A Spline Theory of Deep Networks

Randall Balestriero [1] Richard G. Baraniuk [1]

Abstract

We build a rigorous bridge between deep networks (DNs) and approximation theory via spline functions and operators. Our key result is that a large class of DNs can be written as a composition of *max-affine spline operators* (MASOs), which provide a powerful portal through which to view and analyze their inner workings. For instance, conditioned on the input signal, the output of a MASO DN can be written as a simple affine transformation of the input. This implies that a DN constructs a set of signal-dependent, class-specific templates against which the signal is compared via a simple inner product; we explore the links to the classical theory of optimal classification via matched filters and the effects of data memorization. Going further, we propose a simple penalty term that can be added to the cost function of any DN learning algorithm to force the templates to be orthogonal with each other; this leads to significantly improved classification performance and reduced overfitting with no change to the DN architecture. The spline partition of the input signal space opens up a new geometric avenue to study how DNs organize signals in a hierarchical fashion. As an application, we develop and validate a new distance metric for signals that quantifies the difference between their partition encodings.

1. Introduction

Deep learning has significantly advanced our ability to address a wide range of difficult machine learning and signal processing problems. Today's machine learning landscape is dominated by *deep (neural) networks* (DNs), which are compositions of a large number of simple parameterized linear and nonlinear transforms. An all-too-common story of late is that of plugging a deep network into an application as a black box, training it on copious training data, and then significantly improving performance over more classical approaches.

[1]ECE Department, Rice University, Houston, TX, USA. Correspondence to: Randall B. <randallbalestriero@gmail.com>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

Despite this empirical progress, the precise mechanisms by which deep learning works so well remain relatively poorly understood, adding an air of mystery to the entire field. Ongoing attempts to build a rigorous mathematical framework fall roughly into five camps: (i) probing and measuring networks to visualize their inner workings (Zeiler & Fergus, 2014); (ii) analyzing their properties such as expressive power (Cohen et al., 2016), loss surface geometry (Lu & Kawaguchi, 2017; Soudry & Hoffer, 2017), nuisance management (Soatto & Chiuso, 2016), sparsification (Papyan et al., 2017), and generalization abilities; (iii) new mathematical frameworks that share some (but not all) common features with DNs (Bruna & Mallat, 2013); (iv) probabilistic generative models from which specific DNs can be derived (Arora et al., 2013; Patel et al., 2016); and (v) information theoretic bounds (Tishby & Zaslavsky, 2015).

In this paper, we build a rigorous bridge between DNs and *approximation theory* via *spline functions and operators*. We prove that a large class of DNs — including convolutional neural networks (CNNs) (LeCun, 1998), residual networks (ResNets) (He et al., 2015; Targ et al., 2016), skip connection networks (Srivastava et al., 2015), fully connected networks (Pal & Mitra, 1992), recurrent neural networks (RNNs) (Graves, 2013), and beyond — can be written as spline operators. In particular, when these networks employ current standard-practice piecewise-linear, convex nonlinearities (e.g., ReLU, max-pooling, etc.) they can be written as the composition of *max-affine spline operators* (MASOs) (Magnani & Boyd, 2009; Hannah & Dunson, 2013). We focus on such nonlinearities here but note that our framework applies also to non-piecewise-linear nonlinearities through a standard approximation argument.

The max-affine spline connection provides a powerful portal through which to view and analyze the inner workings of a DN using tools from approximation theory and functional analysis. Here is a summary of our key contributions:

[C1] From our proof that a large class of DNs can be written as a composition of MASOs, it follows immediately that, *conditioned on the input signal, the output of a DN is a simple affine transformation of the input.* We illustrate in Section 4 by deriving a closed-form expression for the input/output mapping of a CNN.

[C2] The affine mapping formula enables us to interpret a MASO DN as constructing a set of *signal-dependent, class-specific templates* against which the signal is compared via a simple inner product. In Section 5 we relate DNs directly to the classical theory of optimal classification via matched filters and provide insights into the effects of *data memorization* (Zhang et al., 2016).

[C3] We propose a simple penalty term that can be added to the cost function of any DN learning algorithm to force the templates to be *orthogonal* to each other. In Section 6, we show that this leads to significantly improved classification performance and reduced overfitting on standard test data sets like CIFAR100 *with no change to the DN architecture.*

[C4] The *partition of the input space* induced by a MASO links DNs to the theory of *vector quantization* (VQ) and *K-means clustering*, which opens up a new geometric avenue to study how DNs cluster and organize signals in a hierarchical fashion. Section 7 studies the properties of the MASO partition.

[C5] Leveraging the fact that a DN considers two signals to be similar if they lie in the same MASO partition region, we develop a *new signal distance* in Section 7.3 that measures the difference between their partition encodings. The distance is easily computed via backpropagation.

A number of appendices in the Supplementary Material (SM) contain the mathematical setup and proofs. A significantly extended account of these events with numerous new results is available in (Balestriero & Baraniuk, 2018).

2. Background on Deep Networks

A *deep network* (DN) is an operator $f_\Theta : \mathbb{R}^D \to \mathbb{R}^C$ that maps an input signal[1] $\boldsymbol{x} \in \mathbb{R}^D$ to an output prediction $\widehat{\boldsymbol{y}} \in \mathbb{R}^C$ as $f_\Theta : \mathbb{R}^D \to \mathbb{R}^C$. All current DNs can be written as a composition of L intermediate mappings called *layers*

$$f_\Theta(\boldsymbol{x}) = \left(f^{(L)}_{\theta^{(L)}} \circ \cdots \circ f^{(1)}_{\theta^{(1)}}\right)(\boldsymbol{x}), \tag{1}$$

where $\Theta = \left\{\theta^{(1)}, \ldots, \theta^{(L)}\right\}$ is the collection of the network's parameters from each layer. This composition of mappings is nonlinear and non-commutative, in general.

A DN *layer* at level ℓ is an operator $f^{(\ell)}_{\theta^{(\ell)}}$ that takes as input the vector-valued signal $\boldsymbol{z}^{(\ell-1)}(\boldsymbol{x}) \in \mathbb{R}^{D^{(\ell-1)}}$ and produces the vector-valued output $\boldsymbol{z}^{(\ell)}(\boldsymbol{x}) \in \mathbb{R}^{D^{(\ell)}}$. We will assume that $\boldsymbol{x}$ and $\boldsymbol{z}^{(\ell)}$ are column vectors. We initialize with $\boldsymbol{z}^{(0)}(\boldsymbol{x}) = \boldsymbol{x}$ and will denote $\boldsymbol{z}^{(L)}(\boldsymbol{x}) =: \boldsymbol{z}$ for convenience.

[1] For concreteness, we focus here on processing K-channel images x, such as color digital photographs. But our analysis and techniques apply to signals of any index-dimensionality, including speech and audio signals, video signals, etc.

