On Lipschitz Bounds of General Convolutional Neural Networks

Dongmian Zou, Radu Balan, and Maneesh Singh

***Abstract*— Many convolutional neural networks (CNN's) have a feed-forward structure. In this paper, we model a general framework for analyzing the Lipschitz bounds of CNN's and propose a linear program that estimates these bounds. Several CNN's, including the scattering networks, the AlexNet and the GoogleNet, are studied numerically. In these practical numerical examples, estimations of local Lipschitz bounds are compared to these theoretical bounds. Based on the Lipschitz bounds, we next establish concentration inequalities for the output distribution with respect to a stationary random input signal. The Lipschitz bound is further used to perform nonlinear discriminant analysis that measures the separation between features of different classes.**

***Index Terms*— Lipschitz bounds, convolutional neural networks, scattering networks, linear programming, adversarial perturbation.**

I. Introduction

CONVOLUTIONAL neural networks (CNN's) have proved to be an effective tool in various image processing tasks. The convolutional layers at different levels are capable of extracting different details from images. As a feature extractor, a CNN is stable to small variations from the input and therefore performs well in a variety of classification, detection and segmentation problems.

The scattering transform [1], [2] is a special type of CNN that can be represented with a multilayer structure (thus also called a scattering network). Although the filters are designed wavelets rather than learned, the scattering transform proves to be an effective feature extractor. In the mathematical analysis of scattering network, it is proved [1, Th. 2.10] that the scattering transform is invariant to translation. However, this is true only if we take the full representation where the limiting scale $J \to \infty$. In practice, we take a finite J and therefore only have stability with respect to translation. The mathematical analysis for the stability properties of scattering networks is not limited to wavelets: for instance, it is generalized by using semi-discrete frames as filters in [3], [4], and time-frequency atoms as filters in [5]. In all these cases, the scattering transforms are Lipschitz continuous with Lipschitz constant $L = 1$, which is an important factor for the provable stability properties.

A scattering network extracts features from every convolutional layer. This is not the case for a general CNN. In [6] a CNN is defined as a neural network which has at least one convolution unit. Many widely-adapted CNN models have either a sequential structure (e.g. the AlexNet [7]) or a more complex feed-forward structure (e.g. the GoogleNet [8]). For those models, stability is still an important issue. Intuitively, keeping the same energy in the feature, we should train the network so that the features are as stable as possible to small perturbations before using dense layers to do the classification. In [9], the authors use the large Lipschitz bound of each single layer to illustrate that the AlexNet could be very unstable with respect to small perturbation on the input image. In fact, changing a small number of pixels could "fool" the network so that it produces wrong classification results. In general, a small Lipschitz bound of the entire transform implies the robustness of a CNN to small perturbations.

"Fooling" networks is naturally connected to adversarial networks. Indeed, Lipschitz bounds are already used in training adversarial networks other than just quantitatively showing the robustness. In [10], the authors propose an objective function for training generative adversarial networks where they use (the distance between) the Lipschitz constant (and 1) as a penalty term. However, there is no direct way to impose it. Later in [11], the authors use a gradient penalty inspired by the fact that a function is 1-Lipschitz if its gradient is bounded by 1.

Although it plays an important role in deep learning, the study of Lipschitz bounds is not completely addressed by existing literature. The frameworks in [1]–[5] analyze the 1-Lipschitz transformations but are limited to the scattering transforms and do not generalize automatically to general CNN's. Reference [9] provides a Lipschitz bound using the product of Bessel bounds of each layer, but in general lacks tightness for non-sequential models such as the scattering network. Our paper fills in the gap between these approaches, by providing a unified stability analysis that applies to both the scattering networks (as in [1]–[5]) and to the more gen-

Manuscript received July 28, 2017; revised July 19, 2019; accepted November 12, 2019. Date of publication December 23, 2019; date of current version February 14, 2020. The work of Dongmian Zou was supported in part by NSF under Grant DMS-1413249. The work of Radu Balan was supported in part by NSF under Grant DMS-1413249 and Grant DMS-1816608, in part by Army Research Office (ARO) under Grant W911NF-16-1-0008, and in part by Laboratory for Telecommunication Sciences (LTS) under Grant H9823031D00560049.

Dongmian Zou was with the Department of Mathematics, University of Maryland, College Park, MD 20742 USA, and also with the Institute for Mathematics and its Applications, University of Minnesota, Minneapolis, MN 55455 USA. He is now with the School of Mathematics, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: dzou@umn.edu).

Radu Balan is with the Department of Mathematics, University of Maryland, College Park, MD 20742 USA (e-mail: rvbalan@cscamm.umd.edu).

Maneesh Singh is with Verisk Analytics, Jersey City, NJ 07310 USA (e-mail: Maneesh.Singh@verisk.com).

Communicated by E. Abbe, Associate Editor for Machine Learning.

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2019.2961812

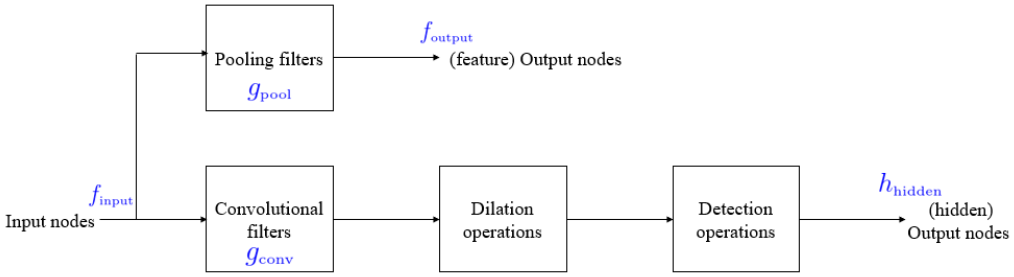


Fig. 1. The structure of a network layer. The network we consider consists of a number of layers, which makes the structure "deep".

eral convolutional networks. Our framework is flexible and compatible with architectures that may or may not generate outputs from hidden layers. The results presented in this paper are optimal for scattering networks and in general tighter than taking the product of Bessel bounds in each layer. Our focus is on estimation of these Lipschitz bounds, and how they relate to stochastic processes. We discuss how the Lipschitz bounds can be used for classification, but we do not focus on extending these results to generative adversarial networks. Instead we study numerically a few examples, including the AlexNet and the GoogleNet.

For practical CNN's such as the AlexNet and the GoogleNet, we discovered that the estimated bounds are about three orders of magnitude more conservative than the numerically estimated local Lipschitz bounds. We give a detailed discussion on the source of looseness in the main text. Surprisingly, even the local Lipschitz bounds are not close to the empirical bounds evaluated over pairs of inputs. Specifically, the empirical bounds are still three orders of magnitude smaller: the largest local Lipschitz bound is obtained numerically to be of order 1, whereas on an extensive study using ImageNet [12] images, the ratio between the energy of output variation to the energy of input variation is of the order 10^{-3}. To bridge this gap, we observe the change of the effect of ReLU units and max poolings, and propose a simple model that estimate the "empirical" ReLU units and max poolings. Interestingly, the resulting estimate based on the local Lipschitz bound is much closer to the empirical bound.

Before discussing our framework in mathematical details, we first overview the CNN architecture considered in this paper (the details are given in the main text) and provide some guidance to the notations. The framework is applicable to the scattering network [1], [2], the AlexNet [7] and the GoogleNet [8]. It can also be used to analyze models such as Long-Short Term Memory [13]. We state the theory for continuous signals, but explain how to adapt it for the discrete case (which is the case for AlexNet and GoogleNet). We focus on the feature extraction part of the network and do not discuss the fully connected layers that are usually put on top of the structure, though the fully connected layers can be regarded as a special case of convolutional layers. The CNN that we consider has a feed-forward structure and consists of different layers (it is possible to use infinitely many layers to represent a feedback structure). We define the layers according to the convolutions. Specifically, each layer consists of input nodes, convolutional filters, detection / merge operations, pooling filters, output (feature) nodes and (hidden) output nodes.

- The *input nodes* are signals passed to the current layer. That could come from the hidden output nodes in the previous layer, or the input signal to the network.
- The *convolutional filters* are the filters that perform convolution with the signal from the input nodes. Suppose y is the signal in an input node, and g is the convolutional filter, the output is

$$z(t) = y * g(t) = \int y(t-s)g(s)ds = \int y(s)g(t-s)ds \ .$$

- The *pooling filters* are low-pass convolutional filters that lower the complexity before the feature is extracted as output. Note that these are still linear translation-invariant operations which are commonly used in scattering networks. The nonlinear operations such as max pooling and average pooling are contained in the detection operations.
- The *(feature) output nodes* are outputs of the convolutional neural network. As we specified earlier, these nodes form a subset of the representation. Once the representation is extracted, the specific machine learning tasks, such as classification and prediction, will be performed on the representation.
- The *dilation* operations are "changes of scale" on the space variables. A dilation operation on a signal $f(x)$, $x \in \mathbb{R}^d$, can be represented using a $d \times d$ invertible matrix D. The dilated signal is $f(Dx)$.
- The *detection* operations are nonlinear operations that apply pointwise to the output of the convolutional filters. The nonlinearities have Lipschitz constant 1 (e.g. ReLU functions). In addition to applying the nonlinearity, the outputs can be aggregated by *merge* operations to produce a single output for dimensionality reduction. The max pooling and average pooling are modeled in this manner.
- The *(hidden) output nodes* are signals that propagate to the next layer. The signals at the output nodes are identical to those at the input nodes of the next layer.

In this paper, unless otherwise specified, we use f to denote the input and output signals of a CNN, h to denote the hidden features, and g to denote filters. The input signal on the d-dimensional Euclidean space has finite energy, that is, $f \in L^2(\mathbb{R}^d)$. The Fourier transform of f, denoted by $\hat{f}$, is defined formally to be

$$\hat{f}(\omega) = \int_{\mathbb{R}^d} f(x)e^{-2\pi i\omega\cdot x}dx \ , \quad \omega \in \mathbb{R}^d \ .$$

TABLE I
SUMMARY OF THE THREE NOTIONS OF LIPSCHITZ CONSTANTS

Lipschitz constant	Method for computation	Remarks
Analytical estimate	Linear program involving several types of Bessel bounds	Optimal for scattering networks but conservative in other cases
Local linearization	Linearize activation functions and pooling operations around an input	Accurate locally but very local and expensive to compute
Stochastic estimate	Consider stochastic model and an average effect of nonlinearities	Agrees with empirical divided differences but inaccurate locally

and we refer the readers to [14] for rigorous definitions for f when $f \in L^2(\mathbb{R}^d)$ or when f is a generalized function. The filters of CNN are taken from the Banach Algebra of tempered distributions with an essentially bounded Fourier Transform, that is,

$$\mathcal{B} = \left\{ g \in \mathcal{S}'(\mathbb{R}^d), \|\hat{g}\|_\infty < \infty \right\} \ . \tag{1}$$

We have a detailed discussion of this algebra in Appendix C. We use $\|\cdot\|_p$ to denote the L^p-norm corresponding to the Lebesgue integral. For a matrix A, A^t denotes its transpose, and A^* denotes its conjugate transpose. We use $\|A\|_{\text{op}} = \max_{\|x\|_2=1} \|Ax\|_2$ to denote the operator norm of A, $\|A\|_* = \text{trace}(\sqrt{A^*A})$ to denote its nuclear norm, and $\|A\|_{\text{Fr}} = \sqrt{\text{trace}(A^*A)}$ to denote its Frobenius norm.

A. Contribution of the Work

In this paper, we analyze the Lipschitz bound of a general CNN and its application in stationary processes and nonlinear discriminant analysis. We first introduce a general framework which is able to model CNN specific operations. According to the framework, we derive a linear program of which the optimal value is a Lipschitz bound of the CNN with respect to the Bessel bounds of the layers.

For large classes of scattering networks the linear program yields an optimal estimate of the Lipschitz bound. In other feed-forward networks, the estimate is usually conservative. To address this issue, two different local estimates are proposed. The first estimate is based on local linearization around the operating input. The second estimate takes into account long-range interactions between activation maps for two different inputs. Extensive experiments were performed to compare the three Lipschitz constants with empirical divided differences from CNN outputs corresponding to input samples.

For clarity, the three notions of Lipschitz constants are summarized in Table I.

In this paper, Lipschitz constant is defined with respect to changes in the input. Such Lipschitz constants are then used to perform nonlinear discriminant analysis. In contrast, [11], [15], [16] utilize the gradient with respect to the input instead of the Lipschitz constant. It is worth noting also that many other papers on neural networks discuss gradients with respect to the network parameters, for instance, the neural tangent kernel [17], [18] and the mean-field analysis [19]. This, however, is different than the approach in the current paper.

The paper is organized as follows. Section II sets up the mathematical problem by defining the layers of a CNN. Section III states the results on estimating the Lipschitz bounds. Section IV illustrates examples from the scattering network to the AlexNet and the GoogleNet. Section V discusses how the Lipschitz bounds relate to concentration results for stationary processes on CNN's. Section VI discusses using the Lipschitz bounds to construct a nonlinear discriminant.

II. DEFINING A CNN

The overall structure of an M-layer CNN is illustrated in Figure 2. The picture shows how an input propagates through the layers while generating outputs at each layer. The details of the layers are described in the following two subsections. If no merge operation is present at a certain layer, the convolutional layer is modeled as a linear operation followed by nonlinearity; if there are merge operations, different types of merge operations are modeled separately.

A. A Layer Without Merge Operations

If a certain layer does not contain any merging, we can model the filters as a linear transform from signals on all the input nodes. In the m-th layer, the set of input nodes is denoted by $\mathcal{I}_m = \{N_{m,1}, N_{m,2}, \cdots, N_{m,n_m}\}$ and the set of output nodes by $\mathcal{O}_m = \{N'_{m,1}, N'_{m,2}, \cdots, N'_{m,n'_m}\}$. Further, the set of output generating nodes is denoted by $\mathcal{V}_m = \{V_{m,1}, V_{m,2}, \cdots, V_{m,n_m}\}$. With this notation, let $h_{m,1}, h_{m,2}, \cdots, h_{m,n_m}$ be the signals on the input nodes, a linear operator $T^{(m)}$ is a n'_m-by-n_m array of filters $T^{(m)}_{n',n}$ in $\mathcal{B}$ such that

$$h^{\spadesuit}_{m,n'} = \sum_{n=1}^{n_m} T^{(m)}_{n',n} * h_{m,n} \ , \quad 1 \le n' \le n'_m,$$

is received before downsampled by the d-by-d invertible matrix $D_{m,n'}$ and sent into a nonlinearity $\sigma_{m,n'}$ to output

$$h'_{m,n'}(x) \ = \sigma_{m,n'} \left(h^{\spadesuit}_{m,n'}(D_{m,n'}x) \right) \ .$$

Moreover, let $\phi_{m,1}, \cdots, \phi_{m,n_m}$ define the filters for the output generating nodes. The signals at the feature output nodes are

$$f_{m,n} = h_{m,n} * \phi_{m,n} \ .$$

For the m-th layer, we define three types of Bessel bounds as follows. For each $\omega \in \mathbb{R}^d$, denote $\hat{T}^{(m)}(\omega)$ to be the $n'_m \times n_m$ matrix that contains the Fourier transform $\hat{T}^{(m)}_{n',n}$ of $T^{(m)}_{n',n}$ at ω, for $1 \le n \le n_m$, $1 \le n' \le n'_m$. Also for each ω, denote $\hat{\Psi}^{(m)}(\omega)$ to be the $n_m \times n_m$ diagonal matrix that has $\hat{\phi}_{m,n}(\omega)$, the Fourier transform of the convolutional filter at ω, as its (n,n) entry. Let $\Delta^{(m)}$ be the $n'_m \times n'_m$ diagonal matrix

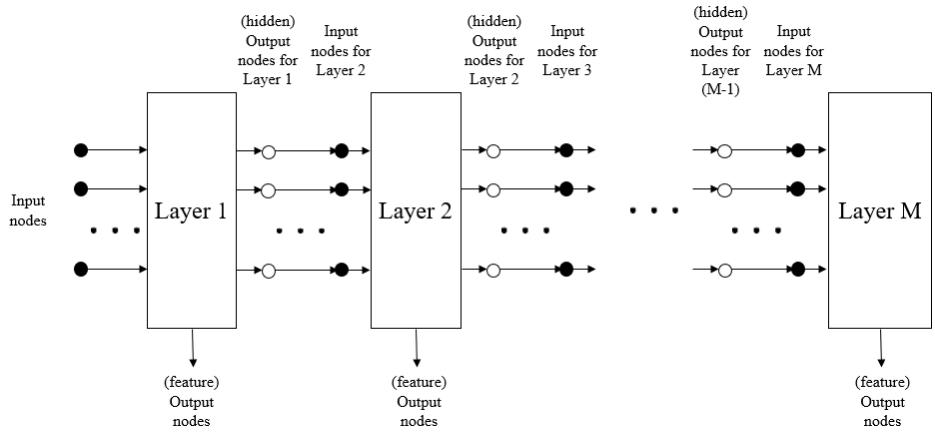


Fig. 2. The detail of an M-layer CNN. The signals at output nodes are identical as at input nodes in the next layer. There may or may not be output generation in each layer.

