JMLR: Workshop and Conference Proceedings vol 49:1–31, 2016

On the Expressive Power of Deep Learning: A Tensor Analysis

Nadav Cohen COHENNADAV@CS.HUJI.AC.IL
Or Sharir OR.SHARIR@CS.HUJI.AC.IL
Amnon Shashua SHASHUA@CS.HUJI.AC.IL
The Hebrew University of Jerusalem

Abstract

It has long been conjectured that hypotheses spaces suitable for data that is compositional in nature, such as text or images, may be more efficiently represented with deep hierarchical networks than with shallow ones. Despite the vast empirical evidence supporting this belief, theoretical justifications to date are limited. In particular, they do not account for the locality, sharing and pooling constructs of convolutional networks, the most successful deep learning architecture to date. In this work we derive a deep network architecture based on arithmetic circuits that inherently employs locality, sharing and pooling. An equivalence between the networks and hierarchical tensor factorizations is established. We show that a shallow network corresponds to CP (rank-1) decomposition, whereas a deep network corresponds to Hierarchical Tucker decomposition. Using tools from measure theory and matrix algebra, we prove that besides a negligible set, all functions that can be implemented by a deep network of polynomial size, require exponential size in order to be realized (or even approximated) by a shallow network. Since log-space computation transforms our networks into SimNets, the result applies directly to a deep learning architecture demonstrating promising empirical performance. The construction and theory developed in this paper shed new light on various practices and ideas employed by the deep learning community.

Keywords: *Deep Learning*, *Expressive Power*, *Arithmetic Circuits*, *Tensor Decompositions*

1. Introduction

The expressive power of neural networks is achieved through depth. There is mounting empirical evidence that for a given budget of resources (e.g. neurons), the deeper one goes, the better the eventual performance will be. However, existing theoretical arguments that support this empirical finding are limited. There have been many attempts to theoretically analyze function spaces generated by network architectures, and their dependency on network depth and size. The prominent approach for justifying the power of depth is to show that deep networks can efficiently express functions that would require shallow networks to have super-polynomial size. We refer to such scenarios as instances of *depth efficiency*. Unfortunately, existing results dealing with depth efficiency (e.g. Hastad (1986); Håstad and Goldmann (1991); Delalleau and Bengio (2011); Martens and Medabalimi (2014)) typically apply to specific network architectures that do not resemble ones commonly used in practice. In particular, none of these results apply to convolutional networks (LeCun and Bengio (1995)), which represent the most empirically successful and widely used deep learning architecture to date. A further limitation of current results is that they merely show *existence* of depth efficiency (i.e. of functions that are efficiently realizable with a certain depth but cannot be efficiently realized with shallower depths), without providing any information as to how frequent this property is. These shortcomings of current theory are the ones that motivated our work.

The architectural features that specialize convolutional networks compared to classic feed-forward fully-connected networks are threefold. The first feature, *locality*, refers to the connection of a neuron only to neighboring neurons in the preceding layer, as opposed to having the entire layer drive it. In the context of image processing (the most common application of convolutional networks), locality is believed to reflect the inherent compositional structure of data – the closer pixels are in an image, the more likely they are to be correlated. The second architectural feature of convolutional networks is *sharing*, which means that different neurons in the same layer, connected to different neighborhoods in the preceding layer, share the same weights. Sharing, which together with locality gives rise to convolution, is motivated by the fact that in natural images, the semantic meaning of a pattern often does not depend on its location (i.e. two identical patterns appearing in different locations of an image often convey the same semantic content). Finally, the third architectural idea of convolutional networks is *pooling*, which is essentially an operator that decimates layers, replacing neural activations in a spatial window by a single value (e.g. their maximum or average). In the context of images, pooling induces invariance to translations (which often do not affect semantic content), and in addition is believed to create a hierarchy of abstraction in the patterns neurons respond to. The three architectural elements of locality, sharing and pooling, which have facilitated the great success of convolutional networks, are all lacking in existing theoretical studies of depth efficiency.

In this paper we introduce a *convolutional arithmetic circuit* architecture that incorporates locality, sharing and pooling. Arithmetic circuits (also known as Sum-Product Networks, Poon and Domingos (2011)) are networks with two types of nodes: sum nodes, which compute a weighted sum of their inputs, and product nodes, computing the product of their inputs. We use sum nodes to implement convolutions (locality with sharing), and product nodes to realize pooling. The models we arrive at may be viewed as convolutional networks with product pooling and linear point-wise activation. They are attractive on three accounts. First, as discussed in app. E, convolutional arithmetic circuits are equivalent to SimNets, a new deep learning architecture that has recently demonstrated promising empirical results on various image recognition benchmarks (Cohen et al. (2016)). Second, as we show in sec. 3, convolutional arithmetic circuits are realizations of hierarchical tensor decompositions (see Hackbusch (2012)), opening the door to various mathematical and algorithmic tools for their analysis and implementation. Third, the depth efficiency of convolutional arithmetic circuits, which we analyze in sec. 4, was shown in the subsequent work of Cohen and Shashua (2016) to be superior to the depth efficiency of the popular convolutional rectifier networks, namely convolutional networks with rectified linear (ReLU) activation and max or average pooling.

Employing machinery from measure theory and matrix algebra, made available through their connection to hierarchical tensor decompositions, we prove a number of fundamental results concerning the depth efficiency of our convolutional arithmetic circuits. Our main theoretical result (thm. 1 and corollary 2) states that *besides a negligible (zero measure) set, all functions that can be realized by a deep network of polynomial size, require exponential size in order to be realized, or even approximated, by a shallow network*. When translated to the viewpoint of tensor decompositions, this implies that *almost all* tensors realized by Hierarchical Tucker (HT) decomposition (Hackbusch and Kühn (2009)) cannot be efficiently realized by the classic CP (rank-1) decomposition. To the best of our knowledge, this result is unknown to the tensor analysis community, in which the advantage of HT over CP is typically demonstrated through *specific examples* of tensors that can be efficiently realized by the former and not by the latter. Following our main result, we present a generalization (thm. 3 and corollary 4) that compares networks of arbitrary depths, show-

ing that the amount of resources one has to pay in order to maintain representational power while trimming down layers of a network grows double exponentially w.r.t. the number of layers cut off. We also characterize cases in which dropping a single layer bears an exponential price.

The remainder of the paper is organized as follows. In sec. 2 we briefly review notations and mathematical background required in order to follow our work. This is followed by sec. 3, which presents our convolutional arithmetic circuits and establishes their equivalence with tensor decompositions. Our theoretical analysis is covered in sec. 4. Finally, sec. 5 concludes. In order to keep the manuscript at a reasonable length, we defer our detailed survey of related work to app. D, covering works on the depth efficiency of boolean circuits, arithmetic circuits and neural networks, as well as different applications of tensor analysis in the field of deep learning.

2. Preliminaries

We begin by establishing notational conventions that will be used throughout the paper. We denote vectors using bold typeface, e.g. $\mathbf{v} \in \mathbb{R}^s$. The coordinates of such a vector are referenced with regular typeface and a subscript, e.g. $v_i \in \mathbb{R}$. This is not to be confused with *bold* typeface and a subscript, e.g. $\mathbf{v}_i \in \mathbb{R}^s$, which represents a vector that belongs to some sequence. Tensors (multi-dimensional arrays) are denoted by the letters "A" and "B" in calligraphic typeface, e.g. $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{M_1 \times \cdots \times M_N}$. A specific entry in a tensor will be referenced with subscripts, e.g. $\mathcal{A}_{d_1 \ldots d_N} \in \mathbb{R}$. Superscripts will be used to denote individual objects within a collection. For example, $\mathbf{v}^{(i)}$ stands for vector i and $\mathcal{A}^y$ stands for tensor y. In cases where the collection of interest is indexed by multiple coordinates, we will have multiple superscripts referencing individual objects, e.g. $\mathbf{a}^{l,j,\gamma}$ will stand for vector (l, j, γ). As shorthand for the Cartesian product of the Euclidean space $\mathbb{R}^s$ with itself N times, we will use the notation $(\mathbb{R}^s)^N$. Finally, for a positive integer k we use the shorthand $[k]$ to denote the set $\{1, \ldots, k\}$.

We now turn to establish a baseline, i.e. to present basic definitions and results, in the broad and comprehensive field of tensor analysis. We list here only the essentials required in order to follow the paper, referring the interested reader to Hackbusch (2012) for a more complete introduction to the field [1]. The most straightforward way to view a tensor is simply as a multi-dimensional array: $\mathcal{A}_{d_1,\ldots,d_N} \in \mathbb{R}$ where $i \in [N], d_i \in [M_i]$. The number of indexing entries in the array, which are also called *modes*, is referred to as the *order* of the tensor. The term *dimension* stands for the number of values an index can take in a particular mode. For example, the tensor $\mathcal{A}$ appearing above has order N and dimension M_i in mode i, $i \in [N]$. The space of all possible configurations $\mathcal{A}$ can take is called a *tensor space* and is denoted, quite naturally, by $\mathbb{R}^{M_1 \times \cdots \times M_N}$.

A central operator in tensor analysis is the *tensor product*, denoted $\otimes$. This operator intakes two tensors $\mathcal{A}$ and $\mathcal{B}$ of orders P and Q respectively, and returns a tensor $\mathcal{A} \otimes \mathcal{B}$ of order $P + Q$, defined by: $(\mathcal{A} \otimes \mathcal{B})_{d_1 \ldots d_{P+Q}} = \mathcal{A}_{d_1 \ldots d_P} \cdot \mathcal{B}_{d_{P+1} \ldots d_{P+Q}}$. Notice that in the case $P = Q = 1$, the tensor product reduces to an outer product between vectors. Specifically, $\mathbf{v} \otimes \mathbf{u}$ – the tensor product between $\mathbf{u} \in \mathbb{R}^{M_1}$ and $\mathbf{v} \in \mathbb{R}^{M_2}$, is no other than the rank-1 matrix $\mathbf{v}\mathbf{u}^\top \in \mathbb{R}^{M_1 \times M_2}$. In this context, we will often use the shorthand $\otimes_{i=1}^N \mathbf{v}^{(i)}$ to denote the joint tensor product $\mathbf{v}^{(1)} \otimes \cdots \otimes \mathbf{v}^{(N)}$.

Tensors of the form $\otimes_{i=1}^N \mathbf{v}^{(i)}$ are called *pure* or *elementary*, and are regarded as having *rank-1* (assuming $\mathbf{v}^{(i)} \neq 0 \ \forall i$). It is not difficult to see that any tensor can be expressed as a sum of rank-1

1. The definitions we give are concrete special cases of the more abstract algebraic definitions given in Hackbusch (2012). We limit the discussion to these special cases since they suffice for our needs and are easier to grasp.

tensors:

$$\mathcal{A} = \sum_{z=1}^{Z} \mathbf{v}_z^{(1)} \otimes \cdots \otimes \mathbf{v}_z^{(N)} \quad , \mathbf{v}_z^{(i)} \in \mathbb{R}^{M_i} \tag{1}$$

A representation as above is called a CANDECOMP/PARAFAC decomposition of $\mathcal{A}$, or in short, a *CP decomposition* [2]. The *CP-rank* of $\mathcal{A}$ is defined as the minimum number of terms in a CP decomposition, i.e. as the minimal Z for which eq. 1 can hold. Notice that for a tensor of order 2, i.e. a matrix, this definition of CP-rank coincides with that of standard matrix rank.

A *symmetric tensor* is one that is invariant to permutations of its indices. Formally, a tensor $\mathcal{A}$ of order N which is symmetric will have equal dimension M in all modes, and for every permutation $\pi : [N] \to [N]$ and indices $d_1 \ldots d_N \in [M]$, the following equality will hold: $\mathcal{A}_{d_{\pi(1)} \ldots d_{\pi(N)}} = \mathcal{A}_{d_1 \ldots d_N}$. Note that for a vector $\mathbf{v} \in \mathbb{R}^M$, the tensor $\otimes_{i=1}^{N} \mathbf{v} \in \mathbb{R}^{M \times \cdots \times M}$ is symmetric. Moreover, every symmetric tensor may be expressed as a linear combination of such (symmetric rank-1) tensors: $\mathcal{A} = \sum_{z=1}^{Z} \lambda_z \cdot \mathbf{v}_z \otimes \cdots \otimes \mathbf{v}_z$. This is referred to as a *symmetric CP decomposition*, and the *symmetric CP-rank* is the minimal Z for which such a decomposition exists. Since a symmetric CP decomposition is in particular a standard CP decomposition, the symmetric CP-rank of a symmetric tensor is always greater or equal to its standard CP-rank. Note that for the case of symmetric matrices (order-2 tensors) the symmetric CP-rank and the original CP-rank are always equal.

A repeating concept in this paper is that of *measure zero*. More broadly, our analysis is framed in measure theoretical terms. While an introduction to the field is beyond the scope of the paper (the interested reader is referred to Jones (2001)), it is possible to intuitively grasp the ideas that form the basis to our claims. When dealing with subsets of a Euclidean space, the standard and most natural measure in a sense is called the *Lebesgue measure*. This is the only measure we consider in our analysis. A set of (Lebesgue) measure zero can be thought of as having zero "volume" in the space of interest. For example, the interval between $(0, 0)$ and $(1, 0)$ has zero measure as a subset of the 2D plane, but has positive measure as a subset of the 1D x-axis. An alternative way to view a zero measure set S follows the property that if one draws a random point in space by some continuous distribution, the probability of that point hitting S is necessarily zero. A related term that will be used throughout the paper is *almost everywhere*, which refers to an entire space excluding, at most, a set of zero measure.

3. Convolutional Arithmetic Circuits

We consider the task of classifying an instance $X = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$, $\mathbf{x}_i \in \mathbb{R}^s$, into one of the categories $\mathcal{Y} := \{1, \ldots, Y\}$. Representing instances as collections of vectors is natural in many applications. In the case of image processing for example, X may correspond to an image, and $\mathbf{x}_1 \ldots \mathbf{x}_N$ may correspond to vector arrangements of (possibly overlapping) patches around pixels. As customary, classification is carried out through maximization of per-label score functions $\{h_y\}_{y \in \mathcal{Y}}$, i.e. the predicted label for the instance X will be the index $y \in \mathcal{Y}$ for which the score value $h_y(X)$ is maximal. Our attention is thus directed to functions over the instance space $\mathcal{X} := \{(\mathbf{x}_1, \ldots, \mathbf{x}_N) : \mathbf{x}_i \in \mathbb{R}^s\} = (\mathbb{R}^s)^N$. We define our hypotheses space through the following

2. CP decomposition is regarded as the classic and most basic tensor decomposition, dating back to the beginning of the 20'th century (see Kolda and Bader (2009) for a historic survey).

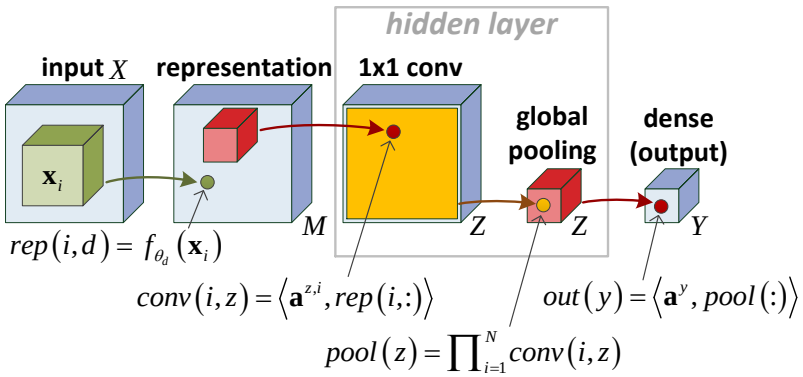

Figure 1: CP model – convolutional arithmetic circuit implementing CP (rank-1) decomposition.

representation of score functions:

$$h_y(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \sum_{d_1 \ldots d_N = 1}^{M} \mathcal{A}^y_{d_1,\ldots,d_N} \prod_{i=1}^{N} f_{\theta_{d_i}}(\mathbf{x}_i) \tag{2}$$

$f_{\theta_1} \ldots f_{\theta_M} : \mathbb{R}^s \to \mathbb{R}$ are referred to as *representation functions*, selected from a parametric family $\mathcal{F} = \{f_\theta : \mathbb{R}^s \to \mathbb{R}\}_{\theta \in \Theta}$. Natural choices for this family are wavelets, radial basis functions (*Gaussians*), and affine functions followed by point-wise activation (*neurons*). The *coefficient tensor* $\mathcal{A}^y$ has order N and dimension M in each mode. Its entries correspond to a basis of M^N point-wise product functions $\{(\mathbf{x}_1, \ldots, \mathbf{x}_N) \mapsto \prod_{i=1}^{N} f_{\theta_{d_i}}(\mathbf{x}_i)\}_{d_1 \ldots d_N \in [M]}$. We will often consider fixed linearly independent representation functions $f_{\theta_1} \ldots f_{\theta_M}$. In this case the point-wise product functions are linearly independent as well (see app. C.1), and we have a one to one correspondence between score functions and coefficient tensors. To keep the manuscript concise, we defer the derivation of our hypotheses space (eq. 2) to app. C, noting here that it arises naturally from the notion of tensor products between L^2 spaces.

