(\frac{\bar{L}(t_0)}{\log 2}\right)^{\phi}. \tag{41}$$

We wish to find a bound on good sets for all η^* with $0 < \eta^* < -\log(3\Lambda\bar{L}(t_0))$, not just $\eta^* = \eta_k$ for some k. By the monotonicity of good sets, if $\eta_{k+1} < \eta^* \leq \eta_k$,

$$\nu\left(G^C_{\eta_{k+1}}(t)\right) \leq \nu\left(G^C_{\eta^*}(t)\right) \leq \nu\left(G^C_{\eta_k}(t)\right). \tag{42}$$

Since $\eta_k = 2\eta_{k+1}$, $1/2\eta^* < 1/\eta_k$. Applying (41) to (42), we have

$$\nu\left(G^C_{\eta^*}(t)\right) \leq \left(\frac{3}{4}\right)^{-\frac{\log(3\Lambda\bar{L}(t_0))}{\eta_k}} + 3\Lambda\left(\frac{\bar{L}(t_0)}{\log 2}\right)^{\phi} \leq \left(\frac{3}{4}\right)^{-\frac{\log(3\Lambda\bar{L}(t_0))}{2\eta^*}} + 3\Lambda\left(\frac{\bar{L}(t_0)}{\log 2}\right)^{\phi}.$$

Since $\nu(G_{\eta^*}(t)) = 1 - \nu(G_{\eta^*}(t))$, for all η^* with $0 < \eta^* < \log(3\Lambda\bar{L}(t_0))$,

$$\nu\left(G_{\eta^*}(t)\right) > 1 - \left(\frac{3}{4}\right)^{-\frac{\log(3\Lambda\bar{L}(t_0))}{2\eta^*}} - 3\Lambda\left(\frac{\bar{L}(t_0)}{\log 2}\right)^{\phi}, \tag{43}$$

and since $G_0(t) = \cup_{\eta^*>0} G_{\eta^*}(t)$,

$$\begin{aligned}
\mathrm{acc}(t) &= \nu\left(G_0(t)\right) \\
&= \lim_{\eta^*\to 0} \nu\left(G_{\eta^*}(t)\right) \\
&> \lim_{\eta^*\to\infty} \left(1 - \left(\frac{3}{4}\right)^{-\frac{\log(3\Lambda\bar{L}(t_0))}{2\eta^*}} - 3\Lambda\left(\frac{\bar{L}(t_0)}{\log 2}\right)^{\phi}\right) \\
&= 1 - 3\Lambda\left(\frac{\bar{L}(t_0)}{\log 2}\right)^{\phi}.
\end{aligned} \tag{44}$$

Finally, by choosing η_0 sufficiently large and $\eta > \eta_0$, we can make $\bar{L}(t_0)$ sufficiently large that $3\Lambda(\bar{L}(t_0)/\log 2)^{\phi} < \varepsilon$. Therefore,

$$\mathrm{acc}(t) > 1 - \varepsilon$$

and

$$\nu\left(G_{\eta^*}(t)\right) > 1 - \epsilon - \left(\frac{3}{4}\right)^{-\frac{\log(3\Lambda\bar{L}(t_0))}{2\eta^*}},$$

concluding the proof.

3.4 Upper bound on the loss for condition B

Just as for Theorem 2.1, the first step in the proof of Theorem 2.2 is to derive an upper bound on the loss function based now on condition B.

Lemma 3.4. *Suppose that $T \subset \mathbb{R}^n$ with weights $\nu(s)$ is a training set for a softmax DNN which satisfies condition B in the sense of definition 2.2, at time t_0 for m_0 and some constants κ, $\delta > 0$ and $\sigma > 0$. If for some $\delta_0 > 0$, and $\delta_0 < \delta$,*

$$\nu(G_\eta(t_0)) > 1 - \delta_0 \tag{45}$$

and

$$B_{-1}(t_0) = \emptyset, \tag{46}$$

then the cross-entropy loss is bounded by:

$$\bar{L}(t_0) \leq C\, m_0 + (K-1)\left(e^{-\eta} + \delta_0\, e^{-\eta/\kappa} + \delta_0\, \frac{\sigma+1}{\eta^\gamma}\, (e + 2\,\kappa^\gamma)\right), \tag{47}$$

for any $0 < \gamma \leq \min\{1, \log(1+\sigma)/\log\kappa\}$.

Proof. Consider $x \in [-1, \eta)$ and $\ell > 0$ such that $I := [x - \ell, x + \ell) \subset [-1, \eta)$. By condition B, we have either that

$$\nu\left(\{s \in T : \delta X(s, \alpha(t_0)) \in I\}\right) \leq m_0,$$

or that

$$\nu\left(\{s \in T : \delta X(s, \alpha(t_0)) \in \kappa\, I\}\right) \geq \min\{\delta,\ (1+\sigma)\,\nu\left(\{s \in T : \delta X(s, \alpha(t_0)) \in I\}\right)\}.$$

Applying condition B repeatedly j times in this last case, we conclude that

$$\nu\left(\left\{s \in T : \delta X(s, \alpha(t_0)) \in \kappa^j\, I\right\}\right) \geq \min\{\delta,\ (1+\sigma)^j \nu\left(\{s \in T : \delta X(s, \alpha(t_0)) \in I\}\right)\}.$$

By (45), $\nu\left(\{s \in T : \delta X(s, \alpha(t_0)) < \eta\}\right) \leq \delta_0$. Therefore, provided $\kappa^j I \subset (-\infty,\ \eta)$,

$$\delta_0 \geq \nu\left(\left\{s \in T : \delta X(s, \alpha(t_0)) \in \kappa^j\, I\right\}\right) \geq \min\{\delta,\ (1+\sigma)^j \nu\left(\{s \in T : \delta X(s, \alpha(t_0)) \in I\}\right)\}.$$