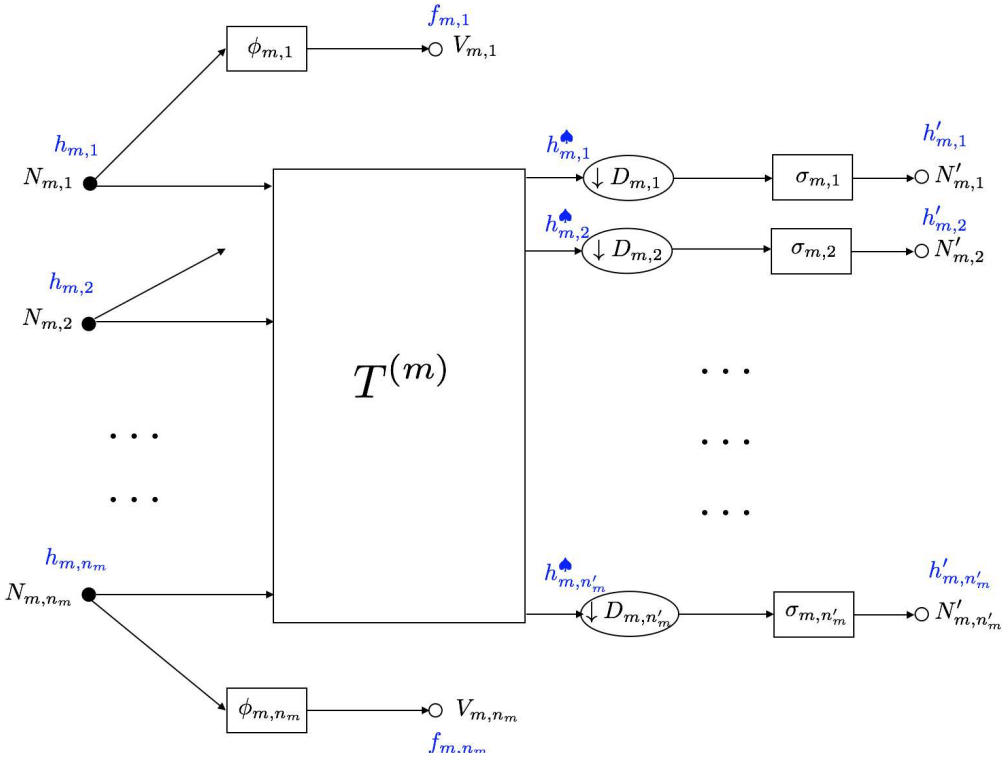


Fig. 3. The detail of the m-th layer with no merge operations. $N_{m,n}$ denote the input nodes, $N'_{m,n'}$ denote the hidden output nodes, $V_{m,n}$ denote the feature output nodes. $\phi_{m,n}$ denote the pooling filters, $D_{m,n'}$ denote the dilation factors, and $\sigma_{m,n'}$ denote the 1-Lipschitz nonlinearities. The notations in blue represent the signals at each node. $h_{m,n}$ denote the input signals of the layer. $h'_{m,n'}$ denote the hidden output signals that are passed to the next layers. $h^{\spadesuit}_{m,n'}$ denote the signals received after passing the linear operator $T^{(m)}$. $f_{m,n}$ denote the signals at the feature output nodes.

with $(\det D_{m,n'})^{-1/2}$ as its (n', n') entry. The 1st type Bessel bound for the m-th layer is defined to be

$$B_m^{(1)} = \sup_{\omega\in\mathbb{R}^d} \left\| \begin{bmatrix} \Delta^{(m)}\hat{T}^{(m)}(\omega) \\ \hat{\Psi}^{(m)}(\omega) \end{bmatrix} \right\|_{\text{op}}^2, \tag{2}$$

the 2nd type Bessel bound for the m-th layer is defined to be

$$B_m^{(2)} = \sup_{\omega\in\mathbb{R}^d} \left\| \Delta^{(m)}\hat{T}^{(m)}(\omega) \right\|_{\text{op}}^2, \tag{3}$$

and the 3rd type Bessel bound is defined to be

$$B_m^{(3)} = \sup_{\omega\in\mathbb{R}^d} \left\| \hat{\Psi}^{(m)}(\omega) \right\|_{\text{op}}^2. \tag{4}$$

In general, the Bessel bound quantifies how the energy is magnified by convolution. The bound is finite if the filters form semi-discrete frames (see [4, Appendix A]). Our definition acts in the spectral domain and it naturally yields estimates of the the Lipschitz bounds: see (30) in Appendix A. The need for three types of Bessel bounds is related to different types of energy mixing: input-to-combined hidden and feature

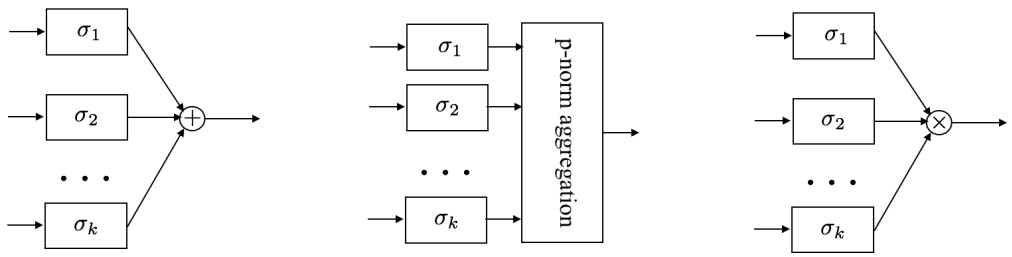
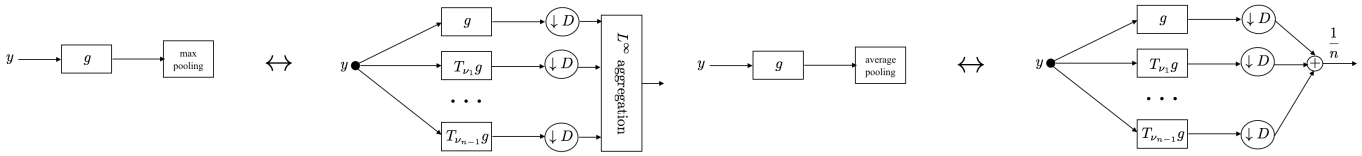


Fig. 4. The three types of merge. Left: Type I - taking sum of the inputs; middle: Type II - taking p-norm aggregation of the inputs; right: Type III - taking pointwise product of the inputs.



(a) max pooling (b) average pooling

Fig. 5. In the continuous case, the max pooling is modeled as Type II aggregation for $p = \infty$, and the average pooling is modeled as Type I aggregation. Here T_ν denotes translation by ν: $T_\nu g(x) := g(x - \nu)$.

output nodes, input-to-hidden output nodes, and input-to-feature output nodes. Intuitively, $B_M^{(1)}$ is the Bessel bound for the frame composed of both $T_{n',n}^{(m)}$ and $\phi_{m,n}$, $B_M^{(2)}$ is for the frame of $T_{n',n}^{(m)}$ and $B_M^{(3)}$ is for the frame of $\phi_{m,n}$ only. For a layer with merge operations, the Bessel bounds share the same intuition, but their estimates have different mathematical representations. We describe that in the next section.

B. A Layer With Merge Operations

There are three types of merging. Type I takes inputs $y_1, \cdots, y_k$ from k channels, applies a nonlinearity function $\sigma_1, \cdots, \sigma_k$ respectively, and then sums them up. That is, the output is

$$z = \sum_{j=1}^{k} \sigma_j(y_j) \ . \tag{5}$$

Type II takes inputs $y_1, \cdots, y_k$ from k channels, apply a nonlinearity on each signal, and then aggregates them by a pointwise p-norm. That is, the output is

$$z = \left(\sum_{j=1}^{k} |\sigma_j(y_j)|^p \right)^{1/p} \ , \text{ if } p < \infty \ ; \tag{6}$$

and

$$z = \max_{j=1,\cdots,k} |\sigma_j(y_j)| \ , \text{ if } p = \infty \ . \tag{7}$$

Type III takes inputs $y_1, \cdots, y_k$ from k channels, apply a nonlinearity on each signal, and then performs a pointwise multiplication. The nonlinearity σ_j should satisfy $\|\sigma_j\|_\infty \leq 1$ for each j. The output is

$$z = \prod_{j=1}^{k} \sigma_j(y_j) \ . \tag{8}$$

We point out that the standard pooling operations in most discrete CNN's can be modeled in the continuous case by these merge operations. Specifically, *max pooling* is the operation of taking the maximal element among those in the same sub-regions. We can use translations and dilations to separate elements in a sub-region to distinct channels, as illustrated in Figure 5a. Then the L^∞-aggregation select the largest element and performs the max pooling. *Average pooling* replaces "taking the max" by "taking the average". Similarly to max pooling, it can be done by taking the sum as illustrated in Figure 5b. A concrete example illustrates max pooling as implemented by this framework. Similar implementation can realize average pooling. Consider the finite signal $(1, 3, 4, 2, 1, 5, 6, 7)$ in Figure 6 for which we want to apply max pooling with size $=$ 2 and stride $= 2$. Then the max pooled signal is $(3, 4, 5, 7)$, where each entry is the larger value within each pair. Consider now the (circular) translation by 1 pixel of the first signal, that is $(3, 4, 2, 1, 5, 6, 7, 1)$ together with the original signal (the middle two signals in the figure). Apply the dilation operator where we discard the second pixel in each consecutive pair of pixels. Thus we obtain $(1, 4, 1, 6)$ and $(3, 2, 5, 7)$ respectively. Now a Type II aggregation with $p = \infty$ selects the larger value between two pixels at the same position, and therefore results in $(3, 4, 5, 7)$, which is the same as the max pooling operation applied on the original signal.

Suppose there are n_m nodes in the m-th layer (this works for $m < M$ but $m = M$ is a similar case in which there is no hidden output node). The set of these input nodes is denoted by $\mathcal{I}_m = \{N_{m,1}, N_{m,2}, \cdots, N_{m,n_m}\}$. Within the layer, each node is connected to several filters. The filter can be either a pooling filter, or a convolutional filter. Associated with $N_{m,n}$ for $1 \leq k \leq n_m$, the pooling filter is denoted to be $\phi_{m,n}$, and the convolutional filters to be $G_{m,n} = \{g_{m,n;1}, \cdots g_{m,n;k_{m,n}}\}$. The set of filters in the m-th layer is thus

$$G_m = \cup_{n=1}^{n_m} G_{m,n} \ . \tag{9}$$

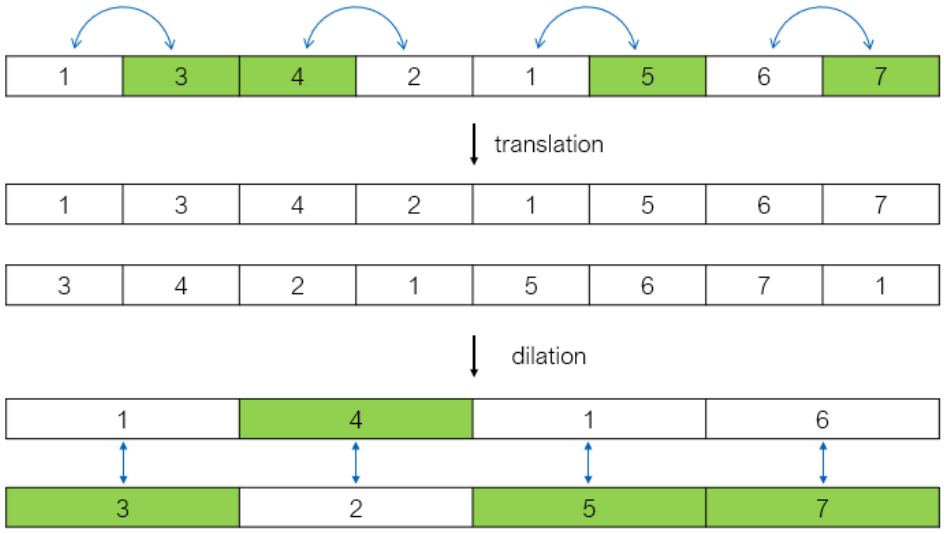


Fig. 6. A concrete example for the max pooling.

Each filter $g_{m,n;k_{m,n}}$ is naturally classified into one of three categories according to the three types of merging: if a filter is merged using Type I operation, then it is classified as a Type I filter; in the same manner we define Type II and Type III filters. If a filter is not merged with other filters, we classify it as Type I (with $k = 1$ in the first picture in Figure 4). We denote the sets of all Type-I, II, III filters by τ_1, τ_2, τ_3, respectively.

Note that each filter is associated with one and only one output node. Let $\mathcal{O}_m = \{N'_{m,1}, N'_{m,2}, \cdots, N'_{m,n'_m}\}$ denote the set of output nodes of the m-th layer. Note that $n'_m = n_{m+1}$ and there is a one-one correspondence between $\mathcal{O}_m$ and $\mathcal{I}_{m+1}$. The output nodes automatically divide G_m into n'_m disjoint subsets $G_m = \cup_{n'=1}^{n'_m} G'_{m,n'}$, where $G'_{m,n'}$ is the set of filters merged into $N'_{m,n'}$. Further, $\mathcal{V}_m = \{V_{m,1}, V_{m,2}, \cdots, V_{m,n_m}\}$ denote the set of output generating nodes. The detail of one layer is illustrated in Figure 7.

For each filter $g_{m,n;k}$, we define the associated multiplier $l_{m,n;k}$ in the following way: suppose $g_{m,n;k} \in G'_{m,n'}$, let $K = \left|G'_{m,n'}\right|$ denote the cardinality of $G'_{m,n'}$. Then

$$l_{m,n;k} = \begin{cases} K & , \text{if } g_{m,n;k} \in \tau_1 \cup \tau_3 \\ K^{\max\{0,2/p-1\}} & , \text{if } g_{m,n;k} \in \tau_2 \end{cases} \tag{10}$$

We define the 1st type Bessel bound for the node $N_{m,n}$ to be

$$B^{(1)}_{m,n} = \left\| \left|\hat{\phi}_{m,n}\right|^2 + \sum_{k=1}^{k_{m,n}} l_{m,n;k} D^{-d}_{m,n;k} \left|\hat{g}_{m,n;k}\right|^2 \right\|_\infty , \tag{11}$$

the 2nd type Bessel bound to be

$$B^{(2)}_{m,n} = \left\| \sum_{k=1}^{k_{m,n}} l_{m,n;k} D^{-d}_{m,n;k} \left|\hat{g}_{m,n;k}\right|^2 \right\|_\infty , \tag{12}$$

and the 3rd type Bessel bound to be

$$B^{(3)}_{m,n} = \left\|\hat{\phi}_{m,n}\right\|^2_\infty . \tag{13}$$

Further, we define the 1st type Bessel bound for the m-th layer to be

$$B^{(1)}_m = \max_{1\le n\le n_m} B^{(1)}_{m,n} , \tag{14}$$

the 2nd type Bessel bound to be

$$B^{(2)}_m = \max_{1\le n\le n_m} B^{(2)}_{m,n} , \tag{15}$$

and the 3rd type Bessel bound to be

$$B^{(3)}_m = \max_{1\le n\le n_m} B^{(3)}_{m,n} . \tag{16}$$

III. Calculating the Lipschitz Bound

Suppose we are given a CNN within the framework given in Section II. For any input signal f and $\tilde{f}$, let f_N be the output for f from the node N, and $\tilde{f}_N$ be the output for $\tilde{f}$ from the node N. Let $\mathcal{V} = \cup_{m=1}^{M} \mathcal{V}_m$ be the collection of all output generating nodes. We say L is a *Lipschitz bound* for the CNN if

$$\sum_{N\in\mathcal{V}} \left\| f_N - \tilde{f}_N \right\|_2^2 \le L \left\| f - \tilde{f} \right\|_2^2 . \tag{17}$$

The map $\Phi : L^2(\mathbb{R}^d) \to [L^2(\mathbb{R}^d)]^{|\mathcal{V}|}$ induced by the CNN is defined by

$$\Phi(f) = (f_N)_{N\in\mathcal{V}} . \tag{18}$$

A norm $|||\cdot|||$ defined on $[L^2(\mathbb{R}^d)]^{|\mathcal{V}|}$ by

$$\left|\left|\left|(f_N)_{N\in V}\right|\right|\right| = \left(\sum_{N\in\mathcal{V}} \|f_N\|_2^2\right)^{1/2}$$

is well defined and $L_c = \sqrt{L}$ is a *Lipschitz constant* in the sense that

$$\left|\left|\left|\Phi(f) - \Phi(\tilde{f})\right|\right|\right| \le L_c \left\| f - \tilde{f} \right\|_2 . \tag{19}$$

We have the following theorem for calculating the Lipschitz bound.