The signals $\boldsymbol{z}^{(\ell)}(\boldsymbol{x})$ are typically called *feature maps*; it is easy to see that

$$\boldsymbol{z}^{(\ell)}(\boldsymbol{x}) = \left(f^{(\ell)}_{\theta^{(\ell)}} \circ \cdots \circ f^{(1)}_{\theta^{(1)}}\right)(\boldsymbol{x}), \;\; \ell \in \{1, \ldots, L\}. \tag{2}$$

We briefly overview the basic DN operators and layers we consider in this paper; more details and additional layers are provided in (Goodfellow et al., 2016) and (Balestriero & Baraniuk, 2018). A *fully connected operator* performs an arbitrary affine transformation by multiplying its input by the dense matrix $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{D^{(\ell)} \times D^{(\ell-1)}}$ and adding the arbitrary bias vector $\boldsymbol{b}^{(\ell)}_{\boldsymbol{W}} \in \mathbb{R}^{D^{(\ell)}}$, as in $f^{(\ell)}_{\boldsymbol{W}}\left(\boldsymbol{z}^{(\ell-1)}(\boldsymbol{x})\right) := \boldsymbol{W}^{(\ell)}\boldsymbol{z}^{(\ell-1)}(\boldsymbol{x}) + \boldsymbol{b}^{(\ell)}_{\boldsymbol{W}}$. A *convolution operator* reduces the number of parameters in the affine transformation by replacing the unconstrained matrix with a multichannel convolution matrix, as in $f^{(\ell)}_{\boldsymbol{C}}\left(\boldsymbol{z}^{(\ell-1)}(\boldsymbol{x})\right) := \boldsymbol{C}^{(\ell)}\boldsymbol{z}^{(\ell-1)}(\boldsymbol{x}) + \boldsymbol{b}^{(\ell)}_{\boldsymbol{C}}$.

An *activation operator* applies a scalar nonlinear activation function σ separately to each entry of its input, as in $\left[f^{(\ell)}_{\sigma}\left(\boldsymbol{z}^{(\ell-1)}(\boldsymbol{x})\right)\right]_k := \sigma\left([\boldsymbol{z}^{(\ell-1)}(\boldsymbol{x})]_k\right)$, $k = 1, \ldots, D^{(\ell)}$. Nonlinearities are crucial to DNs, since otherwise the entire network would collapse to a single global affine transform. Three popular activation functions are the *rectified linear unit* (ReLU) $\sigma_{\text{ReLU}}(u) := \max(u, 0)$, the *leaky ReLU* $\sigma_{\text{LReLU}}(u) := \max(\eta u, u)$, $\eta > 0$, and the *absolute value* $\sigma_{\text{abs}}(u) := |u|$. These three functions are both piecewise affine and convex. Other popular activation functions include the *sigmoid* $\sigma_{\text{sig}}(u) := \frac{1}{1+e^{-u}}$ and *hyperbolic tangent* $\sigma_{\tanh}(u) := 2\sigma_{\text{sig}}(2u) - 1$. These two functions are neither piecewise affine nor convex.

A *pooling operator* subsamples its input to reduce its dimensionality according to a sub-sampling policy ρ over a collection of input indices $\{\mathcal{R}_k\}_{k=1}^{K^{(\ell)}}$ (typically a small patch), e.g., *max pooling* $\left[f^{(\ell)}_{\rho}\left(\boldsymbol{z}^{(\ell-1)}(\boldsymbol{x})\right)\right]_k := \max_{d \in \mathcal{R}^{(\ell)}_k} \left[\boldsymbol{z}^{(\ell-1)}(\boldsymbol{x})\right]_d$, $k = 1, \ldots, D^{(\ell)}$. See (Balestriero & Baraniuk, 2018) for the definitions of *average pooling*, *skip connections* and *recurrent layers*.

Definition 1. *A DN* layer $f^{(\ell)}_{\theta^{(\ell)}}$ *comprises a single nonlinear DN operator (non-affine to be precise) composed with any preceding affine operators lying between it and the preceding nonlinear operator.*

This definition yields a single, unique layer decomposition of any DN, and the complete DN is then the composition of its layers per (1). For example, in a standard CNN, there are two different layers types: i) convolution-activation and ii) max-pooling.

We form the prediction $\widehat{\boldsymbol{y}}$ by feeding $f_\Theta(\boldsymbol{x})$ through a final nonlinearity $g : \mathbb{R}^{D^{(L)}} \to \mathbb{R}^{D^{(L)}}$ as in $\widehat{\boldsymbol{y}} = g(f_\Theta(\boldsymbol{x}))$. In classification, g is typically the *softmax* nonlinearity, which arises naturally from posing the classification inference as a multinomial logistic regression problem (Bishop, 1995). In regression, typically no g is applied.

We *learn* the DN parameters Θ for a particular prediction task in a supervised setting using a labeled data set $\mathcal{D} = (\boldsymbol{x}_n, \boldsymbol{y}_n)_{n=1}^N$, a loss function, and a learning policy to update the parameters Θ in the predictor $f_\Theta(\boldsymbol{x})$. For classification problems, the loss function is typically the negative cross-entropy $\mathcal{L}_{\mathrm{CE}}(\boldsymbol{x}, \boldsymbol{y})$ (Bishop, 1995). For regression problems, the loss function is typically is the squared error. Since the layer-by-layer operations in a DN are differentiable almost everywhere with respect to their parameters and inputs, we can use some flavor of first-order optimization such as gradient descent to optimize the parameters Θ with respect to the loss function. Moreover, the gradients for all internal parameters can be computed efficiently by *backpropagation* (Hecht-Nielsen et al., 1988), which follows from the chain rule of calculus.

3. Background on Spline Operators

Approximation theory is the study of how and how well functions can best be approximated using simpler functions (Powell, 1981). A classical example of a simpler function is a *spline* $s : \mathbb{R}^D \to \mathbb{R}$ (Schmidhuber, 1994). *For concreteness, we will work exclusively with affine splines in this paper (aka "linear splines"), but our ideas generalize naturally to higher-order splines.*

Multivariate Affine Splines. Consider a *partition* of a domain $\mathbb{R}^D$ into a set of regions $\Omega = \{\omega_1, \dots, \omega_R\}$ and a set of local mappings $\Phi = \{\phi_1, \dots, \phi_R\}$ that map each region in the partition to $\mathbb{R}$ via $\phi_r(\boldsymbol{x}) := \langle [\alpha]_{r,\cdot}, \boldsymbol{x}\rangle + [\beta]_r$ for $\boldsymbol{x} \in \omega_r$.[2] The parameters are: $\alpha \in \mathbb{R}^{R\times D}$, a matrix of hyperplane "slopes," and $\beta \in \mathbb{R}^R$, a vector of hyperplane "offsets" or "biases". We will use the terms offset and bias interchangeably in the sequel. The notation $[\alpha]_{r,\cdot}$ denotes the column vector formed from the r^{th} row of α.