Since $\delta_0 < \delta$, we conclude that

$$\nu\left(\{s \in T : \delta X(s, \alpha(t_0)) \in I\}\right) \leq \max\left(m_0,\ \frac{\delta_0}{(\sigma+1)^j}\right) \tag{48}$$

for all j so that $\kappa^j I \subset (-\infty, \eta)$, or equivalently, for all j so that $x + \kappa^j \ell \leq \eta$. Obviously, (48) is best for j as large as possible with the largest value given by

$$j_{\max} = \left\lfloor \frac{\log\left(\frac{\eta - x}{\ell}\right)}{\log \kappa} \right\rfloor > \frac{\log\left(\frac{\eta - x}{\ell}\right)}{\log \kappa} - 1.$$

Therefore,

$$\begin{aligned}
\nu\left(\{s \in T : \delta X(s, \alpha(t_0)) \in I\}\right) &\leq m_0 + \frac{\delta_0}{(\sigma+1)^{j_{\max}}} \\
&< m_0 + \frac{\delta_0}{(\sigma+1)^{\frac{\log\left(\frac{\eta - x}{\ell}\right)}{\log \kappa} - 1}}
\end{aligned}$$

We have thus proved the bound

$$\nu\left(\{s \in T : \delta X(s, \alpha(t_0)) \in I\}\right) < m_0 + \delta_0\,(\sigma+1)\left(\frac{\ell}{\eta - x}\right)^{\gamma}, \tag{49}$$

where $\gamma = \log(\sigma+1)/\log\kappa$. If condition B holds for some σ, it also holds for all smaller σ, so we may assume without loss of generality that $\sigma < \kappa - 1$, and therefore $0 < \gamma < 1$. Similarly we may assume that $\kappa \geq 2$.

Now we will apply (49) to explicit intervals. Let $p = \lfloor \log(\eta)/\log\kappa \rfloor$ and let $I_i = [\kappa^i, \kappa^{i+1})$ for all $i \in \mathbb{N}$ with $i < p$. We also define $I_{-1} = [-1, 1)$ and $I_p = [\kappa^p,\ \eta)$. For $i \geq 0$, I_i is centered at $x_i = \frac{1+\kappa}{2} \cdot \kappa^i$ and has half-width $\ell_i = \frac{\kappa - 1}{2} \cdot \kappa^i$. Therefore by (49),

$$\nu\left(\{s \in T : \delta X(s, \alpha(t_0)) \in I_i\}\right) \leq m_0 + \delta_0\,(1+\sigma)\left(\frac{\ell_i}{\eta - x_i}\right)^{\gamma}. \tag{50}$$

The interval I_{-1} is centered at 0 and has width 1, so

$$\nu\left(\{s \in T : \delta X(s, \alpha(t_0)) \in I_{-1}\}\right) \leq m_0 + \delta_0\,(1+\sigma)\left(\frac{1}{\eta}\right)^{\gamma}. \tag{51}$$

Finally since $\kappa^p > \kappa^{\log\eta/\log\kappa - 1}$, we simply bound for I_p

$$\nu\left(\{s \in T : \delta X(s, \alpha(t_0)) \in I_p\}\right) \leq \delta_0. \tag{52}$$

For each $s \in T$, either $\delta X(s, \alpha(t_0)) \geq \eta$, or $\delta X(s, \alpha(t_0)) \in I_i$ for integer $i \geq -1$. Therefore,

$$\begin{aligned}
\bar{L}(t_0) &\leq \sum_{s \in T} \nu(s) \log\left(1 + (K-1)e^{-\delta X(s, \alpha(t_0))}\right) \\
&= \sum_{\substack{s \in T \\ \delta X(s, \alpha(t_0)) \geq \eta}} \nu(s) \log\left(1 + (K-1)e^{-\delta X(s, \alpha(t_0))}\right) + \sum_{i=-1}^{p-1} \sum_{\substack{s \in T \\ \delta X(s, \alpha(t_0)) \in I_i}} \nu(s) \log\left(1 + (K-1)e^{-\delta X(s, \alpha(t_0))}\right) \\
&\leq \log(1 + (K-1)e^{-\eta}) \sum_{\substack{s \in T \\ \delta X(s, \alpha(t_0)) \geq \eta}} \nu(s) + \sum_{i=-1}^{p-1} \log\left(1 + (K-1)e^{-\inf I_i}\right) \sum_{\substack{s \in T \\ \delta X(s, \alpha(t_0)) \in I_i}} \nu(s).
\end{aligned}$$

Of course by decomposing

$$\begin{aligned}\bar{L}(t_0) &\le (K-1)e^{-\eta}\nu(G_\eta(t_0)) + \sum_{i=-1}^{p-1} \log\left(1+(K-1)e^{-\inf I_i}\right)\nu\left(\{s\in T : \delta X(s,\alpha(t_0))\in I_i\}\right)\\ &\le (K-1)\,e^{-\eta} + \log\left(1+(K-1)e\right)\nu\left(\{s\in T : \delta X(s,\alpha(t_0))\in I_{-1}\}\right)\\ &+ (K-1)\,e^{-\eta/\kappa}\,\nu\left(\{s\in T : \delta X(s,\alpha(t_0))\in I_p\}\right)\\ &+ (K-1)\sum_{i=0}^{p-1} e^{-\kappa^i}\nu\left(\{s\in T : \delta X(s,\alpha(t_0))\in I_i\}\right).\end{aligned}$$