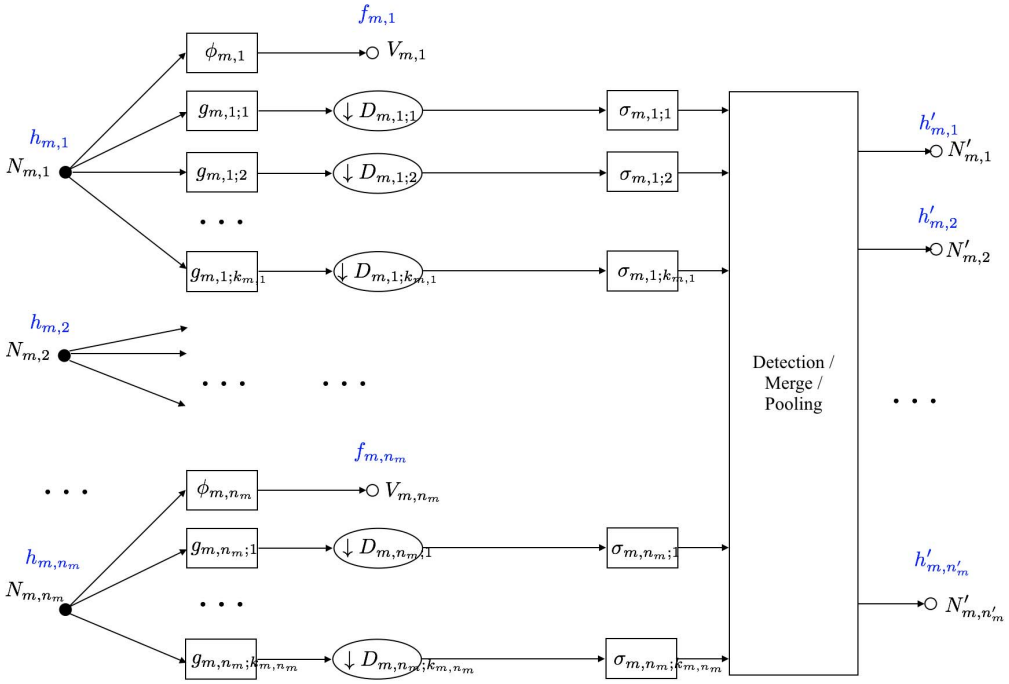


Fig. 7. The detail of one layer with merging. $N_{m,n}$ denote the input nodes, $N'_{m,n'}$ denote the output nodes, $V_{m,n}$ denote the output generating nodes. $\phi_{m,n}$ and $g_{m,n}$ denote the filters, $D_{m,n;k}$ denote the dilation factors. $\sigma_{m,n;k}$ denote the 1-Lipschitz nonlinearities (for illustration we put them outside the merge box, but they belong to the merge operations where we defined the three types of merge). The notations in blue represent the signals at the nodes. $h_{m,n}$ denote the input signals of the layer. $h'_{m,n'}$ denote the output signals that are passed to the next layers. $f_{m,n}$ denote the signals at the feature output nodes.

Theorem III.1. *Consider a CNN in the framework of Section II, with M layers and in the m-th layer it has 1st type Bessel bound $B_m^{(1)}$, 2nd type Bessel bound $B_m^{(2)}$ and 3rd type Bessel bound $B_m^{(3)}$. Then the CNN induces a nonlinear map Φ that is Lipschitz continuous, and its Lipschitz bound is given by the optimal value of the following linear program*:

$$\begin{aligned} \max \quad & \sum_{m=1}^{M} z_m \\ \text{s.t.} \quad & y_0 = 1 \\ & y_m + z_m \le B_m^{(1)} y_{m-1}, \quad 1 \le m \le M-1 \\ & y_m \le B_m^{(2)} y_{m-1}, \quad 1 \le m \le M-1 \\ & z_m \le B_m^{(3)} y_{m-1}, \quad 1 \le m \le M \\ & y_m, z_m \ge 0, \quad \textit{for all } m \ . \end{aligned} \tag{20}$$

The proof of Theorem III.1 is given in Appendix A. We remark here that the linear program presented as (20) is feasible, since one obvious feasible point is $y_m = 0$ for $1 \le m \le M-1$ and $z_m = 0$ for $1 \le m \le M$. Moreover, the solution is bounded since all z_m's are bounded by $B_m^{(3)} \prod_{m'=1}^{m-1} B_{m'}^{(2)}$ according to the third and fourth inequalities in (20). In practice, either the simplex method or the interior method (see, for instance [20, Ch. 13 and 14]) can be used to solve this linear program, and they run in polynomial time with respect to the number of layers. If we are in the discrete case, say for pixel images, then we need to compute the Bessel bounds, which relies on the Fast Fourier Transforms that grows as $O(N \log N)$ with the dimensionality of filters. Although the complexity is not high, a Lipschitz bound computed via a linear program is still not intuitive. We give more explicit estimates of the Lipschitz bound in the following corollaries.

Corollary III.2. *Consider a CNN in the framework of Section II, with M layers and in the m-th layer it has 1st type Bessel bound $B_m^{(1)}$. Then the CNN induces a nonlinear map that is Lipschitz continuous, and its Lipschitz bound is given by*

$$\prod_{m=1}^{M} \max\{1, B_m^{(1)}\} \ . \tag{21}$$

Corollary III.3. *Consider a CNN in the framework of Section II, with M layers and in the m-th layer it has 2nd type Bessel bound $B_m^{(2)}$ and generating bound $B_m^{(3)}$. Then the CNN induces a nonlinear map that is Lipschitz continuous, and its Lipschitz bound is given by*

$$B_1^{(3)} + \sum_{m=2}^{M} B_m^{(3)} \prod_{m'=1}^{m-1} B_{m'}^{(2)} \ . \tag{22}$$

The proof of Corollary III.3 is an immediate consequence of Theorem III.1, specifically from the third and fourth inequalities of (20). The proof of Corollary III.2 is given in Appendix B. We remark here that both corollaries give a more conservative bound compared to the linear program (20) because both results restrict the variables to a subset of the feasible region. The idea of using Bessel bounds is also addressed in [9] where the authors compute the Bessel bounds of each layer of the AlexNet, and in [4] where the authors set $B_m \le 1$ to make the CNN a 1-Lipschitz map. We return to the AlexNet in the following section.

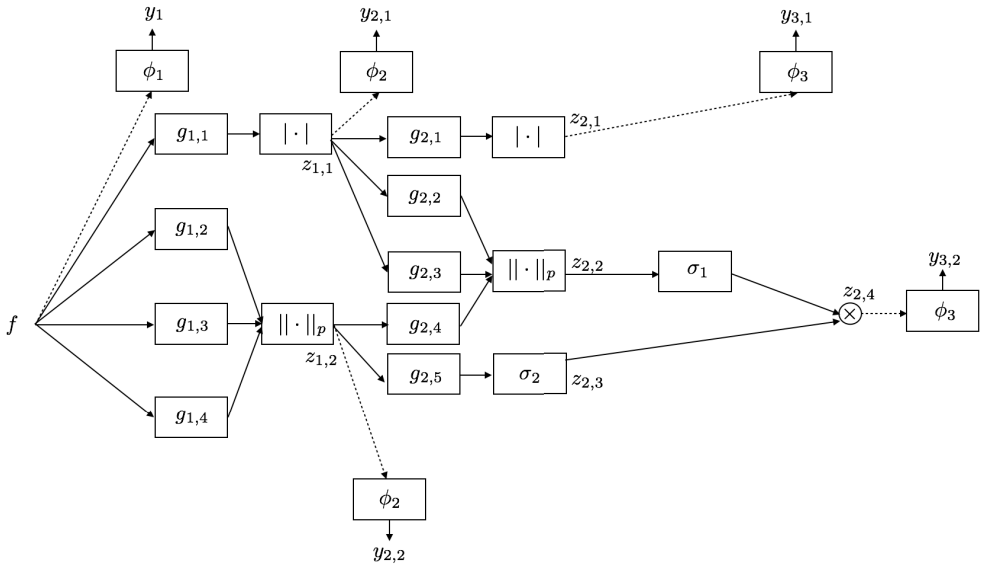


Fig. 8. The toy example that also appears in [21]. Note that we have different choices of filters in the numerical experiment.

Subject to the knowledge of the three types of Bessel bounds in each layer, the estimate given by the linear program (20) is tight. However three issues may prevent its tightness. First, except for the scattering network when defined for continuous inputs, most of CNN's consider discrete time inputs only. Second, even subject to the same Bessel bounds, different filters may produce much smaller Lipschitz bounds. Suboptimality occurs in cases where the signal that achieves the Bessel bound for Layer $m+1$ is not in the range of Layer m. Third, in some practical applications when signals are modeled as samples drawn from certain distributions, then the emphasis is on local stability around the operating distributions, whereas the global Lipschitz bound may be irrelevant.

We address these issues by looking at three examples: the scattering network, a toy network that includes all three types of merge operations we consider in this paper, and the well-known AlexNet and GoogleNet.

IV. Examples

A. Scattering Network

The scattering network in [1], [2] is a 1-Lipschitz map. In each layer the filters are designed to form Parseval wavelet frames using multi-resolution analysis. Such design leads to $B_{m,n}^{(1)} = B_{m,n}^{(2)} = B_{m,n}^{(3)} = 1$, for all m, n. Then Corollary III.2 simply yields a Lipschitz bound $L = 1$ which is tight. We refer the readers to [21, Sec. 4.1] for a detailed discussion.

B. A Toy Example That Contains Merge Operations

The scattering network enjoys $B_{m,n}^{(1)} = B_{m,n}^{(2)} = B_{m,n}^{(3)} = 1$ for all m, n since it is tightly related to wavelet decompositions. In many CNN's we don't have feature output from hidden layers and therefore $B_{m,n}^{(1)} = B_{m,n}^{(2)}$, whence the results in Corollary III.2 coincide with the optimal value by the linear program (20). However, Corollary III.2 can be suboptimal. To see this, we take a toy example of CNN that contains merge operations. The same network structure appears also in [21] with different filter weights. The parameter p is set to $p = 2$.

Figure 8 is an illustration of the CNN. According to Appendix C, we can translate it into a CNN within our framework, as illustrated in Figure 9.

Define the smooth "gate" function on the Fourier domain supported on $(-1, 1)$ as

$$\begin{aligned} F(\omega) =& \exp\left(\frac{4\omega^2+4\omega+1}{4\omega^2+4\omega}\right)\chi_{(-1,-1/2)}(\omega)+ \\ & \chi_{(-1/2,1/2)}(\omega)+ \\ & \exp\left(\frac{4\omega^2-4\omega+1}{4\omega^2-4\omega}\right)\chi_{(1/2,1)}(\omega)\ . \end{aligned} \tag{23}$$

With this, we define the Fourier transforms of the filters to be C^∞ gate functions

$$\begin{aligned} \hat{\phi}_1(\omega) &= F(\omega) \\ \hat{g}_{1,j}(\omega) &= F(\omega+2j-1/2)+F(\omega-2j+1/2), \\ &\qquad j=1,2,3,4. \\ \hat{\phi}_2(\omega) &= \exp\left(\frac{4\omega^2+12\omega+9}{4\omega^2+12\omega+8}\right)\chi_{(-2,-3/2)}(\omega)+ \\ &\quad \chi_{(-3/2,3/2)}(\omega)+ \\ &\quad \exp\left(\frac{4\omega^2-12\omega+9}{4\omega^2-12\omega+8}\right)\chi_{(3/2,2)}(\omega) \\ \hat{g}_{2,j}(\omega) &= F(\omega+2j)+F(\omega-2j), \\ &\qquad j=1,2,3. \\ \hat{g}_{2,4}(\omega) &= F(\omega+2)+F(\omega-2) \\ \hat{g}_{2,5}(\omega) &= F(\omega+5)+F(\omega-5) \\ \hat{\phi}_3(\omega) &= \exp\left(\frac{4\omega^2+20\omega+25}{4\omega^2+20\omega+24}\right)\chi_{(-3,-5/2)}(\omega)+ \\ &\quad \chi_{(-5/2,5/2)}(\omega)+ \\ &\quad \exp\left(\frac{4\omega^2-20\omega+25}{4\omega^2-20\omega+25}\right)\chi_{(5/2,3)}(\omega)\ . \end{aligned}$$

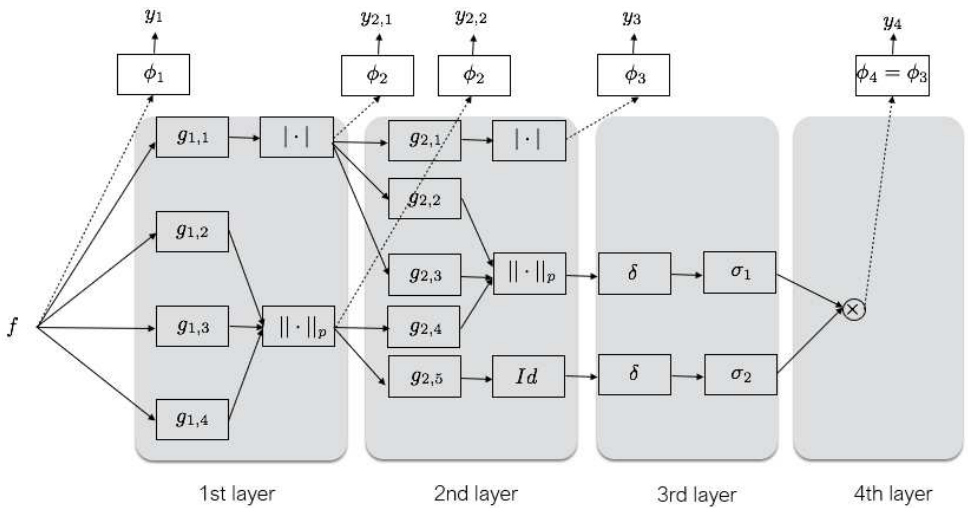


Fig. 9. Equivalent representation of the CNN in Fig. 8. The four layers of the network are illustrated in blocks.

TABLE II
THE BESSEL BOUNDS OF THE EXAMPLE IN FIGURE 8

m	1	2	3	4
$B_m^{(1)}$	$2e^{-1/3}$	$2e^{-1/3}$	2	1
$B_m^{(2)}$	1	1	2	0
$B_m^{(3)}$	1	1	1	1

Table II lists the Bessel bounds for all the layers. The optimal value of the linear program (20) gives a Lipschitz bound of $L = 2.866$; the Lipschitz bound as derived in Corollary III.2 is $L = 8[\exp(-1/3)]^2 = 4.102$; Corollary III.3 gives an estimate of the Lipschitz bound of $L = 5$. We see that the output of the linear program (20) is more optimal than the product given in Corollary III.2 and III.3.

C. AlexNet and GoogleNet

In this subsection, we analyze the Lipschitz properties of AlexNet and GoogleNet. First, we apply the analytical results derived in earlier sections to these networks and compare their results to empirical estimates. To accomplish this, we need to extend the theory hitherto developed to processing of discrete signals. Second, we construct a local Lipschitz analysis theory and explain the gap between the analytical and empirical estimates. In this process, we obtain additional information on local stability and robustness of the network, which we exploit in the third part of this subsection where we apply these results to adversarial perturbations.