Our eventual aim is to realize score functions h_y with a layered network architecture. As a first step along this path, we notice that $h_y(\mathbf{x}_1, \ldots, \mathbf{x}_N)$ is fully determined by the activations of the M representation functions $f_{\theta_1} \ldots f_{\theta_M}$ on the N input vectors $\mathbf{x}_1 \ldots \mathbf{x}_N$. In other words, given $\{f_{\theta_d}(\mathbf{x}_i)\}_{d \in [M], i \in [N]}$, the score $h_y(\mathbf{x}_1, \ldots, \mathbf{x}_N)$ is independent of the input. It is thus natural to consider the computation of these $M \cdot N$ numbers as the first layer of our networks. This layer, referred to as the *representation layer*, may be conceived as a convolutional operator with M channels, each corresponding to a different function applied to all input vectors (see fig. 1).

Once we have constrained our score functions to have the structure depicted in eq. 2, learning a classifier reduces to estimation of the parameters $\theta_1 \ldots \theta_M$, and the coefficient tensors $\mathcal{A}^1 \ldots \mathcal{A}^Y$. The computational challenge is that the latter tensors are of order N (and dimension M in each mode), having an exponential number of entries (M^N each). In the next subsections we utilize tensor decompositions (factorizations) to address this computational challenge, and show how they are naturally realized by convolutional arithmetic circuits.

3.1. Shallow Network as a CP Decomposition of $\mathcal{A}^y$

The most straightforward way to factorize a tensor is through a CP (rank-1) decomposition (see sec. 2). Consider a joint CP decomposition for the coefficient tensors $\{\mathcal{A}^y\}_{y\in\mathcal{Y}}$:

$$\mathcal{A}^y = \sum_{z=1}^{Z} a^y_z \cdot \mathbf{a}^{z,1} \otimes \cdots \otimes \mathbf{a}^{z,N} \tag{3}$$

where $\mathbf{a}^y \in \mathbb{R}^Z$ for $y \in \mathcal{Y}$ (a^y_z stands for entry z of $\mathbf{a}^y$), and $\mathbf{a}^{z,i} \in \mathbb{R}^M$ for $i \in [N], z \in [Z]$. The decomposition is joint in the sense that the same vectors $\mathbf{a}^{z,i}$ are shared across all classes y. Clearly, if we set $Z = M^N$ this model is universal, i.e. any tensors $\mathcal{A}^1 \ldots \mathcal{A}^Y$ may be represented.

Substituting our CP decomposition (eq. 3) into the expression for the score functions in eq. 2, we obtain:

$$h_y(X) = \sum_{z=1}^{Z} a^y_z \prod_{i=1}^{N} \left(\sum_{d=1}^{M} a^{z,i}_d f_{\theta_d}(\mathbf{x}_i) \right)$$

From this we conclude that the network illustrated in fig. 1 implements a classifier (score functions) under the CP decomposition in eq. 3. We refer to this network as *CP model*. The network consists of a representation layer followed by a single hidden layer, which in turn is followed by the output. The hidden layer begins with a 1×1 *conv* operator, which is simply a 3D convolution with Z channels and receptive field 1×1. The convolution may operate without coefficient sharing, i.e. the filters that generate feature maps by sliding across the previous layer may have different coefficients at different spatial locations. This is often referred to in the deep learning community as a locally-connected operator (see Taigman et al. (2014)). To obtain a standard convolutional operator, simply enforce coefficient sharing by constraining the vectors $\mathbf{a}^{z,i}$ in the CP decomposition (eq. 3) to be equal to each other for different values of i (this setting is discussed in sec. 3.3). Following conv operator, the hidden layer includes global product pooling. Feature maps generated by conv are reduced to singletons through multiplication of their entries, creating a vector of dimension Z. This vector is then mapped into the Y network outputs through a final dense linear layer.

To recap, CP model (fig. 1) is a shallow (single hidden layer) convolutional arithmetic circuit that realizes the CP decomposition (eq. 3). It is universal, i.e. it can realize any coefficient tensors with large enough size (Z). Unfortunately, since the CP-rank of a generic tensor is exponential in its order (see Hackbusch (2012)), the size required for CP model to be universal is exponential (Z exponential in N).

3.2. Deep Network as a Hierarchical Decomposition of $\mathcal{A}^y$

In this subsection we present a deep network that corresponds to the recently introduced Hierarchical Tucker tensor decomposition (Hackbusch and Kühn (2009)), which we refer to in short as *HT decomposition*. The network, dubbed *HT model*, is universal. Specifically, any set of tensors $\mathcal{A}^y$ represented by CP model can be represented by HT model with only a polynomial penalty in terms of resources. The advantage of HT model, as we show in sec. 4, is that in almost all cases it generates tensors that require an exponential size in order to be realized, or even approximated, by CP model. Put differently, if one draws the weights of HT model by some continuous distribution, with probability one, the resulting tensors cannot be approximated by a polynomial CP model. Informally, this implies that HT model is exponentially more expressive than CP model.

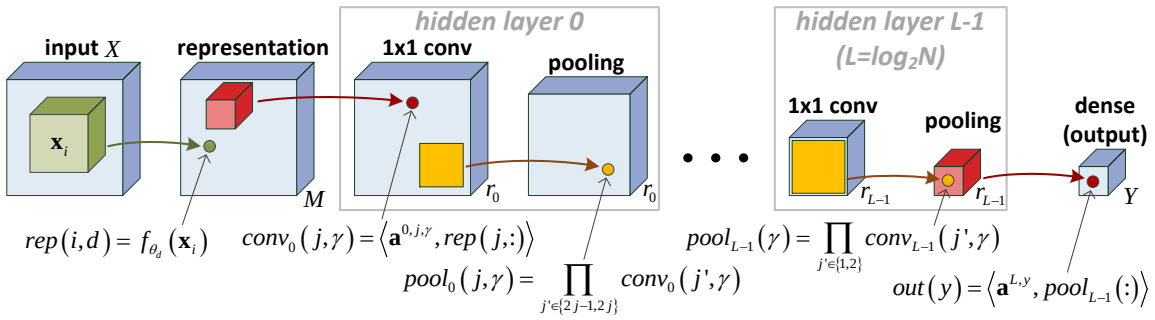

Figure 2: HT model – convolutional arithmetic circuit implementing hierarchical decomposition.

HT model is based on the hierarchical tensor decomposition in eq. 4, which is a special case of the HT decomposition as presented in Hackbusch and Kühn (2009) (in the latter's terminology, we restrict the matrices $A^{l,j,\gamma}$ to be diagonal). Our construction and theoretical results apply to the general HT decomposition as well, with the specialization done merely to bring forth a network that resembles current convolutional networks [3].

$$\begin{aligned}
\phi^{1,j,\gamma} &= \sum_{\alpha=1}^{r_0} a_\alpha^{1,j,\gamma} \mathbf{a}^{0,2j-1,\alpha} \otimes \mathbf{a}^{0,2j,\alpha} \\
&\cdots \\
\phi^{l,j,\gamma} &= \sum_{\alpha=1}^{r_{l-1}} a_\alpha^{l,j,\gamma} \underbrace{\phi^{l-1,2j-1,\alpha}}_{\text{order } 2^{l-1}} \otimes \underbrace{\phi^{l-1,2j,\alpha}}_{\text{order } 2^{l-1}} \\
&\cdots \\
\phi^{L-1,j,\gamma} &= \sum_{\alpha=1}^{r_{L-2}} a_\alpha^{L-1,j,\gamma} \underbrace{\phi^{L-2,2j-1,\alpha}}_{\text{order } \frac{N}{4}} \otimes \underbrace{\phi^{L-2,2j,\alpha}}_{\text{order } \frac{N}{4}} \\
\mathcal{A}^y &= \sum_{\alpha=1}^{r_{L-1}} a_\alpha^{L,y} \underbrace{\phi^{L-1,1,\alpha}}_{\text{order } \frac{N}{2}} \otimes \underbrace{\phi^{L-1,2,\alpha}}_{\text{order } \frac{N}{2}}
\end{aligned} \tag{4}$$

The decomposition in eq. 4 recursively constructs the coefficient tensors $\{\mathcal{A}^y\}_{y\in[Y]}$ by assembling vectors $\{\mathbf{a}^{0,j,\gamma}\}_{j\in[N],\gamma\in[r_0]}$ into tensors $\{\phi^{l,j,\gamma}\}_{l\in[L-1],j\in[N/2^l],\gamma\in[r_l]}$ in an incremental fashion. The index l stands for the level in the decomposition, j represents the "location" within level l, and γ corresponds to the individual tensor in level l and location j. r_l is referred to as *level-l rank*, and is defined to be the number of tensors in each location of level l (we denote for completeness $r_L := Y$). The tensor $\phi^{l,j,\gamma}$ has order 2^l, and we assume for simplicity that N – the order of $\mathcal{A}^y$, is a power of 2 (this is merely a technical assumption also made in Hackbusch and Kühn (2009), it does not limit the generality of our analysis).

The parameters of the decomposition are the final level weights $\{\mathbf{a}^{L,y} \in \mathbb{R}^{r_{L-1}}\}_{y\in[Y]}$, the intermediate levels' weights $\{\mathbf{a}^{l,j,\gamma} \in \mathbb{R}^{r_{l-1}}\}_{l\in[L-1],j\in[N/2^l],\gamma\in[r_l]}$, and the first level vectors $\{\mathbf{a}^{0,j,\gamma} \in \mathbb{R}^M\}_{j\in[N],\gamma\in[r_0]}$. This totals at $N \cdot M \cdot r_0 + \sum_{l=1}^{L-1} \frac{N}{2^l} \cdot r_{l-1} \cdot r_l + Y \cdot r_{l-1}$ individual parame-

3. If we had not constrained $A^{l,j,\gamma}$ to be diagonal, pooling operations would involve entries from different channels.

ters, and if we assume equal ranks $r := r_0 = \cdots = r_{L-1}$, the number of parameters becomes $N \cdot M \cdot r + N \cdot r^2 + Y \cdot r$.

The hierarchical decomposition (eq. 4) is universal, i.e. with large enough ranks r_l it can represent any tensors. Moreover, it is a super-set of the CP decomposition (eq. 3). That is to say, all tensors representable by a CP decomposition having Z components are also representable by a hierarchical decomposition with ranks $r_0 = r_1 = \cdots = r_{L-1} = Z$ [4]. Note that this comes with a polynomial penalty – the number of parameters increases from $N \cdot M \cdot Z + Z \cdot Y$ in the CP decomposition, to $N \cdot M \cdot Z + Z \cdot Y + N \cdot Z^2$ in the hierarchical decomposition. However, as we show in sec. 4, the gain in expressive power is exponential.

Plugging the expression for $\mathcal{A}^y$ in our hierarchical decomposition (eq. 4) into the score function h_y given in eq. 2, we obtain the network displayed in fig. 2 – HT model. This network includes a representation layer followed by $L = \log_2 N$ hidden layers which in turn are followed by the output. As in the shallow CP model (fig. 1), the hidden layers consist of 1×1 conv operators followed by product pooling. The difference is that instead of a single hidden layer collapsing the entire spatial structure through global pooling, hidden layers now pool over size-2 windows, decimating feature maps by a factor of two (no overlaps). After $L = \log_2 N$ such layers feature maps are reduced to singletons, and we arrive at a 1D structure with r_{L-1} nodes. This is then mapped into Y network outputs through a final dense linear layer. We note that the network's size-2 pooling windows (and the resulting number of hidden layers $L = \log_2 N$) correspond to the fact that our hierarchical decomposition (eq. 4) is based on a full binary tree over modes, i.e. it combines (through tensor product) two tensors at a time. We focus on this setting solely for simplicity of presentation, and since it is the one presented in Hackbusch and Kühn (2009). Our analysis (sec. 4) could easily be adapted to hierarchical decompositions based on other trees (taking tensor products between more than two tensors at a time), and that would correspond to networks with different pooling window sizes and resulting depths.

HT model (fig. 2) is conceptually divided into two parts. The first is the representation layer, transforming input vectors $\mathbf{x}_1 \ldots \mathbf{x}_N$ into $N \cdot M$ real-valued scalars $\{f_{\theta_d}(\mathbf{x}_i)\}_{i \in [N], d \in [M]}$. The second and main part of the network, which we view as an "inference" engine, is the convolutional arithmetic circuit that takes the $N \cdot M$ measurements produced by the representation layer, and accordingly computes Y class scores at the output layer.

To recap, we have now a deep network (fig. 2), which we refer to as HT model, that computes the score functions h_y (eq. 2) with coefficient tensors $\mathcal{A}^y$ hierarchically decomposed as in eq. 4. The network is universal in the sense that with enough channels r_l, any tensors may be represented. Moreover, the model is a super-set of the shallow CP model presented in sec. 3.1. The question of depth efficiency now naturally arises. In particular, we would like to know if there are functions that may be represented by a polynomially sized deep HT model, yet require exponential size from the shallow CP model. The answer, as described in sec. 4, is that almost all functions realizable by HT model meet this property. In other words, the set of functions realizable by a polynomial CP model has *measure zero* in the space of functions realizable by a given polynomial HT model.

4. To see this, simply assign the first level vectors $\mathbf{a}^{0,j,\gamma}$ with CP's basis vectors, the last level weights with CP's per-class weights, and the intermediate levels' weights with indicator vectors.

3.3. Shared Coefficients for Convolution

The 1×1 conv operator in our networks (see fig. 1 and 2) implements a local linear transformation with coefficients generally being location-dependent. In the special case where coefficients do not depend on location, i.e. remain fixed across space, the local linear transformation becomes a standard convolution. We refer to this setting as coefficient *sharing*. Sharing is a widely used structural constraint, one of the pillars behind the successful convolutional network architecture. In the context of image processing (prominent application of convolutional networks), sharing is motivated by the observation that in natural images, the semantic content of a pattern often does not depend on its location. In this subsection we explore the effect of sharing on the expressiveness of our networks, or more specifically, on the coefficient tensors $\mathcal{A}^y$ they can represent.

For CP model, coefficient sharing amounts to setting $\mathbf{a}^z := \mathbf{a}^{z,1} = \cdots = \mathbf{a}^{z,N}$ in the CP decomposition (eq. 3), transforming the latter to a symmetric CP decomposition:

$$\mathcal{A}^y = \sum_{z=1}^{Z} a_z^y \cdot \underbrace{\mathbf{a}^z \otimes \cdots \otimes \mathbf{a}^z}_{N \text{ times}} \quad , \mathbf{a}^z \in \mathbb{R}^M, \mathbf{a}^y \in \mathbb{R}^Z$$

CP model with sharing is not universal (not all tensors $\mathcal{A}^y$ are representable, no matter how large Z is allowed to be) – it can only represent symmetric tensors.

In the case of HT model, sharing amounts to applying the following constraints on the hierarchical decomposition in eq. 4: $\mathbf{a}^{l,\gamma} := \mathbf{a}^{l,1,\gamma} = \cdots = \mathbf{a}^{l,N/2^l,\gamma}$ for every $l = 0 \ldots L-1$ and $\gamma = 1 \ldots r_l$. Note that in this case universality is lost as well, but nonetheless generated tensors are not limited to be symmetric, already demonstrating an expressive advantage of deep models over shallow ones. In sec. 4 we take this further by showing that the shared HT model is exponentially more expressive than CP model, even if the latter is not constrained by sharing.

4. Theorems of Network Capacity

The first contribution of this paper, presented in sec. 3, is the equivalence between deep learning architectures successfully employed in practice, and tensor decompositions. Namely, we showed that convolutional arithmetic circuits as in fig. 2, which are in fact SimNets that have demonstrated promising empirical performance (see app. E), may be formulated as hierarchical tensor decompositions. As a second contribution, we make use of the established link between arithmetic circuits and tensor decompositions, combining theoretical tools from these two worlds, to prove results that are of interest to both deep learning and tensor analysis communities. This is the focus of the current section.

