Convolutional Rectifier Networks as Generalized Tensor Decompositions

Nadav Cohen
The Hebrew University of Jerusalem
cohennadav@cs.huji.ac.il

Amnon Shashua
The Hebrew University of Jerusalem
shashua@cs.huji.ac.il

Abstract

Convolutional rectifier networks, i.e. convolutional neural networks with rectified linear activation and max or average pooling, are the cornerstone of modern deep learning. However, despite their wide use and success, our theoretical understanding of the expressive properties that drive these networks is partial at best. On the other hand, we have a much firmer grasp of these issues in the world of arithmetic circuits. Specifically, it is known that convolutional arithmetic circuits possess the property of "complete depth efficiency", meaning 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 implemented (or even approximated) by a shallow network.

In this paper we describe a construction based on generalized tensor decompositions, that transforms convolutional arithmetic circuits into convolutional rectifier networks. We then use mathematical tools available from the world of arithmetic circuits to prove new results. First, we show that convolutional rectifier networks are universal with max pooling but not with average pooling. Second, and more importantly, we show that depth efficiency is weaker with convolutional rectifier networks than it is with convolutional arithmetic circuits. This leads us to believe that developing effective methods for training convolutional arithmetic circuits, thereby fulfilling their expressive potential, may give rise to a deep learning architecture that is provably superior to convolutional rectifier networks but has so far been overlooked by practitioners.

1. Introduction

Deep neural networks are repeatedly proving themselves to be extremely effective machine learning models, providing state of the art accuracies on a wide range of tasks (see [17, 9]). Arguably, the most successful deep learning architecture to date is that of convolutional neural networks (*ConvNets*, [16]), which prevails in the field of computer vision, and is recently being harnessed for many other application domains as well (*e.g.* [25, 31, 2]). Modern ConvNets are formed by stacking layers one after the other, where each layer consists of a linear convolutional operator followed by Rectified Linear Unit (*ReLU* [21]) activation ($\sigma(z) = \max\{0, z\}$), which in turn is followed by max or average pooling ($P\{c_j\} = \max\{c_j\}$ or $P\{c_j\} = \mathrm{mean}\{c_j\}$ respectively). Such models, which we refer to as *convolutional rectifier networks*, have driven the resurgence of deep learning ([15]), and represent the cutting edge of the ConvNet architecture ([28, 27]).

Despite their empirical success, and the vast attention they are receiving, our theoretical understanding of convolutional rectifier networks is partial at best. It is believed that they enjoy *depth efficiency*, *i.e.* that when allowed to go deep, such networks can implement with polynomial size computations that would require super-polynomial size if the networks were shallow. However, formal arguments that support this are scarce. It is unclear to what extent convolutional rectifier networks leverage depth efficiency, or more formally, what is the proportion of weight settings that would lead a deep network to implement a computation that cannot be efficiently realized by a shallow network. We refer to the most optimistic situation, where this takes place for all weight settings but a negligible (zero measure) set, as *complete depth efficiency*.

Compared to convolutional rectifier networks, our theoretical understanding of depth efficiency for arithmetic circuits, and in particular for convolutional arithmetic circuits, is much more developed. *Arithmetic circuits* (also known as Sum-Product Networks, [24]) 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. The depth efficiency of arithmetic circuits has been studied by the theoretical computer science community for the last five decades, long before the resurgence of deep learning. Although many problems in the area remain open, significant progress has been made over the years, making use of various mathematical tools. *Convolutional arithmetic circuits* form a specific sub-class of arithmetic circuits. Namely, these are ConvNets with linear activation ($\sigma(z) = z$) and product pooling ($P\{c_j\} = \prod c_j$). Recently, [5] analyzed convolutional arithmetic circuits through tensor decompositions, essentially proving, for the type of networks considered, that *depth efficiency holds completely*. Although convolutional arithmetic circuits are known to be equivalent to SimNets ([3]), a new deep learning architecture that has recently demonstrated promising empirical performance ([4]), they are fundamentally different from convolutional rectifier networks. Accordingly, the result established in [5] does not apply to the models most commonly used in practice.

In this paper we present a construction, based on the notion of *generalized tensor decompositions*, that transforms convolutional arithmetic circuits of the type described in [5] into convolutional rectifier networks. We then use the available mathematical tools from the world of arithmetic circuits to prove new results concerning the expressive power and depth efficiency of convolutional rectifier networks. Namely, we show that with ReLU activation, average pooling leads to loss of universality, whereas max pooling is universal but enjoys depth efficiency to a lesser extent than product pooling with linear activation (convolutional arithmetic circuits). These results indicate that from the point of view of expressive power and depth efficiency, convolutional arithmetic circuits (SimNets) have an advantage over the prevalent convolutional rectifier networks (ConvNets with ReLU activation and max or average pooling). This leads us to believe that developing effective methods for training convolutional arithmetic circuits, thereby fulfilling their expressive potential, may give rise to a deep learning architecture that is provably superior to convolutional rectifier networks but has so far been overlooked by practitioners.

The remainder of the paper is organized as follows. In sec. 2 we review existing works relating to depth efficiency of arithmetic circuits and networks with ReLU activation. Sec. 3 presents our definition of generalized tensor decompositions, followed by sec. 4 which employs this concept to frame convolutional rectifier networks. In sec. 5 we make use of this framework for an analysis of the expressive power and depth efficiency of such networks. Finally, sec. 6 concludes.