Then the *multivariate affine spline* is defined as

$$\begin{aligned} s[\alpha,\beta,\Omega](\boldsymbol{x}) &= \sum_{r=1}^{R} \left(\langle [\alpha]_{r,\cdot}, \boldsymbol{x}\rangle + [\beta]_r\right) \mathbf{1}(\boldsymbol{x} \in \omega_r) \\ &=: \langle \alpha[\boldsymbol{x}], \boldsymbol{x}\rangle + \beta[\boldsymbol{x}], \end{aligned} \tag{3}$$

where $\mathbf{1}(\boldsymbol{x} \in \omega_r)$ is the indicator function. The second line of (3) introduces the streamlined notation $\alpha[\boldsymbol{x}] = [\alpha]_{r,\cdot}$ when $\boldsymbol{x} \in \omega_r$; the definition for $\beta[\boldsymbol{x}]$ is similar. Such a spline is piecewise affine and hence piecewise convex. However, in general, it is neither globally affine nor globally convex unless $R = 1$, a case we denote as a *degenerate spline*, since it corresponds simply to an affine mapping.

[2] To make the connection between splines and DNs more immediately obvious, here $\boldsymbol{x}$ is interpreted as a point in $\mathbb{R}^D$, which plays the rôle of the space of signals in the other sections.

Max-Affine Spline Functions. A major complication of function approximation with splines in general is the need to jointly optimize both the spline parameters α, β and the input domain partition Ω (the "knots" for a 1D spline) (Bennett & Botkin, 1985). However, if a multivariate affine spline is constrained to be *globally convex*, then it can be rewritten as a *max-affine spline* (Magnani & Boyd, 2009; Hannah & Dunson, 2013)

$$s[\alpha,\beta,\Omega](\boldsymbol{x}) = \max_{r=1,\dots,R} \langle [\alpha]_{r,\cdot}, \boldsymbol{x}\rangle + [\beta]_r \, . \tag{4}$$

An extremely useful feature of such a spline is that *it is completely determined by its parameters α and β without needing to specify the partition Ω.* As such, we denote a max-affine spline simply as $s[\alpha, \beta]$. Changes in the parameters α, β of a max-affine spline automatically induce changes in the partition Ω, meaning that they are *adaptive partitioning splines* (Magnani & Boyd, 2009).

Max-Affine Spline Operators. A natural extension of an affine spline function is an *affine spline operator* (ASO) $S[A, B, \Omega^S]$ that produces a multivariate output. It is obtained simply by concatenating K affine spline functions from (3). The details and a more general development are provided in the SM and (Balestriero & Baraniuk, 2018).

We are particularly interested in the *max-affine spline operator* (MASO) $S[A, B] : \mathbb{R}^D \to \mathbb{R}^K$ formed by concatenating K independent max-affine spline functions from (4). A MASO with slope parameters $A \in \mathbb{R}^{K\times R\times D}$ and offset parameters $B \in \mathbb{R}^{K\times R}$ is defined as

$$\begin{aligned} S[A,B](\boldsymbol{x}) &= \begin{bmatrix} \max_{r=1,\dots,R} \langle [A]_{1,r,\cdot}, \boldsymbol{x}\rangle + [B]_{1,r} \\ \vdots \\ \max_{r=1,\dots,R} \langle [A]_{K,r,\cdot}, \boldsymbol{x}\rangle + [B]_{K,r} \end{bmatrix} \\ &=: A[\boldsymbol{x}]\, \boldsymbol{x} + B[\boldsymbol{x}]. \end{aligned} \tag{5}$$

The second line of (5) introduces the streamlined notation in terms of the signal-dependent matrix $A[\boldsymbol{x}]$ and signal-dependent vector $B[\boldsymbol{x}]$, where $[A[\boldsymbol{x}]]_{k,\cdot} := [A]_{k,r_k(\boldsymbol{x}),\cdot}$ and $[B[\boldsymbol{x}]]_k := [B]_{k,r_k(\boldsymbol{x})}$ with $r_k(\boldsymbol{x}) = \arg\max_r \langle [A]_{k,r,\cdot}, \boldsymbol{x}\rangle + [B]_{k,r}$.

Max-affine spline functions and operators are always a piecewise affine and globally convex (and hence also continuous). Conversely, any piecewise affine and globally convex function/operator can be written as a max-affine spline. Moverover, using standard approximation arguments, it is easy to show that a MASO can approximate arbitrarily closely any (nonlinear) operator that is convex in each output dimension.

4. Deep Networks are Compositions of Spline Operators

While a MASO is appropriate only for approximating convex functions/operators, we now show that virtually all of today's DNs can be written as a composition of MASOs, one for each layer. Such a composition is, in general, nonconvex and hence can approximate a much larger class of functions/operators. Interestingly, under certain broad conditions, the composition remains a piecewise affine spline operator, which enables a variety of insights into DNs.

4.1. Deep Network Operators are MASOs

We now state our main theoretical results, which are proved in the SM and elaborated in (Balestriero & Baraniuk, 2018).

Proposition 1. *An arbitrary fully connected operator $f^{(\ell)}_{\boldsymbol{W}}$ is an affine mapping and hence a degenerate MASO $S\Big[A^{(\ell)}_{\boldsymbol{W}}, B^{(\ell)}_{\boldsymbol{W}}\Big]$, with $R = 1$, $[A^{(\ell)}_{\boldsymbol{W}}]_{k,1,\cdot} = \big[\boldsymbol{W}^{(\ell)}\big]_{k,\cdot}$ and $[B^{(\ell)}_{\boldsymbol{W}}]_{k,1} = \Big[\boldsymbol{b}^{(\ell)}_{\boldsymbol{W}}\Big]_k$, leading to $\boldsymbol{W}^{(\ell)}\boldsymbol{z}^{(\ell-1)}(\boldsymbol{x}) + b^{(\ell)}_{\boldsymbol{W}} = A^{(\ell)}_{\boldsymbol{W}}[\boldsymbol{x}]\boldsymbol{z}^{(\ell-1)}(\boldsymbol{x}) + B^{(\ell)}_{\boldsymbol{W}}[\boldsymbol{x}]$. The same is true of a convolution operator with $\boldsymbol{W}^{(\ell)}, \boldsymbol{b}^{(\ell)}_{\boldsymbol{W}}$ replaced by $\boldsymbol{C}^{(\ell)}, \boldsymbol{b}^{(\ell)}_{\boldsymbol{C}}$.*

Proposition 2. *Any activation operator $f^{(\ell)}_{\sigma}$ using a piecewise affine and convex activation function is a MASO $S\Big[\boldsymbol{A}^{(\ell)}_{\sigma}, \boldsymbol{B}^{(\ell)}_{\sigma}\Big]$ with $R = 2$, $\Big[\boldsymbol{B}^{(\ell)}_{\sigma}\Big]_{k,1} = \Big[\boldsymbol{B}^{(\ell)}_{\sigma}\Big]_{k,2} = 0\ \forall k$, and for ReLU $\Big[\boldsymbol{A}^{(\ell)}_{\sigma}\Big]_{k,1,\cdot} = 0$, $\Big[\boldsymbol{A}^{(\ell)}_{\sigma}\Big]_{k,2,\cdot} = \boldsymbol{e}_k\ \forall k$; for leaky ReLU $\Big[\boldsymbol{A}^{(\ell)}_{\sigma}\Big]_{k,1,\cdot} = \nu\boldsymbol{e}_k$, $\Big[\boldsymbol{A}^{(\ell)}_{\sigma}\Big]_{k,2,\cdot} = \boldsymbol{e}_k\ \forall k, \nu > 0$; and for absolute value $\Big[\boldsymbol{A}^{(\ell)}_{\sigma}\Big]_{k,1,\cdot} = -\boldsymbol{e}_k$, $\Big[\boldsymbol{A}^{(\ell)}_{\sigma}\Big]_{k,2,\cdot} = \boldsymbol{e}_k\ \forall k$, where $\boldsymbol{e}_k$ represents the k^{th} canonical basis element of $\mathbb{R}^{D^{(\ell)}}$.*