The fundamental theoretical result proven in this paper is the following:

Theorem 1 *Let $\mathcal{A}^y$ be a tensor of order N and dimension M in each mode, generated by the recursive formulas in eq. 4. Define $r := \min\{r_0, M\}$, and consider the space of all possible configurations for the parameters of the composition – $\{\mathbf{a}^{l,j,\gamma}\}_{l,j,\gamma}$. In this space, the generated tensor $\mathcal{A}^y$ will have CP-rank of at least $r^{N/2}$ almost everywhere (w.r.t. Lebesgue measure). Put differently, the configurations for which the CP-rank of $\mathcal{A}^y$ is less than $r^{N/2}$ form a set of measure zero. The exact same result holds if we constrain the composition to be "shared", i.e. set $\mathbf{a}^{l,j,\gamma} \equiv \mathbf{a}^{l,\gamma}$ and consider the space of $\{\mathbf{a}^{l,\gamma}\}_{l,\gamma}$ configurations.*

From the perspective of deep learning, thm. 1 leads to the following corollary:

Corollary 2 *Given linearly independent representation functions* $\{f_{\theta_d}\}_{d\in[M]}$*, randomizing the weights of HT model (sec. 3.2) by a continuous distribution induces score functions* h_y *that with probability one, cannot be approximated arbitrarily well (in* L^2 *sense) by a CP model (sec. 3.1) with less than* $\min\{r_0, M\}^{N/2}$ *hidden channels. This result holds even if we constrain HT model with weight sharing (sec. 3.3) while leaving CP model in its general form.*

That is to say, *besides a negligible set, all functions that can be realized by a polynomially sized HT model (with or without weight sharing), require exponential size in order to be realized, or even approximated, by CP model*. Most of the previous works relating to depth efficiency (see app. D) merely show *existence* of functions that separate depths (i.e. that are efficiently realizable by a deep network yet require super-polynomial size from shallow networks). Corollary 2 on the other hand establishes depth efficiency for *almost all* functions that a deep network can implement. Equally importantly, it applies to deep learning architectures that are being successfully employed in practice (SimNets – see app. E).

Adopting the viewpoint of tensor analysis, thm. 1 states that besides a negligible set, all tensors realized by HT (Hierarchical Tucker) decomposition cannot be represented by the classic CP (rank-1) decomposition if the latter has less than an exponential number of terms [5]. To the best of our knowledge, this result has never been proved in the tensor analysis community. In the original paper introducing HT decomposition (Hackbusch and Kühn (2009)), as a motivating example, the authors present a specific tensor that is efficiently realizable by HT decomposition while requiring an exponential number of terms from CP decomposition [6]. Our result strengthens this motivation considerably, showing that it is not just one specific tensor that favors HT over CP, but rather, almost all tensors realizable by HT exhibit this preference. Taking into account that any tensor realized by CP can also be realized by HT with only a polynomial penalty in the number of parameters (see sec. 3.2), this implies that in an asymptotic sense, HT decomposition is exponentially more efficient than CP decomposition.

4.1. Proof Sketches

The complete proofs of thm. 1 and corollary 2 are given in app. B. We provide here an outline of the main tools employed and arguments made along these proofs.

To prove thm. 1 we combine approaches from the worlds of circuit complexity and tensor decompositions. The first class of machinery we employ is *matrix algebra*, which has proven to be a powerful source of tools for analyzing the complexity of circuits. For example, arithmetic circuits have been analyzed through what is called the partial derivative matrix (see Raz and Yehudayoff (2009)), and for boolean circuits a widely used tool is the communication matrix (see Karchmer (1989)). We gain access to matrix algebra by arranging tensors that take part in the CP and HT decompositions as matrices, a process often referred to as *matricization*. With matricization, the tensor product translates to the Kronecker product, and the properties of the latter become readily available. The second tool-set we make use of is *measure theory*, which prevails in the study of tensor decompositions, but is much less frequent in analyses of circuit complexity. In order to frame

5. As stated in sec. 3.2, the decomposition in eq. 4 to which thm. 1 applies is actually a special case of HT decomposition as introduced in Hackbusch and Kühn (2009). However, the theorem and its proof can easily be adapted to account for the general case. We focus on the special case merely because it corresponds to convolutional arithmetic circuit architectures used in practice.
6. The same motivating example is given in a more recent textbook introducing tensor analysis (Hackbusch (2012)).

a problem in measure theoretical terms, one obviously needs to define a measure space of interest. For tensor decompositions, the straightforward space to focus on is that of the decomposition variables. For general circuits on the other hand, it is often unclear if defining a measure space is at all appropriate. However, when circuits are considered in the context of machine learning they are usually parameterized, and defining a measure space on top of these parameters is an effective approach for studying the prevalence of various properties in hypotheses spaces.

Our proof of thm. 1 traverses through the following path. We begin by showing that matricizing a rank-1 tensor produces a rank-1 matrix. This implies that the matricization of a tensor generated by a CP decomposition with Z terms has rank at most Z. We then turn to show that the matricization of a tensor generated by the HT decomposition in eq. 4 has rank at least $\min\{r_0, M\}^{N/2}$ almost everywhere. This is done through induction over the levels of the decomposition ($l = 1 \ldots L$). For the first level ($l = 1$), we use a combination of measure theoretical and linear algebraic arguments to show that the generated matrices have maximal rank ($\min\{r_0, M\}$) almost everywhere. For the induction step, the facts that under matricization tensor product translates into Kronecker product, and that the latter increases ranks multiplicatively [7], imply that matricization ranks in the current level are generally equal to those in the previous level squared. Measure theoretical claims are then made to ensure that this indeed takes place almost everywhere.

To prove corollary 2 based on thm. 1, we need to show that the inability of CP model to realize a tensor generated by HT model, implies that the former cannot approximate score functions produced by the latter. In general, the set of tensors expressible by a CP decomposition is not topologically closed [8], which implies that a-priori, it may be that CP model can approximate tensors generated by HT model even though it cannot realize them. However, since the proof of thm. 1 was achieved through separation of matrix rank, distances are indeed positive and CP model cannot approximate HT model's tensors almost always. To translate from tensors to score functions, we simply note that in a finite-dimensional Hilbert space convergence in norm implies convergence in coefficients under any basis. Therefore, in the space of score functions (eq. 2) convergence in norm implies convergence in coefficients under the basis $\{(\mathbf{x}_1, \ldots, \mathbf{x}_N) \mapsto \prod_{i=1}^{N} f_{\theta_{d_i}}(\mathbf{x}_i)\}_{d_1 \ldots d_N \in [M]}$. That is to say, it implies convergence in coefficient tensors.

4.2. Generalization

Thm. 1 and corollary 2 compare the expressive power of the deep HT model (sec. 3.2) to that of the shallow CP model (sec. 3.1). One may argue that such an analysis is lacking, as it does not convey information regarding the importance of each individual layer. In particular, it does not shed light on the advantage of very deep networks, which at present provide state of the art recognition accuracy, compared to networks of more moderate depth. For this purpose we present a generalization, specifying the amount of resources one has to pay in order to maintain representational power while layers are incrementally cut off from a deep network. For conciseness we defer this analysis to app. A, and merely state here our final conclusions. We find that the representational penalty is double exponential w.r.t. the number of layers removed. In addition, there are certain cases where the removal of even a single layer leads to an exponential inflation, falling in line with the suggestion of Bengio (2009).

7. If $\odot$ denotes the Kronecker product, then for any matrices A and B: $rank(A \odot B) = rank(A) \cdot rank(B)$.
8. Hence the definition of *border rank*, see Hackbusch (2012).

5. Discussion

In this work we address a fundamental issue in deep learning – the expressive efficiency of depth. There have been many attempts to theoretically analyze this question, but from a practical machine learning perspective, existing results are limited. Most of the results apply to very specific types of networks that do not resemble ones used in practice, and none of the results account for the locality-sharing-pooling paradigm which forms the basis for convolutional networks – the most successful deep learning architecture to date. In addition, current analyses merely show *existence* of depth efficiency, i.e. of functions that are efficiently realizable by deep networks but not by shallow ones. The practical implications of such findings are arguably slight, as a-priori, it may be that only a small fraction of the functions realizable by deep networks enjoy depth efficiency, and for all the rest shallow networks suffice.

Our aim in this paper was to develop a theory that facilitates an analysis of depth efficiency for networks that incorporate the widely used structural ingredients of locality, sharing and pooling. We consider the task of classification into one of a finite set of categories $\mathcal{Y} = \{1...Y\}$. Our instance space is defined to be the Cartesian product of N vector spaces, in compliance with the common practice of representing natural data through ordered local structures (e.g. images through patches). Each of the N vectors that compose an instance is represented by a descriptor of length M, generated by running the vector through M "representation" functions. As customary, classification is achieved through maximization of score functions h_y, one for every category $y \in \mathcal{Y}$. Each score function is a linear combination over the M^N possible products that may be formed by taking one descriptor entry from every input vector. The coefficients for these linear combinations conveniently reside in tensors $\mathcal{A}^y$ of order N and dimension M along each axis. We construct networks that compute score functions h_y by decomposing (factorizing) the coefficient tensors $\mathcal{A}^y$. The resulting networks are convolutional arithmetic circuits that incorporate locality, sharing and pooling, and operate on the $N \cdot M$ descriptor entries generated from the input.

We show that a shallow (single hidden layer) network realizes the classic CP (rank-1) tensor decomposition, whereas a deep network with $\log_2 N$ hidden layers realizes the recently introduced Hierarchical Tucker (HT) decomposition (Hackbusch and Kühn (2009)). Our fundamental result, presented in thm. 1 and corollary 2, states that randomizing the weights of a deep network by some continuous distribution will lead, *with probability one*, to score functions that cannot be approximated by a shallow network if the latter's size is not exponential (in N). We extend this result (thm. 3 and corollary 4) by deriving analogous claims that compare two networks of any depths, not just deep vs. shallow.

To further highlight the connection between our networks and ones used in practice, we show (app. E) that translating convolution and product pooling computations to log-space (for numerical stability) gives rise to SimNets – a recently proposed deep learning architecture which has been shown to produce state of the art accuracy in computationally limited settings (Cohen et al. (2016)).

Besides the central line of our work discussed above, the construction and theory presented in this paper shed light on various conjectures and practices employed by the deep learning community. First, with respect to the *pooling* operation, our analysis points to the possibility that perhaps it has more to do with factorization of computed functions than it does with translation invariance. This may serve as an explanation for the fact that pooling windows in state of the art convolutional networks are typically very small (see for example Simonyan and Zisserman (2014)), often much smaller than the radius of translation one would like to be invariant to. Indeed, in our framework, as

we show in app. A, pooling over large windows and trimming down a network's depth may bring to an exponential decrease in expressive efficiency.

The second point our theory sheds light on is *sharing*. As discussed in sec. 3.3, introducing weight sharing to a shallow network (CP model) considerably limits its expressive power. The network can only represent symmetric tensors, which in turn means that it is location invariant w.r.t. input vectors (patches). In the case of a deep network (HT model) the limitation posed by sharing is not as strict. Generated tensors need not be symmetric, implying that the network is capable of modeling location – a crucial ability in almost any real-world task. The above findings suggest that the sharing constraint is increasingly limiting as a network gets shallower, to the point where it causes complete ignorance to location. This could serve as an argument supporting the empirical success of deep convolutional networks – they bind together the statistical and computational advantages of sharing with many layers that mitigate its expressive limitations.

Lastly, our construction advocates *locality*, or more specifically, 1×1 receptive fields. Recent convolutional networks providing state of the art recognition performance (e.g. Lin et al. (2014); Szegedy et al. (2015)) make extensive use of 1×1 linear transformations, proving them to be very successful in practice. In view of our model, such 1×1 operators factorize tensors while providing universality with a minimal number of parameters. It seems reasonable to conjecture that for this task of factorizing coefficient tensors, larger receptive fields are not significantly helpful, as they lead to redundancy which may deteriorate performance in presence of limited training data. Investigation of this conjecture is left for future work.

Acknowledgments

Amnon Shashua would like to thank Tomaso Poggio and Shai S. Shwartz for illuminating discussions during the preparation of this manuscript. We would also like to thank Tomer Galanti, Tamir Hazan and Lior Wolf for commenting on draft versions of the paper. The work is partly funded by Intel grant ICRI-CI no. 9-2012-6133 and by ISF Center grant 1790/12. Nadav Cohen is supported by a Google Fellowship in Machine Learning.

References

Animashree Anandkumar, Rong Ge, Daniel Hsu, Sham M Kakade, and Matus Telgarsky. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15(1):2773–2832, 2014.

Richard Bellman, Richard Ernest Bellman, Richard Ernest Bellman, and Richard Ernest Bellman. *Introduction to matrix analysis*, volume 960. SIAM, 1970.

Yoshua Bengio. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning*, 2(1): 1–127, 2009.

Monica Bianchini and Franco Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *Neural Networks and Learning Systems, IEEE Transactions on*, 25(8):1553–1565, 2014.

Joan Bruna and Stéphane Mallat. Invariant Scattering Convolution Networks. *IEEE TPAMI*, 2012.

Richard Caron and Tim Traynor. The zero set of a polynomial. *WSMR Report 05-02*, 2005.

Nadav Cohen and Amnon Shashua. Simnets: A generalization of convolutional networks. *Advances in Neural Information Processing Systems (NIPS), Deep Learning Workshop*, 2014.

Nadav Cohen and Amnon Shashua. Convolutional rectifier networks as generalized tensor decompositions. *International Conference on Machine Learning (ICML)*, 2016.

Nadav Cohen, Or Sharir, and Amnon Shashua. Deep simnets. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

G Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

Olivier Delalleau and Yoshua Bengio. Shallow vs. deep sum-product networks. In *Advances in Neural Information Processing Systems*, pages 666–674, 2011.

Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. *arXiv preprint arXiv:1512.03965*, 2015.

F Girosi and T Poggio. Networks and the best approximation property. *Biological cybernetics*, 63(3):169–176, 1990.

W Hackbusch and S Kühn. A New Scheme for the Tensor Representation. *Journal of Fourier Analysis and Applications*, 15(5):706–722, 2009.

Wolfgang Hackbusch. *Tensor Spaces and Numerical Tensor Calculus*, volume 42 of *Springer Series in Computational Mathematics*. Springer Science & Business Media, Berlin, Heidelberg, February 2012.

Benjamin D Haeffele and René Vidal. Global Optimality in Tensor Factorization, Deep Learning, and Beyond. *CoRR abs/1202.2745*, cs.NA, 2015.

András Hajnal, Wolfgang Maass, Pavel Pudlák, Márló Szegedy, and György Turán. Threshold circuits of bounded depth. In *Foundations of Computer Science, 1987., 28th Annual Symposium on*, pages 99–110. IEEE, 1987.

Johan Hastad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 6–20. ACM, 1986.

Johan Håstad and Mikael Goldmann. On the power of small-depth threshold circuits. *Computational Complexity*, 1(2):113–129, 1991.

Kurt Hornik, Maxwell B Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Brian Hutchinson, Li Deng, and Dong Yu. Tensor Deep Stacking Networks. *IEEE Trans. Pattern Anal. Mach. Intell. ()*, 35(8):1944–1957, 2013.

Majid Janzamin, Hanie Sedghi, and Anima Anandkumar. Beating the Perils of Non-Convexity: Guaranteed Training of Neural Networks using Tensor Methods. *CoRR abs/1506.08473*, 2015.

Frank Jones. *Lebesgue integration on Euclidean space*. Jones & Bartlett Learning, 2001.

Mauricio Karchmer. Communication complexity a new approach to circuit depth. 1989.

Tamara G Kolda and Brett W Bader. Tensor Decompositions and Applications. *SIAM Review ()*, 51(3): 455–500, 2009.

Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan V Oseledets, and Victor S Lempitsky. Speeding-up Convolutional Neural Networks Using Fine-tuned CP-Decomposition. *CoRR abs/1202.2745*, cs.CV, 2014.

Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10), 1995.

Min Lin, Qiang Chen, and Shuicheng Yan. Network In Network. *International Conference on Learning Representations*, 2014.

Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir. On the computational efficiency of training neural networks. *Advances in Neural Information Processing Systems*, 2014.

Wolfgang Maass, Georg Schnitger, and Eduardo D Sontag. *A comparison of the computational power of sigmoid and Boolean threshold circuits*. Springer, 1994.

James Martens and Venkatesh Medabalimi. On the expressive efficiency of sum product networks. *arXiv preprint arXiv:1411.7717*, 2014.

James Martens, Arkadev Chattopadhya, Toni Pitassi, and Richard Zemel. On the representational efficiency of restricted boltzmann machines. In *Advances in Neural Information Processing Systems*, pages 2877–2885, 2013.

Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2924–2932, 2014.

Alexander Novikov, Anton Rodomanov, Anton Osokin, and Dmitry Vetrov. Putting MRFs on a Tensor Train. *ICML*, pages 811–819, 2014.