We now use (50), (51) and (52) to derive

$$\begin{aligned}\bar{L}(t_0) \le& C\,m_0 + (K-1)e^{-\eta} + (K-1)\,\delta_0\,e^{-\eta/\kappa} + \delta_0\,(\sigma+1)\,\frac{\log(1+e\,(K-1))}{\eta^\gamma}\\ &+ \delta_0\,(\sigma+1)(K-1)\sum_{i=0}^{p-1}\left(\frac{(\kappa-1)\,\kappa^i}{2\eta-(1+\kappa)\,\kappa^i}\right)^\gamma e^{-\kappa^i}.\end{aligned} \tag{53}$$

We need to estimate the value of the sum above. The smallest value of the denominator occurs when i is at its maximum value of $p-1=\log_2\eta-1$. Thus, since $\gamma<1$,

$$\sum_{i=1}^{p-1}\left(\frac{(\kappa-1)\,\kappa^i}{2\eta-(1+\kappa)\,\kappa^i}\right)^\gamma e^{-\kappa^i} \le \sum_{i=0}^{p-1}\left(\frac{(\kappa-1)\,\kappa^i}{2\eta-\eta\,(1+1/\kappa)}\right)^\gamma e^{-\kappa^i} = \frac{\kappa^\gamma}{\eta^\gamma}\sum_{i=0}^{p-1}\kappa^{\gamma\,i}e^{-\kappa^i} \le \frac{\kappa^\gamma}{\eta^\gamma}\sum_{i=0}^{p-1}\kappa^i e^{-\kappa^i}.$$

By Lemma 3.2, this yields that

$$\sum_{0=1}^{p-1}\left(\frac{(\kappa-1)\,\kappa^i}{2\eta-(1+\kappa)\,\kappa^i}\right)^\gamma e^{-\kappa^i} \le 2\,\frac{\kappa^\gamma}{\eta^\gamma}. \tag{54}$$

Applying (54) to (53), we arrive at

$$\bar{L}(t_0) \le C\,m_0 + (K-1)\left(e^{-\eta} + \delta_0\,e^{-\eta/\kappa} + \delta_0\,\frac{\sigma+1}{\eta^\gamma}\,(e+2\,\kappa^\gamma)\right),$$

which finishes the proof. □

3.5 Proof of Theorem 2.2

We start with the trivial bound derived from (30) by a sort of Chebyshev inequality

$$\sum_{s\in G_{\eta^*}^c(t)}\nu(s)\,\log(1+e^{-\eta^*}) \le \bar{L}(t) \le \bar{L}(t_0). \tag{55}$$

As a consequence, we obtain from Lemma 3.4 that provided $G_{-1}^c(t_0)=\emptyset$ and $\nu(G_\eta(t_0))>1-\delta_0$,

$$\nu(G_0^c(t)) \le C\,m_0 + \frac{K-1}{\log 2}\,\left(e^{-\eta} + \delta_0\,e^{-\eta/\kappa} + \delta_0\,\frac{\sigma+1}{\eta^\gamma}\,(e+2\,\kappa^\gamma)\right).$$

We can make sure that the right-hand side is less than ε if for some constant $C(K,\sigma,\kappa)$

$$m_0 \le \frac{\varepsilon}{C}, \quad \eta \ge \log\frac{1}{\varepsilon} + C, \quad \delta_0 \le C\,\varepsilon\,\eta^\gamma.$$

Of course one could even be rather explicit on C

$$C = \max\left(\log\frac{K-1}{\log 2},\ \frac{K-1}{\log 2}\,(\sigma+1)\,(e+2\,\kappa^\gamma)\right).$$

This immediately proves the first part of Theorem 2.2.

For the second part, we note that the above choice of η and δ_0 also guarantees that

$$\bar{L}(t_0) \le \varepsilon\,\log 2.$$

Therefore by (55) and for $\eta^* \geq 0$, we have that

$$\nu(G_{\eta^*}^C(t)) \leq \varepsilon \, \frac{\log 2}{\log(1+e^{-\eta^*})} \leq 2 \, \log 2 \, \varepsilon \, e^{\eta^*}.$$

We finish the proof by a technical remark which may, in some cases, improve the estimates. Define a strip

$$S = \{s \, : \, x - \ell \leq \delta X(s, \alpha(t)) \leq x + \ell\}.$$

We may directly apply the bound (49), proved previously, which we recall below: For any $\eta > x + \ell$,

$$\nu(S) < m_0 + \nu(G_\eta^c(t)) \, (\sigma + 1) \, \frac{\ell^\gamma}{(\eta - x)^\gamma}.$$

This implies that

$$\nu(S) < m_0 + 2 \, \log 2 \, \varepsilon \, e^\eta \, (\sigma + 1) \, \frac{\ell^\gamma}{(\eta - x)^\gamma}.$$

One may optimize in η by finding the minimum of

$$f(\eta) = \frac{e^\eta}{(\eta - x)^\gamma} = e^x \, \frac{e^{\eta - x}}{(\eta - x)^\gamma},$$

which is obtained at $\eta - x = \gamma$. Therefore

$$\nu(S) < \begin{cases} m_0 + 2 \, \log 2 \, \varepsilon \, \ell^\gamma \, e^x \, (\sigma + 1) \, \dfrac{e^\gamma}{\gamma^\gamma} & \text{if } \ell \leq \gamma, \\ m_0 + 2 \, \log 2 \, \varepsilon \, \ell^\gamma \, e^{x+\ell} \, (\sigma + 1) & \text{if } \ell > \gamma. \end{cases} \tag{56}$$

This of course has to be compared with the trivial bound

$$\nu(S) \leq \nu(G_{x+\ell}^c) \leq 2 \, \log 2 \, \varepsilon \, e^{x+\ell},$$

which makes it obvious that (56) is only useful if ℓ is small enough.