Proposition 3. *Any pooling operator $f^{(\ell)}_{\rho}$ that is piecewise-affine and convex is a MASO $S\Big[\boldsymbol{A}^{(\ell)}_{\rho}, \boldsymbol{B}^{(\ell)}_{\rho}\Big]$.*[3] *Max-pooling has $R = \#\mathcal{R}_k$ (typically a constant over all output dimensions k), $\Big[\boldsymbol{A}^{(\ell)}_{\rho}\Big]_{k,\cdot,\cdot} = \{\boldsymbol{e}_i, i \in \mathcal{R}_k\}$, and $\Big[\boldsymbol{B}^{(\ell)}_{\rho}\Big]_{k,r} = 0\ \forall k, r$. Average-pooling is a degenerate MASO with $R = 1$, $\Big[\boldsymbol{A}^{(\ell)}_{\rho}\Big]_{k,1,\cdot} = \frac{1}{\#(\mathcal{R}_k)}\sum_{i\in\mathcal{R}_k}\boldsymbol{e}_i$, and $\Big[\boldsymbol{B}^{(\ell)}_{\rho}\Big]_{k,1} = 0\ \forall k$.*

Proposition 4. *A DN layer constructed from an arbitrary composition of fully connected/convolution operators followed by one activation or pooling operator is a MASO $S[A^{(\ell)}, B^{(\ell)}]$ such that*

$$f^{(\ell)}(\boldsymbol{z}^{(\ell-1)}(\boldsymbol{x})) = A^{(\ell)}[\boldsymbol{x}]\boldsymbol{z}^{(\ell-1)}(\boldsymbol{x}) + B^{(\ell)}[\boldsymbol{x}]. \quad (6)$$

Consequently, a large class of DNs boil down to a composition of MASOs. We prove the following in the SM and in (Balestriero & Baraniuk, 2018) for CNNs, ResNets, skip connection nets, fully connected nets, and RNNs.

Theorem 1. *A DN constructed from an arbitrary composition of fully connected/convolution, activation, and pooling operators of the types in Propositions 1–3 is a composition of MASOs that is equivalent to a global affine spline operator.*

Note carefully that, while the component layers of each of the DNs stated in Theorem 1 are MASOs, the composition of several layers is not necessarily a MASO. Indeed, a composition of MASOs remains a MASO if and only if all of its component operators (except the first) are non-decreasing with respect to each of their output dimensions (Boyd & Vandenberghe, 2004). Interestingly, ReLU and max-pooling are both non-decreasing, while leaky ReLU is strictly increasing. The culprits causing non-convexity of the composition of layers are negative entries in the fully connected or convolution operators, which destroy the required non-increasing property. A DN where these culprits are thwarted is an interesting special case, because it is convex with respect to its input (Amos et al., 2016) and multiconvex (Xu & Yin, 2013) with respect to its parameters.

Theorem 2. *A DN whose layers $\ell = 2, \dots, L$ consist of an arbitrary composition of fully connected and convolution operators with nonnegative weights, i.e., $\boldsymbol{W}^{(\ell)}_{k,j} \geq 0$, $\boldsymbol{C}^{(\ell)}_{k,j} \geq 0$; non-decreasing, piecewise-affine, and convex activation operators; and non-decreasing, piecewise-affine, and convex pooling operators is globally a MASO and thus also globally convex with respect to each of its output dimensions.*

The above results pertain to DNs using convex, affine operators. Other popular non-convex DN operators (e.g., the sigmoid and arctan activation functions) can be approximated arbitrarily closely by an affine spline operator but not by a MASO.

DNs are Signal-Dependent Affine Transformations. A common theme of the above results is that, for DNs constructed from fully connected/convolution, activation, and pooling operators of from Propositions 1–3, the operator/layer outputs $\boldsymbol{z}^{(\ell)}(\boldsymbol{x})$ are always a *signal-dependent affine function* of the input $\boldsymbol{x}$ (recall (5)). The particular affine mapping applied to $\boldsymbol{x}$ depends on which partition of the spline it falls in $\mathbb{R}^D$. More on this in Section 7 below.

DN Learning and MASO Parameters. Given labeled training data $(\boldsymbol{x}_n, \boldsymbol{y}_n)_{n=1}^N$, learning in a DN that meets the conditions of Theorem 1 (i.e., optimizing its parameters Θ) is equivalent to optimally approximating the mapping from input $\boldsymbol{x}$ to output $\widehat{\boldsymbol{y}} = g\big(\boldsymbol{z}^{(L)}(\boldsymbol{x})\big)$ using an appropriate cost function (e.g., cross-entropy for classification or squared error for regression) by learning the parameters $\theta^{(\ell)}$ of the layers. In general the overall optimization problem is nonconvex (it is actually piecewise multi-convex (Rister, 2016)).

[3] This result is agnostic of the pooling type (spatial or channel).

$$\boldsymbol{z}^{(L)}_{\text{CNN}}(\boldsymbol{x}) = \boldsymbol{W}^{(L)}\underbrace{\left(\prod_{\ell=L-1}^{1} A^{(\ell)}_{\rho}[\boldsymbol{x}]A^{(\ell)}_{\sigma}[\boldsymbol{x}]\boldsymbol{C}^{(\ell)}\right)}_{A_{\text{CNN}}[\boldsymbol{x}]}\boldsymbol{x} + \boldsymbol{W}^{(L)}\underbrace{\sum_{\ell=1}^{L-1}\left(\prod_{j=L-1}^{\ell+1} A^{(j)}_{\rho}[\boldsymbol{x}]A^{(j)}_{\sigma}[\boldsymbol{x}]\boldsymbol{C}^{(j)}\right)\left(A^{(\ell)}_{\rho}[\boldsymbol{x}]A^{(\ell)}_{\sigma}[\boldsymbol{x}]\boldsymbol{b}^{(\ell)}_{\boldsymbol{C}}\right)}_{B_{\text{CNN}}[\boldsymbol{x}]} + \boldsymbol{b}^{(L)}_{\boldsymbol{W}} \quad (7)$$

4.2. Application: DN Affine Mapping Formula

Combining Propositions 1–3 and Theorem 1 and substituting (5) into (2), we can write an explicit formula for the output of any layer $\boldsymbol{z}^{(\ell)}(\boldsymbol{x})$ of a DN in terms of the input $\boldsymbol{x}$ for a variety of different architectures. The formula for a standard CNN (using ReLU activation and max-pooling) is given in (7) above; we derive this formula and analogous formulas for ResNets and RNNs in (Balestriero & Baraniuk, 2018). In (7), $A^{(\ell)}_{\sigma}[\boldsymbol{x}]$ are the signal-dependent matrices corresponding to the ReLU activations, $A^{(\ell)}_{\rho}[\boldsymbol{x}]$ are the signal-dependent matrices corresponding to max-pooling, and the biases $\boldsymbol{b}^{(L)}_{\boldsymbol{W}}, \boldsymbol{b}^{(\ell)}_{\boldsymbol{C}}$ arise directly from the fully connected and convolution operators. The absence of $B^{(\ell)}_{\sigma}[\boldsymbol{x}]$, $B^{(\ell)}_{\rho}[\boldsymbol{x}]$ is due to the absence of bias in the ReLU (recall (2)) and max-pooling operators (recall (3)).