1) Extending to Discrete Signal Processing: The AlexNet and the GoogleNet have filters trained on specific datasets, with no closed form parametric description of their weights such as the wavelets. Therefore, instead of using approximations of the continuous signal theory to these networks, we extend the theory to discrete signal processing. We do this by computing the Bessel bounds using the Discrete Fourier Transform along the lines in [9, Sec. 4.3] subsequent to the computation of the operator norms of the discrete linear operators. Given the Bessel bound, the Lipschitz bound is computed using the same estimates derived earlier to the case of continuous signals.

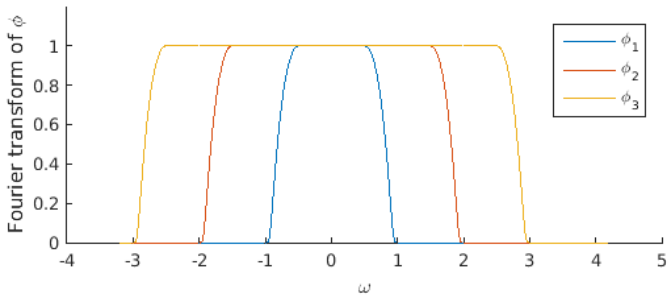


Fig. 10. Illustration of the filters ϕ_1, ϕ_2 and ϕ_3 in the frequency domain. Note that they are all C^∞ smooth functions.

First we compute the Bessel bounds. Note that both networks do not generate feature outputs in hidden layers. Therefore, $B_m^{(1)} = B_m^{(2)}$, $B_m^{(3)} = 0$ for each $1 \leq m \leq M-1$. Now the fourth line of (20) forces $z_m = 0$ for $m = 1, \cdots, M-1$, which makes the second and third lines equivalent. Note that Corollary III.2 is derived from the third and fourth lines of (20). Consequently, the linear program (20) and Corollary III.2 provide the same Lipschitz bounds $L = B_M^{(3)} \prod_{m=1}^{M-1} B_m^{(2)}$.

There are several versions of trained networks for AlexNet and GoogleNet. We consider the MatConvNet [22] pretrained networks that are trained using ImageNet (ILSVRC2012) dataset [12] (the trained networks for both AlexNet and GoogleNet are retrievable at http://www.vlfeat.org/matconvnet/pretrained/). For both pretrained models, there are no cross-channel response normalizations (which appears in the original model [7]). The features are extracted after the last convolution layer in each network.

We present the Bessel bounds ($B_m^{(2)}$ for $1 \leq m \leq M-1$ and $B_m^{(3)}$ for $m = M$) for each layer of the AlexNet

TABLE III

THE BESSEL CONSTANTS (= SQUARE ROOT OF BESSEL BOUNDS) FOR EACH LAYER OF THE ALEXNET

Layer	Lip const
conv1	0.2628
conv2	6.7761
conv3	6.5435
conv4	13.3898
conv5	16.0937

TABLE IV

THE BESSEL CONSTANTS (= SQUARE ROOT OF BESSEL BOUNDS) FOR EACH LAYER OF THE GOOGLENET

Layer	Lip const	Layer	Lip const
conv1	5.8608	icp3reduce	2.6642
reduce2	3.4147	icp3conv	6.0129
conv2	3.0309	icp4reduce	2.6403
icp1reduce	3.6571	icp4conv	5.1029
icp1conv	5.2917	icp5reduce	2.9825
icp2reduce	3.7994	icp5conv	5.5389
icp2conv	7.6367	icp6reduce	3.1758

Layer	Lip const
icp6conv	6.7737
icp7reduce	2.2093
icp7conv	6.5312
icp8reduce	2.2947
icp8conv	5.5561
icp9reduce	2.8567
icp9conv	7.0353

in Table III and the GoogleNet in Table IV. Since we are in the discrete case (previous sections discuss signals $f \in L^2(\mathbb{R}^d)$ and continuous convolutions), we need to adjust the way the Bessel bounds in (3) and (4) are computed. The adjusted computations for the AlexNet follows [9, Sec. 4.3], which uses the Discrete Fourier Transform and takes striding into account. For the GoogleNet, we treat the inception modules (see [8, Fig. 2(b)]) as two layers: the first layer is the scattering with the dimension reductions (denoted as "icpxreduce" in Table IV), and the second layer is the merging after taking convolutions (denoted as "icpxconv" in Table IV).

Using the computed Bessel constants and Corollary III.2, the estimated Lipschitz constant is 2.51×10^3 for the AlexNet and 9.67×10^{12} for the GoogleNet. Subject to the Bessel computed computed above (which are tight), the CNN Lipschitz bound estimates cannot be improved analytically. Instead we perform an empirical study. Specifically, we randomly take two images f_1 and f_2 from ImageNet, and compute the ratio $|||\Phi(f_1) - \Phi(f_2)|||/\|f_1 - f_2\|_2$, where Φ is the Lipschitz map induced by the network. The empirical Lipschitz constant is the largest ratio among all samples that we take. We sample 10^6 pairs for this experiment. The resulting empirical constant is 7.32×10^{-3} for the AlexNet, and 4.84×10^{-2} for the GoogleNet.

The empirical constants are of significantly smaller order than the analytical constants. In general, two factors explain the gap between the analytical and empirical Lipschitz constant estimates: first, the principal singular vector that optimizes the operator norm in a given layer in not in the range of signals reachable by the previous layer; second, whenever we have ReLU nonlinearity and max pooling, the distance between two vectors tends to shrink.

The first factor can be partially addressed by considering the norm of tensorial product of all layers instead of considering the product of tensor norms in each layer individually (similar to computing the operator norm of a product of matrices directly instead of upper bounding it by the product of operator norms of each matrix). Both this and the second factor can be addressed by a framework that locally linearizes the network for analysis. We demonstrate how to do this in the following subsection.

2) Local Lipschitz Analysis: To begin with, we estimate the Lipschitz constants without the ReLU functions to illustrate the impact of the nonlinearity. We construct the AlexNet and GoogleNet without the ReLU units by replacing them with the identity functions, and repeat the experiments of taking ratios from pairwise random samples. Empirically, the ratio estimated in this way is 9.08×10^{-2} for the AlexNet and 1.10×10^3 for the GoogleNet. Note that these constants are larger than the empirical Lipschitz constants for the networks with the ReLU units.

The nonlinearities also have a non-negligible impact on the Lipschitz constant. To handle them in the analysis, we linearize them locally and compute the local Lipschitz constants. The local Lipschitz constant of Φ at $f \in \mathcal{D}$ for ϵ-neighborhood is defined by

$$L^{\text{loc}}(f, \epsilon) := \sup_{\substack{f' \in \mathcal{D} \\ \|f' - f\|_2 < \epsilon}} \frac{|||\Phi(f') - \Phi(f)|||}{\|f' - f\|_2}. \qquad (24)$$

Note that for the case of the AlexNet and the GoogleNet (and similarly for all other discrete networks), the input signal is from a compact domain $\mathcal{D} = \mathfrak{I}^D$ where $\mathfrak{I}$ is the interval for the pixel values, and D is the dimensionality (the number of pixels). Since $\mathcal{D}$ is convex, the Lipschitz constant of Φ is the maximum of the local Lipschitz constants on $\mathcal{D}$. The rigorous proof of this claim is given in Appendix E.

Using the linearization formulas, we estimate numerically the local Lipschitz constants. The procedure is described as follows. We vectorize [1] the input image and the output feature vector. Also, we use Toepliz matrices $T_1, T_2, \cdots, T_M$ to represent filters in each layer. For any input sample f, the CNN generates the output feature vector $\Phi(f)$ by propagating f through T_m's and the nonlinearities that activate only a subset of the pixels for the hidden layer outputs. For the m-th layer, we delete (remove) the rows that correspond to the pixels not activated by the ReLU units and max pooling (if they exist) in T_m, and the corresponding columns in T_{m+1}. In this way we obtain matrices $T'_1, T'_2, \cdots, T'_M$. The product $T'[f] = T'_M T'_{M-1} \cdots T'_2 \; T'_1$ represents the locally linearized operator for the CNN acting at f. For a small ϵ, the local Lipschitz constant at f is estimated by $L^{\text{loc}}(f, \epsilon) \approx \sigma_{\max}(T'[f])$, the largest singular value of T'. Specifically, when ϵ is so small that the effect of ReLU units and max pooling does not change

[1] For a matrix $A = [a_1|a_2|\cdots|a_D] \in \mathbb{R}^{D \times D}$ where $a_1, a_2, \cdots, a_D$ are D-dimensional vectors, we vectorize A to be $A^{\text{vec}} = [a_1^t|a_2^t|\cdots|a_D^t]^t$

in the ϵ-neighborhood of f, we have $L^{\text{loc}}(f,\epsilon) = \sigma_{\max}(T'[f])$. To see the reason, note that $\Phi(f') = T'[f]f'$ for any f' such that $\|f' - f\|_2 \leq \epsilon$. Consequently, $|||\Phi(f') - \Phi(f)||| = |||T'[f]f' - T'[f]f||| \leq \sigma_{\max}(T'[f])\,\|f' - f\|_2$, in which the equality is achieved when $f' - f$ is in the direction of the principal singular vector. Therefore, $\sigma_{\max}(T'[f])$ is indeed the local Lipschitz constant.

As a result, the Lipschitz constant for Φ is estimated by

$$L_c = \max_{f \in \mathfrak{I}^D} L^{\text{loc}}(f,\epsilon) = \max_{f \in \mathfrak{I}^D} \sigma_{\max}(T'[f]),$$

where the second equality follows if we take the maximum over the entire compact convex set $\mathfrak{I}^D$ (see Appendix E). However, for numerical reasons, we replace $\mathfrak{I}^D$ with a finite number of samples $\mathcal{F}$ thus obtaining an approximate (lower) bound:

$$L_c \approx \max_{f \in \mathcal{F}} \sigma_{\max}(T'[f]).$$

Before discussing the numerical results based on this method, we remark here that there are two limitations of this local Lipschitz analysis. First, since the nonlinearities have very different effects for different samples, it requires a different computation of the equivalent Toepliz matrix on each input sample and in practice it is slow if the size of $\mathcal{F}$ is large. Second, it is based on local linearization and the linearized region is small in practice, which causes a difference between the local bounds and the empirical bounds since a pair of images from the dataset are usually far from each other.

We follow the procedure described above to estimate the Lipschitz constant for the AlexNet, with $\mathcal{F}$ having 500 random samples drawn from the ImageNet (ILSVRC2012) dataset. Figure 11 illustrates the histogram of these results. We see that the local Lipschitz constants in our case are between 0.2 and 1.6, hence of order 1. Table V summarizes the results of the analytical, empirical and numerical local Lipschitz constants analysis for the AlexNet.

One would naturally ask if the 500 random samples we chose for this analysis are sufficient to infer an accurate estimate of the Lipschitz constant. To address this question we performed two sets of experiments. First we test if the local Lipschitz constant is narrowly distributed over samples in each class and whether the distribution changes for random input signals (i.e. artificial noise input images). Figure 12 depicts the histogram of local Lipschitz constants for images from class "tench" (left plot), and compares it with the histogram of local Lipschitz constants for i.i.d. Gaussian noise images (right plot). We note that the local Lipschitz constants for Gaussian noise are much more concentrated around a significantly smaller mean than for the class "tench". This implies that the AlexNet behaves differently for different ImageNet samples from the same class. On the other hand the distribution of local Lipschitz constants for images from same class reflects the same range of values as the distribution over all 500 images considered in Figure 11. This experiment gives us confidence that the estimated Lipschitz constant over the 500 ImageNet images is nearly tight.

On the other hand, as observed from Table V, the Lipschitz constant computed by taking the maximum of the local

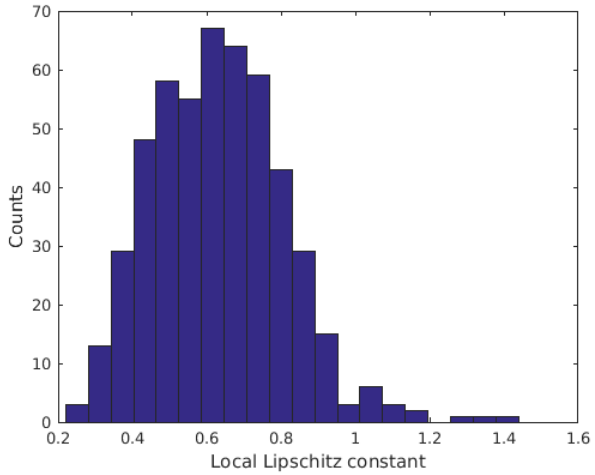


Fig. 11. The histogram of the local Lipschitz constants for the AlexNet for 500 sample images taken from the ImageNet dataset.

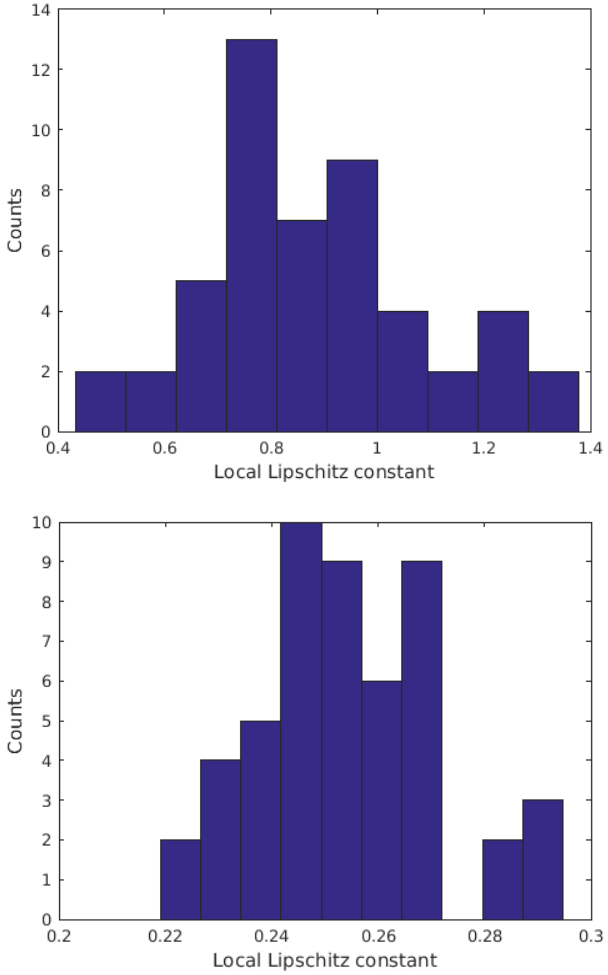


Fig. 12. Two histograms of local Lipschitz constants for the AlexNet: the left plot contains the results of 50 samples from the class "tench"; the right plot contains the results from 50 samples from i.i.d. Gaussian distribution of same size ($224 \times 224 \times 3$).

Lipschitz constant is about 3 orders of magnitude larger than the empirically computed constant. This surprising observation implies that the direction of maximum variation (the principal

TABLE V
THE LIPSCHITZ CONSTANT ESTIMATION USING THREE METHODS FOR THE ALEXNET

Method	Lip const
Analytical estimate: compute Bessel bounds and follow Corollary III.2	2.51×10^3
Empirical bound: take quotient from pairs of samples	7.32×10^{-3}
Numerical approximation: compute local Lipschitz constants and take the maximum	1.44

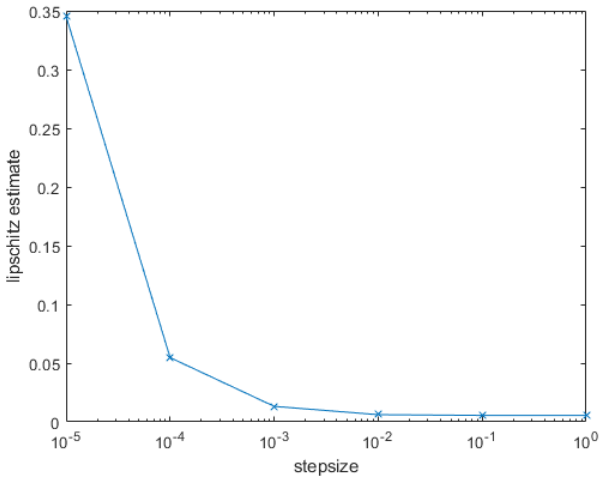


Fig. 13. The ratio $|||\Phi(f + h \cdot v) - \Phi(f)|||/h$ for different h.

singular vector) varies significantly from one ImageNet sample to another. This variation is caused by different effects of ReLU and max pooling on different image samples. To bridge the gap between the local Lipschitz bound and the empirical bound, one could assume an "average" effect of ReLU and max pooling, and conclude with an estimated empirical bound via a corresponding linear version of the CNN. We describe this method in Appendix D and conclude with an estimated bound of 1.78×10^{-2}.