Razvan Pascanu, Guido Montufar, and Yoshua Bengio. On the number of inference regions of deep feed forward networks with piece-wise linear activations. *arXiv preprint arXiv*, 1312, 2013.

Allan Pinkus. Approximation theory of the MLP model in neural networks. *Acta Numerica*, 8:143–195, January 1999.

Hoifung Poon and Pedro Domingos. Sum-product networks: A new deep architecture. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 689–690. IEEE, 2011.

Ran Raz and Amir Yehudayoff. Lower bounds and separations for constant depth multilinear circuits. *Computational Complexity*, 18(2):171–207, 2009.

Benjamin Rossman, Rocco A Servedio, and Li-Yang Tan. An average-case depth hierarchy theorem for boolean circuits. *arXiv preprint arXiv:1504.03398*, 2015.

Walter Rudin. Functional analysis. international series in pure and applied mathematics, 1991.

Thomas Serre, Lior Wolf, and Tomaso Poggio. Object Recognition with Features Inspired by Visual Cortex. *CVPR*, 2:994–1000, 2005.

Hendra Setiawan, Zhongqiang Huang, Jacob Devlin, Thomas Lamar, Rabih Zbib, Richard M Schwartz, and John Makhoul. Statistical Machine Translation Features with Multitask Tensor Networks. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, cs.CL, 2015.

Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3–4):207–388, 2010.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Michael Sipser. *Borel sets and circuit complexity*. ACM, New York, New York, USA, December 1983.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Y Ng. Reasoning With Neural Tensor Networks for Knowledge Base Completion. *Advances in Neural Information Processing Systems*, pages 926–934, 2013.

Le Song, Mariya Ishteva, Ankur P Parikh, Eric P Xing, and Haesun Park. Hierarchical Tensor Decomposition of Latent Tree Graphical Models. *ICML*, pages 334–342, 2013.

Maxwell Stinchcombe and Halbert White. Universal approximation using feedforward networks with non-sigmoid hidden layer activation functions. *International Joint Conference on Neural Networks*, pages 613–617 vol.1, 1989.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. *CVPR*, 2015.

Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *CVPR '14: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, June 2014.

Matus Telgarsky. Representation benefits of deep feedforward networks. *arXiv preprint arXiv:1509.08101*, 2015.

Y Yang and D B Dunson. Bayesian conditional tensor factorizations for high-dimensional classification. *Journal of the American Statistical*, 2015.

Dong Yu, Li Deng, and Frank Seide. Large Vocabulary Speech Recognition Using Deep Tensor Neural Networks. *INTERSPEECH*, pages 6–9, 2012.

Daniel Zoran and Yair Weiss. "Natural Images, Gaussian Mixtures and Dead Leaves". *Advances in Neural Information Processing Systems*, pages 1745–1753, 2012.

Appendix A. Generalized Theorem of Network Capacity

In sec. 4 we presented our fundamental theorem of network capacity (thm. 1 and corollary 2), showing that besides a negligible set, all functions that can be realized by a polynomially sized HT model (with or without weight sharing), require exponential size in order to be realized, or even approximated, by CP model. In terms of network depth, CP and HT models represent the extremes – the former has only a single hidden layer achieved through global pooling, whereas the latter has $L = \log_2 N$ hidden layers achieved through minimal (size-2) pooling windows. It is of interest to generalize the fundamental result by establishing a comparison between networks of intermediate depths. This is the focus of the current appendix.

We begin by defining a truncated version of the hierarchical tensor decomposition presented in eq. 4:

$$\begin{aligned}
\phi^{1,j,\gamma} &= \sum_{\alpha=1}^{r_0} a_\alpha^{1,j,\gamma} \mathbf{a}^{0,2j-1,\alpha} \otimes \mathbf{a}^{0,2j,\alpha} \\
&\vdots \\
\phi^{l,j,\gamma} &= \sum_{\alpha=1}^{r_{l-1}} a_\alpha^{l,j,\gamma} \underbrace{\phi^{l-1,2j-1,\alpha}}_{\text{order } 2^{l-1}} \otimes \underbrace{\phi^{l-1,2j,\alpha}}_{\text{order } 2^{l-1}} \\
&\vdots \\
\mathcal{A} &= \sum_{\alpha=1}^{r_{L_c-1}} a_\alpha^{L_c} \bigotimes_{j=1}^{2^{L-L_c+1}} \underbrace{\phi^{L_c-1,j,\alpha}}_{\text{order } 2^{L_c-1}}
\end{aligned} \tag{5}$$

The only difference between this decomposition and the original is that instead of completing the full process with $L := \log_2 N$ levels, we stop after $L_c \leq L$. At this point remaining tensors are binded together to form the final order-N tensor. The corresponding network will simply include a premature global pooling stage that shrinks feature maps to 1×1, and then a final linear layer that performs classification. As before, we consider a shared version of the decomposition in which $\mathbf{a}^{l,j,\gamma} \equiv \mathbf{a}^{l,\gamma}$. Notice that this construction realizes a continuum between CP and HT models, which correspond to the extreme cases $L_c = 1$ and $L_c = L$ respectively.

The following theorem, a generalization of thm. 1, compares a truncated decomposition having L_1 levels, to one with $L_2 < L_1$ levels that implements the same tensor, quantifying the penalty in terms of parameters:

Theorem 3 *Let $\mathcal{A}^{(1)}$ and $\mathcal{A}^{(2)}$ be tensors of order N and dimension M in each mode, generated by the truncated recursive formulas in eq. 5, with L_1 and L_2 levels respectively. Denote by $\{r_l^{(1)}\}_{l=0}^{L_1-1}$ and $\{r_l^{(2)}\}_{l=0}^{L_2-1}$ the composition ranks of $\mathcal{A}^{(1)}$ and $\mathcal{A}^{(2)}$ respectively. Assuming w.l.o.g. that $L_1 > L_2$, we define $r := \min\{r_0^{(1)}, ..., r_{L_2-1}^{(1)}, M\}$, and consider the space of all possible configurations for the parameters of $\mathcal{A}^{(1)}$'s composition – $\{\mathbf{a}^{(1),l,j,\gamma}\}_{l,j,\gamma}$. In this space, almost everywhere (w.r.t. Lebesgue measure), the generated tensor $\mathcal{A}^{(1)}$ requires that $r_{L_2-1}^{(2)} \geq (r)^{2^{L-L_2}}$ if one wishes that $\mathcal{A}^{(2)}$ be equal to $\mathcal{A}^{(1)}$. Put differently, the configurations for which $\mathcal{A}^{(1)}$ can be realized by $\mathcal{A}^{(2)}$ with $r_{L_2-1}^{(2)} < (r)^{2^{L-L_2}}$ form a set of measure zero. The exact same result holds if we constrain the composition of $\mathcal{A}^{(1)}$ to be "shared", i.e. set $\mathbf{a}^{(1),l,j,\gamma} \equiv \mathbf{a}^{(1),l,\gamma}$ and consider the space of $\{\mathbf{a}^{(1),l,\gamma}\}_{l,\gamma}$ configurations.*

In analogy with corollary 2, we obtain the following generalization:

Corollary 4 *Suppose we are given linearly independent representation functions $f_{\theta_1} \dots f_{\theta_M}$, and consider two networks that correspond to the truncated hierarchical tensor decomposition in eq. 5, with L_1 and L_2 hidden layers respectively. Assume w.l.o.g. that $L_1 > L_2$, i.e. that network 1 is deeper than network 2, and define r to be the minimal number of channels across the representation layer and the first L_2 hidden layers of network 1. Then, if we randomize the weights of network 1 by a continuous distribution, we obtain, with*

probability one, score functions h_y that cannot be approximated arbitrarily well (in L^2 sense) by network 2 if the latter has less than $(r)^{2^{L-L_2}}$ channels in its last hidden layer. The result holds even if we constrain network 1 with weight sharing while leaving network 2 in its general form.

Proofs of thm. 3 and corollary 4 are given in app. B. Hereafter, we briefly discuss some of their implications. First, notice that we indeed obtain a generalization of the fundamental theorem of network capacity (thm. 1 and corollary 2), which corresponds to the extreme case $L_1 = L$ and $L_2 = 1$. Second, note that for the baseline case of $L_1 = L$, i.e. a full-depth network has generated the target score function, approximating this with a truncated network draws a price that grows *double exponentially* w.r.t. the number of missing layers. Third, and most intriguingly, we see that when L_1 is considerably smaller than L, i.e. when a significantly truncated network is sufficient to model our problem, cutting off even a single layer leads to an exponential price, and this price is *independent* of L_1. Such scenarios of exponential penalty for trimming down a single layer were discussed in Bengio (2009), but only in the context of specific functions realized by networks that do not resemble ones used in practice (see Håstad and Goldmann (1991) for an example of such result). We prove this in a much broader, more practical setting, showing that for convolutional arithmetic circuit (SimNet – see app. E) architectures, almost any function realized by a significantly truncated network will exhibit this behavior. The issue relates to empirical practice, supporting the common methodology of designing networks that go as deep as possible. Specifically, it encourages extending network depth by pooling over small regions, avoiding significant spatial decimation that brings network termination closer.

We conclude this appendix by stressing once more that our construction and theoretical approach are not limited to the models covered by our theorems (CP model, HT model, truncated HT model). These are merely exemplars deemed most appropriate for initial analysis. The fundamental and generalized theorems of network capacity are similar in spirit, and analogous theorems for networks with different pooling window sizes and depths (corresponding to different tensor decompositions) may easily be derived.

Appendix B. Proofs

B.1. Proof of Theorems 1 and 3

Our proof of thm. 1 and 3 relies on basic knowledge in measure theory, or more specifically, Lebesgue measure spaces. We do not provide here a comprehensive background on this field (the interested reader is referred to Jones (2001)), but rather supplement the brief discussion given in sec. 2, with a list of facts we will be using which are not necessarily intuitive:

- A union of countably (or finitely) many sets of zero measure is itself a set of zero measure.
- If p is a polynomial over d variables that is not identically zero, the set of points in $\mathbb{R}^d$ in which it vanishes has zero measure (see Caron and Traynor (2005) for a short proof of this).
- If $S \subset \mathbb{R}^{d_1}$ has zero measure, then $S \times \mathbb{R}^{d_2} \subset \mathbb{R}^{d_1+d_2}$, and every set contained within, have zero measure as well.

In the above, and in the entirety of this paper, the only measure spaces we consider are Euclidean spaces equipped with Lebesgue measure. Thus when we say that a set of d-dimensional points has zero measure, we mean that its Lebesgue measure in the d-dimensional Euclidean space is zero.

Moving on to some preliminaries from matrix and tensor theory, we denote by $[\mathcal{A}]$ the *matricization* of an order-N tensor $\mathcal{A}$ (for simplicity, N is assumed to be even), where rows correspond to odd modes and columns correspond to even modes. Namely, if $\mathcal{A} \in \mathbb{R}^{M_1 \times \cdots \times M_N}$, the matrix $[\mathcal{A}]$ has $M_1 \cdot M_3 \cdot \ldots \cdot M_{N-1}$ rows and $M_2 \cdot M_4 \cdot \ldots \cdot M_N$ columns, rearranging the entries of the tensor such that $\mathcal{A}_{d_1 \ldots d_N}$ is stored in row index $1 + \sum_{i=1}^{N/2} (d_{2i-1} - 1) \prod_{j=i+1}^{N/2} M_{2j-1}$ and column index $1 + \sum_{i=1}^{N/2} (d_{2i} - 1) \prod_{j=i+1}^{N/2} M_{2j}$. To distinguish from the tensor product operation $\otimes$, we denote the Kronecker product between matrices by $\odot$. Specifically, for two matrices $A \in \mathbb{R}^{M_1 \times M_2}$ and $B \in \mathbb{R}^{N_1 \times N_2}$, $A \odot B$ is the matrix in $\mathbb{R}^{M_1 N_1 \times M_2 N_2}$ that holds $A_{ij} B_{kl}$ in row index $(i-1)N_1 + k$ and column index $(j-1)N_2 + l$. The basic relation that binds together tensor

product, matricization and Kronecker product is $[\mathcal{A} \otimes \mathcal{B}] = [\mathcal{A}] \odot [\mathcal{B}]$, where $\mathcal{A}$ and $\mathcal{B}$ are tensors of even orders. Two additional facts we will make use of are that the matricization is a linear operator (i.e. for scalars $\alpha_1 \ldots \alpha_r$ and tensors with the same size $\mathcal{A}_1 \ldots \mathcal{A}_r$: $[\sum_{i=1}^{r} \alpha_i \mathcal{A}_i] = \sum_{i=1}^{r} \alpha_i [\mathcal{A}_i]$), and less trivially, that for any matrices A and B, the rank of $A \odot B$ is equal to $rank(A) \cdot rank(B)$ (see Bellman et al. (1970) for a proof). These two facts, along with the basic relation laid out above, lead to the conclusion that:

$$rank\left[\mathbf{v}_1^{(z)} \otimes \cdots \otimes \mathbf{v}_{2^L}^{(z)}\right] = \prod_{i=1}^{2^L/2} rank \overbrace{\left[\mathbf{v}_{2i-1}^{(z)} \otimes \mathbf{v}_{2i}^{(z)}\right]}^{\mathbf{v}_{2i-1}^{(z)} \mathbf{v}_{2i}^{(z)\top}} = 1$$

and thus:

$$rank\left[\sum_{z=1}^{Z} \lambda_z \mathbf{v}_1^{(z)} \otimes \cdots \otimes \mathbf{v}_{2^L}^{(z)}\right] = rank \sum_{z=1}^{Z} \lambda_z \left[\mathbf{v}_1^{(z)} \otimes \cdots \otimes \mathbf{v}_{2^L}^{(z)}\right] \leq \sum_{z=1}^{Z} rank\left[\mathbf{v}_1^{(z)} \otimes \cdots \otimes \mathbf{v}_{2^L}^{(z)}\right] = Z$$

In words, an order-2^L tensor given by a CP-decomposition (see sec. 2) with Z terms, has matricization with rank at most Z. Thus, *to prove that a certain order-2^L tensor has CP-rank of at least R, it suffices to show that its matricization has rank of at least R.*

We now state and prove two lemmas that will be needed for our proofs of thm. 1 and 3.

Lemma 5 *Let $M, N \in \mathbb{N}$, and define the following mapping taking $\mathbf{x} \in \mathbb{R}^{2MN+N}$ to three matrices: $A(\mathbf{x}) \in \mathbb{R}^{M \times N}$, $B(\mathbf{x}) \in \mathbb{R}^{M \times N}$ and $D(\mathbf{x}) \in \mathbb{R}^{N \times N}$. $A(\mathbf{x})$ simply holds the first MN elements of $\mathbf{x}$, $B(\mathbf{x})$ holds the following MN elements of $\mathbf{x}$, and $D(\mathbf{x})$ is a diagonal matrix that holds the last N elements of $\mathbf{x}$ on its diagonal. Define the product matrix $U(\mathbf{x}) := A(\mathbf{x})D(\mathbf{x})B(\mathbf{x})^\top \in \mathbb{R}^{M \times M}$, and consider the set of points $\mathbf{x}$ for which the rank of $U(\mathbf{x})$ is different from $r := \min\{M, N\}$. This set of points has zero measure. The result will also hold if the points $\mathbf{x}$ reside in $\mathbb{R}^{MN+N}$, and the same elements are used to assign $A(\mathbf{x})$ and $B(\mathbf{x})$ ($A(\mathbf{x}) \equiv B(\mathbf{x})$).*

Proof Obviously $rank(U(\mathbf{x})) \leq r$ for all $\mathbf{x}$, so it remains to show that $rank(U(\mathbf{x})) \geq r$ for all $\mathbf{x}$ but a set of zero measure. Let $U_r(\mathbf{x})$ be the top-left $r \times r$ sub-matrix of $U(\mathbf{x})$. If $U_r(\mathbf{x})$ is non-singular then of course $rank(U(\mathbf{x})) \geq r$ as required. It thus suffices to show that the set of points $\mathbf{x}$ for which $\det U_r(\mathbf{x}) = 0$ has zero measure. Now, $\det U_r(\mathbf{x})$ is a polynomial in the entries of $\mathbf{x}$, and so it either vanishes on a set of zero measure, or it is the zero polynomial (see Caron and Traynor (2005)). All that is left is to disqualify the latter option, and that can be done by finding a specific point $\mathbf{x}_0$ for which $\det U_r(\mathbf{x}_0) \neq 0$. Indeed, we may choose $\mathbf{x}_0$ such that $D(\mathbf{x}_0)$ is the identity matrix and $A(\mathbf{x}_0), B(\mathbf{x}_0)$ hold 1 on their main diagonal and 0 otherwise. This selection implies that $U_r(\mathbf{x}_0)$ is the identity matrix, and in particular $\det U_r(\mathbf{x}_0) \neq 0$. ■

Lemma 6 *Assume we have p continuous mappings from $\mathbb{R}^d$ to $\mathbb{R}^{M \times N}$ taking the point $\mathbf{y}$ to the matrices $A_1(\mathbf{y}) \ldots A_p(\mathbf{y})$. Assume that under these mappings, the points $\mathbf{y}$ for which every $i \in [p]$ satisfies $rank(A_i(\mathbf{y})) < r$ form a set of zero measure. Define a mapping from $\mathbb{R}^p \times \mathbb{R}^d$ to $R^{M \times N}$ given by $(\mathbf{x}, \mathbf{y}) \mapsto A(\mathbf{x}, \mathbf{y}) := \sum_{i=1}^{p} x_i \cdot A_i(\mathbf{y})$. Then, the points $(\mathbf{x}, \mathbf{y})$ for which $rank(A(\mathbf{x}, \mathbf{y})) < r$ form a set of zero measure.*

Proof Denote $S := \{(\mathbf{x}, \mathbf{y}) : rank(A(\mathbf{x}, \mathbf{y})) < r\} \subset \mathbb{R}^p \times \mathbb{R}^d$. We would like to show that this set has zero measure. We first note that since $A(\mathbf{x}, \mathbf{y})$ is a continuous mapping, and the set of matrices $A \in \mathbb{R}^{M \times N}$ which have rank less than r is closed, S is a closed set and in particular measurable. Our strategy for computing its measure will be as follows. For every $\mathbf{y} \in \mathbb{R}^d$ we define the marginal set $S^{\mathbf{y}} := \{\mathbf{x} : rank(A(\mathbf{x}, \mathbf{y})) < r\} \subset \mathbb{R}^p$. We will show that for every $\mathbf{y}$ but a set of zero measure, the measure of $S^{\mathbf{y}}$ is zero. An application of Fubini's theorem will then prove the desired result.