Inspection of (7) reveals the exact form of the signal-dependent, piecewise affine mapping linking $\boldsymbol{x}$ to $\boldsymbol{z}^{(L)}_{\text{CNN}}(\boldsymbol{x})$. Moreover, this formula can be collapsed into

$$\boldsymbol{z}^{(L)}_{\text{CNN}}(\boldsymbol{x}) = \boldsymbol{W}^{(L)}\left(A_{\text{CNN}}[\boldsymbol{x}]\,\boldsymbol{x} + B_{\text{CNN}}[\boldsymbol{x}]\right) + \boldsymbol{b}^{(L)}_{\boldsymbol{W}} \quad (8)$$

from which we can recognize

$$\boldsymbol{z}^{(L-1)}_{\text{CNN}}(\boldsymbol{x}) = A_{\text{CNN}}[\boldsymbol{x}]\,\boldsymbol{x} + B_{\text{CNN}}[\boldsymbol{x}] \quad (9)$$

as an explicit, signal-dependent, affine formula for the *featurization process* that aims to convert $\boldsymbol{x}$ into a set of (hopefully) linearly separable features that are then input to the linear classifier in layer $\ell = L$ with parameters $\boldsymbol{W}^{(L)}$ and $\boldsymbol{b}^{(L)}_{\boldsymbol{W}}$. Of course, the final prediction $\widehat{\boldsymbol{y}}$ is formed by running $\boldsymbol{z}^{(L)}_{\text{CNN}}(x)$ through a softmax nonlinearity g, but this merely rescales its entries to create a probability distribution.

5. DNs are Template Matching Machines

We now dig deeper into (8) in order to bridge DNs and classical optimal classification theory. While we focus on CNNs and classification for concreteness, our analysis holds for any DN meeting the conditions of Theorem 1.

5.1. Template Matching

An alternate interpretation of (8) is that $\boldsymbol{z}^{(L)}_{\text{CNN}}(\boldsymbol{x})$ is the output of a bank of linear *matched filters* (plus a set of biases). That is, the c^{th} element of $\boldsymbol{z}^{(L)}(\boldsymbol{x})$ equals the inner product between the signal $\boldsymbol{x}$ and the matched filter for the c^{th} class, which is contained in the c^{th} row of the matrix $\boldsymbol{W}^{(L)}A[\boldsymbol{x}]$. The bias $\boldsymbol{W}^{(L)}B[\boldsymbol{x}] + \boldsymbol{b}^{(L)}_{\boldsymbol{W}}$ can be used to account for the fact that some classes might be more likely than others (i.e., the prior probability over the classes). It is well-known that a matched filterbank is the optimal classifier for deterministic signals in additive white Gaussian noise (Rabiner & Gold, 1975). Given an input $\boldsymbol{x}$, the class decision is simply the index of the largest element of $\boldsymbol{z}^{(L)}(\boldsymbol{x})$.[4]

Yet another interpretation of (8) is that $\boldsymbol{z}^{(L)}(\boldsymbol{x})$ is computed not in a single matched filter calculation but hierarchically as the signal propagates through the DN layers. Abstracting (5) to write the per-layer maximization process as $\boldsymbol{z}^{(\ell)}(\boldsymbol{x}) = \max_{r^{(\ell)}} A^{(\ell)}_{r^{(\ell)}}\boldsymbol{z}^{(\ell-1)}(\boldsymbol{x}) + B^{(\ell)}_{r^{(\ell)}}$ and cascading, we obtain a formula for the end-to-end DN mapping

$$\boldsymbol{z}^{(L)}(\boldsymbol{x}) = \boldsymbol{W}^{(L)}\max_{r^{(L-1)}}\left(\boldsymbol{A}^{(L-1)}_{r^{(L-1)}}\max_{r^{(2)}}\left(\boldsymbol{A}^{(2)}_{r^{(2)}}\cdots \max_{r^{(1)}}\left(\boldsymbol{A}^{(1)}_{r^{(1)}}\boldsymbol{x} + \boldsymbol{B}^{(1)}_{r^{(1)}}\right) + \boldsymbol{B}^{(2)}_{r^{(2)}}\right)\cdots + \boldsymbol{B}^{(L-1)}_{r^{(L-1)}}\right) + b^{(L)}_{W}. \quad (10)$$

This formula elucidates that a DN performs a *hierarchical, greedy template matching* on its input, a computationally efficient yet sub-optimal template matching technique. Such a procedure is globally optimal when the DN is globally convex.

Corollary 1. *For a DN abiding by the requirements of Theorem 2, the computation (10) collapses to the following globally optimal template matching*

$$\boldsymbol{z}^{(L-1)}(\boldsymbol{x}) = \boldsymbol{W}^{(L)}\max_{r^{(L-1)},r^{(2)},\dots,r^{(1)}}\left(\boldsymbol{A}^{(L-1)}_{r^{(L-1)}}\left(\boldsymbol{A}^{(2)}_{r^{(2)}}\cdots\left(\boldsymbol{A}^{(1)}_{r^{(1)}}\boldsymbol{x} + \boldsymbol{B}^{(1)}_{r^{(1)}}\right) + \boldsymbol{B}^{(2)}_{r^{(2)}}\right)\cdots + \boldsymbol{B}^{(L-1}_{r^{(L-1)}}\right) + b^{(L)}_{W}. \quad (11)$$

5.2. Template Visualization Examples

Since the complete DN mapping (up to the final softmax) can be expressed as in (8), given a signal $\boldsymbol{x}$, we can compute the signal-dependent template for class c via $A[\boldsymbol{x}]_c = \frac{[\boldsymbol{z}^{(L)}(\boldsymbol{x})]_c}{d\boldsymbol{x}}$, which can be efficiently computed via backpropogation (Hecht-Nielsen et al., 1988).[5] Once the template $A[\boldsymbol{x}]_c$ has been computed, the bias term $b[\boldsymbol{x}]_c$ can

[4]Again, since the softmax merely rescales the entries of $\boldsymbol{z}^{(L)}(\boldsymbol{x})$ into a probability distribution, it does not affect the location of its largest element.