3.6 Proof of Theorem 2.3

The proof is performed by induction on the number of layers in the network. For this reason it is worth taking a more general perspective on the doubling assumption behind condition B.

For any measure μ on $\mathbb{R}^d$, we consider the following condition

$$\forall u \in \mathbb{R}^d \setminus \{0\}, \; \forall s \in \mathbb{R}, \quad \mu\left(\{x, \; |u \cdot x - s| \leq \kappa\}\right) = \min\Big\{\delta, \; \max\big\{m_0, \; (1+\sigma) \, \mu\left(\{x, \; |u \cdot x - s| \leq 1\}\right)\big\}\Big\}. \tag{57}$$

We denote by $L : \mathbb{R}^N \to \mathbb{R}^d$ any non-linear function that is a combination of a shift, linear operation and as a non-linear function the absolute value; namely

$$L(x)_i = \left| \sum_{j=1}^N M_{ij}(x_j + s_j) \right|, \tag{58}$$

where $s \in \mathbb{R}^N$ is the shift and $M \in M_{N,d}(\mathbb{R})$ is a matrix.

We then have Theorem 2.3 as a consequence of

Theorem 3.5. *Assume that the measure μ satisfies* (57) *and L is given by* (58)*. Then the pushforward $L_\# \mu$ also satisfies* (57) *though with the new constants m_0', δ', σ', κ'.*

We recall that $L_\# \mu$ is defined by $L_\# \mu(O) = \mu(L^{-1}(O))$.

Remark 3.1. Theorem 3.5 allows us to propagate condition B backwards. That is, we transfer condition B on the values taken by the last layer of a DNN before softmax to a similar condition on the second to last layer, then the third to last layer, until we reach a condition on the training set.

To prove Theorem 3.5, we decompose L into a linear part and the absolute value with propositions on each.

Proposition 3.6. *Assume that the measure μ satisfies* (57) *and that $M \in M_{N,d}(\mathbb{R})$. Then the pushforward $M_\# \mu$ also satisfies* (57) *with the same constants.*

Proof. We simply observe that if $|u \cdot x - s| \leq \kappa$ then any y s.t. $M\, y = x$ also satisfies that

$$|(M^T\, u) \cdot y - s| \leq \kappa.$$

Hence

$$M_\# \mu\left(\{x,\ |u \cdot x - s| \leq \kappa\}\right) = \mu\left(\{x,\ |(M^T\, u) \cdot x - s| \leq \kappa\}\right),$$
$$M_\# \mu\left(\{x,\ |u \cdot x - s| \leq 1\}\right) = \mu\left(\{x,\ |(M^T\, u) \cdot x - s| \leq 1\}\right).$$

Since (57) holds on μ for all u, it trivially holds on $M_\#\, \mu$. □

The second and last part consists in handling the absolute value with

Proposition 3.7. *Denote by $A : \mathbb{R}^N \to \mathbb{R}^N$ the absolute value function $A(x) = (|x_1|, x_2 \ldots, x_N)$. Assume that μ solves* (57) *then $A_\#\, \mu$ solves* (57) *with the new constants $\sigma' = \sigma/2$, $\delta' = \delta$, $m_0' = m_0\,(1 + \max(\sigma/4,\ 4/\sigma))$, $\kappa' = \kappa^k$ with $k = 1 + \bar{k}\, \log \frac{1}{\delta}$ for some universal constant $\bar{k}$.*

Proof. Consider any strip

$$S = \{|u \cdot x - a| \leq 1\}.$$

The inverse image $A^{-1}(S)$ consists of S and of

$$S' = \{|u' \cdot x - a| \leq 1\}, \quad u' = (-u_1, u_2, \ldots, u_N).$$

For any κ', we have of course that $A^{-1}(\kappa'\, S) = \kappa'\, S \cup \kappa'\, S'$ and typically we want to apply (57) to those two strips. The issue however is that the intersections $\kappa'\, S \cap S'$ or $\kappa'\, S' \cap S$ may be non empty.

To be more precise, let us decompose

$$S_1 = (\kappa'\, S) \cap S', \quad S_2 = (\kappa'\, S) \setminus S', \quad S_1' = (\kappa'\, S') \cap S, \quad S_2' = (\kappa'\, S') \setminus S.$$

The assumption (57) guarantees that $\mu(S_1) + \mu(S_2) \geq \sigma\, \mu(S)$ for example already for $\kappa' = \kappa$. But we could have that $\mu(S_2) << \mu(S_1)$ leading to an issue since S_1 is contained in S' and hence in $A^{-1}(S)$.

We hence treat differently several cases for some constant C to be chosen later

Case 1: If $\mu(S) \geq C\, \mu(S')$ or $\mu(S') \geq C\, \mu(S)$. Those are equivalent so we may freely assume $\mu(S) \geq C\, \mu(S')$. This is the simplest case where we may take $\kappa' = \kappa$. Applying (57) to S, we find that either $\mu(S) \leq m_0$, $\mu(\kappa\, S) \geq \delta$ or

$$\mu(\kappa\, S) = \mu(S_1) + \mu(S_2) \geq (1 + \sigma)\, \mu(S).$$

If $\mu(S) \leq m_0$ then $\mu(S) + \mu(S') \leq m_0\,(1 + 1/C)$. If $\mu(\kappa\, S) \geq \delta$ then $\mu(\kappa\, S \cup \kappa\, S') \geq \delta$.