Let C be the set of points $\mathbf{y} \in \mathbb{R}^d$ for which $\forall i \in [p] : rank(A_i(\mathbf{y})) < r$. By assumption, C has zero measure. We now show that for $\mathbf{y}_0 \in \mathbb{R}^d \setminus C$, the measure of $S^{\mathbf{y}_0}$ is zero. By the definition of C there exists an $i \in [p]$ such that $rank(A_i(\mathbf{y}_0)) \geq r$. W.l.o.g., we assume that $i = 1$, and that the top-left $r \times r$ sub-matrix of $A_1(\mathbf{y}_0)$ is non-singular. Regarding $\mathbf{y}_0$ as fixed, the determinant of the top-left $r \times r$ sub-matrix of $A(\mathbf{x}, \mathbf{y}_0)$ is a polynomial in the elements of $\mathbf{x}$. It is not the zero polynomial, as setting $x_1 = 1, x_2 = \cdots = x_p = 0$ yields $A(\mathbf{x}, \mathbf{y}_0) = A_1(\mathbf{y}_0)$, and the determinant of the latter's top-left $r \times r$ sub-matrix is non-zero. As a non-zero polynomial, the determinant of the top-left $r \times r$ sub-matrix of $A(\mathbf{x}, \mathbf{y}_0)$ vanishes only on a set of zero measure (Caron and Traynor (2005)). This implies that indeed the measure of $S^{\mathbf{y}_0}$ is zero.

We introduce a few notations towards our application of Fubini's theorem. First, the symbol $\mathbf{1}$ will be used to represent indicator functions, e.g. $\mathbf{1}_S$ is the function from $\mathbb{R}^p \times \mathbb{R}^d$ to $\mathbb{R}$ that receives 1 on S and 0 elsewhere. Second, we use a subscript of $n \in \mathbb{N}$ to indicate that the corresponding set is intersected with the hyper-rectangle of radius n. For example, S_n stands for the intersection between S and $[-n, n]^{p+d}$, and $\mathbb{R}^d_n$ stands for the intersection between $\mathbb{R}^d$ and $[-n, n]^d$ (which is equal to the latter). All the sets we consider are measurable, and those with subscript n have finite measure. We may thus apply Fubini's theorem to get:

$$\int_{(\mathbf{x},\mathbf{y})} \mathbf{1}_{S_n} = \int_{(\mathbf{x},\mathbf{y}) \in \mathbb{R}^{p+d}_n} \mathbf{1}_S = \int_{\mathbf{y} \in \mathbb{R}^d_n} \int_{\mathbf{x} \in \mathbb{R}^p_n} \mathbf{1}_{S^y} = \int_{\mathbf{y} \in \mathbb{R}^d_n \cap C} \int_{\mathbf{x} \in \mathbb{R}^p_n} \mathbf{1}_{S^y} + \int_{\mathbf{y} \in \mathbb{R}^d_n \setminus C} \int_{\mathbf{x} \in \mathbb{R}^p_n} \mathbf{1}_{S^y}$$

Recall that the set $C \in \mathbb{R}^d$ has zero measure, and for every $\mathbf{y} \notin C$ the measure of $S^y \in \mathbb{R}^p$ is zero. This implies that both integrals in the last expression vanish, and thus $\int \mathbf{1}_{S_n} = 0$. Finally, we use the monotone convergence theorem to compute $\int \mathbf{1}_S$:

$$\int \mathbf{1}_S = \int \lim_{n \to \infty} \mathbf{1}_{S_n} = \lim_{n \to \infty} \int \mathbf{1}_{S_n} = \lim_{n \to \infty} 0 = 0$$

This shows that indeed our set of interest S has zero measure. ■

With all preliminaries and lemmas in place, we turn to prove thm. 1, establishing an exponential efficiency of HT decomposition (eq. 4) over CP decomposition (eq. 3).

Proof [of theorem 1] We begin with the case of an "unshared" composition, i.e. the one given in eq. 4 (as opposed to the "shared" setting of $\mathbf{a}^{l,j,\gamma} \equiv \mathbf{a}^{l,\gamma}$). Denoting for convenience $\phi^{L,1,1} := \mathcal{A}^y$ and $r_L = 1$, we will show by induction over $l = 1, ..., L$ that almost everywhere (at all points but a set of zero measure) w.r.t. $\{\mathbf{a}^{l,j,\gamma}\}_{l,j,\gamma}$, all CP-ranks of the tensors $\{\phi^{l,j,\gamma}\}_{j \in [N/2^l], \gamma \in [r_l]}$ are at least $r^{2^l/2}$. In accordance with our discussion in the beginning of this subsection, it suffices to consider the matricizations $[\phi^{l,j,\gamma}]$, and show that these all have ranks greater or equal to $r^{2^l/2}$ almost everywhere.

For the case $l = 1$ we have:

$$\phi^{1,j,\gamma} = \sum_{\alpha=1}^{r_0} a^{1,j,\gamma}_\alpha \mathbf{a}^{0,2j-1,\alpha} \otimes \mathbf{a}^{0,2j,\alpha}$$

Denote by $A \in \mathbb{R}^{M \times r_0}$ the matrix with columns $\{\mathbf{a}^{0,2j-1,\alpha}\}_{\alpha=1}^{r_0}$, by $B \in \mathbb{R}^{M \times r_0}$ the matrix with columns $\{\mathbf{a}^{0,2j,\alpha}\}_{\alpha=1}^{r_0}$, and by $D \in \mathbb{R}^{r_0 \times r_0}$ the diagonal matrix with $\mathbf{a}^{1,j,\gamma}$ on its diagonal. Then, we may write $[\phi^{1,j,\gamma}] = ADB^\top$, and according to lemma 5 the rank of $[\phi^{1,j,\gamma}]$ equals $r := \min\{r_0, M\}$ almost everywhere w.r.t. $\left(\{\mathbf{a}^{0,2j-1,\alpha}\}_\alpha, \{\mathbf{a}^{0,2j,\alpha}\}_\alpha, \mathbf{a}^{1,j,\gamma}\right)$. To see that this holds almost everywhere w.r.t. $\{\mathbf{a}^{l,j,\gamma}\}_{l,j,\gamma}$, one should merely recall that for any dimensions $d_1, d_2 \in \mathbb{N}$, if the set $S \subset \mathbb{R}^{d_1}$ has zero measure, so does any subset of $S \times \mathbb{R}^{d_2} \subset \mathbb{R}^{d_1+d_2}$. A finite union of zero measure sets has zero measure, thus the fact that $rank[\phi^{1,j,\gamma}] = r$ holds almost everywhere individually for any $j \in [N/2]$ and $\gamma \in [r_1]$, implies that it holds almost everywhere jointly for all j and γ. This proves our inductive hypothesis (unshared case) for $l = 1$.

Assume now that almost everywhere $rank[\phi^{l-1,j',\gamma'}] \geq r^{2^{l-1}/2}$ for all $j' \in [N/2^{l-1}]$ and $\gamma' \in [r_{l-1}]$. For some specific choice of $j \in [N/2^l]$ and $\gamma \in [r_l]$ we have:

$$\phi^{l,j,\gamma} = \sum_{\alpha=1}^{r_{l-1}} a^{l,j,\gamma}_\alpha \phi^{l-1,2j-1,\alpha} \otimes \phi^{l-1,2j,\alpha} \implies [\phi^{l,j,\gamma}] = \sum_{\alpha=1}^{r_{l-1}} a^{l,j,\gamma}_\alpha [\phi^{l-1,2j-1,\alpha}] \odot [\phi^{l-1,2j,\alpha}]$$

Denote $M_\alpha := [\phi^{l-1,2j-1,\alpha}] \odot [\phi^{l-1,2j,\alpha}]$ for $\alpha = 1 \ldots r_{l-1}$. By our inductive assumption, and by the general property $rank(A \odot B) = rank(A) \cdot rank(B)$, we have that almost everywhere the ranks of all matrices M_α are at least $r^{2^{l-1}/2} \cdot r^{2^{l-1}/2} = r^{2^l/2}$. Writing $[\phi^{l,j,\gamma}] = \sum_{\alpha=1}^{r_{l-1}} a_\alpha^{l,j,\gamma} \cdot M_\alpha$, and noticing that $\{M_\alpha\}$ do not depend on $\mathbf{a}^{l,j,\gamma}$, we turn our attention to lemma 6. The lemma tells us that $rank[\phi^{l,j,\gamma}] \geq r^{2^l/2}$ almost everywhere. Since a finite union of zero measure sets has zero measure, we conclude that almost everywhere $rank[\phi^{l,j,\gamma}] \geq r^{2^l/2}$ holds jointly for all $j \in [N/2^l]$ and $\gamma \in [r_l]$. This completes the proof of the theorem in the unshared case.

Proving the theorem in the shared case may be done in the exact same way, except that for $l = 1$ one needs the version of lemma 5 for which $A(\mathbf{x})$ and $B(\mathbf{x})$ are equal. ■

We now head on to prove thm. 3, which is a generalization of thm. 1. The proof will be similar in nature to that of thm. 1, yet slightly more technical. In short, the idea is to show that in the generic case, expressing $\mathcal{A}^{(1)}$ as a sum of tensor products between tensors of order 2^{L_2-1} requires at least $r^{N/2^{L_2}}$ terms. Since $\mathcal{A}^{(2)}$ is expressed as a sum of r_{L_2-1} such terms, demanding $\mathcal{A}^{(2)} = \mathcal{A}^{(1)}$ implies $r_{L_2-1} \geq r^{N/2^{L_2}}$.

To gain technical advantage and utilize known results from matrix theory (as we did when proving thm. 1), we introduce a new tensor "squeezing" operator φ. For $q \in \mathbb{N}$, φ_q is an operator that receives a tensor with order divisible by q, and returns the tensor obtained by merging together the latter's modes in groups of size q. Specifically, when applied to the tensor $\mathcal{A} \in \mathbb{R}^{M_1 \times \cdots \times M_{c \cdot q}}$ ($c \in \mathbb{N}$), φ_q returns a tensor of order c which holds $\mathcal{A}_{d_1 \ldots d_{c \cdot q}}$ in the location defined by the following index for every mode $t \in [c]$: $1 + \sum_{i=1}^{q} (d_{i+q(t-1)} - 1) \prod_{j=i+1}^{q} M_{j+q(t-1)}$. Notice that when applied to a tensor of order q, φ_q returns a vector. Also note that if $\mathcal{A}$ and $\mathcal{B}$ are tensors with orders divisible by q, and λ is a scalar, we have the desirable properties:

- $\varphi_q(\mathcal{A} \otimes \mathcal{B}) = \varphi_q(\mathcal{A}) \otimes \varphi_q(\mathcal{B})$
- $\varphi_q(\lambda\mathcal{A} + \mathcal{B}) = \lambda\varphi_q(\mathcal{A}) + \varphi_q(\mathcal{B})$

For the sake of our proof we are interested in the case $q = 2^{L_2-1}$, and denote for brevity $\varphi := \varphi_{2^{L_2-1}}$.

As stated above, we would like to show that in the generic case, expressing $\mathcal{A}^{(1)}$ as $\sum_{z=1}^{Z} \phi_1^{(z)} \otimes \cdots \otimes \phi_{N/2^{L_2-1}}^{(z)}$, where $\phi_i^{(z)}$ are tensors of order 2^{L_2-1}, implies $Z \geq r^{N/2^{L_2}}$. Applying φ to both sides of such a decomposition gives: $\varphi(\mathcal{A}^{(1)}) = \sum_{z=1}^{Z} \varphi(\phi_1^{(z)}) \otimes \cdots \otimes \varphi(\phi_{N/2^{L_2-1}}^{(z)})$, where $\varphi(\phi_i^{(z)})$ are now vectors. Thus, to prove thm. 3 it suffices to show that in the generic case, the CP-rank of $\varphi(\mathcal{A}^{(1)})$ is at least $r^{N/2^{L_2}}$, or alternatively, that the rank of the matricization $[\varphi(\mathcal{A}^{(1)})]$ is at least $r^{N/2^{L_2}}$. This will be our strategy in the following proof:

Proof [of theorem 3] In accordance with the above discussion, it suffices to show that in the generic case $rank[\varphi(\mathcal{A}^{(1)})] \geq r^{N/2^{L_2}}$. To ease the path for the reader, we reformulate the problem using slightly simpler notations. We have an order-N tensor $\mathcal{A}$ with dimension M in each mode, generated as follows:

$$
\begin{aligned}
\phi^{1,j,\gamma} &= \sum_{\alpha=1}^{r_0} a_\alpha^{1,j,\gamma} \mathbf{a}^{0,2j-1,\alpha} \otimes \mathbf{a}^{0,2j,\alpha} \\
&\vdots \\
\phi^{l,j,\gamma} &= \sum_{\alpha=1}^{r_{l-1}} a_\alpha^{l,j,\gamma} \underbrace{\phi^{l-1,2j-1,\alpha}}_{\text{order } 2^{l-1}} \otimes \underbrace{\phi^{l-1,2j,\alpha}}_{\text{order } 2^{l-1}} \\
&\vdots \\
\mathcal{A} &= \sum_{\alpha=1}^{r_{L_1-1}} a_\alpha^{L_1,1,1} \bigotimes_{j=1}^{2^{L-L_1+1}} \underbrace{\phi^{L_1-1,j,\alpha}}_{\text{order } 2^{L_1-1}}
\end{aligned}
$$

where:

- $L_1 \leq L := \log_2 N$
- $r_0, ..., r_{L_1-1} \in \mathbb{N}_{>0}$
- $\mathbf{a}^{0,j,\alpha} \in \mathbb{R}^M$ for $j \in [N]$ and $\alpha \in [r_0]$
- $\mathbf{a}^{l,j,\gamma} \in \mathbb{R}^{r_{l-1}}$ for $l \in [L_1 - 1]$, $j \in [N/2^l]$ and $\gamma \in [r_l]$
- $\mathbf{a}^{L_1,1,1} \in \mathbb{R}^{r_{L_1-1}}$

Let L_2 be a positive integer smaller than L_1, and let φ be the tensor squeezing operator that merges groups of 2^{L_2-1} modes. Define $r := \min\{r_0, ..., r_{L_2-1}, M\}$. With $[\cdot]$ being the matricization operator defined in the beginning of the appendix, our task is to prove that $rank[\varphi(\mathcal{A})] \geq r^{N/2^{L_2}}$ almost everywhere w.r.t. $\{\mathbf{a}^{l,j,\gamma}\}_{l,j,\gamma}$. We also consider the case of shared parameters – $\mathbf{a}^{l,j,\gamma} \equiv \mathbf{a}^{l,\gamma}$, where we would like to show that the same condition holds almost everywhere w.r.t. $\{\mathbf{a}^{l,\gamma}\}_{l,\gamma}$.