[5]In fact, we can use the same backpropagation procedure used for computing the gradient with respect to a fully connected or

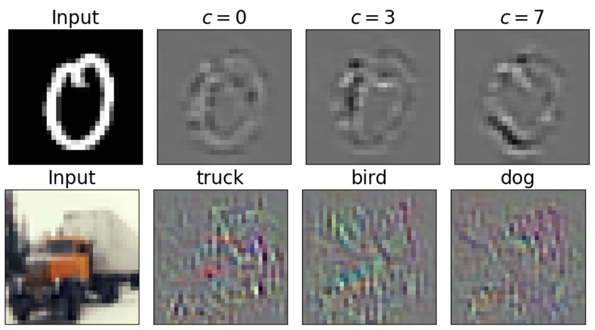


Figure 1. Visualization of the matched filter templates generated by a CNN using ReLU activation and max-pooling. At top, we show the input image x of the digit '0' from the MNIST dataset along with the three signal-dependent templates $A_{\text{CNN}}[x]_{('0',\cdot)}$, $A_{\text{CNN}}[x]_{('3',\cdot)}$, and $A_{\text{CNN}}[x]_{('7',\cdot)}$. The inner products between x and the three templates are 32.7, -27.7, -27.8, respectively (c.f. (12)). Below, we repeat the same experiment with the input image x of a 'truck' from the CIFAR10 dataset along with the three signal-dependent templates corresponding to the classes 'truck,' 'bird,' and 'dog.' The inner products between x and the three templates are 30.6, -12.5, -16.2, respectively. (The final classification will involve not only these inner products but also the relevant biases and softmax transformation.)

be computed as $b[\boldsymbol{x}]_c = \boldsymbol{z}^{(L)}(\boldsymbol{x})_c - \langle A[\boldsymbol{x}]_{c,\cdot}, \boldsymbol{x}\rangle$. Figure 1 plots various signal-dependent templates for two CNNs trained on the MNIST and CIFAR10 datasets.

5.3. Collinear Templates and Data Set Memorization

Under the matched filterbank interpretation of a DN developed in Section 5.1, the optimal template for an image $\boldsymbol{x}$ of class c is a scaled version of $\boldsymbol{x}$ itself. But what are the optimal templates for the other (incorrect) classes? In an idealized setting, we can answer this question.

Proposition 5. *Consider an idealized DN consisting of a composition of MASOs that has sufficient approximation power to span arbitrary MASO matrices $A[\boldsymbol{x}_n]$ from (9) for any input $\boldsymbol{x}_n$ from the training set. Train the DN to classify among C classes using the training data $\mathcal{D} = (\boldsymbol{x}_n, y_n)_{n=1}^N$ with normalized inputs $\|\boldsymbol{x}_n\|_2 = 1\ \forall n$ and the cross-entropy loss $\mathcal{L}_{\text{CE}}(y_n, f_\Theta(\boldsymbol{x}_n))$ with the addition of the regularization constraint that $\sum_c \|A[\boldsymbol{x}_n]_{c,\cdot}\|_2 < \alpha$ with $\alpha > 0$. At the global minimum of this constrained optimization problem, the rows of $A^\star[\boldsymbol{x}_n]$ (the optimal templates) have the form:*

$$[A^\star[\boldsymbol{x}_n]]_{c,\cdot} = \begin{cases} +\sqrt{\frac{(C-1)\alpha}{C}}\,\boldsymbol{x}_n, & c = y_n \\ -\sqrt{\frac{\alpha}{C(C-1)}}\,\boldsymbol{x}_n, & c \neq y_n \end{cases} \tag{12}$$

In short, the idealized CNN in the proposition will *memorize*

convolution weight but instead with the input x. This procedure is becoming increasingly popular in the study of adversarial examples (Szegedy et al., 2013).

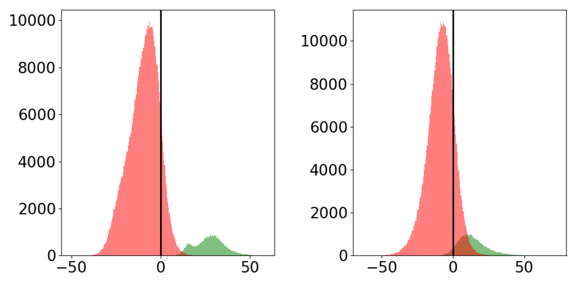


Figure 2. Empirical study of the implications of Proposition 5. We illustrate the bimodality of $\langle [A^\star[\boldsymbol{x}_n]]_{c,\cdot}, \boldsymbol{x}_n\rangle$ for the *largeCNN* matched filterbank trained on (a) MNIST and (b) CIFAR10. Training used batch normalization and bias units. Results are similar when trained with bias and no batch-normalization. The green histogram summarizes the output values for the correct class (top half of (12)), while the red histogram summarizes the output values for the incorrect classes (bottom half of (12)). The easier the classification problem (MNIST), the more bimodal the distribution.

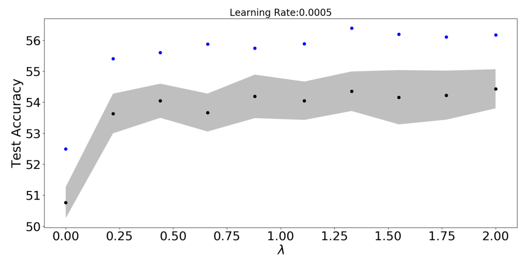
a set of collinear templates whose bimodal outputs force the softmax output to a Dirac delta function (aka *1-hot representation*) that peaks at the correct class. Figure 2 confirms this bimodal behavior on the MNIST and CIFAR10 datasets.

6. New DNs with Orthogonal Templates

While a DN's signal-dependent matched filterbank (8) is optimized for classifying signals immersed in additive white Gaussian noise, such a statistical model is overly simplistic for most machine learning problems of interest. In practice, errors will arise not just from random noise but also from nuisance variations in the inputs such as arbitrary rotations, positions, and modalities of the objects of interest. The effects of these nuisances are only poorly approximated as Gaussian random errors. Limited work has been done on filterbanks for classification in nonGaussian noise; one promising direction involves using not matched but rather *orthogonal* templates (Eldar & Oppenheim, 2001).

For a MASO DN's templates to be orthogonal for all inputs, it is sufficient that the rows of the matrix $\boldsymbol{W}^{(L)}$ in the final linear classifer layer be orthogonal. This weak constraint on the DN still enables the earlier layers to create a high-performance, class-agnostic, featurized representation (recall the discussion just below (9)). To create orthogonal templates during learning, we simply add to the standard (potentially regularized) cross-entropy loss function $\mathcal{L}_{\text{CE}}$ a term that penalizes non-zero off-diagonal entries in the matrix $\boldsymbol{W}^{(L)}(\boldsymbol{W}^{(L)})^T$ leading to the new loss with the additional penalty

$$\mathcal{L}_{\text{CE}} + \lambda \sum_{c_1 \neq c_2} \left| \left\langle \left[\boldsymbol{W}^{(L)}\right]_{c_1,\cdot}, \left[\boldsymbol{W}^{(L)}\right]_{c_2,\cdot} \right\rangle \right|^2. \tag{13}$$

(a) 

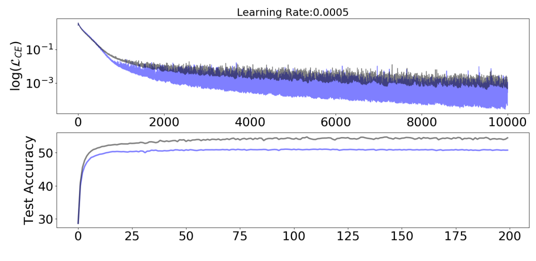
(b) 

Figure 3. Orthogonal templates significantly boost DN performance with *no change to the architecture*. (a) Classification performance of the *largeCNN* trained on CIFAR100 for different values of the orthogonality penalty λ in (13). We plot the average (back dots), standard deviation (gray shade), and maximum (blue dots) of the test set accuracy over 15 runs. (b, top) Training set error. The blue/black curves corresponds to $\lambda = 0/1$. (b, bottom) Test set accuracy over the course of the learning.