On the other hand $\mu(S_1) \leq \mu(S') \leq \mu(S)/C$ so

$$\begin{aligned}\mu(\kappa\, S \cup \kappa S') &\geq \mu(\kappa\, S) \geq (1 + \sigma)\, \mu(S) \geq (1 + \sigma)(1 - 1/C)\, (\mu(S) + \mu(S')) \\ &\geq (1 + \sigma)(1 - 1/C)\, \mu(S \cap S').\end{aligned}$$

This lets us immediately conclude if we choose $1/C = \sigma/4$ for example in which case (57) holds for $A_\#\mu$ with $\delta' = \delta$, $m_0' = m_0(1 + \sigma/4)$, $\sigma' = \sigma/2$ and $\kappa' = \kappa$.

Case 2: If $\mu(S')/C \leq \mu(S) \leq C\, \mu(S')$. We now look at $\kappa^k\, S$ or $\kappa' = \kappa^k$ for some k. Observe that by using (57) k times, we have that

$$\mu(\kappa^k\, S) \geq (1 + \sigma)^k\, \mu(S), \quad \text{or } \mu(\kappa^k\, S) \geq \delta, \quad \text{or finally } \mu(S) \leq m_0.$$

In the second case, we are done again. In the third case, we have that $\mu(S) + \mu(S') \leq (1 + C)\, \mu_0 = (1 + 4/\sigma)\, \mu_0$.

And in the first case, we may deduce that

$$\mu(\kappa^k\, (S \cup \kappa S')) \geq (1 + \sigma)^k\, \mu(S) \geq \frac{(1 + \sigma)^k}{C + 1}\, (\mu(S) + \mu(S')) \geq \frac{(1 + \sigma)^k}{C + 1}\, \mu(S \cup S').$$

We recall that we chose $C = 4/\sigma$ for the previous case so

$$\frac{(1 + \sigma)^k}{C + 1} = \frac{(1 + \sigma)^k}{1 + 4/\sigma} \geq 1 + \frac{\sigma}{2}$$

provided that

$$k \leq \frac{\log(1+\sigma/2)+\log(1+4/\sigma)}{\log(1+\sigma)} \leq 1+\bar{k}\,\log\frac{1}{\sigma},$$

for some constant $\bar{k}$.

Therefore using either $\kappa' = \kappa$ or $\kappa' = \kappa^k \geq \kappa$ with $k = 1+\bar{k}\,\log\frac{1}{\sigma}$, we have that

$$\mu(\kappa'\,(S \cup S')) \geq (1+\sigma/2)\,\mu(S \cup S').$$

Hence (57) holds $A_{\#}\mu$ with $\delta' = \delta$, $m_0' = m_0(1+4/\sigma)$, $\sigma' = \sigma/2$ and $\kappa' = \kappa^k$. □

4 Appendix

4.1 Derivations of Conditions A and B

The goal co conditions A and B is to ensure that $\{\delta X(s,\alpha(t_0)) : s \in G_\eta^C(t_0)\}$ does not concentrate near its minimum β. Conditions A and B accomplish this goal in different ways.

4.1.1 Condition A

If there is no concentration of $\delta X(s,\alpha(t_0))$ near β, we expect that for a small interval whose left endpoint is β, more δX values are in the right half of this interval than in the left. In other words, for small $a > 0$,

$$\nu\left(\{s \in T : \beta \leq \delta X(s,\alpha(t_0)) < \beta + a\}\right) < \nu\left(\{s \in T : \beta + a \leq \delta X(s,\alpha(t_0)) < \beta + 2a\}\right). \tag{59}$$

Since $\beta = \min_{s\in T}\delta X(s,\alpha(t_0))$, we may write (59) equivalently as

$$\nu\left(\{s \in T : \delta X(s,\alpha(t_0)) < \beta + a\}\right) < \frac{1}{2}\nu\left(\{s \in T : \delta X(s,\alpha(t_0)) < \beta + 2a\}\right). \tag{60}$$

Now replace $1/2$ with a continuous parameter Λ:

$$\nu\left(\{s \in T : \delta X(s,\alpha(t_0)) < \beta + a\}\right) < \Lambda\nu\left(\{s \in T : \delta X(s,\alpha(t_0)) < \beta + 2a\}\right). \tag{61}$$

Since all masses are less than 1, we can control concentration near β better by increasing the exponent on the right side of (61):

$$\nu\left(\{s \in T : \delta X(s,\alpha(t_0)) < \beta + a\}\right) < \Lambda\nu\left(\{s \in T : \delta X(s,\alpha(t_0)) < \beta + 2a\}\right)^{\psi+1}, \quad \psi > 0. \tag{62}$$

By letting $x_1 = \beta$ and $x_2 = \beta + 2a$, we may write this condition as

$$\nu\left(\left\{s \in T : \delta X(s,\alpha(t_0)) < \frac{x_1+x_2}{2}\right\}\right) < \Lambda\nu\left(\{s \in T : \delta X(s,\alpha(t_0)) < x_2\}\right)^{\psi+1}, \quad \psi > 0. \tag{63}$$

We will see in the proof of Lemma 3.3 that (63) leads to an excellent bound on $\bar{L}(t_0)$.