Our strategy for proving the claim is inductive. We show that for $l = L_2 \ldots L_1 - 1$, almost everywhere it holds that for all j and all γ: $rank[\varphi(\phi^{l,j,\gamma})] \geq r^{2^{l-L_2}}$. We then treat the special case of $l = L_1$, showing that indeed $rank[\varphi(\mathcal{A})] \geq r^{N/2^{L_2}}$. We begin with the setting of unshared parameters ($\mathbf{a}^{l,j,\gamma}$), and afterwards attend the scenario of shared parameters ($\mathbf{a}^{l,\gamma}$) as well.

Our first task is to treat the case $l = L_2$, i.e. show that $rank[\varphi(\phi^{L_2,j,\gamma})] \geq r$ almost everywhere jointly for all j and all γ (there is actually no need for the matricization $[\cdot]$ here, as $\varphi(\phi^{L_2,j,\gamma})$ are already matrices). Since a union of finitely many zero measure sets has zero measure, it suffices to show that this condition holds almost everywhere when specific j and γ are chosen. Denote by $\mathbf{e}_i$ a vector holding 1 in entry i and 0 elsewhere, by $\mathbf{0}$ a vector of zeros, and by $\mathbf{1}$ a vector of ones. Suppose that for every j we assign $\mathbf{a}^{0,j,\alpha}$ to be $\mathbf{e}_\alpha$ when $\alpha \leq r$ and $\mathbf{0}$ otherwise. Suppose also that for all $1 \leq l \leq L_2 - 1$ and all j we set $\mathbf{a}^{l,j,\gamma}$ to be $\mathbf{e}_\gamma$ when $\gamma \leq r$ and $\mathbf{0}$ otherwise. Finally, assume we set $\mathbf{a}^{L_2,j,\gamma} = \mathbf{1}$ for all j and all γ. These settings imply that for every j, when $\gamma \leq r$ we have $\phi^{L_2-1,j,\gamma} = \otimes_{j=1}^{2^{L_2-2}}(\mathbf{e}_\gamma \otimes \mathbf{e}_\gamma)$, i.e. the tensor $\phi^{L_2-1,j,\gamma}$ holds 1 in location $(\gamma, ..., \gamma)$ and 0 elsewhere. If $\gamma > r$ then $\phi^{L_2-1,j,\gamma}$ is the zero tensor. We conclude from this that there are indices $1 \leq i_1 < ... < i_r \leq M^{L_2-1}$ such that $\varphi(\phi^{L_2-1,j,\gamma}) = \mathbf{e}_{i_\gamma}$ for $\gamma \leq r$, and that for $\gamma > r$ we have $\varphi(\phi^{L_2-1,j,\gamma}) = \mathbf{0}$. We may thus write:

$$\varphi(\phi^{L_2,j,\gamma}) = \varphi\left(\sum_{\alpha=1}^{r_{L_2-1}} \phi^{L_2-1,2j-1,\alpha} \otimes \phi^{L_2-1,2j,\alpha}\right) = \sum_{\alpha=1}^{r_{L_2-1}} \varphi(\phi^{L_2-1,2j-1,\alpha}) \otimes \varphi(\phi^{L_2-1,2j,\alpha}) = \sum_{\alpha=1}^{r} \mathbf{e}_{i_\alpha}\mathbf{e}_{i_\alpha}^\top$$

Now, since $i_1 \ldots i_r$ are different from each other, the matrix $\varphi(\phi^{L_2,j,\gamma})$ has rank r. This however does not prove our inductive hypothesis for $l = L_2$. We merely showed a specific parameter assignment for which it holds, and we need to show that it is met almost everywhere. To do so, we consider an $r \times r$ sub-matrix of $\varphi(\phi^{L_2,j,\gamma})$ which is non-singular under the specific parameter assignment we defined. The determinant of this sub-matrix is a polynomial in the elements of $\{\mathbf{a}^{l,j,\gamma}\}_{l,j,\gamma}$ which we know does not vanish with the specific assignments defined. Thus, this polynomial vanishes at subset of $\{\mathbf{a}^{l,j,\gamma}\}_{l,j,\gamma}$ having zero measure (see Caron and Traynor (2005)). That is to say, the sub-matrix of $\varphi(\phi^{L_2,j,\gamma})$ has rank r almost everywhere, and thus $\varphi(\phi^{L_2,j,\gamma})$ has rank at least r almost everywhere. This completes our treatment of the case $l = L_2$.

We now turn to prove the propagation of our inductive hypothesis. Let $l \in \{L_2 + 1, ..., L_1 - 1\}$, and assume that our inductive hypothesis holds for $l - 1$. Specifically, assume that almost everywhere w.r.t. $\{\mathbf{a}^{l,j,\gamma}\}_{l,j,\gamma}$, we have that $rank[\varphi(\phi^{l-1,j,\gamma})] \geq r^{2^{l-1-L_2}}$ jointly for all $j \in [N/2^{l-1}]$ and all $\gamma \in [r_{l-1}]$. We would like to show that almost everywhere, $rank[\varphi(\phi^{l,j,\gamma})] \geq r^{2^{l-L_2}}$ jointly for all $j \in [N/2^l]$ and all $\gamma \in [r_l]$. Again, the fact that a finite union of zero measure sets has zero measure implies that we may prove the condition for specific $j \in [N/2^l]$ and $\gamma \in [r_l]$. Applying the squeezing operator φ followed by

matricization $[\cdot]$ to the recursive expression for $\phi^{l,j,\gamma}$, we get:

$$\begin{aligned}[\varphi(\phi^{l,j,\gamma})] &= \left[\varphi\left(\sum_{\alpha=1}^{r_{l-1}} a_\alpha^{l,j,\gamma}\phi^{l-1,2j-1,\alpha}\otimes\phi^{l-1,2j,\alpha}\right)\right] = \left[\sum_{\alpha=1}^{r_{l-1}} a_\alpha^{l,j,\gamma}\varphi(\phi^{l-1,2j-1,\alpha})\otimes\varphi(\phi^{l-1,2j,\alpha})\right] \\ &= \sum_{\alpha=1}^{r_{l-1}} a_\alpha^{l,j,\gamma}[\varphi(\phi^{l-1,2j-1,\alpha})]\odot[\varphi(\phi^{l-1,2j,\alpha})]\end{aligned}$$

For $\alpha = 1\ldots r_{l-1}$, denote the matrix $[\varphi(\phi^{l-1,2j-1,\alpha})]\odot[\varphi(\phi^{l-1,2j,\alpha})]$ by M_α. The fact that the Kronecker product multiplies ranks, along with our inductive assumption, imply that almost everywhere $rank(M_\alpha) \geq r^{2^{l-1-L_2}}\cdot r^{2^{l-1-L_2}} = r^{2^{l-L_2}}$. Noting that the matrices M_α do not depend on $\mathbf{a}^{l,j,\gamma}$, we apply lemma 6 and conclude that almost everywhere $rank[\varphi(\phi^{l,j,\gamma})] \geq r^{2^{l-L_2}}$, which completes the prove of the inductive propagation.

Next, we treat the special case $l = L_1$. We assume now that almost everywhere $rank[\varphi(\phi^{L_1-1,j,\gamma})] \geq r^{2^{L_1-1-L_2}}$ jointly for all j and all γ. Again, we apply the squeezing operator φ followed by matricization $[\cdot]$, this time to both sides of the expression for $\mathcal{A}$:

$$[\varphi(\mathcal{A})] = \sum_{\alpha=1}^{r_{L_1-1}} a_\alpha^{L_1,1,1}\bigodot_{j=1}^{2^{L-L_1+1}}[\varphi(\phi^{L_1-1,j,\alpha})]$$

As before, denote $M_\alpha := \odot_{j=1}^{2^{L-L_1+1}}[\varphi(\phi^{L_1-1,j,\alpha})]$ for $\alpha = 1\ldots r_{L_1-1}$. Using again the multiplicative rank property of the Kronecker product along with our inductive assumption, we get that almost everywhere $rank(M_\alpha) \geq \prod_{j=1}^{2^{L-L_1+1}} r^{2^{L_1-1-L_2}} = r^{L-L_2}$. Noticing that $\{M_\alpha\}_{\alpha\in[r_{L_1-1}]}$ do not depend on $\mathbf{a}^{L_1,1,1}$, we apply lemma 6 for the last time and get that almost everywhere (w.r.t. $\{\mathbf{a}^{l,j,\gamma}\}_{l,j,\gamma}$), the rank of $[\varphi(\mathcal{A})]$ is at least r^{L-L_2}. This completes our proof in the case of unshared parameters.

Proving the theorem in the case of shared parameters ($\mathbf{a}^{l,j,\gamma} \equiv \mathbf{a}^{l,\gamma}$) can be done in the exact same way as above. In fact, all one has to do is omit the references to j and the proof will apply. Notice in particular that the specific parameter assignment we defined to handle $l = L_2$ was completely symmetric, i.e. it did not include any dependence on j. ■

B.2. Proof of Corollaries 2 and 4

Corollaries 2 and 4 are a direct continuation of thm. 1 and 3 respectively. In the theorems, we have shown that almost all coefficient tensors generated by a deep network cannot be realized by a shallow network if the latter does not meet a certain minimal size requirement. The corollaries take this further, by stating that given linearly independent representation functions $f_{\theta_1}\ldots f_{\theta_M}$, not only is efficient realization of coefficient tensors generally impossible, but also efficient approximation of score functions. To prove this extra step, we recall from the proofs of thm. 1 and 3 (app. B.1) that in order to show separation between the coefficient tensor of a deep network and that of a shallow network, we relied on matricization rank. Specifically, we derived constants $R^D, R^S \in \mathbb{N}$, $R^D > R^S$, such that the matricization of a deep network's coefficient tensor had rank greater or equal to R^D, whereas the matricization of a shallow network's coefficient tensor had rank smaller or equal to R^S. Given this observation, corollaries 2 and 4 readily follow from lemma 7 below (the lemma relies on basic concepts and results from the topic of L^2 Hilbert spaces – see app. C.1 for a brief discussion on the matter).

Lemma 7 *Let $f_{\theta_1}\ldots f_{\theta_M}\in L^2(\mathbb{R}^s)$ be a set of linearly independent functions, and denote by $\mathcal{T}$ the (Euclidean) space of tensors with order N and dimension M in each mode. For a given tensor $\mathcal{A}\in\mathcal{T}$, denote*

by $h(\mathcal{A})$ the function in $L^2\left((\mathbb{R}^s)^N\right)$ defined by:

$$(\mathbf{x}_1,\ldots,\mathbf{x}_N) \overset{h(\mathcal{A})}{\mapsto} \sum_{d_1,\ldots,d_N=1}^{M} \mathcal{A}_{d_1\ldots d_N} \prod_{i=1}^{N} f_{\theta_{d_i}}(\mathbf{x}_i)$$

Let $\{\mathcal{A}^\lambda\}_{\lambda\in\Lambda} \subset \mathcal{T}$ be a family of tensors, and $\mathcal{A}^*$ be a certain target tensor that lies outside the family. Assume that for all $\lambda \in \Lambda$ we have $rank([\mathcal{A}^\lambda]) < rank([\mathcal{A}^*])$, where $[\cdot]$ is the matricization operator defined in app. B.1. Then, the distance in $L^2\left((\mathbb{R}^s)^N\right)$ between $h(\mathcal{A}^*)$ and $\{h(\mathcal{A}^\lambda)\}_{\lambda\in\Lambda}$ is strictly positive, i.e. there exists an $\epsilon > 0$ such that:

$$\forall\lambda\in\Lambda : \int \left|h(\mathcal{A}^\lambda) - h(\mathcal{A}^*)\right|^2 > \epsilon$$

Proof The fact that $\{f_{\theta_d}(\mathbf{x})\}_{d\in[M]}$ are linearly independent in $L^2(\mathbb{R}^s)$ implies that the product functions $\{\prod_{i=1}^N f_{\theta_{d_i}}(\mathbf{x}_i)\}_{d_1\ldots d_N\in[M]}$ are linearly independent in $L^2\left((\mathbb{R}^s)^N\right)$ (see app. C.1). Let $(h^{(t)})_{t=1}^\infty$ be a sequence of functions that lie in the span of $\{\prod_{i=1}^N f_{\theta_{d_i}}(\mathbf{x}_i)\}_{d_1\ldots d_N\in[M]}$, and for every $t\in\mathbb{N}$ denote by $\mathcal{A}^{(t)}$ the coefficient tensor of $h^{(t)}$ under this basis, i.e. $\mathcal{A}^{(t)} \in \mathcal{T}$ is defined by:

$$h^{(t)}(\mathbf{x}_1,\ldots,\mathbf{x}_N) = \sum_{d_1,\ldots,d_N=1}^{M} \mathcal{A}^{(t)}_{d_1,\ldots,d_N} \prod_{i=1}^{N} f_{\theta_{d_i}}(\mathbf{x}_i)$$

Assume that $(h^{(t)})_{t=1}^\infty$ converges to $h(\mathcal{A}^*)$ in $L^2\left((\mathbb{R}^s)^N\right)$:

$$\lim_{t\to\infty} \int \left|h^{(t)} - h(\mathcal{A}^*)\right|^2 = 0$$

In a finite-dimensional Hilbert space, convergence in norm implies convergence in representation coefficients under any preselected basis. We thus have:

$$\forall d_1\ldots d_N \in [M] : \mathcal{A}^{(t)}_{d_1,\ldots,d_N} \xrightarrow{t\to\infty} \mathcal{A}^*_{d_1,\ldots,d_N}$$

This means in particular that in the tensor space $\mathcal{T}$, $\mathcal{A}^*$ lies in the closure of $\{\mathcal{A}^{(t)}\}_{t=1}^\infty$. Accordingly, in order to show that the distance in $L^2\left((\mathbb{R}^s)^N\right)$ between $h(\mathcal{A}^*)$ and $\{h(\mathcal{A}^\lambda)\}_{\lambda\in\Lambda}$ is strictly positive, it suffices to show that the distance in $\mathcal{T}$ between $\mathcal{A}^*$ and $\{\mathcal{A}^\lambda\}_{\lambda\in\Lambda}$ is strictly positive, or equivalently, that the distance between the matrix $[\mathcal{A}^*]$ and the family of matrices $\{[\mathcal{A}^\lambda]\}_{\lambda\in\Lambda}$ is strictly positive. This however is a direct implication of the assumption $\forall\lambda\in\Lambda : rank([\mathcal{A}^\lambda]) < rank([\mathcal{A}^*])$. ■

Appendix C. Derivation of Hypotheses Space

In order to keep the body of the paper at a reasonable length, the presentation of our hypotheses space (eq. 2) in sec. 3 did not provide the grounds for its definition. In this appendix we derive the hypotheses space step by step. After establishing basic preliminaries on the topic of L^2 spaces, we utilize the notion of tensor products between such spaces to reach a universal representation as in eq. 2 but with $M\to\infty$. We then make use of empirical studies characterizing the statistics of natural images, to argue that in practice a moderate value of M ($M\in\Omega(100)$) suffices.

C.1. Preliminaries on L^2 Spaces

When dealing with functions over scalars, vectors or collections of vectors, we consider L^2 spaces, or more formally, the Hilbert spaces of Lebesgue measurable square-integrable real functions equipped with standard

(point-wise) addition and scalar multiplication, as well as the inner-product defined by integral over point-wise multiplication. The topic of L^2 function spaces lies at the heart of functional analysis, and requires basic knowledge in measure theory. We present here the bare necessities required to follow this appendix, referring the interested reader to Rudin (1991) for a more comprehensive introduction.

For our purposes, it suffices to view an L^2 space as a vector space of all functions f satisfying $\int f^2 < \infty$. This vector space is infinite dimensional, and a set of functions $\mathcal{F} \subset L^2$ is referred to as *total* if the closure of its span covers the entire space, i.e. if for any function $g \in L^2$ and $\epsilon > 0$, there exist functions $f_1 \ldots f_K \in \mathcal{F}$ and coefficients $c_1 \ldots c_K \in \mathbb{R}$ such that $\int |\sum_{i=1}^{K} c_i \cdot f_i - g|^2 < \epsilon$. $\mathcal{F}$ is regarded as *linearly independent* if all of its finite subsets are linearly independent, i.e. for any $f_1 \ldots f_K \in \mathcal{F}$, $f_i \neq f_j$, and $c_1 \ldots c_K \in \mathbb{R}$, if $\sum_{i=1}^{K} c_i \cdot f_i = 0$ then $c_1 = \cdots = c_K = 0$. A non-trivial result states that L^2 spaces in general must contain total and linearly independent sets, and moreover, for any $s \in \mathbb{N}$, $L^2(\mathbb{R}^s)$ contains a *countable* set of this type. It seems reasonable to draw an analogy between total and linearly independent sets in L^2 space, and bases in a finite dimensional vector space. While this analogy is indeed appropriate from our perspective, total and linearly independent sets are not to be confused with *bases* for L^2 spaces, which are typically defined to be orthonormal.