Furthermore, the local Lipschitz constant is large only in a small neighborhood around each sample. In order to estimate the largest perturbation that achieves the local Lipschitz bound we performed the following experiment. For input signal f, let v denote the principal singular vector of norm $\|v\|_2 = 1$ that corresponds to the largest singular value $\sigma_{\max}$. By definition, we have

$$\lim_{\epsilon \to 0} L^{\text{loc}}(f, \epsilon) = \lim_{t \to 0} \frac{|||\Phi(f + t \cdot v) - \Phi(f)|||}{t} = \sigma_{\max}.$$

Figure 13 shows how the quotient $|||\Phi(f + h \cdot v) - \Phi(f)|||/h$ changes with h. Note that the convergence as h approaches 0 is very slow. In particular, this experiment confirms that the local Lipschitz constant is achievable, hance the numerical estimates in Table V are not just numerical artifacts, but actual achievable ratios. On the other hand, Figure 13 shows that the largest relative variation of the output (i.e. the ratio $|||\Phi(f) - \Phi(\tilde{f})|||/||f - \tilde{f}||_2$) is achieved by small perturbations only. In general, given a pair of different image samples from ImageNet, their l^2-distance is much larger than 10^{-5}, so they cannot reflect the local oscillation of Φ.

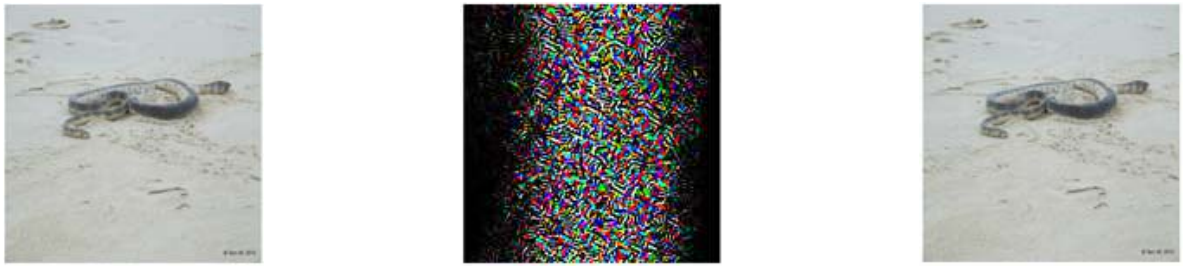
3) Adversarial Perturbation Induced by the Local Lipschitz Constants: CNN's such as the AlexNet and the GoogleNet are shown to be vulnerable to small perturbations [9], [23], [24]. This kind of instability of those deep networks not only leads to difficulties in cross-model generalization, but also causes serious security problems in practice [25], [26]. An adversarial perturbation is a small perturbation of the input signal that changes the classification decision of the CNN. The perturbation can be constructed by solving an optimization problem where the wrong classification is considered as a loss in the objective function, as described in [9]. Various optimization settings can be found in [23], [24] where specific restriction on the perturbation is required.

The local Lipschitz analysis carried out in the previous section characterizes the impact of varying the direction of signals perturbations on the output of the CNN. It can be seen that for the same amount of input perturbation, different directions can be chosen to achieve a better adversarial impact on the network performance. We use this observation to create adversarial perturbations below. We show that a relative change of the order of 10^{-2} can lead the network to wrongly characterize the input image.

Since a local Lipschitz constant is associated with a singular vector v_0 with $\|v_0\|_2 = 1$ which is the direction that Φ varies the most at f, we expect this direction gives a perturbation that "fools" the CNN more than other directions. The task is to find the smallest h for which f and $f' = f + h \cdot v_0$ are labeled differently by the CNN. We use the AlexNet and empirically search for h. For each sample, we find the smallest h that fools the AlexNet. One such example is given in Figure 14. We take 50 samples and find that the optimal h_{opt}'s have order of magnitude 10^3, which is relatively small compared to $\|f\|_2$ (we have $227 \times 227 \times 3$ input with pixel values in $[0, 255]$, so the relative change is of the order 10^{-2}). Note that this order of h is also observed in [23], where the 2-norm of the perturbation is chosen to be 2000. Further, for each sample, we take 1000 random directions v_{rand}, and compare the labels given by the AlexNet for f and $f + (h_{\text{opt}} + \Delta h) \cdot v_{\text{rand}}$ for a set of different values of Δh. We plot the percentage of directions that fools the AlexNet on average for these samples in Figure 15. Surprisingly, the direction informed by the local Lipschitz constant performs better than most directions, although at for $h > 10^3$ the quotient $|||\Phi(f + h \cdot v) - \Phi(f)|||/h$ is much smaller than the Lipschitz constant at f. Empirically, this implies that the local Lipschitz constant is still important although it decreases fast outside a small region.

V. LIPSCHITZ BOUNDS IN STATIONARY PROCESSES

Signals (audio or image) are often modeled as random processes. In our case, there are two ways to model the



(a) The original image (b) The perturbation (scaled for visibility) (c) The perturbed image

Fig. 14. An example of the perturbation along the direction of the singular vector. The left is the original image, the middle is the perturbation which is amplified 1000 times for clear illustration, and the right is the perturbed image. The AlexNet recognizes the original image as "king snake" but the perturbed one as "loggerhead turtle".

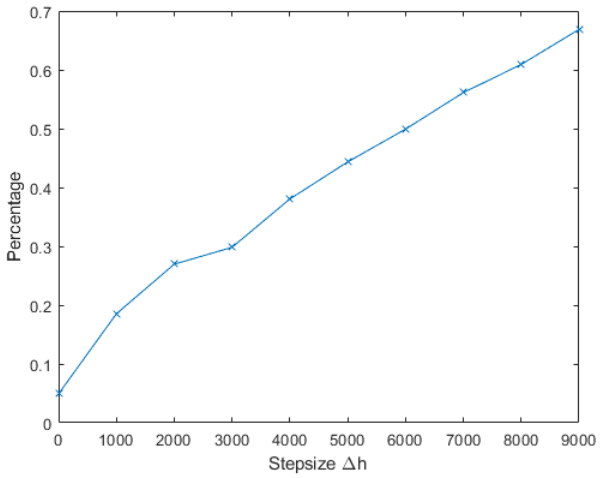


Fig. 15. Average percentage of successful perturbations in 1000 random directions. $\Delta h = 0$ is the smallest stepsize where the perturbation along the direction informed by the local Lipschitz constants successfully fools the AlexNet.

input signal of a CNN: one is to consider $X(t)$ as a random process (field) with some underlying probability space $(\Omega, \mathfrak{F}, \mathbb{P})$ with finite second-order moments (see [1, Ch. 4]); the other is to regard X as a random variable such that

$$X : (\Omega, \mathfrak{F}, \mathbb{P}) \to L^2(\mathbb{R}^d) \ .$$

We first present the former model for our framework in Section II. In the following, we use the notation $X(t)$ to emphasize the time (space) variable $t \in \mathbb{R}^d$ and $X_t(\omega)$ to emphasize $\omega \in \Omega$. We are interested in studying stationary signals. Fix a realization $X(t) = X_\omega(t)$ for some $\omega \in \Omega$. $X(t)$ is said to be strict-sense-stationary (SSS) (see, for instance, [27], Chapter 16) if all of its finite-order moments are time-invariant (its cumulative distribution does not change with time). The output of a CNN is SSS provided that the input X is SSS. This is stated as the following lemma.

Lemma V.1. *Consider a CNN in the framework of Section II in which there is no dilation operation. Let Φ be the induced Lipschitz continuous map as defined in (18). If X is an SSS process, then so is $\Phi(X)$.*

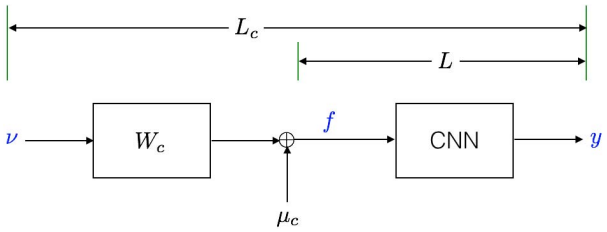


Fig. 16. Illustration of the Lipschitz bounds L_c. Suppose f is an image filtered by W_c (and a bias μ_c) from a white Gaussian noise $\nu \sim N(0, I)$. Then the Lipschitz bound L_c for the class c considers both processes of W_c and the CNN. This bound is not the same for different classes since it depends not only on the CNN but also on W_c.

Remark 1. *In general, if we apply dilations for random processes, the signals are no longer stationary after the merge operations. To see a concrete example, let θ be a random variable taking values uniformly in $[0, 2\pi)$. Consider $X(t) = \cos(t + \theta)$ which has i.i.d. distribution over time and is thus SSS. Note that $Y(t) := X(t) + X(3t) = \cos(t+\theta) + \cos(3t+\theta) = 2\cos(2t + \theta)\cos(t)$ has different distributions at $t = 0$ and $t = \pi/2$, and is thus not SSS. Therefore, throughout this section, we assume that there is no dilation operation in our CNN.*

Now we state the result that connects the Lipschitz bound derived in Section III with stationary processes.

Theorem V.2. *Consider a CNN in the framework of Section II in which there is no dilation operation. Let X and Y be SSS processes with finite second-order moments. Then*

$$\mathbb{E}\left(\left|\left|\left|\Phi(X) - \Phi(Y)\right|\right|\right|^2\right) \leq L \cdot \mathbb{E}\left(|X - Y|^2\right) \ . \qquad (25)$$

In particular, $|||\Phi(X)|||^2 \leq L \cdot \mathbb{E}\left(|X|^2\right)$.

The proof for Theorem V.2 parallels that of Section III. We present it in Appendix E.

As mentioned above, we can also follow the second way to model the signal as a random variable $X : \Omega \to L^2(\mathbb{R}^d)$. In this case, we have a random variable with values in a Banach space (see a detailed discussion of such random processes in [28], [29]). In particular, let Φ be the map induced by the CNN, and $L_c = \sqrt{L}$ be the Lipschitz constant.

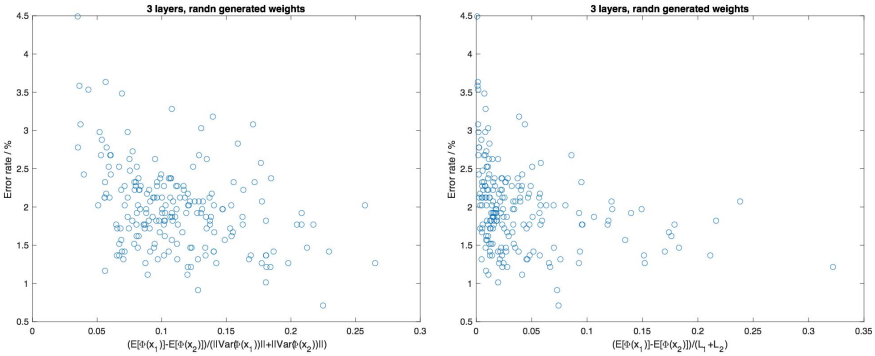


Fig. 17. Plots of error rate versus discriminant for a three-layer CNN with randomly (normal distributed) generated weights.

Denote $Y = \Phi \circ X$ to be the received random variable. Then by [29, Proposition 1.2], we have the concentration function $\alpha(L^2(\mathbb{R}^d), \mathbb{P}_Y)(r) \leq \alpha(L^2(\mathbb{R}^d), \mathbb{P}_X)(r/L_c)$. Suppose X is Gaussian (see [28, Ch. 2] for the definition in this case) and let $\sigma = \sigma(X) = \sup(\mathbb{E}\,\|X\|_2^2)^{1/2}$. Then similar to the concentration inequality in [29, Lemma 3.1], there exists a median $\mathfrak{M} > 0$ for which we have both

$$\mathbb{P}_Y(\|Y - \mathbb{E}(Y)\|_2 \leq \mathfrak{M}) \geq 1/2$$

and

$$\mathbb{P}_Y(\|Y - \mathbb{E}(Y)\|_2 \leq \mathfrak{M}) \leq 1/2\ ;$$

and we have

$$\mathbb{P}\left\{|\|Y - \mathbb{E}(Y)\|_2 - \mathfrak{M}| > t\right\} \leq \exp\left(-\frac{t^2}{2\sigma^2 L}\right)\ . \qquad (26)$$

In signal classification tasks, if we view signals in each class as realizations from a common distribution, then we have the same $\mathbb{E}(Y)$ for all signals in this class. If the feature Y generated by the CNN is concentrated around $\mathbb{E}(Y)$, and $\mathbb{E}(Y)$'s are separated for different classes, then features from different classes will naturally form clusters. Although we do not have exact concentration ($\mathfrak{M} = 0$), Inequality (26) demonstrates that Y concentrates in a "thin" shell of radius $\mathfrak{M}$ around $\mathbb{E}(Y)$ provided that we have a small Lipschitz bound L. We further promotes making the Lipschitz bound small in designing CNN's in the next section.

VI. Lipschitz Bound in Classification

In addition to analyzing stochastic processes, we present here another application of the Lipschitz bounds which is similar to the linear discriminant analysis (LDA) (see, e.g. [30], [31]). In LDA, it is desired to maximize the "separation", or the "discriminant", which is the variance between classes divided by the variance within each class (see [31, eq. (1)] and the discussion that follows). We use a similar notion in our (nonlinear discriminant) analysis, albeit its nature of nonlinearity. We define the *discriminant* of two classes C_1 and C_2 to be

$$S = \frac{|||\mathbb{E}[\Phi(f)|f \in C_1] - \mathbb{E}[\Phi(f)|f \in C_2]|||^2}{\|\mathrm{Cov}(\Phi(f)|f \in C_1)\|_* + \|\mathrm{Cov}(\Phi(f)|f \in C_2)\|_*}\ , \qquad (27)$$

in which Φ is the nonlinear map induced by the CNN, as defined in (18), $\|\cdot\|_*$ denotes the nuclear norm, and Cov denotes the covariance matrix.

To see how the Lipschitz bound is associated with the separation S, we look at the nature of the variance of the output feature $\Phi(f)$. Suppose we have a Gaussian noise $\nu \sim N(0, I)$ and apply a linear transform A, then $A\nu$ is also Gaussian with covariance AA^t. The nuclear norm of its variance is given by

$$\|\mathrm{Cov}(A\nu)\|_* = \mathrm{trace}AA^t = \|A\|_{\mathrm{Fr}}^2\ ,$$

where $\|\cdot\|_{\mathrm{Fr}}$ denotes the Frobenius norm. Since A is linear, its Lipschitz constant is given by $\|A\|_{\mathrm{op}}$ and its Lipschitz bound is given by $\|A\|_{\mathrm{op}}^2$. Note that the Frobenious norm and the operator norm are equivalent norms, since $\|A\|_{\mathrm{op}} \leq \|A\|_{\mathrm{Fr}} \leq \sqrt{n}\,\|A\|_{\mathrm{op}}$.

Motivated by the linear case, we look into replacing $\|\mathrm{Cov}(\cdot)\|_*$ in (27) with the Lipschitz bound for general CNN's. We consider a CNN with a Gaussian white noise input $\nu \sim N(0, I)$. We assume two classes of signals, C_1 and C_2 where each class C_c ($c = 1, 2$) contains samples from a colored Gaussian noise $\nu_c \sim N(\mu_c, W_c W_c^t)$. We use L_c to denote the Lipschitz bound for the whole system, as illustrated in Figure 16.

We define the *Lipschitz discriminant* to be

$$\tilde{S} = \frac{|||\mathbb{E}[\Phi(f)|f \in C_1] - \mathbb{E}[\Phi(f)|f \in C_2]|||^2}{L_1 + L_2}\ , \qquad (28)$$

where L_1 and L_2 are the Lipschitz bounds for Class 1 and Class 2, respectively.