The inequality (12) provides a lower bound for $G_\ell(t)$ via $\bar{L}(t_0)$ for $\ell > 0$, but by adjusting condition A slightly, we can improve this inequality. In particular, we would like to apply a condition like (63) with $x_1 = 0$ and $x_2 = 2\ell$ to obtain a bound on $\nu(G_\ell^C(t)) = \nu(\{s \in T : \delta X(s,\alpha(t)) < \ell\})$:

$$1 - \nu(G_\ell(t)) = \nu(\{s \in T : \delta X(s,\alpha(t)) < \ell\}) \leq \Lambda\nu(\{s \in T : \delta X(s,\alpha(t)) < 2\ell\})^{\psi+1} = \Lambda(1-\nu(G_{2\ell}))^{\psi+1} \tag{64}$$

Assuming that the map $\ell \mapsto \nu(G_\ell(t))$ is continuous at $\ell = 0$ (i.e., there is no $s \in T$ with $\delta X(s,\alpha(t)) = 0$), then in the limit as $\ell \to 0$:

$$1 - \nu(G_0(t)) \leq \Lambda(1-\nu(G_0(t)))^{\psi+1}. \tag{65}$$

Dividing both sides of (65) by $1-\nu(G_0(t))$ and using (13), we obtain

$$1 \leq \Lambda(1-\nu(G_0(t)))^{\psi} \leq \Lambda\left(\frac{\bar{L}(t_0)}{\log 2}\right)^{\psi}. \tag{66}$$

If $\bar{L}(t_0)$ is small, (66) may not be satisfied. We do not want to exclude distributions $\{\delta X(s,\alpha(t_0)) : s \in T\}$ with small $\bar{L}(t_0)$, so (63) is insufficient. A simple solution to this problem is to add a new term depending on x_1 to the right side of (63). The new term must vanish when $x_1 = \beta$ so that the loss bound obtained from (63) still

holds. The new term must also not exclude distributions with small $\bar{L}(t_0)$ when $x_1 = 0$. The obvious candidate is $\nu(\{s \in T : \delta X(s, \alpha(t)) < x_1)\})^\phi$ for some power $\phi > 0$, so the TDSM condition becomes

$$\begin{aligned}\nu\left(\left\{s \in T : \delta X(s, \alpha(t_0)) < \frac{x_1 + x_2}{2}\right\}\right) < \Lambda\nu\left(\{s \in T : \delta X(s, \alpha(t_0)) < x_2\}\right)^{\psi+1} \\ + \Lambda\nu\left(\{s \in T : \delta X(s, \alpha(t_0)) < x_1\}\right)^{\phi}, \quad \psi, \phi > 0.\end{aligned} \tag{67}$$

Applying the analysis used to obtain (66) to (67), we get

$$1 \leq \Lambda\left(\frac{\bar{L}(t_0)}{\log 2}\right)^{\psi} + \Lambda\left(\frac{\bar{L}(t_0)}{\log 2}\right)^{\phi-1}. \tag{68}$$

This is satisfied trivially as long as $\Lambda \geq 1$ and $0 < \phi < 1$.

4.1.2 Condition B

Consider a small interval $J = [\beta - x, \beta + x]$ for $x > 0$. There may be some $\delta X(s, \alpha(t_0))$ values in the right half of I, but if the $\delta X(s, \alpha(t_0))$ values do not cluster near β, then there should be more $\delta X(s, \alpha(t_0))$ values to the right of J. In other words, If we increase the width of J from $2x$ to $2\kappa x$ for some $\kappa > 1$, leaving its center in place, the number of $\delta X(s, \alpha(t_0))$ values it contains should increase:

$$\nu\left(\{s \in T : \delta X(s, \alpha(t_0)) \in [\beta - \kappa x, \beta + \kappa x]\}\right) > \nu\left(\{s \in T : \delta X(s, \alpha(t_0)) \in [\beta - x, \beta + x]\}\right). \tag{69}$$

Now let I be any interval. Denoting by κI the interval whose center is the same as I but whose width is increased by a factor of κ, (69) becomes

$$\nu\left(\{s \in T : \delta X(s, \alpha(t_0)) \in \kappa I\}\right) > \nu\left(\{s \in T : \delta X(s, \alpha(t_0)) \in I\}\right). \tag{70}$$

We can strengthen (70) by introducing a factor of $(1 + \sigma)$ on the right hand side, where $\sigma > 0$:

$$\nu\left(\{s \in T : \delta X(s, \alpha(t_0)) \in \kappa I\}\right) > (1 + \sigma)\nu\left(\{s \in T : \delta X(s, \alpha(t_0)) \in I\}\right). \tag{71}$$

Of course T is a finite set, so one can take I arbitrarily small containing a single element, in which case κI may still contain only that same element. Therefore, it is necessary to introduce m_0, a small mass which accounts for when I is so small that (71) may not hold. When the left right hand side of (71) is too small, (71) need not hold:

$$\nu\left(\{s \in T : \delta X(s, \alpha(t_0)) \in \kappa I\}\right) > \max\left\{m_0, (1 + \sigma)\nu\left(\{s \in T : \delta X(s, \alpha(t_0)) \in I\}\right)\right\}. \tag{72}$$

Finally, If I is too large, it is to be expected that increasing its width does not increase its mass much, e.g., if I contains all $\delta X(s, \alpha(t_0))$ values. Moreover, we will use condition B to control the distribution of only a small number of misclassified objects. Therefore, if we introduce δ which is the maximum mass of intervals we will consider with condition B:

$$\nu\left(\{s \in T : \delta X(s, \alpha(t_0)) \in \kappa I\}\right) > \min\left\{\delta, \max\left\{m_0, (1 + \sigma)\nu\left(\{s \in T : \delta X(s, \alpha(t_0)) \in I\}\right)\right\}\right\}. \tag{73}$$

This completes the derivation of condition B.