It can be shown (see for example Hackbusch (2012)) that for any natural numbers s and N, if $\{f_d(\mathbf{x})\}_{d\in\mathbb{N}}$ is a total or a linearly independent set in $L^2(\mathbb{R}^s)$, then $\{(\mathbf{x}_1, \ldots, \mathbf{x}_N) \mapsto \prod_{i=1}^{N} f_{d_i}(\mathbf{x}_i)\}_{d_1 \ldots d_N \in \mathbb{N}}$, the induced point-wise product functions on $(\mathbb{R}^s)^N$, form a set which is total or linearly independent, respectively, in $L^2\left((\mathbb{R}^s)^N\right)$. As we now briefly outline, this result actually emerges from a deep relation between tensor products and Hilbert spaces. The definitions given in sec. 2 for a tensor, tensor space, and tensor product, are actually concrete special cases of much deeper, abstract algebraic concepts. A more formal line of presentation considers multiple vector spaces $V_1 \ldots V_N$, and defines their tensor product space $V_1 \otimes \cdots \otimes V_N$ to be a specific quotient space of the space freely generated by their Cartesian product set. For every combination of vectors $\mathbf{v}^{(i)} \in V_i$, $i \in [N]$, there exists a corresponding element $\mathbf{v}^{(1)} \otimes \cdots \otimes \mathbf{v}^{(N)}$ in the tensor product space, and moreover, elements of this form span the entire space. If $V_1 \ldots V_N$ are Hilbert spaces, it is possible to equip $V_1 \otimes \cdots \otimes V_N$ with a natural inner-product operation, thereby turning it too into a Hilbert space. It may then be shown that if the sets $\{\mathbf{v}_\alpha^{(i)}\}_\alpha \subset V_i$, $i \in [N]$, are total or linearly independent, elements of the form $\mathbf{v}_{\alpha_1}^{(1)} \otimes \cdots \otimes \mathbf{v}_{\alpha_N}^{(N)}$ are total or linearly independent, respectively, in $V_1 \otimes \cdots \otimes V_N$. Finally, when the underlying Hilbert spaces are $L^2(\mathbb{R}^s)$, the point-wise product mapping $f_1(\mathbf{x}) \otimes \cdots \otimes f_N(\mathbf{x}) \mapsto \prod_{i=1}^{N} f_i(\mathbf{x}_i)$ from the tensor product space $\left(L^2(\mathbb{R}^s)\right)^{\otimes N} := L^2(\mathbb{R}^s) \otimes \cdots \otimes L^2(\mathbb{R}^s)$ to $L^2\left((\mathbb{R}^s)^N\right)$, induces an isomorphism of Hilbert spaces.

C.2. Construction

Recall from sec. 3 that our instance space is defined as $\mathcal{X} := (\mathbb{R}^s)^N$, in accordance with the common practice of representing natural data through ordered local structures (for example images are often represented through small patches around their pixels). We classify instances into categories $\mathcal{Y} := \{1 \ldots Y\}$ via maximization of per-label score functions $\{h_y : (\mathbb{R}^s)^N \to \mathbb{R}\}_{y\in\mathcal{Y}}$. Our hypotheses space $\mathcal{H}$ is defined to be the subset of $L^2\left((\mathbb{R}^s)^N\right)$ from which score functions may be taken.

In app. C.1 we stated that if $\{f_d(\mathbf{x})\}_{d\in\mathbb{N}}$ is a total set in $L^2(\mathbb{R}^s)$, i.e. if every function in $L^2(\mathbb{R}^s)$ can be arbitrarily well approximated by a linear combination of a finite subset of $\{f_d(\mathbf{x})\}_{d\in\mathbb{N}}$, then the point-wise products $\{(\mathbf{x}_1, \ldots, \mathbf{x}_N) \mapsto \prod_{i=1}^{N} f_{d_i}(\mathbf{x}_j)\}_{d_1,\ldots,d_N\in\mathbb{N}}$ form a total set in $L^2\left((\mathbb{R}^s)^N\right)$. Accordingly, in a universal hypotheses space $\mathcal{H} = L^2\left((\mathbb{R}^s)^N\right)$, any score function h_y may be arbitrarily well approximated by finite linear combinations of such point-wise products. A possible formulation of this would be as follows. Assume we are interested in ϵ-approximation of the score function h_y, and consider a formal tensor $\mathcal{A}^y$ having N modes and a countable infinite dimension in each mode $i \in [N]$, indexed by $d_i \in \mathbb{N}$. Then, there exists such a tensor, with all but a finite number of entries set to zero, for which:

$$h_y\left(\mathbf{x}_1, \ldots, \mathbf{x}_N\right) \approx \sum_{d_1 \ldots d_N \in \mathbb{N}} A^y_{d_1,\ldots,d_N} \prod_{i=1}^{N} f_{d_i}(\mathbf{x}_i) \tag{6}$$

Given that the set of functions $\{f_d(\mathbf{x})\}_{d\in\mathbb{N}}\subset L^2(\mathbb{R}^s)$ is total, eq. 6 defines a universal hypotheses space. There are many possibilities for choosing a total set of functions. Wavelets are perhaps the most obvious choice, and were indeed used in a deep network setting by Bruna and Mallat (2012). The special case of Gabor wavelets has been claimed to induce features that resemble representations in the visual cortex (Serre et al. (2005)). Two options we pay special attention to due to their importance in practice are:

- *Gaussians* (with diagonal covariance):

$$f_\theta(\mathbf{x}) = \mathcal{N}\left(\mathbf{x};\boldsymbol{\mu},diag(\boldsymbol{\sigma}^2)\right) \tag{7}$$

  where $\theta = (\boldsymbol{\mu}\in\mathbb{R}^s, \boldsymbol{\sigma}^2\in\mathbb{R}^s_{++})$.

- *Neurons*:

$$f_\theta(\mathbf{x}) = \sigma\left(\mathbf{x}^\top\mathbf{w}+b\right) \tag{8}$$

  where $\theta = (\mathbf{w}\in\mathbb{R}^s, b\in\mathbb{R})$ and σ is a point-wise non-linear activation such as threshold $\sigma(z) = \mathbb{1}[z>0]$, rectified linear unit (ReLU) $\sigma(z)=\max\{z,0\}$ or sigmoid $\sigma(z)=1/(1+e^{-z})$.

In both cases, there is an underlying parametric family of functions $\mathcal{F} = \{f_\theta:\mathbb{R}^s\to\mathbb{R}\}_{\theta\in\Theta}$ of which a countable total subset may be chosen. The fact that Gaussians as above are total in $L^2(\mathbb{R}^s)$ has been proven in Girosi and Poggio (1990), and is a direct corollary of the Stone-Weierstrass theorem. To achieve countability, simply consider Gaussians with rational parameters (mean and variances). In practice, the choice of Gaussians (with diagonal covariance) give rises to a "similarity" operator as described by the SimNet architecture (Cohen and Shashua (2014); Cohen et al. (2016)). For the case of neurons we must restrict the domain $\mathbb{R}^s$ to some bounded set, otherwise the functions are not integrable. This however is not a limitation in practice, and indeed neurons are widely used across many application domains. The fact that neurons are total has been proven in Cybenko (1989) and Hornik et al. (1989) for threshold and sigmoid activations. More generally, it has been proven in Stinchcombe and White (1989) for a wide class of activation functions, including linear combinations of ReLU. See Pinkus (1999) for a survey of such results. For countability, we may again restrict parameters (weights and bias) to be rational.

In the case of Gaussians and neurons, we argue that a *finite* set of functions suffices, i.e. that it is possible to choose $f_{\theta_1}\ldots f_{\theta_M}\in\mathcal{F}$ that will suffice in order to represent score functions required for natural tasks. Moreover, we claim that M need not be large (e.g. on the order of 100). Our argument relies on statistical properties of natural images, and is fully detailed in app. C.3. It implies that under proper choice of $\{f_{\theta_d}(\mathbf{x})\}_{d\in[M]}$, the finite set of point-wise product functions $\{(\mathbf{x}_1,\ldots,\mathbf{x}_N)\mapsto\prod_{i=1}^N f_{\theta_{d_i}}(\mathbf{x}_i)\}_{d_1,\ldots,d_N\in[M]}$ spans the score functions of interest, and we may define for each label y a tensor $\mathcal{A}^y$ of order N and dimension M in each mode, such that:

$$h_y(\mathbf{x}_1,\ldots,\mathbf{x}_N) = \sum_{d_1,\ldots,d_N=1}^{M}\mathcal{A}^y_{d_1,\ldots,d_N}\prod_{i=1}^{N}f_{\theta_{d_i}}(\mathbf{x}_i) \tag{2}$$

which is exactly the hypotheses space presented in sec. 3. Notice that if $\{f_{\theta_d}(\mathbf{x})\}_{d\in[M]}\subset L^2(\mathbb{R}^s)$ are linearly independent (there is no reason to choose them otherwise), then so are the product functions $\{(\mathbf{x}_1,\ldots,\mathbf{x}_N)\mapsto\prod_{i=1}^N f_{\theta_{d_i}}(\mathbf{x}_i)\}_{d_1,\ldots,d_N\in[M]}\subset L^2\left((\mathbb{R}^s)^N\right)$ (see app. C.1), and a score function h_y uniquely determines the coefficient tensor $\mathcal{A}^y$. In other words, two score functions $h_{y,1}$ and $h_{y,2}$ are identical if and only if their coefficient tensors $\mathcal{A}^{y,1}$ and $\mathcal{A}^{y,2}$ are the same.

C.3. Finite Function Bases for Classification of Natural Data

In app. C.2 we laid out the framework of classifying instances in the space $\mathcal{X} := \{(\mathbf{x}_1,\ldots,\mathbf{x}_N):\mathbf{x}_i\in\mathbb{R}^s\} = (\mathbb{R}^s)^N$ into labels $\mathcal{Y} := \{1,\ldots,Y\}$ via maximization of per-label score functions $h_y:\mathcal{X}\to\mathbb{R}$:

$$\hat{y}(\mathbf{x}_1,\ldots,\mathbf{x}_N) = \operatorname*{argmax}_{y\in\mathcal{Y}} h_y(\mathbf{x}_1,\ldots,\mathbf{x}_N)$$

where $h_y(\mathbf{x}_1, \ldots, \mathbf{x}_N)$ is of the form:

$$h_y\left(\mathbf{x}_1, \ldots, \mathbf{x}_N\right) = \sum_{d_1,\ldots,d_N=1}^{M} A^y_{d_1,\ldots,d_N} \prod_{i=1}^{N} f_{\theta_{d_i}}(\mathbf{x}_i) \tag{2}$$

and $\{f_\theta\}_{d\in[M]}$ are selected from a parametric family of functions $\mathcal{F} = \{f_\theta : \mathbb{R}^s \to \mathbb{R}\}_{\theta\in\Theta}$. For universality, i.e. for the ability of score functions h_y to approximate any function in $L^2(\mathcal{X})$ as $M \to \infty$, we required that it be possible to choose a countable subset of $\mathcal{F}$ that is total in $L^2(\mathbb{R}^s)$. We noted that the families of Gaussians (eq. 7) and neurons (eq. 8) meet this requirement.

In this subsection we formalize our argument that a *finite* value for M is sufficient when $\mathcal{X}$ represents natural data, and in particular, natural images. Based on empirical studies characterizing the statistical properties of natural images, and in compliance with the number of channels in a typical convolutional network layer, we find that M on the order of 100 typically suffices.

Let $\mathcal{D}$ be a distribution of labeled instances $(X, \bar{y})$ over $\mathcal{X} \times \mathcal{Y}$ (we use bar notation to distinguish the label $\bar{y}$ from the running index y), and $\mathcal{D}_\mathcal{X}$ be the induced marginal distribution of instances X over $\mathcal{X}$. We would like to show, given particular assumptions on $\mathcal{D}$, that there exist functions $f_{\theta_1}, \ldots, f_{\theta_M} \in \mathcal{F}$ and tensors $\mathcal{A}^1, \ldots, \mathcal{A}^Y$ of order N and dimension M in each mode, such that the score functions h_y defined in eq. 2 achieve low classification error:

$$L^{0-1}_\mathcal{D}(h_1, \ldots, h_Y) := \mathbb{E}_{(X,\bar{y})\sim\mathcal{D}}\left[\mathbb{1}\left[\bar{y} \neq \operatorname*{argmax}_{y\in\mathcal{Y}} h_y(X)\right]\right] \tag{9}$$

$\mathbb{1}[\cdot]$ here stands for the indicator function, taking the value 1 when its argument is true, and 0 otherwise.

Let $\{h^*_y\}_{y\in\mathcal{Y}}$ be a set of "ground truth" score functions for which optimal prediction is achieved, or more specifically, for which the expected hinge-loss (upper bounds the 0-1 loss) is minimal:

$$(h^*_1, \ldots, h^*_Y) = \operatorname*{argmin}_{h'_1,\ldots,h'_Y:\mathcal{X}\to\mathbb{R}} L^{hinge}_\mathcal{D}(h'_1, \ldots, h'_Y)$$

where:

$$L^{hinge}_\mathcal{D}(h'_1, \ldots, h'_Y) := \mathbb{E}_{(X,\bar{y})\sim\mathcal{D}}\left[\max_{y\in\mathcal{Y}}\left\{\mathbb{1}\left[y \neq \bar{y}\right] + h'_y(X)\right\} - h'_{\bar{y}}(X)\right] \tag{10}$$

Our strategy will be to select score functions h_y of the format given in eq. 2, that approximate h^*_y in the sense of low expected maximal absolute difference:

$$\mathcal{E} := \mathbb{E}_{X\sim\mathcal{D}_\mathcal{X}}\left[\max_{y\in\mathcal{Y}}\left|h_y(X) - h^*_y(X)\right|\right] \tag{11}$$

We refer to $\mathcal{E}$ as the *score approximation error* obtained by h_y. The 0-1 loss of h_y with respect to the labeled example $(X, \bar{y}) \in \mathcal{X} \times \mathcal{Y}$ is bounded as follows:

$$\begin{aligned}
\mathbb{1}\left[\bar{y} \neq \operatorname*{argmax}_{y\in\mathcal{Y}} h_y(X)\right] &\leq \max_{y\in\mathcal{Y}}\left\{\mathbb{1}\left[y \neq \bar{y}\right] + h_y(X)\right\} - h_{\bar{y}}(X) \\
&= \max_{y\in\mathcal{Y}}\left\{\mathbb{1}\left[y \neq \bar{y}\right] + h^*_y(X) + h_y(X) - h^*_y(X)\right\} - h^*_{\bar{y}}(X) + h^*_{\bar{y}}(X) - h_{\bar{y}}(X) \\
&\leq \max_{y\in\mathcal{Y}}\left\{\mathbb{1}\left[y \neq \bar{y}\right] + h^*_y(X)\right\} - h^*_{\bar{y}}(X) + \max_{y\in\mathcal{Y}}\left\{h_y(X) - h^*_y(X)\right\} + h^*_{\bar{y}}(X) - h_{\bar{y}}(X) \\
&\leq \max_{y\in\mathcal{Y}}\left\{\mathbb{1}\left[y \neq \bar{y}\right] + h^*_y(X)\right\} - h^*_{\bar{y}}(X) + 2\max_{y\in\mathcal{Y}}\left\{\left|h_y(X) - h^*_y(X)\right|\right\}
\end{aligned}$$

Taking expectation of the first and last terms above with respect to $(X, \bar{y}) \sim \mathcal{D}$, and recalling the definitions given in eq. 9, 10 and 11, we get:

$$L^{0-1}_\mathcal{D}(h_1, \ldots, h_Y) \leq L^{hinge}_\mathcal{D}(h^*_1, \ldots, h^*_Y) + 2\mathcal{E}$$

In words, the classification error of the score functions h_y is bounded by the optimal expected hinge-loss plus a term equal to twice their score approximation error. Recall that we did not constrain the optimal score functions h_y^* in any way. Thus, assuming a label is deterministic given an instance, the optimal expected hinge-loss is essentially zero, and the classification error of h_y is dominated by their score approximation error $\mathcal{E}$ (eq. 11). Our problem thus translates to showing that h_y can be selected such that $\mathcal{E}$ is small.