In Figure 17 – 20, we report the experiments on the discriminative behavior of randomly generated CNN's. We take two classes (number "3" and "8") of test images from the well-known MNIST database [32], and randomly build CNN's with three or four convolutional layers and record their discriminant according to (27) (plotted on the left-hand-side in each figure) and (28) (plotted on the right-hand-side in each figure). We then train a linear SVM for each network and plot the error rate of classification against the discriminants. The purpose of this experiment is to show that larger discriminants lead to better classification results. The reason we use SVM's is to examine the quality of the CNN (feature extractor) given different discriminants, and therefore we choose to train

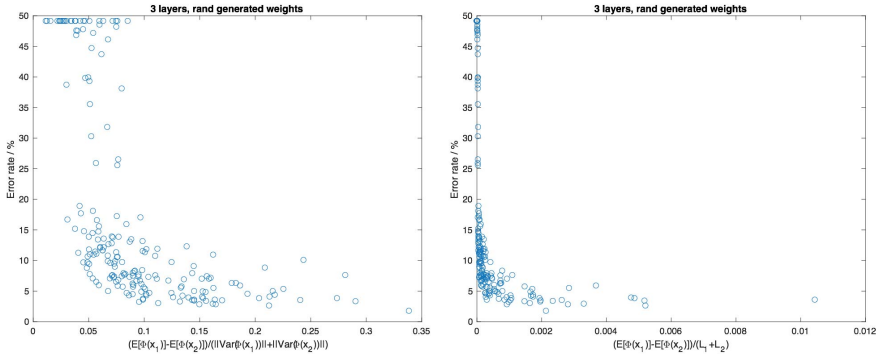


Fig. 18. Plots of error rate versus discriminant for a three-layer CNN with randomly (uniformly distributed) generated weights.

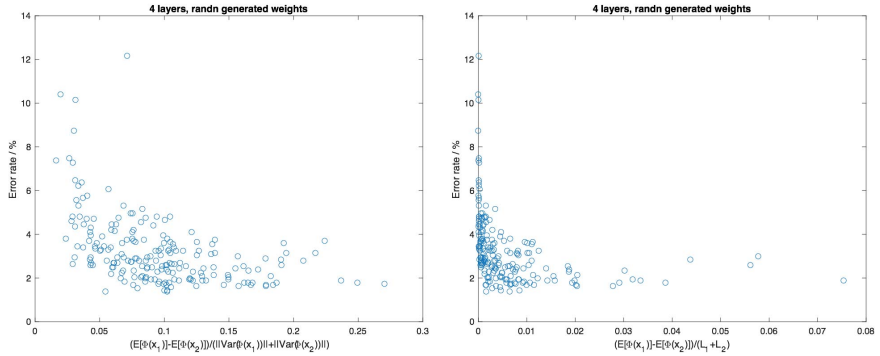


Fig. 19. Plots of error rate versus discriminant for a four-layer CNN with randomly (normal distributed) generated weights.

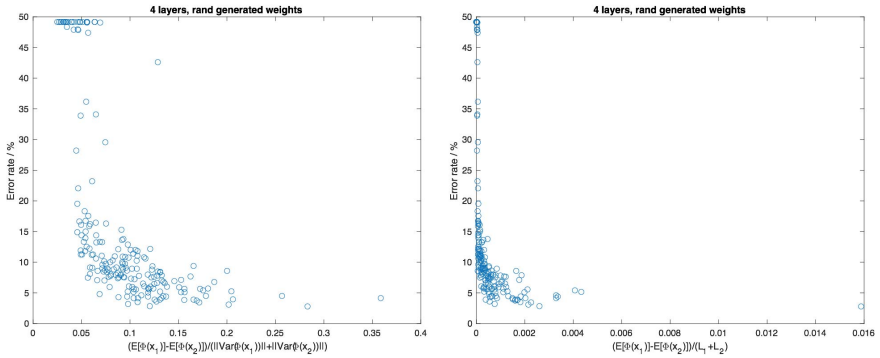


Fig. 20. Plots of error rate versus discriminant for a four-layer CNN with randomly (uniformly distributed) generated weights.

linear SVM's (which works for two classes) with the same regularization parameter. The numerical implementation is done using MATLAB 2016b. We use MatConvNet [22] for constructing the CNN, and the Machine Learning Toolbox in MATLAB for training the SVM's.

As seen from the results, the error rate tends to decrease as the discriminant (27) and the Lipschitz discriminant (28) increase. The trend is clearer when we have more layers. Therefore, either the discriminant or the Lipschitz discriminant is a reasonable penalty term for the training objective function of the CNN. Our analysis in previous chapters can be effectively used to estimate the Lipschitz discriminant for these optimization problems. However, it remains open how to design a training algorithm using the discriminants since the weights appear in both the numerators and denominators in (27) and (28).

VII. Conclusion

In this paper we proposed a general framework for Lipschitz analysis of CNN's. We showed that after calculating the Bessel bounds for each layer, the Lipschitz bound can be calculated by solving a linear program. We also demonstrated that the Lipschitz bounds play a significant role in the second order statistical description of CNN's. Further, we illustrated that the Lipschitz bounds of CNN's can be used to form a discriminant that works effectively in classification systems.

In addition to the Lipschitz bounds derived from the Bessel bounds, we discussed the Lipschitz bounds from local singular value and from empirical ratios by taking pairs of samples. The Bessel bound method can be over conservative due to looseness from cascading the upper bound and from neglecting the effect of nonlinearities. However, the local Lipschitz bound also has limitations due to the expensive

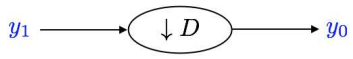


Fig. 21. The dilation operation. $y_1 \in \mathbb{R}^d$ is the input and y_0 is the output given as $y_0(x) = y_1(Dx)$.

local computations and the rapid variation of the effect of nonlinearities. We believe the hope to overcome these limitations may come from stochastic models. From the numerical experiments in Section IV, we found interesting results on gaps between these bounds. We provided a simple stochastic model that bridges the gap between the local bounds in the worst-case sense and empirical bounds, but we believe the analysis needed for completely understanding the behaviors of these bounds are more complicated and requires further treatment. Given the importance of Lipschitz bounds in understanding the stability and adversarial perturbation, we believe this deserves future work.

In future works, we will pursue more systematic analysis based on the stochastic models for both the upper bound cascading and the effect of nonlinearities. Specifically, this requires modeling both the filters and nonlinearities. Note that if we adopt the same model as given in the text for the nonlinearites, then we can linearize a CNN by looking at the product of the corresponding Toepliz matrices. It is likely that an estimation of the principal singular value can be estimated given some assumptions on the distribution of the entries of the filters. Potentially, the distribution of Lipschitz constants will be reached given these models. We will also seek for the application of these analyses in training CNN's robust to adversarial perturbation.

Appendix A
Proof of Theorem III.1

We are going to show that the optimal value for the linear program (20) is a Lipschitz bound. In particular, we study $\sum_{N\in\mathcal{V}} \left\| f_N - \tilde{f}_N \right\|_2^2$ as $\sum_{m=1}^{M} \sum_{N\in\mathcal{V}_m} \left\| f_N - \tilde{f}_N \right\|_2^2$.

For the m-th layer, we mark the signals at the input nodes to be $h_{m,1}, \cdots, h_{m,n_m}$ and the signals at the output nodes to be $h'_{m,1}, \cdots, h'_{m,n'_m}$. We estimate the Lipschitz bound by comparing the output nodes and input nodes for each layer, and then derive a relation between the outputs and the input at the very first layer. Note that with our notation here, $h_{1,1} = f$ and $\tilde{h}_{1,1} = \tilde{f}$.

We first look at the case of no merging. Before we study the input-output relation, note that for the dilation operation illustrated in Figure 21, for two outputs $y_0, \tilde{y}_0$ from inputs $y_1, \tilde{y}_1 \in \mathbb{R}^d$ respectively, we have

$$\begin{aligned} \|y_0 - \tilde{y}_0\|_2^2 &= \int |y_1(Dx) - \tilde{y}_1(Dx)|^2 \, dx \\ &= (\det D)^{-1} \|y_1 - \tilde{y}_1\|_2^2 \ . \end{aligned} \tag{29}$$

Now we look at the illustration in Figure 3. Since the nonlinearity $\sigma_{m,n'}$ is 1-Lipschitz, and also according to (29), we have

$$\left\| h'_{m,n'} - \tilde{h}'_{m,n'} \right\|_2^2 \le (\det D_{m,n'})^{-1} \left\| h^{\spadesuit}_{m,n'} - \tilde{h}^{\spadesuit}_{m,n'} \right\|_2^2 \ .$$

Therefore,

$$\begin{aligned} & \sum_{n'=1}^{n'_m} \left\| h_{m,n'} - \tilde{h}_{m,n'} \right\|_2^2 + \sum_{n=1}^{n_m} \left\| f_{m,n} - \tilde{f}_{m,n} \right\|_2^2 \\ \le & \sum_{n'=1}^{n'_m} (\det D_{m,n'})^{-1} \left\| h^{\spadesuit}_{m,n'} - \tilde{h}^{\spadesuit}_{m,n'} \right\|_2^2 + \\ & \sum_{n=1}^{n_m} \left\| f_{m,n} - \tilde{f}_{m,n} \right\|_2^2 \\ = & \sum_{n'=1}^{n'_m} (\det D_{m,n'})^{-1} \left\| \hat{h}^{\spadesuit}_{m,n'} - \tilde{\hat{h}}^{\spadesuit}_{m,n'} \right\|_2^2 + \\ & \sum_{n=1}^{n_m} \left\| \hat{f}_{m,n} - \tilde{\hat{f}}_{m,n} \right\|_2^2 \\ = & \sum_{n'=1}^{n'_m} \int \left| \left\{ \begin{bmatrix} \Delta^{(m)} \hat{T}^{(m)}(\omega) \\ \hat{\Psi}^{(m)}(\omega) \end{bmatrix} \left(\hat{h}^{(m)}(\omega) - \tilde{\hat{h}}^{(m)}(\omega) \right) \right\}_{n'} \right|^2 d\omega \\ \le & \left(\sup_{\omega\in\mathbb{R}^d} \left\| \begin{bmatrix} \Delta^{(m)} \hat{T}^{(m)}(\omega) \\ \hat{\Psi}^{(m)}(\omega) \end{bmatrix} \right\|_{op}^2 \right) \cdot \\ & \left(\sum_{n=1}^{n_m} \left\| h_{m,n} - \tilde{h}_{m,n} \right\|_2^2 \right) \\ = & B_m^{(1)} \sum_{n=1}^{n_m} \left\| h_{m,n} - \tilde{h}_{m,n} \right\|_2^2 \ , \end{aligned} \tag{30}$$

where in the last two steps, $\hat{h}^{(m)}$ is the column vector whose n-th entry is $\hat{h}_{m,n}$ (and similarly for $\tilde{\hat{h}}^{(m)}$), and $\{\cdot\}_n$ denotes the n-th entry of a vector.

In the same manner, we have

$$\sum_{n'=1}^{n'_m} \left\| h_{m,n'} - \tilde{h}_{m,n'} \right\|_2^2 \le B_m^{(2)} \sum_{n=1}^{n_m} \left\| h_{m,n} - \tilde{h}_{m,n} \right\|_2^2 \ ,$$

and

$$\sum_{n=1}^{n_m} \left\| f_{m,n} - \tilde{f}_{m,n} \right\|_2^2 \le B_m^{(3)} \sum_{n=1}^{n_m} \left\| h_{m,n} - \tilde{h}_{m,n} \right\|_2^2 \ .$$

We have completed the analysis of one layer without merging. Now we focus on the merging case, in which the definition of the corresponding Bessel bounds will be clear immediately after we study the three types of merging. Now we look at the relation between the output and input of the merging blocks.

For Type I, as illustrated in Figure 22, we have

$$y_0 = \sum_{k=1}^{K} \sigma_k(y_k) \ , \tag{31}$$

and

$$\tilde{y}_0 = \sum_{k=1}^{K} \sigma_k(\tilde{y}_k) \ . \tag{32}$$

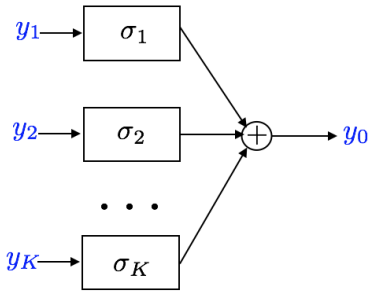


Fig. 22. Type I merging. y_0 is the sum of $\sigma_1(y_1), \cdots, \sigma_K(y_K)$.

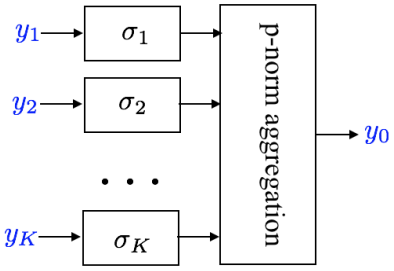


Fig. 23. Type II merging. y_0 is the aggregate of $\sigma_1(y_1), \cdots, \sigma_K(y_K)$ using p-norm.

Therefore

$$\begin{aligned}\|y_0-\tilde{y}_0\|_2^2 &= \left\|\sum_{k=1}^K \sigma_k(y_k)-\sigma_k(\tilde{y}_k)\right\|_2^2\\ &\le K\sum_{k=1}^K \|\sigma_k(y_k)-\sigma_k(\tilde{y}_k)\|_2^2\\ &\le K\sum_{k=1}^K \|y_k-\tilde{y}_k\|_2^2 \ . \end{aligned} \tag{33}$$

For Type II, as illustrated in Figure 23, we have

$$y_0=\left(\sum_{k=1}^K |\sigma_k(y_k)|^p\right)^{1/p}, \tag{34}$$

and

$$\tilde{y}_0=\left(\sum_{k=1}^K |\sigma_k(\tilde{y}_k)|^p\right)^{1/p}, \tag{35}$$

Therefore if $p\le 2$ we have

$$\begin{aligned}&\|y_0-\tilde{y}_0\|_2^2\\ =&\left\|\left(\sum_{k=1}^K |\sigma_k(y_k)|^p\right)^{1/p}-\left(\sum_{k=1}^K |\sigma_k(\tilde{y}_k)|^p\right)^{1/p}\right\|_2^2\\ \le&\left\|\Big(\sum_{k=1}^K |\sigma_k(y_k)-\sigma_k(\tilde{y}_k)|^p\Big)^{1/p}\right\|_2^2\\ \le& K^{2/p-1}\cdot\left\|\Big(\sum_{k=1}^K |\sigma_k(y_k)-\sigma_k(\tilde{y}_k)|^2\Big)^{1/2}\right\|_2^2\\ =& K^{2/p-1}\cdot\sum_{k=1}^K \|\sigma_k(y_k)-\sigma_k(\tilde{y}_k)\|_2^2\\ \le& K^{2/p-1}\cdot\sum_{k=1}^K \|y_k-\tilde{y}_k\|_2^2 \ ; \end{aligned}$$

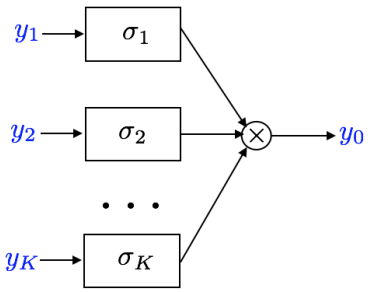


Fig. 24. Type III merging. y_0 is the product of $\sigma_1(y_1), \cdots, \sigma_K(y_K)$. Here $\|\sigma_j\|_\infty \le 1$ for $j=1,\cdots,K$.

and if $p>2$ we have

$$\begin{aligned}&\|y_0-\tilde{y}_0\|_2^2\\ =&\left\|\left(\sum_{k=1}^K |\sigma_k(y_k)|^p\right)^{1/p}-\left(\sum_{k=1}^K |\sigma_k(\tilde{y}_k)|^p\right)^{1/p}\right\|_2^2\\ \le&\left\|\Big(\sum_{k=1}^K |\sigma_k(y_k)-\sigma_k(\tilde{y}_k)|^p\Big)^{1/p}\right\|_2^2\\ \le&\left\|\Big(\sum_{k=1}^K |\sigma_k(y_k)-\sigma_k(\tilde{y}_k)|^2\Big)^{1/2}\right\|_2^2\\ =&\sum_{k=1}^K \|\sigma_k(y_k)-\sigma_k(\tilde{y}_k)\|_2^2\\ \le&\sum_{k=1}^K \|y_k-\tilde{y}_k\|_2^2 \ . \end{aligned}$$