4.2 Proof of Lemma 3.1

Here we present the proof of Lemma 3.1

Proof. For simplicity, write $\varepsilon K = x$. Since $0 < \varepsilon < 1/(2K)$, we consider $0 < x < 1/2$. First, observe that

$$\sum_{k \geq 0} e^{2^k log(x) - \frac{\eta+1}{2^{k+1}}} - Ce^{-\sqrt{2|\log(x)|(\eta+1)}} \leq 0$$

if and only if

$$C \geq \sum_{k \geq 0} e^{2^k log(x) - \frac{\eta+1}{2^{k+1}} + \sqrt{2|\log(x)|(\eta+1)}}.$$

To simplify the problem, let $y = \sqrt{-\log(x)}$ and $z = \frac{1}{y}\sqrt{\frac{\eta+1}{2}}$. Then

$$2^k \log(x) - \frac{\eta + 1}{2^{k+1}} + \sqrt{2|\log(x)|(\eta + 1)} = -2^{-k}(z - 2^k)^2 y^2.$$

Since $0 < x < 1/2$ and $\eta > 1$, we have $y > \sqrt{\log 2}$ and $z > \frac{1}{y\sqrt{2}}$ or equivalently, $y \geq \max\{\sqrt{\log 2}, 1/(z\sqrt{2})\}$. The problem is reduced to finding C such that

$$C \geq h(y,z) := \sum_{k\geq 0} e^{-2^{-k}(z-2^k)^2 y^2}$$

for all $y \geq \max\{\sqrt{\log 2}, 1/(z\sqrt{2})\}$. We proceed to find

$$C_0 = \max_{y\geq \max\{\sqrt{\log 2}, 1/(z\sqrt{2})\},\ z>0} h(y,z)$$

so that it is always possible choose $C \leq C_0$.

It is easy to calculate

$$\frac{\partial}{\partial y} h(y,z) = -2y \sum_{k\geq 0} 2^{-k} \left(2^k - z\right)^2 e^{-2^{-k}\left(z-2^k\right)^2 y^2} < 0$$

for $y, z > 0$. Thus, for fixed $z = z'$,

$$\max_{y\geq \max\{\sqrt{\log 2}, 1/(z'\sqrt{2})\}} h(y,z') = h(\max\{\sqrt{\log 2}, 1/(z'\sqrt{2})\}, z').$$

Therefore, $C_0 = \max\{a, b\}$ where

$$a = \max_{z\geq 1/\sqrt{2\log(2)}} h(\sqrt{\log 2}, z), \quad \text{and} \quad b = \max_{0<z\leq 1/\sqrt{2\log 2}} h(1/(z\sqrt{2}), z).$$

Therefore, we will estimate a and b.

First we will estimate a. Let

$$p_k(z) = 2^{-2^{-k}(2^k - z)^2}$$

so that $h(\sqrt{\log 2}, z) = \sum_{k\geq 0} p_k(z)$. Note $0 \leq p_k(z) \leq 1$ for all $z \in \mathbb{R}$ and $k \geq 0$. Observe also that $(2^k - z)^2$ is a convex function, so it is bounded below by any tangent line. In particular,

$$(2^k - z)^2 \geq 2^k \left(\frac{3}{4} 2^k - z\right)$$

and

$$(2^k - z)^2 \geq 2^{k+1} \left(z - \frac{3}{4} 2^{k+1}\right).$$

From these, we obtain two upper bounds on $p_k(z)$:

$$p_k(z) \leq \frac{2^k}{z} 2^{-\left(\frac{3}{4}2^k - z\right)}$$

and

$$p_k(z) \leq \frac{2^k}{z} 2^{-2\left(z - \frac{3}{4}2^{k+1}\right)}.$$

The former is useful for $z \leq 2^{k-1}$. The latter is useful for $z \geq 2^{k+1}$. Recall also the following identities:

$$\sum_{k=1}^{n} a^k = \frac{a\left(a^n - 1\right)}{a - 1}$$

and

$$\sum_{k=n}^{\infty} a^k = \frac{a^n}{1-a} \quad \text{for } a < 1.$$

First consider $z < 2$. We may write

$$\begin{aligned}
h(\sqrt{\log 2}, z) &= p_0(z) + p_1(z) + \sum_{k=2}^{\infty} p_k(z) \\
&\le 2 + \sum_{k=2}^{\infty} 2^{-\left(\frac{3}{4}2^k - z\right)} \\
&\le 2 + 4\sum_{k=2}^{\infty} 2^{-\frac{3}{4}2^k} \\
&\le 2 + 4\sum_{k=4}^{\infty} 2^{-\frac{3}{4}k} \\
&= 2 + 4\frac{2^{-\frac{3}{4}\times 4}}{1 - 2^{-3/4}} \\
&\le 2 + \frac{5}{4} \\
&= 3.25
\end{aligned}$$

Now, suppose $z \ge 2$. Then there is an integer $m \ge 2$ so that $2^{m-1} \le z < 2^m$. We write

$$h(\sqrt{\log 2}, z) = p_{m-1}(z) + p_m(z) + \sum_{k=0}^{m-2} p_k(z) + \sum_{k=m+1}^{\infty} p_k(z),$$

and estimate both sums:

$$\begin{aligned}
\sum_{k=0}^{m-2} p_k(z) &\le \sum_{k=0}^{m-2} 2^{-2\left(z - \frac{3}{4}2^{k+1}\right)} \\
&\le 2^{-2^m} \sum_{k=0}^{m-2} 2^{\frac{3}{2}2^{k+1}} \\
&\le 2^{-2^m} \sum_{k=1}^{2^{m-2}} 8^k \\
&= 2^{-2^m} \frac{8(8^{2^{m-2}} - 1)}{7} \\
&= \frac{8}{7} 2^{-2^{m-2}} \\
&\le \frac{4}{7},
\end{aligned}$$

$$\begin{aligned}
\sum_{k=m+1}^{\infty} p_k(z) &\le \sum_{k=m+1}^{\infty} 2^{-\left(\frac{3}{4}2^k - z\right)} \\
&\le 2^{2^m} \sum_{k=m+1}^{\infty} 2^{-\frac{3}{4}2^k} \\
&\le 2^{2^m} \sum_{k=2^{m+1}}^{\infty} 2^{-\frac{3}{4}k} \\
&= 2^{2^m} \frac{2^{-\frac{3}{4}2^{m+1}}}{1 - 2^{-3/4}} \\
&= \frac{2^{-2^{m-1}}}{1 - 2^{-3/4}} \\
&\le \frac{5}{8}.
\end{aligned}$$

Since clearly $p_{m-1}(z)$ and $p_m(z)$ are less than 1, we have

$$h(\sqrt{\log 2}, z) \le 2 + \frac{4}{7} + \frac{5}{8} \approx 3.196$$

for $z \ge 2$. Thus, $a \le 3.25$.