At this point we introduce our main assumption on the distribution $\mathcal{D}$, or more specifically, on the marginal distribution of instances $\mathcal{D}_{\mathcal{X}}$. According to various studies, in natural settings, the marginal distribution of individual vectors in $\mathcal{X}$, e.g. of small patches in images, may be relatively well captured by a Gaussian Mixture Model (*GMM*) with a moderate number (on the order of 100 or less) of distinct components. For example, it was shown in Zoran and Weiss (2012) that natural image patches of size 2×2, 4×4, 8×8 or 16×16, can essentially be modeled by GMMs with 64 components (adding more components barely improved the log-likelihood). This complies with the common belief that a moderate number of low-level templates suffices in order to model the vast majority of local image patches. Following this line, we model the marginal distribution of $\mathbf{x}_i$ with a GMM having M components with means $\boldsymbol{\mu}_1 \ldots \boldsymbol{\mu}_M \in \mathbb{R}^s$. We assume that the components are well localized, i.e. that their standard deviations are small compared to the distances between means, and also compared to the variation of the target functions h_y^*. In the context of images for example, the latter two assumptions imply that a local patch can be unambiguously assigned to a template, and that the assignment of patches to templates determines the class of an image. Returning to general instances X, their probability mass will be concentrated in distinct regions of the space $\mathcal{X}$, in which for every $i \in [N]$, the vector $\mathbf{x}_i$ lies near $\boldsymbol{\mu}_{c_i}$ for some $c_i \in [M]$. The score functions h_y^* are approximately constant in each such region. It is important to stress here that we do *not* assume statistical independence of $\mathbf{x}_i$'s, only that their possible values can be quantized into M templates $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_M$.

Under our idealized assumptions on $\mathcal{D}_{\mathcal{X}}$, the expectation in the score approximation error $\mathcal{E}$ can be discretized as follows:

$$\mathcal{E} := \mathbb{E}_{X\sim\mathcal{D}_{\mathcal{X}}}\left[\max_{y\in\mathcal{Y}}\left|h_y(X) - h_y^*(X)\right|\right] = \sum_{c_1,\ldots,c_N=1}^{M} P_{c_1,\ldots,c_N} \max_{y\in\mathcal{Y}}\left|h_y(\mathcal{M}_{c_1,\ldots,c_N}) - h_y^*(\mathcal{M}_{c_1,\ldots,c_N})\right| \tag{12}$$

where $\mathcal{M}_{c_1,\ldots,c_N} := (\boldsymbol{\mu}_{c_1}, \ldots, \boldsymbol{\mu}_{c_N})$ and $P_{c_1,\ldots,c_N}$ stands for the probability that $\mathbf{x}_i$ lies near $\boldsymbol{\mu}_{c_i}$ for every $i \in [N]$ ($P_{c_1,\ldots,c_N} \geq 0$, $\sum_{c_1,\ldots,c_N} P_{c_1,\ldots,c_N} = 1$).

We now turn to show that $f_{\theta_1} \ldots f_{\theta_M}$ can be chosen to separate GMM components, i.e. such that for every $c, d \in [M]$, $f_{\theta_d}(\boldsymbol{\mu}_c) \neq 0$ if and only if $c = d$. If the functions f_θ are Gaussians (eq. 7), we can simply set the mean of f_{θ_d} to $\boldsymbol{\mu}_d$, and its standard deviations to be low enough such that the function effectively vanishes at $\boldsymbol{\mu}_c$ when $c \neq d$. If f_θ are neurons (eq. 8), an additional requirement is needed, namely that the GMM component means $\boldsymbol{\mu}_1 \ldots \boldsymbol{\mu}_M$ be linearly separable. In other words, we require that for every $d \in [M]$, there exist $\mathbf{w}_d \in \mathbb{R}^s$ and $b_d \in \mathbb{R}$ for which $\mathbf{w}_d^\top \boldsymbol{\mu}_c + b_d$ is positive if $c = d$ and negative otherwise. This may seem like a strict assumption at first glance, but notice that the dimension s is often as large, or even larger, then the number of components M. In addition, if input vectors $\mathbf{x}_i$ are normalized to unit length (a standard practice with image patches for example), $\boldsymbol{\mu}_1 \ldots \boldsymbol{\mu}_M$ will also be normalized, and thus linear separability is trivially met. Assuming we have linear separability, one may set $\theta_d = (\mathbf{w}_d, b_d)$, and for threshold or ReLU activations we indeed get $f_{\theta_d}(\boldsymbol{\mu}_c) \neq 0 \iff c = d$. With sigmoid activations, we may need to scale $(\mathbf{w}_d, b_d)$ so that $\mathbf{w}_d^\top \boldsymbol{\mu}_c + b_d \ll 0$ when $c \neq d$, and that would ensure that in this case $f_{\theta_d}(\boldsymbol{\mu}_c)$ effectively vanishes.

Assuming we have chosen $f_{\theta_1} \ldots f_{\theta_M}$ to separate GMM components, and plugging-in the format of h_y given in eq. 2, we get the following convenient form for $h_y(\mathcal{M}_{c_1,\ldots,c_N})$:

$$h_y(\mathcal{M}_{c_1,\ldots,c_N}) = A^y_{c_1,\ldots,c_N} \prod_{i=1}^{N} f_{\theta_{c_i}}(\boldsymbol{\mu}_{c_i})$$

Assigning the coefficient tensors through the following rule:

$$A^y_{c_1,\ldots,c_N} = \frac{h_y^*(\mathcal{M}_{c_1,\ldots,c_N})}{\prod_{i=1}^{N} f_{\theta_{c_i}}(\boldsymbol{\mu}_{c_i})}$$

implies:

$$h_y(\mathcal{M}_{c_1,\ldots,c_N}) = h^*_y(\mathcal{M}_{c_1,\ldots,c_N})$$

for every $y \in \mathcal{Y}$ and $c_1 \ldots c_N \in [M]$. Plugging this into eq. 12, we get a score approximation error of zero.

To recap, we have shown that when the parametric functions f_θ are Gaussians (eq. 7) or neurons (eq. 8), not only are the score functions h_y given in eq. 2 universal when $M \to \infty$ (see app. C.2), but they can also achieve zero classification error (eq. 9) with a moderate value of M (on the order of 100) if the underlying data distribution $\mathcal{D}$ is "natural". In this context, $\mathcal{D}$ is regarded as natural if it satisfies two conditions. The first, which is rather mild, requires that a label be completely determined by the instance. For example, an image will belong to one category with probability one, and to the rest of the categories with probability zero. The second condition, which is far more restrictive, states that input vectors composing an instance can be quantized into a moderate number (M) of templates. The assumption that natural images exhibit this property is based on various empirical studies where it is shown to hold approximately. Since it does not hold exactly, our analysis is approximate, and its implication in practice is that the classification error introduced by constraining score functions to have the format given in eq. 2, is negligible compared to other sources of error (factorization of the coefficient tensors, finiteness of training data and difficulty in optimization).

Appendix D. Related Work

The classic approach for theoretically analyzing the power of depth focused on investigation of the computational complexity of *boolean circuits*. An early result, known as the "exponential efficiency of depth", may be summarized as follows: for every integer k, there are boolean functions that can be computed by a circuit comprising alternating layers of AND and OR gates which has depth k and polynomial size, yet if one limits the depth to $k-1$ or less, an exponentially large circuit is required. See Sipser (1983) for a formal statement of this classic result. Recently, Rossman et al. (2015) have established a somewhat stronger result, showing cases where not only are polynomially wide shallow boolean circuits incapable of exact realization, but also of approximation (i.e. of agreeing with the target function on more than a specified fraction of input combinations). Other classical results are related to *threshold circuits*, a class of models more similar to contemporary neural networks than boolean circuits. Namely, they can be viewed as neural networks where each neuron computes a weighted sum of its inputs (possibly including bias), followed by threshold activation ($\sigma(z) = \mathbf{1}[z \geq 0]$). For threshold circuits, the main known result in our context is the existence of functions that separate depth 3 from depth 2 (see Hajnal et al. (1987) for a statement relating to exact realization, and the techniques in Maass et al. (1994); Martens et al. (2013) for extension to approximation).

More recent studies focus on *arithmetic circuits* (Shpilka and Yehudayoff (2010)), whose nodes typically compute either a weighted sum or a product of their inputs [9] (besides their role in studying expressiveness, deep networks of this class have been shown to support provably optimal training Livni et al. (2014)). A special case of this are the Sum-Product Networks (*SPNs*) presented in Poon and Domingos (2011). SPNs are a class of deep generative models designed to efficiently compute probability density functions. Their summation weights are typically constrained to be non-negative (such an arithmetic circuit is called *monotone*), and in addition, in order for them to be valid (i.e. to be able to compute probability density functions), additional architectural constraints are needed (e.g. decomposability and completeness). The most widely known theoretical arguments regarding the efficiency of depth in SPNs were given in Delalleau and Bengio (2011). In this work, two specific families of SPNs were considered, both comprising alternating sum and product layers – a family $\mathcal{F}$ whose nodes form a full binary tree, and a family $\mathcal{G}$ with n nodes per layer (excluding the output), each connected to $n-1$ nodes in the preceding layer. The authors show that functions implemented by these networks require an exponential number of nodes in order to be realized by shallow (single

9. There are different definitions for arithmetic circuits in the literature. We adopt the definition given in Martens and Medabalimi (2014), under which an arithmetic circuit is a directed acyclic graph, where nodes with no incoming edges correspond to inputs, nodes with no outgoing edges correspond to outputs, and the remaining nodes are either labeled as "sum" or as "product". A product node computes the product of its child nodes. A sum node computes a weighted sum of its child nodes, where the weights are parameters linked to its incoming edges.

hidden-layer networks). The limitations of this work are twofold. First, as the authors note themselves, it only analyzes the ability of shallow networks to realize *exactly* functions generated by deep networks, and does not provide any result relating to approximation. Second, the specific SPN families considered in this work are not universal hypothesis classes and do not resemble networks used in practice. Recently, Martens and Medabalimi (2014) proved that there exist functions which can be efficiently computed by decomposable and complete (D&C) SPNs of depth $d+1$, yet require a D&C SPN of depth d or less to have super-polynomial size for exact realization. This analysis only treats approximation in the limited case of separating depth 4 from depth 3 (D&C) SPNs. Additionally, it only deals with specific separating functions, and does not convey information regarding how frequent these are. In other words, according to this analysis, it may be that almost all functions generated by deep networks *can* be efficiently realized by shallow networks, and there are only few pathological functions for which this does not hold. A further limitation of this analysis is that for general d, the separation between depths $d+1$ and d is based on a multilinear circuit result by Raz and Yehudayoff (2009), that translates into a network that once again does not follow the common practices of deep learning.

There have been recent attempts to analyze the efficiency of network depth in other settings as well. The most commonly used type of neural networks these days includes neurons that compute a weighted sum of their inputs (with bias) followed by Rectified Linear Unit (ReLU) activation ($\sigma(z) = \max\{0, z\}$). Pascanu et al. (2013) and Montufar et al. (2014) study the number of linear regions that may be expressed by such networks as a function of their depth and width, thereby showing existence of functions separating deep from shallow (depth 2) networks. Telgarsky (2015) shows a simple construction of a depth d width 2 ReLU network that operates on one-dimensional inputs, realizing a function that cannot be approximated by ReLU networks of depth $o(d/\log d)$ and width polynomial in d. Eldan and Shamir (2015) provides functions expressible by ReLU networks of depth 3 and polynomial width, which can only be approximated by a depth 2 network if the latter's width is exponential. The result in this paper applies not only to ReLU activation, but also to the standard sigmoid ($\sigma(z) = 1/(1+e^{-z})$), and more generally, to any universal activation (see assumption 1 in Eldan and Shamir (2015)). Bianchini and Scarselli (2014) also considers different types of activations, studying the topological complexity (through Betti numbers) of decision regions as a function of network depth, width and activation type. The results in this paper establish the existence of deep vs. shallow separating functions only for the case of polynomial activation. While the above works do address more conventional neural networks, they do not account for the structure of convolutional networks – the most successful deep learning architectures to date, and more importantly, they too prove only existence of *some* separating functions, without providing any insight as to how frequent these are.

We are not the first to incorporate ideas from the field of tensor analysis into deep learning. Socher et al. (2013), Yu et al. (2012), Setiawan et al. (2015), and Hutchinson et al. (2013) all proposed different neural network architectures that include tensor-based elements, and exhibit various advantages in terms of expressiveness and/or ease of training. In Janzamin et al. (2015), an alternative algorithm for training neural networks is proposed, based on tensor decomposition and Fourier analysis, with proven generalization bounds. In Novikov et al. (2014), Anandkumar et al. (2014), Yang and Dunson (2015) and Song et al. (2013), algorithms for tensor decompositions are used to estimate parameters of different graphical models. Notably, Song et al. (2013) uses the relatively new Hierarchical Tucker decomposition (Hackbusch and Kühn (2009)) that we employ in our work, with certain similarities in the formulations. The works differ considerably in their objectives though: while Song et al. (2013) focuses on the proposal of a new training algorithm, our purpose in this work is to analyze the expressive efficiency of networks and how that depends on depth. Recently, Lebedev et al. (2014) modeled the filters in a convolutional network as four dimensional tensors, and used the CP decomposition to construct an efficient and accurate approximation. Another work that draws a connection between tensor analysis and deep learning is the recent study presented in Haeffele and Vidal (2015). This work shows that with sufficiently large neural networks, no matter how training is initialized, there exists a local optimum that is accessible with gradient descent, and this local optimum is approximately equivalent to the global optimum in terms of objective value.

Appendix E. Computation in Log-Space with SimNets

A practical issue one faces when implementing arithmetic circuits is the numerical instability of the product operation – a product node with a large number of inputs is easily susceptible to numerical overflow or underflow. A common solution to this is to perform the computations in log-space, i.e. instead of computing activations we compute their $\log$. This requires the activations to be non-negative to begin with, and alters the sum and product operations as follows. A product simply turns into a sum, as $\log \prod_i \alpha_i = \sum_i \log \alpha_i$. A sum becomes what is known as *log-sum-exp* or *softmax*: $\log \sum_i \alpha_i = \log \sum_i \exp(\log \alpha_i)$.

Turning to our networks, the requirement that all activations be non-negative does not limit their universality. The reason for this is that the functions f_θ are non-negative in both cases of interest – Gaussians (eq. 7) and neurons (eq. 8). In addition, one can always add a common offset to all coefficient tensors $\mathcal{A}^y$, ensuring they are positive without affecting classification. Non-negative decompositions (i.e. decompositions with all weights holding non-negative values) can then be found, leading all network activations to be non-negative. In general, non-negative tensor decompositions may be less efficient than unconstrained decompositions, as there are cases where a non-negative tensor supports an unconstrained decomposition that is smaller than its minimal non-negative decomposition. Nevertheless, as we shall soon see, these non-negative decompositions translate into a proven architecture, which was demonstrated to achieve comparable performance to state of the art convolutional networks, thus in practice the deterioration in efficiency does not seem to be significant.

Naïvely implementing CP or HT model (fig. 1 or 2 respectively) in log-space translates to $\log$ activation following the locally connected linear transformations (convolutions if coefficients are shared, see sec. 3.3), to product pooling turning into sum pooling, and to $\exp$ activation following the pooling. However, applying $\exp$ and $\log$ activations as just described, without proper handling of the inputs to each computational layer, would not result in a numerically stable computation [10].

The SimNet architecture (Cohen and Shashua (2014); Cohen et al. (2016)) naturally brings forth a numerically stable implementation of our networks. The architecture is based on two ingredients – a flexible similarity measure and the *MEX* operator:

$$\text{MEX}_\beta(\mathbf{x}, \mathbf{b}) := \frac{1}{\beta} \log \left(\frac{1}{N} \sum_j \exp(\beta(x_j + b_j)) \right)$$

The similarity layer, capable of computing both the common convolutional operator as well as weighted l_p norm, may realize the representation by computing $\log f_\theta(\mathbf{x}_i)$, whereas MEX can naturally implement both log-sum-exp and sum-pooling ($\lim_{\beta \to 0} \text{MEX}_\beta(\mathbf{x}, \mathbf{0}) = \text{mean}_j\{x_j\}$) in a numerically stable manner.

Not only are SimNets capable of correctly and efficiently implementing our networks, but they have already been demonstrated (Cohen et al. (2016)) to perform as well as state of the art convolutional networks on several image recognition benchmarks, and outperform them when computational resources are limited.

10. Naïve implementation of softmax is not numerically stable, as it involves storing $\alpha_i = \exp(\log \alpha_i)$ directly. This however can be easily corrected by defining $c := \max_i \log \alpha_i$, and computing $\log \sum_i \exp(\log \alpha_i - c) + c$. The result is identical, but now we only exponentiate negative numbers (no overflow), with at least one of these numbers equal to zero (no underflow).