For Type III, as illustrated in Figure 24, we have $y_0=\prod_{k=1}^K \sigma_k(y_k)$ and $\tilde{y}_0=\prod_{k=1}^K \sigma_k(\tilde{y}_k)$. Therefore,

$$\begin{aligned}&\|y_0-\tilde{y}_0\|_2\\ =&\left\|\prod_{k=1}^K \sigma_k(y_k)-\prod_{k=1}^K \sigma_k(\tilde{y}_k)\right\|_2\\ =&\Bigg\|\prod_{k=1}^K \sigma_k(y_k)+\sum_{J=1}^{K-1}\Big[-\prod_{k=1}^J \sigma_k(y_k)\prod_{k=J+1}^K \sigma_k(\tilde{y}_k)+\\ &\prod_{k=1}^J \sigma_k(y_k)\prod_{k=J+1}^K \sigma_k(\tilde{y}_k)\Big]+\prod_{k=1}^K \sigma_k(\tilde{y}_k)\Bigg\|_2\\ =&\Bigg\|\prod_{k=1}^{K-1}\sigma_k(y_k)\cdot(\sigma_K(y_K)-\sigma_K(\tilde{y}_K))+\sum_{J=2}^{K-1}\prod_{k=1}^{J-1}\sigma_k(y_k)\cdot\\ &(\sigma_J(y_J)-\sigma_J(\tilde{y}_J))\cdot\prod_{k=J+1}^K \sigma_k(\tilde{y}_k)+\\ &(\sigma_1(y_1)-\sigma_1(\tilde{y}_1))\cdot\prod_{k=2}^K \sigma_k(\tilde{y}_k)\Bigg\|_2\\ \le&\prod_{k=1}^{K-1}\|\sigma_k(y_k)\|_\infty\cdot\|\sigma_K(y_K)-\sigma_K(\tilde{y}_K)\|_2+\\ &\sum_{J=2}^{K-1}\prod_{k=1}^{J-1}\|\sigma_k(y_k)\|_\infty\cdot\prod_{k=J+1}^K\|\sigma_k(\tilde{y}_k)\|_\infty\cdot \end{aligned}$$

$$
\begin{aligned}
&\|\sigma_J(y_J) - \sigma_J(\tilde{y}_J)\|_2 + \\
&\prod_{k=2}^{K} \|\sigma_k(\tilde{y}_k)\|_\infty \cdot \|\sigma_1(y_1) - \sigma_1(\tilde{y}_1)\|_2 \\
\leq\ & \sum_{k=1}^{K} \|\sigma_k(y_k) - \sigma_k(\tilde{y}_k)\|_2 \\
\leq\ & \sum_{k=1}^{K} \|y_k - \tilde{y}_k\|_2 \ ,
\end{aligned}
$$

and thus

$$
\|y_0 - \tilde{y}_0\|_2^2 \leq K \sum_{k=1}^{K} \|y_k - \tilde{y}_k\|_2^2 \ . \tag{36}
$$

Therefore, when we compare the input nodes and output nodes of the m-th layer for the merging case, using the above relations and the definition of $B_m^{(1)}$, we have (see Figure 7)

$$
\begin{aligned}
&\sum_{n'=1}^{n'_m} \left\| h'_{m,n'} - \tilde{h}'_{m,n'} \right\|_2^2 + \sum_{n=1}^{n_m} \left\| f_{m,n} - \tilde{f}_{m,n} \right\|_2^2 \\
\leq\ & B_m^{(1)} \left\| h_{m,n} - \tilde{h}_{m,n} \right\|_2^2 \ .
\end{aligned}
$$

By the one-one correspondence of the output nodes in the $(m+1)$-th layer and the input nodes in the m-th layer, we know that

$$
\sum_{n=1}^{n_{m+1}} \left\| h_{m+1,n} - \tilde{h}_{m+1,n} \right\|_2^2 = \sum_{n=1}^{n'_m} \left\| h'_{m,n} - \tilde{h}'_{m,n} \right\|_2^2 \ , \tag{37}
$$

and therefore,

$$
\begin{aligned}
&\sum_{n=1}^{n_{m+1}} \left\| h_{m+1,n} - \tilde{h}_{m+1,n} \right\|_2^2 + \sum_{n=1}^{n_m} \left\| f_{m,n} - \tilde{f}_{m,n} \right\|_2^2 \\
\leq\ & B_m^{(1)} \sum_{n=1}^{n_m} \left\| h_{m,n} - \tilde{h}_{m,n} \right\|_2^2 \ ,
\end{aligned} \tag{38}
$$

for $1 \leq m \leq M-1$.

If we do not consider the output generating, then the forward propagation relation is

$$
\sum_{n=1}^{n_m} \left\| h_{m+1,n} - \tilde{h}_{m+1,n} \right\|_2^2 \leq B_m^{(2)} \sum_{n=1}^{n_m} \left\| h_{m,n} - \tilde{h}_{m,n} \right\|_2^2 \ , \tag{39}
$$

for $1 \leq m \leq M-1$, and similarly, considering the output generating nodes alone gives

$$
\sum_{n=1}^{n_m} \left\| f_{m,n} - \tilde{f}_{m,n} \right\|_2^2 \leq B_m^{(3)} \sum_{n=1}^{n_m} \left\| h_{m,n} - \tilde{h}_{m,n} \right\|_2^2 \ , \tag{40}
$$

for $1 \leq m \leq M$.

Since we would like to compare $\sum_{m=1}^{M} \sum_{n=1}^{n_m} \left\| f_{m,n} - \tilde{f}_{m,n} \right\|_2^2$ with $\left\| h_{1,1} - \tilde{h}_{1,1} \right\|_2^2$, by (38)-(40), we see that the maximal value of the linear program (20) gives a Lipschitz bound.

APPENDIX B
PROOF OF COROLLARY III.2

From the definitions of $B_{m,n}^{(1)}$, $B_{m,n}^{(2)}$ and $B_{m,n}^{(3)}$ (11)-(13) it is obvious that

$$
B_{m,n}^{(1)} \leq B_{m,n}^{(2)} + B_{m,n}^{(3)} \tag{41}
$$

and from (14)-(16), as well as (2)-(4), we have hence

$$
B_m^{(1)} \leq B_m^{(2)} + B_m^{(3)} \tag{42}
$$

for each m. Then note that if $\{y_m\}_{m=0}^{M-1}$ and $\{z_m\}_{m=0}^{M-1}$ are the maximums of the linear program (20), then

$$
z_m \leq B_m^{(1)} y_{m-1} - y_m, \qquad 1 \leq m \leq M-1, \tag{43}
$$

and

$$
z_M \leq B_M^{(1)} y_{M-1} \tag{44}
$$

(note that $B_M^{(1)} = B_M^{(3)}$).

We take the sum over all m's to get (denote $y_M = 0$)

$$
\begin{aligned}
\sum_{m=1}^{M} z_m \ &\leq \ \sum_{m=1}^{M} B_m^{(1)} y_{m-1} - y_m \\
&= \ \sum_{m=0}^{M-1} B_{m+1}^{(1)} y_m - \sum_{m=1}^{M-1} y_m \\
&= \ B_1^{(1)} + \sum_{m=1}^{M-1} (B_{m+1}^{(1)} - 1) y_m \ .
\end{aligned} \tag{45}
$$

Also, $y_m \leq B_m^{(2)} y_{m-1}$ implies $y_m \leq B_m^{(1)} y_{m-1}$, so $y_m \leq \prod_{m'=1}^{m} B_{m'}^{(1)}$ and thus

$$
\begin{aligned}
\sum_{m=1}^{M} z_m \ &\leq \ B_1^{(1)} + \sum_{m=1}^{M-1} \Big(\Big(\max\{1, B_{m+1}^{(1)}\} - 1 \Big) \cdot \\
&\qquad \prod_{m'=1}^{m} \max\{1, B_{m'}^{(1)}\} \Big) \\
&= \ B_1^{(1)} - \sum_{m=1}^{M-1} \prod_{m'=1}^{m} \max\{1, B_{m'}^{(1)}\} + \\
&\qquad \sum_{m=1}^{M-1} \left(\max\{1, B_{m+1}^{(1)}\} \prod_{m'=1}^{m} \max\{1, B_{m'}^{(1)}\} \right) \\
&= \ B_1^{(1)} - \sum_{m=1}^{M-1} \prod_{m'=1}^{m} \max\{1, B_{m'}^{(1)}\} + \\
&\qquad \sum_{m=2}^{M} \prod_{m'=1}^{m} \max\{1, B_{m'}^{(1)}\} \\
&\leq \ \prod_{m'=1}^{M} \max\{1, B_{m'}^{(1)}\} = \prod_{m=1}^{M} \max\{1, B_m^{(1)}\} \ .
\end{aligned}
$$

APPENDIX C
THE BANACH ALGEBRA (1)

We first show that we indeed have a Banach algebra in (1).

Lemma C.1. *$\mathcal{B}$ as defined in (1) is a Banach algebra, where the $+$ operation is pointwise addition, and the $\cdot$ operation is the convolution defined by*

$$
f * g = \left(\hat{f} \hat{g} \right)^{\vee} \ , \tag{46}
$$

where " $^{\vee}$ " denotes the inverse Fourier transform.

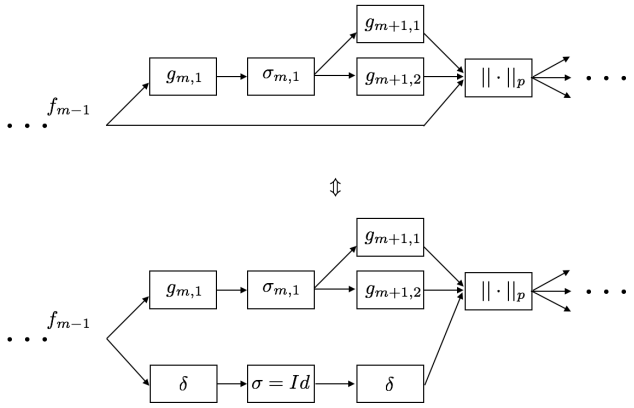


Fig. 25. Use δ function to equivalently represent a CNN.

Proof: Note that $\mathcal{B}$ is closed under the convolution in the sense of (46) because $\hat{f}\hat{g} \in L^\infty(\mathbb{R}^d)$ and therefore is also in $\mathcal{S}'(\mathbb{R}^d)$. Since the Fourier transform is an isomorphism on $\mathcal{S}'(\mathbb{R}^d)$, the inverse Fourier transform of $\hat{f}\hat{g}$ also lies in $\mathcal{S}'(\mathbb{R}^d)$.

After the closedness is clear, it is trivial to check that $\mathcal{B}$ is indeed an algebra. The fact that $\mathcal{B}$ is a Banach algebra is due to the norm inequality

$$\left\|\hat{f}\hat{g}\right\|_\infty \leq \left\|\hat{f}\right\|_\infty \left\|\hat{g}\right\|_\infty . \tag{47}$$

■

The definition of the Banach Algebra becomes natural after the Bessel bounds (11)-(13) are defined. Of course, in practice we can consider only filters lie in the space $L^1(\mathbb{R}^d)$. The Banach Algebra (1) is a larger space, and it also has some practical consideration. Suppose we have a network where there is aggregation of two layers, then we notice that this does not fall in our general model. Nevertheless, we can add several layers of δ-function, to make it fall in our framework. This is illustrated in Figure 25.

In the definition (1), the L^∞ norm is considered in the usual sense, that is, we only consider $\hat{f}$ to be a well-defined ordinary function in $L^\infty(\mathbb{R}^d)$. Then the convolution operation should be understood as $f * g = (\hat{f} \cdot \hat{g})^\vee$. Then obviously the Banach Algebra $\mathcal{B}$ is closed and well-defined under the convolution operation.

Under this definition, if we don't choose a smooth (in the frequency domain) filter, then in the signal domain we do not have good decay and it is possible to have infinite L^1 norm. Even if we choose signals whose Fourier transform is in $C_c^\infty(\mathbb{R}^d)$, we have a coarse approximation by using Young's inequality. Details can be seen in the example given in [21].

APPENDIX D
ESTIMATING EMPIRICAL LIPSCHITZ BOUNDS

In Section IV-C, Table V, we observe that the Lipschitz constant computed by taking the maximum of the local Lipschitz constant is about 3 orders of magnitude larger than the empirically computed constant. It is natural to ask whether this gap can be bridged by looking at the empirical effects of ReLU and max pooling operations.

By the Lebesgue's differentiation theorem, a CNN Φ with ReLU activation map is differentiable almost everywhere. At points of differentiability, its linearization is described by the product

$$F = P_M D_M T_M P_{M-1} D_{M-1} T_{M-1} \cdots P_1 D_1 T_1 , \tag{48}$$

where $T_1, \cdots, T_M$ are the Toeplitz matrices corresponding to the filters; $D_1, \cdots, D_M$ are diagonal matrices whose diagonals consist of entries equal to 1 if activated by the ReLU unit and 0 otherwise; $P_1, \cdots, P_M$ are slanted diagonal matrices whose each row is zero except for one value equal to 1 corresponding to the entry selected by the max pooling. In general, T_m's do not change with the input signal, but D_m's and P_m's are constant locally. Let f_0 and f_1 be two different inputs for which Φ has different local linearizations. Take a point f_t on the line segment between f_0 and f_1: $f_t = f(t) := (1-t)f_0 + tf_1$, $0 \leq t \leq 1$, then F in (48) is defined almost everywhere in t,

$$\begin{aligned} F(t) =& P_M(t) D_M(t) T_M P_{M-1}(t) D_{M-1}(t) T_{M-1} \cdots \\ & P_1(t) D_1(t) T_1, \qquad 0 \leq t \leq 1 . \end{aligned} \tag{49}$$

Consider the partition of $[0,1]$ where F is piecewise constant: let $\{t_0, t_1, \cdots, t_Q\}$ such that $0 = t_0 < t_1 < \cdots < t_Q = 1$ and $F(t) = F_q := F(t_q + 0)$ for $t_q < t < t_{q+1}$, $q = 0, 1, \cdots, Q-1$. Consequently, by continuity of Φ,

$$\Phi(f(t_{q+1})) - \Phi(f(t_q)) = (t_{q+1} - t_q) F_q (f_1 - f_0) . \tag{50}$$

Summing over $q = 0, 1, \cdots, Q-1$,

$$\Phi(f_1) - \Phi(f_0) = \left(\sum_{q=0}^{Q-1} (t_{q+1} - t_q) F_q \right) (f_1 - f_0) . \tag{51}$$

Denote $F_\star := \sum_{q=0}^{Q-1} (t_{q+1} - t_q) F_q$. We rewrite (51) as

$$\Phi(f_1) - \Phi(f_0) = F_\star (f_1 - f_0) . \tag{52}$$

When we compute the empirical Lipschitz constant in Table V, we take the largest quotient of the norm of $\Phi(f_1) - \Phi(f_0)$ and the norm of $f_1 - f_0$ over pairs of samples. For each pair of f_0 and f_1, the quotient will be bounded by the largest singular value of $F_\star$. In contrast to the local singular value, which corresponds to the variation between two inputs arbitrary close to each other, $\sigma_{\max}(F_\star)$ corresponds to the variation between two input images at order 1 distance apart. For this reason we call $F_\star$ the *effective Jacobian* for the input pair (f_0, f_1).

In the following we present a stochastic model designed to compute the effective Jacobian on a specific dataset, and then we compare our model prediction with the empirical Lipschitz constant in Table V.

The first assumption is an ergodic hypothesis: the time average in definition of $F_\star$ can be replaced by an expectation over realizations of P_m's and D_m's:

$$J_\star = \sum_{q=0}^{Q-1} F_q (t_{q+1} - t_q) = \mathbb{E}[F]. \tag{53}$$

The second assumption is independence of D_1, P_1, D_2, P_2, $\cdots$, D_M, P_M. While this assumption is obviously not true when conditioned to a specific input pair (f_0, f_1), the independence between various channels and layers increases when the random variables are analyzed over a large dataset. Consequently, the effective Jacobian is estimated by

$$J_\star \approx (\mathbb{E}P_M)(\mathbb{E}D_M)T_M \cdots (\mathbb{E}P_1)(\mathbb{E}D_1)T_1 \ . \tag{54}$$

Finally, the third assumption is specific to the two types of matrices: (i) In each layer m, the Bernoulli random variables in D_m have the same distribution dependent on the layer index only. Thus $\mathbb{E}[D_m] = p_m I$; (ii) Each row in P_m is a realization of one of 9 possible row vectors (each corresponding to selecting one of the 3×3 entries in the sliding window); The assumption is that these realizations occur with equal probability; The consequence of this is that $\mathbb{E}[P_m]$ can be replaced by the *average pooling* operator.