Now we will calculate $b = \max_{0 < z \le 1/\sqrt{2\log 2}} h(1/(z\sqrt{2}), z)$. Let

$$q_k(z) = e^{-2^{-k-1}\left(1 - \frac{2^k}{z}\right)^2}.$$

Then

$$q_k'(z) = \frac{e^{-2^{-k-1}\left(\frac{2^k}{z} - 1\right)^2}\left(\frac{2^k}{z} - 1\right)}{z^2}.$$

The sign of $q_k'(z)$ is the sign of $(2^k/z - 1)$. For $0 \le z \le 1/\sqrt{2\log 2} < 1$, $(2^k/z - 1) > 0$, so $q_k(z)$ is increasing for all $z \in (0, 1)$ and $k \ge 0$. Therefore,

$$b = h\left(1/\sqrt{2}, 1\right)$$

$$
\begin{aligned}
&= \sum_{k\geq 0} e^{-2^{-k-1}(2^k-1)^2} \\
&= \sum_{k\geq 0} e^{-2^{-k-1}(2^{2k}-2^{k+1}+1)} \\
&= \sum_{k\geq 0} e^{-\frac{1}{2}(2^k-2+2^{-k})} \\
&= \sum_{k\geq 0} e^{-\frac{1}{2}(2^{k/2}-2^{-k/2})^2} \\
&= e^0 + e^{-\frac{1}{2}(2^{1/2}-2^{-1/2})^2} + \sum_{k\geq 2} e^{-\frac{1}{2}(2^{k/2}-2^{-k/2})^2} \\
&\leq \frac{9}{5} + \sum_{k\geq 2} e^{-2^{k-2}} \\
&\leq \frac{9}{5} + \sum_{k\geq 1} e^{-k} \\
&= \frac{9}{5} + \frac{1}{e-1} \\
&< \frac{5}{2}.
\end{aligned}
$$

We conclude that $C_0 = \max\{a, b\} \leq 3.25$ □

References

[1] R. Balan, M. Singh, D. Zou, Lipschitz Properties for Deep Convolutional Networks. To appear in *Contemporary Mathematics* 2018.

[2] L. Berlyand, P.-E. Jabin, On the convergence of formally diverging neural net-based classifiers. *Comptes rendus - Mathématique*, **356** (4), 2018 395–405, DOI: 10.1016/j.crma.2018.03.003.

[3] O. Butkovsky. Subgeometric rates of convergence of Markov processes in the Wasserstein metric. *Ann. Appl. Probab.*, **24** (2), 526–552, 2014.

[4] X. Cheng, X. Chen, S. Mallat, Deep Haar scattering networks. *Inf. Inference* **5** (2016), no. 2, 105–133.

[5] J. Cohen. A Coefficient of Agreement for Nominal Scales. Educational and Psychological Measurement, 20(1), 37–46, 1960.

[6] A. Compte, N. Brunel, P. S. Goldman-Rakic, and X.-J. Wang. Synaptic mechanisms and network dynamics underlying spatial working memory in a cortical network model. *Cerebral Cortex* **10**, 910–923, 2000.

[7] W. Czaja, W. Li, Analysis of time-frequency scattering transforms. To appear in *Applied and Computational Harmonic Analysis* 2018.

[8] A. Durmus, G. Fort, E. Moulines, Subgeometric rates of convergence in Wasserstein distance for Markov chains. *Ann. Inst. Henri Poincaré Probab. Stat.* **52** (2016), no. 4, 1799–1822.

[9] W. Gerstner and W. M. Kistler. Spiking neuron models: Single neurons, populations, plasticity. Cambridge university press, 2002.

[10] M. Hairer, J.C. Mattingly, and M. Scheutzow. Asymptotic coupling and a general form of Harris' theorem with applications to stochastic delay equations. Probability Theory and Related Fields, 149(1-2):223–259, 2011.

[11] G. Hinton et al., Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Process. Mag.* **29**, 82–97, 2012.

[12] A. Krizhevsky, I. Sutskever, G. Hinton, ImageNet classification with deep convolutional neural networks. In *Proc. 26th Annual Conf. on Neural Information Processing Systems, Lake Tahoe,* NV, 2012, 109–1098, 2014.

[13] Y. Le Cun, Y. Bengio, G. Hinton, Deep learning. *Nature* **521**, 436–444, 2015.

[14] Y. Le Cun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, L. Jackelt. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems 2* (ed. DS Touretzky), 396–404. San Francisco, CA: Morgan Kaufmann, 1990.

[15] MK. Leung, HY. Xiong, LJ. Lee, BJ. Frey, Deep learning of the tissue regulated splicing code. *Bioinformatics* **30**, i121–i129, 2014.

[16] S. Mallat, Understanding deep convolutional networks. *Phil. Trans. R. Soc. A* **374**, 2016.

[17] S. Mallat, Group invariant scattering. *Comm. Pure Appl. Math.* **65** (2012), no. 10, 1331–1398.

[18] I. Sutskever, O. Vinyals, Q.V. Le, Sequence to sequence learning with neural networks. *Proc. 28th Annual Conf. on Neural Information Processing Systems*, Montreal, Canada, 8–13 2014.