The parameter λ controls the tradeoff between cross-entropy minimization and orthogonality preservation. Conveniently, when minimizing (13) via backpropagation, the orthogonal rows of $\boldsymbol{W}^{(L)}$ induce orthogonal backpropagation updates for the various classes.

We now empirically demonstrate that orthogonal templates lead to significantly improved classification performance. We conducted a range of experiments with three different conventional DN architectures – *smallCNN*, *largeCNN*, and *ResNet4-4* – trained on three different datasets – SVHN, CIFAR10, and CIFAR100. Each DN employed bias units, ReLU activations, and max-pooling as well as batch-normalization prior each ReLU. The full experimental details are given in the (Balestriero & Baraniuk, 2018). For learning, we used the Adam optimizer with an exponential learning rate decay. All inputs were centered to zero mean and scaled to a maximum value of one. No further pre-processing was performed, such as ZCA whitening (Nam et al., 2014). We assessed how the classification performance of a given DN would change as we varied the orthogonality penalty λ in (13). For each configuration of DN architecture, training dataset, learning rate, and penalty λ, we averaged over 15 runs to estimate the average performance and standard deviation.

We report here on only the CIFAR100 with *largeCNN* experiments. (See (Balestriero & Baraniuk, 2018) for detailed results for all three datasets and the other architectures. The trends for all three datasets are similar and are independent of the learning rate.) The results for CIFAR100 in Figure 3 indicate that the benefits of the orthogonality penalty emerge distinctly as soon as $\lambda > 0$. In addition to improved final accuracy and generalization performance, we see that template orthogonality reduces the temptation of the DN to overfit. (This is is especially visible in the examples in (Balestriero & Baraniuk, 2018).) One explanation is that the orthogonal weights $\boldsymbol{W}^{(L)}$ positively impact not only the prediction but also the backpropagation via orthogonal gradient updates with respect to each output dimension's partial derivatives.

7. DN's Intrinsic Multiscale Partition

Like any spline, it is the interplay between the (affine) spline mappings and the input space partition that work the magic in a MASO DN. Recall from Section 3 that a MASO has the attractive property that it implicitly partitions its input space as a function of its slope and offset parameters. The induced partition Ω opens up a new geometric avenue to study how a DN clusters and organizes signals in a hierarchical fashion.

7.1. Effect of the DN Operators on the Partition

A DN operator at level ℓ directly influences the partitioning of its input space $\mathbb{R}^{D^{(\ell-1)}}$ and indirectly influences the partitioning of the overall signal space $\mathbb{R}^{D}$.

A ReLU activation operator splits each of its input dimensions into two half-planes depending on the sign of the input in each dimension. This partitions $\mathbb{R}^{D^{(\ell-1)}}$ into a combinatorially large number (up to $2^{D^{(\ell)}}$) of regions. Following a fully connected or convolution operator with a ReLU simply rotates the partition in $\mathbb{R}^{D^{(\ell-1)}}$. A max-pooling operator also partitions $\mathbb{R}^{D^{(\ell-1)}}$ into a combinatorially large number (up to $\#R^{D^{(\ell)}}$) of regions, where $\#R$ is the size of the pooling region.

This per-MASO partitioning of each layer's input space constructs an overall partitioning of the input signal space $\mathbb{R}^{D}$. As each MASO is applied, it subdivides the input space $\mathbb{R}^{D}$ into finer and finer partitions. The final partition corresponds to the intersection of all of the intermediate partitions, and hence we can encode the input in terms of the ordered collection of per-layer partition regions into which it falls. This overall process can be interpreted as a *hierarchical vector quantization* (VQ) of the training input signals $\boldsymbol{x}_n$. There are thus many potential connections between DNs and optimal quantization, information theory, and clustering that we leave for future research. See (Balestriero & Baraniuk, 2018) for some early results.

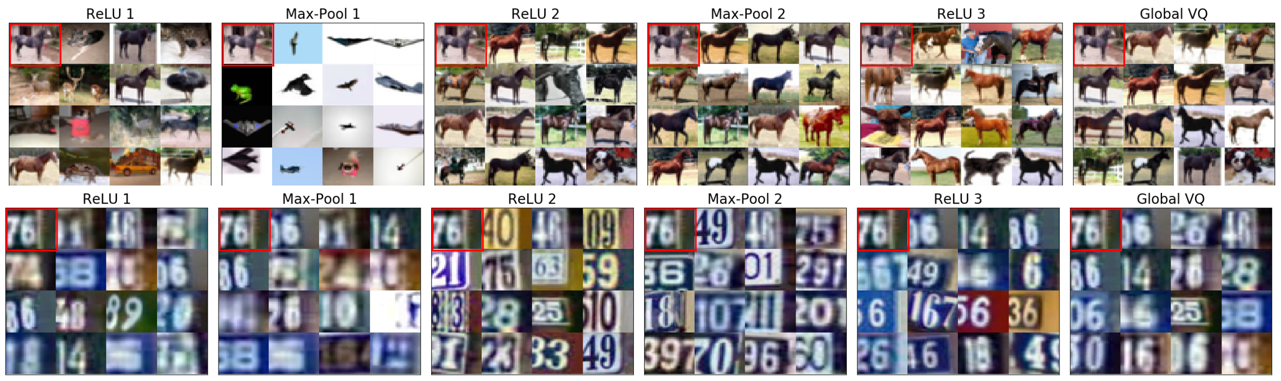


Figure 4. Example of the partition-based distance between images in the CIFAR10 (top) and SVHN (bottom) datasets. In each image, we plot the target image $\boldsymbol{x}$ at top-left and its 15 nearest neighbors in the partition-based distance at layer ℓ of a *smallCNN* trained without batch normalization (see (Balestriero & Baraniuk, 2018) for the details). From left to right, the images display layers $\ell = 1, \dots, 6$, and we see that the images become further and further in Euclidean distance but closer and closer in featurized distance. In particular, the first layer neighbors share similar colors and shapes (and thus are closer in Euclidean distance). Later layer neighbors mainly come from the same class independently of their color or shape.

7.2. Inferring a DN Layer's Intrinsic Partition

Unfortunately there is no simple formula for the partition of the signal space. However, once can obtain the set of inputs signals $\boldsymbol{x}_n$ that fall into the same partition region at each layer of a DN. At layer ℓ, denote the index of the region selected by the input $\boldsymbol{x}$ (recall (6)) by

$$\left[t^{(\ell)}(\boldsymbol{x})\right]_k = \arg\max_r \left\langle [A^{(\ell)}]_{k,r,.}, \boldsymbol{z}^{(\ell-1)}(\boldsymbol{x})\right\rangle + [B^{(\ell)}]_{k,r}, \tag{14}$$

Thus, $[t^{(\ell)}]_k \in \{1, \dots, R^{(\ell)}\}$, with $R^{(\ell)}$ the number of partition region in the layer's input space. Encoding the partition as an ordered collection of integers designating the activate hyperplane parameters from (4), we can now visualize which inputs fall into the same or nearby partitions.