Next we estimate empirically the five constants p_1, p_2, p_3, p_4, p_5 for the AlexNet. We compute the expected terms in (54) based on 10k pairs of image samples. For each pair, we take the line segment between them and sample at $t_{q+1} - t_q = 1$. We compute the empirical D_m's as $D_m = p_m \cdot I_m$ where p is the percentage of entries being activated in the m-th layer and I_m is the identity matrix. The estimation are based on the average of the samples along the line segment and the pairs of images. Numerically, we obtain $p_1 = 0.4115$, $p_2 = 0.3184$, $p_3 = 0.3587$, $p_4 = 0.2733$, $p_5 = 0.1943$. For the five convolutional layers, we get only the 1st, 2nd and 5th layers have max pooling operations. Those max poolings consider 3×3 areas and on average we expect the effect is an average pooling with a multiplier of $1/9$ for each entry. Indeed, from Figure 26, the histogram of average ratio of being activated by max poolings concentrates around the mean of 0.1111. Replacing the expected terms by the matrices described above, the modified effective Jacobian has the largest singular value 1.78×10^{-2}. This value is about twice the empirical Lipschitz constant estimated at 7.32×10^{-3} in Table V.

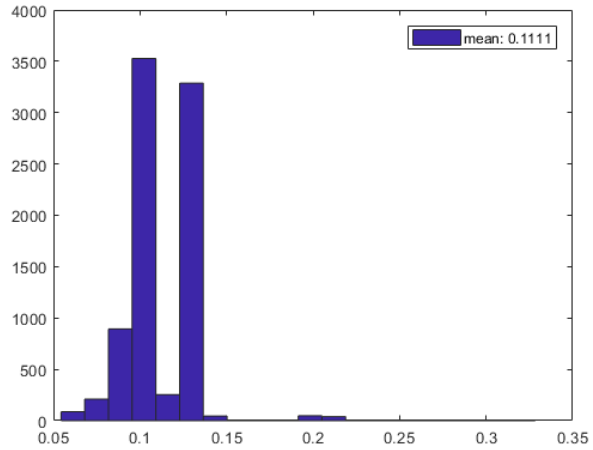


Fig. 26. The histogram of activation ratios of max poolings. Each sample corresponds to an entry of the input of max pooling, and the x-axis is the percentage of time that entry is activated by max pooling among all the samples we take in our experiment.

APPENDIX E
LIPSCHITZ CONSTANTS AND LOCAL LIPSCHITZ CONSTANTS

For CNN's such as the AlexNet and the GoogleNet, the Lipschitz constant is the maximum among all the local Lipschitz constants (see Section IV-C). In particular, we have the following result.

Proposition E.1. *Let $\Phi : \mathcal{D} \to \mathcal{R}$ be a Lipschitz continuous function on a compact convex domain $\mathcal{D} \in \mathbb{R}^D$ with the Lipschitz constant*

$$L_c := \max_{\substack{f,g \in \mathcal{D} \\ f \neq g}} \frac{|||\Phi(f) - \Phi(g)|||}{\|f - g\|_2} \ ,$$

where $|||\cdot|||$ is a well-defined norm on $\mathcal{R}$. Suppose the local Lipschitz constant at $f \in \mathcal{D}$ for some $\epsilon > 0$ is $L^{\text{loc}}(f, \epsilon)$ as defined in (24). Then

$$L_c = \max_{f \in \mathcal{D}} L^{\text{loc}}(f, \epsilon) \ . \tag{55}$$

Proof: Assume on the contrary that (55) is not true. Then $L_c > \max_{f \in \mathcal{D}} L^{\text{loc}}(f, \epsilon)$. Suppose $L_c = 2\delta + \max_{f \in \mathcal{D}} L^{\text{loc}}(f, \epsilon)$. Then there exists $f, g \in \mathcal{D}$ for which

$$\frac{|||\Phi(f) - \Phi(g)|||}{\|f - g\|_2} > \delta + \max_{f \in \mathcal{D}} L^{\text{loc}}(f, \epsilon) \ . \tag{56}$$

Let $I = \{h \mid h = (1-t)f + tg, 0 \le t \le 1\} \subset \mathcal{D}$ be the line segment that joins f and g. Take

$$I' = \{h \mid h = (1-t)f + tg, \ t = 0, \frac{\epsilon}{2}, \epsilon, \frac{3\epsilon}{2}, \cdots, \frac{\epsilon}{2}\left\lfloor \frac{2}{\epsilon} \right\rfloor, 1\} \ .$$

Let $N = |I'|$ denote the number of elements in I'. Let $h_n = (1 - n\epsilon/2)f + (n\epsilon/2)g$ for $n = 1, \cdots, N-1$ and $h_N = g$. Then since $\|h_n - h_{n+1}\|_2 \le \epsilon$,we have

$$|||\Phi(h_n) - \Phi(h_{n+1})||| \le L^{\text{loc}}(h_n, \epsilon) \cdot \|h_n - h_{n+1}\|_2 \ , \quad n = 1, 2, \cdots, N-1 \ .$$

But $L^{\text{loc}}(h_n, \epsilon) \le \max_{f \in \mathcal{D}} L^{\text{loc}}(f, \epsilon)$, so we have

$$|||\Phi(h_n) - \Phi(h_{n+1})||| \le \max_{f \in \mathcal{D}} L^{\text{loc}}(f, \epsilon) \cdot \|h_n - h_{n+1}\|_2 \ , \quad n = 1, 2, \cdots, N-1 \ .$$

Summing over $n = 1, 2, \cdots, N-1$ and applying the triangle inequality for norms, we have

$$\begin{aligned} |||\Phi(f) - \Phi(g)||| &\le \sum_{n=1}^{N-1} |||\Phi(h_n) - \Phi(h_{n+1})||| \\ &\le \max_{f \in \mathcal{D}} L^{\text{loc}}(f, \epsilon) \cdot \sum_{n=1}^{N-1} \|h_n - h_{n+1}\|_2 \\ &= \max_{f \in \mathcal{D}} L^{\text{loc}}(f, \epsilon) \, \|f - g\|_2 \ , \end{aligned}$$

where the last equality come from the fact that h_n's are all on the same line. But this implies

$$\frac{|||\Phi(f) - \Phi(g)|||}{\|f - g\|_2} \le \max_{f \in \mathcal{D}} L^{\text{loc}}(f, \epsilon) \ ,$$

which contradicts (56). Therefore the assumption cannot be true and we conclude with (55). ■

APPENDIX F
PROOF OF LEMMA V.1

The proof of Lemma V.1 lies on the following two facts.
1) If X is SSS, then $\sigma(X(t))$, where σ is a pointwise function, is also SSS;
2) If X is SSS, then $X * g(t)$ defined as

$$(X * g)_\omega(t) = \int X_\omega(t-s)g(s)ds \ , \tag{57}$$

is also SSS. To see 1), we need to show

$$\begin{aligned} &\mathbb{P}\Big\{\sigma(X_{t_1+\tau}) \in A_1, \cdots, \sigma(X_{t_n+\tau}) \in A_n\Big\} \\ =\ &\mathbb{P}\Big\{\sigma(X_{t_1}) \in A_1, \cdots, \sigma(X_{t_n}) \in A_n\Big\} \end{aligned} \tag{58}$$

for any $t_1, \cdots, t_n, \tau \in \mathbb{R}^d$ and any $A_1, \cdots, A_n \in \mathfrak{F}$. Let $B_j = \sigma^{-1}(A_j) = \{c \in \mathbb{C} : \sigma(c) \in A_j\}$ for $j = 1, \cdots, n$. The above equality reads

$$\begin{aligned} &\mathbb{P}\Big\{X_{t_1+\tau} \in B_1, \cdots, X_{t_n+\tau} \in B_n\Big\} \\ =\ &\mathbb{P}\Big\{X_{t_1} \in B_1, \cdots, X_{t_n} \in B_n\Big\} \ , \end{aligned} \tag{59}$$

which holds true due to the assumption that X is SSS.
To see 2, note that since X is SSS there exists a semigroup of measure-preserving transformation

$$\left\{T^t : \Omega \to \Omega\right\}_{t \in \mathbb{R}^d}$$

associated with X such that

$$T^s T^t = T^{s+t}$$

for each $s, t \in \mathbb{R}^d$; and a function f such that

$$f(T^t\omega) = X_t(\omega) \ , \tag{60}$$

for each $\omega \in \Omega$, $t \in \mathbb{R}^d$. Thus

$$X * g(t) = \int f\left(T^{t-s}\omega\right) g(s)ds \ . \tag{61}$$

For any $t_1, \cdots, t_n \in \mathbb{R}^d$, $A_1, \cdots, A_n \in \mathfrak{F}$, let

$$\begin{aligned} \tilde{\Omega}_\tau = &\{\omega \in \Omega : (X * g)_{t_1+\tau}(\omega) \in A_1, \cdots, \\ &(X * g)_{t_n+\tau}(\omega) \in A_n\} \ . \end{aligned} \tag{62}$$

For $\omega \in \tilde{\Omega}_\tau$, note that $T^\tau\omega$ satisfies

$$(X * g)_{t_1}(\omega) \in A_1, \cdots, (X * g)_{t_n}(\omega) \in A_n \ .$$

Since T^τ is measure-preserving, we have $\mathbb{P}(\tilde{\Omega}_\tau) = \mathbb{P}(\tilde{\Omega}_0)$. Thus $X * g$ is SSS.

Given the two facts and that there is no dilation, Lemma V.1 is proved by tracking from the input to each output of the CNN.

APPENDIX G
PROOF OF THEOREM V.2

Since the input X and Y are SSS, so are the signals at all input and output nodes of the CNN. Therefore we can apply the Wiener-Khinchin Theorem to relate the auto-correlation with the power spectrum.

Consider an SSS process Z that are filtered by some fixed $g \in \mathcal{B}$. Denote $W = Z * g$. Then we have $R_W(0) = \int \hat{S}_W(\omega)d\omega$. Note that we have the transfer relation

$$\hat{S}_W(\omega) = \hat{S}_Z(\omega) \cdot |\hat{g}(\omega)|^2 \ . \tag{63}$$

That is to say,

$$\mathbb{E}\left(|W|^2\right) = \int \hat{R}_W(\omega)\, |\hat{g}(\omega)|^2\, d\omega \ . \tag{64}$$

More generally, due to linearity of $\mathbb{E}$, if we have two inputs Z and $\tilde{Z}$ and a family of filters $\{g_j\}_{j \in J}$, we have

$$\begin{aligned} &\mathbb{E}\left(\sum_j \left|Z * g_j - \tilde{Z} * g_j\right|^2\right) \\ =\ &\sum_j \int \hat{S}_{Z-\tilde{Z}}(\omega)\, |\hat{g}_j(\omega)|^2 \ d\omega \\ =\ &\int \hat{S}_{Z-\tilde{Z}}(\omega) \sum_j |\hat{g}_j|^2 (\omega)d\omega \\ \le\ &\int \hat{S}_{Z-\tilde{Z}}(\omega)d\omega \cdot \left\|\sum_j |\hat{g}_j|^2\right\|_\infty \\ =\ &\mathbb{E}\left(\left|Z - \tilde{Z}\right|^2\right) \cdot \left\|\sum_j |\hat{g}_j|^2\right\|_\infty \ . \end{aligned} \tag{65}$$

With this, we can compare the correlation on the first input nodes with the outputs of the CNN similar to what we did in the proof of Theorem III.1. Note that for merging, the inequalities still hold when $\|\cdot\|_2^2$ are replaced with $\mathbb{E}\,|\cdot|^2$.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their careful reading of the manuscript and constructive suggestions.

REFERENCES

[1] S. Mallat, "Group invariant scattering," *Commun. Pure Appl. Math.*, vol. 65, no. 10, pp. 1331–1398, 2012, doi: 10.1002/cpa.21413.
[2] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1872–1886, Aug. 2013.
[3] T. Wiatowski and H. Bölcskei, "Deep convolutional neural networks based on semi-discrete frames," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2015, pp. 1212–1216. [Online]. Available: http://www.nari.ee.ethz.ch/commth//pubs/p/ISIT2015
[4] T. Wiatowski and H. Bölcskei, "A mathematical theory of deep convolutional neural networks for feature extraction," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, pp. 1845–1866, Mar. 2018. [Online]. Available: http://www.nari.ee.ethz.ch/commth//pubs/p/deep-2015
[5] W. Czaja and W. Li, "Analysis of time-frequency scattering transforms," *Appl. Comput. Harmon. Anal.*, vol. 47, no. 1, pp. 149–171, 2019.
[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[8] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.
[9] C. Szegedy *et al.*, "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*. [Online]. Available: http://arxiv.org/abs/1312.6199
[10] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70. D. Precup and Y. W. Teh, Eds., Sydney, NSW, Australia: PMLR, Aug. 2017, pp. 214–223. [Online]. Available: http://proceedings.mlr.press/v70/arjovsky17a.html
[11] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein GANs," 2017, *arXiv:1704.00028*. [Online]. Available: http://arxiv.org/abs/1704.00028
[12] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.
[14] J. J. Benedetto, *Harmonic Analysis and Applications*, vol. 23. Boca Raton, FL, USA: CRC Press, 1996.
[15] C. Finlay, J. Calder, B. Abbasi, and A. Oberman, "Lipschitz regularized deep neural networks generalize and are adversarially robust," 2018, *arXiv:1808.09540*. [Online]. Available: https://arxiv.org/abs/1808.09540
[16] A. Virmaux and K. Scaman, "Lipschitz regularity of deep neural networks: Analysis and efficient estimation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31. Red Hook, NY, USA: Curran Associates, 2018, pp. 3835–3844.
[17] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 8571–8580.
[18] A. Jacot, F. Gabriel, and C. Hongler, "Freeze and chaos for DNNs: An NTK view of batch normalization, checkerboard and boundary effects," 2019, *arXiv:1907.05715*. [Online]. Available: http://arxiv.org/abs/1907.05715
[19] S. Mei, A. Montanari, and P.-M. Nguyen, "A mean field view of the landscape of two-layer neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 115, no. 33, pp. E7665–E7671, 2018. [Online]. Available: https://www.pnas.org/content/115/33/E7665
[20] S. Wright and J. Nocedal, "Numerical optimization," *Springer Sci.*, vol. 35, nos. 67–68, p. 7, 1999.
[21] R. Balan, M. Singh, and D. Zou, "Lipschitz properties for deep convolutional networks," *Contemp. Math.*, vol. 706, pp. 129–151, Jan. 2018.
[22] A. Vedaldi and K. Lenc, "Matconvnet—Convolutional neural networks for MATLAB," in *Proc. ACM Int. Conf. Multimedia*, 2015.
[23] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Honolulu, HI, USA, 2017, pp. 1765–1773.
[24] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 828–841, Oct. 2019.
[25] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2016, pp. 582–597.
[26] F. Zhang, P. P. K. Chan, B. Biggio, D. S. Yeung, and F. Roli, "Adversarial feature selection against evasion attacks," *IEEE Trans. Cybern.*, vol. 46, no. 3, pp. 766–777, Mar. 2016.
[27] L. Koralov and Y. G. Sinai, *Theory of Probability and Random Processes*. Berlin, Germany: Springer, 2007.
[28] M. Ledoux and M. Talagrand, *Probability in Banach Spaces*. Berlin, Germany: Springer-Verlag, 1991.
[29] M. Ledoux, *The Concentration of Measure Phenomenon*, no. 89. Providence, RI, USA: American Mathematical Society, 2005.
[30] P. Xanthopoulos, P. M. Pardalos, and T. B. Trafalis, *Linear Discriminant Analysis*. New York, NY, USA: Springer, 2013, pp. 27–33.
[31] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K.-R. Mullers, "Fisher discriminant analysis with kernels," in *Proc. Neural Netw. Signal Process. IX, IEEE Signal Process. Soc. Workshop*, Aug. 1999, pp. 41–48.
[32] Y. LeCun and C. Cortes. (2010). *MNIST Handwritten Digit Database*. [Online]. Available: http://yann.lecun.com/exdb/mnist/