Due to the very large number of possible regions (up to $2^{D^{(\ell)}}$ for a ReLU at layer ℓ) and the limited amount of training data, in general, most partitions will be empty or contain only a single training data point.

7.3. A New Image Distance based on the DN Partition

To validate the utility of the hierarchical intrinsic clustering induced by a DN, we define a new distance function between the signals $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ that quantifies the similarity of their position encodings $t^{\ell}(\boldsymbol{x}_1)$ and $t^{\ell}(\boldsymbol{x}_2)$ at layer ℓ via

$$\begin{aligned} &d\Big(t^{(\ell)}(\boldsymbol{x}_1), t^{(\ell)}(\boldsymbol{x}_2)\Big) \\ &\quad = 1 - \frac{\sum_{k=1}^{D^{(\ell)}} \mathbf{1}\left([t^{(\ell)}(\boldsymbol{x}_1)]_k = [t^{(\ell)}(\boldsymbol{x}_2)]_k\right)}{D^{(\ell)}}. \end{aligned} \tag{15}$$

For a ReLU MASO, this corresponds simply to counting how many entries of the layer inputs for $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are positive or negative at the same positions. For a max-pooling MASO, this corresponds to counting how many argmax positions are the same in each patch for $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$.

Figure 4 provides a visualization of the nearest neighbors of a test image under this partition-based distance measure. Visual inspection of the figures highlights that, as we progress through the layers of the DN, similar images become closer in the new distance but further in Euclidean distance.

8. Conclusions and Future Work

We have used the theory of splines to build a rigorous bridge between deep networks (DNs) and approximation theory. Our key finding is that, conditioned on the input signal, the output of a DN can be written as a simple affine transformation of the input. This links DNs directly to the classical theory of optimal classification via matched filters and provides insights into the positive effects of data memorization.

There are many avenues for future work, including a more in-depth analysis of the hierarchical MASO partitioning, particularly from the viewpoint of vector quantization and K-means clustering, which are unsupervised learning techniques, and information theory. The spline viewpoint also could inspire the creation of new DN layers that have certain attractive partitioning or approximation capabilities. We have begun exploring some of these directions in (Balestriero & Baraniuk, 2018).[6]

[6] This work supported by DOD Vannevar Bush Faculty Fellowship grant N00014-18-1-2047, NSF grant CCF-1527501, ARO grant W911NF-15-1-0316, AFOSR grant FA9550-14-1-0088, ONR grant N00014-17-1-2551, DARPA grant HR0011-16-C-0028, and an ONR BRC grant for Randomized Numerical Linear Algebra.

References

Amos, B., Xu, L., and Kolter, J. Z. Input convex neural networks. *arXiv preprint arXiv:1609.07152*, 2016.

Arora, S., Bhaskara, A., Ge, R., and Ma, T. Provable bounds for learning some deep representations. *arXiv preprint arXiv:1310.6343*, 2013.

Balestriero, R. and Baraniuk, R. A spline theory of deep networks (extended version). *arXiv preprint arXiv:1805.06576*, 2018.

Bennett, J. and Botkin, M. Structural shape optimization with geometric description and adaptive mesh refinement. *AIAA journal*, 23(3):458–464, 1985.

Bishop, C. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

Boyd, S. and Vandenberghe, L. *Convex Optimization*. Cambridge University Press, 2004.

Bruna, J. and Mallat, S. Invariant scattering convolution networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8): 1872–1886, June 2013.

Cohen, N., Sharir, O., and Shashua, A. On the expressive power of deep learning: A tensor analysis. In *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pp. 698–728. PMLR, 2016.

Eldar, Y. and Oppenheim, A. Orthogonal matched filter detection. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 5, pp. 2837–2840. IEEE, 2001.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. *Deep Learning*, volume 1. MIT Press Cambridge, 2016.

Graves, A. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

Hannah, L. A. and Dunson, D. B. Multivariate convex regression with adaptive partitioning. *The Journal of Machine Learning Research*, 14(1):3261–3294, 2013.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

Hecht-Nielsen, R. et al. Theory of the backpropagation neural network. *Neural Networks*, 1(Supplement-1):445–448, 1988.

LeCun, Y. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Lu, H. and Kawaguchi, K. Depth creates no bad local minima. *arXiv preprint arXiv:1702.08580*, 2017.

Magnani, A. and Boyd, S. P. Convex piecewise-linear fitting. *Optimization and Engineering*, 10(1):1–17, 2009.

Nam, W., Dollár, P., and Han, J. Local decorrelation for improved pedestrian detection. In *Advances in Neural Information Processing Systems*, pp. 424–432, 2014.

Pal, S. K. and Mitra, S. Multilayer perceptron, fuzzy sets, and classification. *IEEE Transactions on neural networks*, 3(5):683–697, 1992.

Papyan, V., Romano, Y., and Elad, M. Convolutional neural networks analyzed via convolutional sparse coding. *Journal of Machine Learning Research*, 18(83):1–52, 2017.

Patel, A. B., Nguyen, T., and Baraniuk, R. G. A probabilistic framework for deep learning. *NIPS*, 2016.

Powell, M. J. D. *Approximation theory and methods*. Cambridge university press, 1981.

Rabiner, L. and Gold, B. Theory and application of digital signal processing. *Englewood Cliffs, NJ, Prentice-Hall, Inc., 1975. 777 p.*, 1975.

Rister, B. Piecewise convexity of artificial neural networks. *arXiv preprint arXiv:1607.04917*, 2016.

Schmidhuber, J. Discovering problem solutions with low kolmogorov complexity and high generalization capability. In *Machine Learning: Proceedings of the Twelfth International Conference*. Citeseer, 1994.

Soatto, S. and Chiuso, A. Visual representations: Defining properties and deep approximations. In *Proc. Int. Conf. Learn. Rep. (ICLR'16)*, Nov. 2016.

Soudry, D. and Hoffer, E. Exponentially vanishing sub-optimal local minima in multilayer neural networks. *arXiv preprint arXiv:1702.05777*, 2017.

Srivastava, R. K., Greff, K., and Schmidhuber, J. Training very deep networks. In *Advances in Neural Information Processing Systems*, pp. 2377–2385, 2015.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Targ, S., Almeida, D., and Lyman, K. Resnet in resnet: generalizing residual architectures. *arXiv preprint arXiv:1603.08029*, 2016.

Tishby, N. and Zaslavsky, N. Deep learning and the information bottleneck principle. In *Information Theory Workshop (ITW), 2015 IEEE*, pp. 1–5. IEEE, 2015.

Xu, Y. and Yin, W. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on Imaging Sciences*, 6(3):1758–1789, 2013.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In *Proc. European Conf. Comp. Vision (ECCV'14)*, pp. 818–833. Sep. 2014.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.