2. Related Work

The literature on the computational complexity of arithmetic circuits is far too wide to cover here, dating back over five decades. Although many of the fundamental questions in the field remain open, significant progress has been made over the years, developing and employing a vast share of mathematical tools from branches of geometry, algebra, analysis, combinatorics, and more. We refer the interested reader to [26] for a survey written in 2010, and mention here the more recent works [7] and [19] studying depth efficiency of arithmetic circuits in the context of deep learning (Sum-Product Networks). Compared to arithmetic circuits, the literature on depth efficiency of neural networks with ReLU activation is far less developed, primarily since these models were only introduced several years ago ([21]). There have been some notable works on this line, but these employ dedicated mathematical machinery, not making use of the plurality of available tools from the world of arithmetic circuits. [22] and [20] use combinatorial arguments to characterize the maximal number of linear regions in functions generated by ReLU networks, thereby establishing existence of depth efficiency. [30] uses semi-algebraic geometry to analyze the number of oscillations in functions realized by neural networks with semi-algebraic activations, ReLU in particular. The fundamental result proven in [30] is the existence, for every $k \in \mathbb{N}$, of functions realizable by networks with $\Theta(k^3)$ layers and $\Theta(1)$ nodes per layer, which cannot be approximated by networks with $\mathcal{O}(k)$ layers unless these are exponentially large (have $\Omega(2^k)$ nodes). The work in [8] makes use of Fourier analysis to show existence of functions that are efficiently computable by depth-3 networks, yet require exponential size in order to be approximated by depth-2 networks. The result applies to various activations, including ReLU. [23] also compares the computational abilities of deep *vs.* shallow networks under different activations that include ReLU. However, the complexity measure considered in [23] is the VC dimension, whereas our interest lies in network size.

None of the analyses above account for convolutional networks [1], thus they do not apply to the deep learning architecture most commonly used in practice. Recently, [5] introduced convolutional arithmetic circuits, which may be viewed as ConvNets with linear activation and product pooling. These networks were shown to correspond to hierarchical tensor decompositions (see [11]). Tools from linear algebra, functional analysis and measure theory were then employed to prove that the networks are universal, and exhibit *complete depth efficiency*. Although similar in structure, convolutional arithmetic circuits are inherently different from convolutional rectifier networks (ConvNets with ReLU activation and max or average pooling). Accordingly, the analysis carried out in [5] does not apply to the networks at the forefront of deep learning.

Closing the gap between the networks analyzed in [5] and convolutional rectifier networks is the topic of this paper. We achieve this by generalizing tensor decompositions, thereby opening the door to mathematical machinery as used in [5], harnessing it to analyze, for the first time, the depth efficiency of convolutional rectifier networks.

[1] By this we mean that in all analyses, the deep networks shown to benefit from depth (*i.e.* to realize functions that require super-polynomial size from shallow networks) are not ConvNets.

3. Generalized Tensor Decompositions

We begin by establishing basic tensor-related terminology and notations. [2] For our purposes, a *tensor* is simply a multi-dimensional array:

$$\mathcal{A}_{d_1,\ldots,d_N} \in \mathbb{R} \quad, d_i \in [M_i]$$

The *order* of a tensor is defined to be the number of indexing entries in the array, which are referred to as *modes*. The term *dimension* stands for the number of values an index can take in a particular mode. For example, the tensor $\mathcal{A}$ 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 $\mathbb{R}^{M_1 \times \cdots \times M_N}$.

The fundamental operator in tensor analysis is the *tensor product*, denoted by $\otimes$. It is an operator that intakes two tensors $\mathcal{A} \in \mathbb{R}^{M_1 \times \cdots \times M_P}$ and $\mathcal{B} \in \mathbb{R}^{M_{P+1} \times \cdots \times M_{P+Q}}$ (orders P and Q respectively), and returns a tensor $\mathcal{A} \otimes \mathcal{B} \in \mathbb{R}^{M_1 \times \cdots \times M_{P+Q}}$ (order $P+Q$) defined by:

$$(\mathcal{A} \otimes \mathcal{B})_{d_1,\ldots,d_{P+Q}} = \mathcal{A}_{d_1,\ldots,d_P} \cdot \mathcal{B}_{d_{P+1},\ldots,d_{P+Q}} \qquad (1)$$

Notice that in the case $P = Q = 1$, the tensor product reduces to the standard outer product between vectors, *i.e.* if $\mathbf{u} \in \mathbb{R}^{M_1}$ and $\mathbf{v} \in \mathbb{R}^{M_2}$, then $\mathbf{u} \otimes \mathbf{v}$ is no other than the rank-1 matrix $\mathbf{u}\mathbf{v}^\top \in \mathbb{R}^{M_1 \times M_2}$.

Tensor decompositions (see [14] for a survey) may be viewed as schemes for expressing tensors using tensor products and weighted sums. For example, suppose we have a tensor $\mathcal{A} \in \mathbb{R}^{M_1 \times \cdots \times M_N}$ given by:

$$\mathcal{A} = \sum_{j_1 \ldots j_N = 1}^{J} c_{j_1 \ldots j_N} \cdot \mathbf{a}^{j_1,1} \otimes \cdots \otimes \mathbf{a}^{j_N,N}$$

This expression is known as a Tucker decomposition, parameterized by the coefficients $\{c_{j_1 \ldots j_N} \in \mathbb{R}\}_{j_1 \ldots j_N \in [J]}$ and vectors $\{\mathbf{a}^{j,i} \in \mathbb{R}^{M_i}\}_{i \in [N], j \in [J]}$. It is different from the *CP* (rank-1) and *Hierarchical Tucker* decompositions our analysis will rely upon (see sec. 4). All decompositions however are closely related, specifically in the fact that they are based on iterating between tensor products and weighted sums.

Our construction and analysis are facilitated by generalizing the tensor product, which in turn generalizes tensor decompositions. For an associative and commutative binary operator g, *i.e.* a function $g : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ such that $\forall a, b, c \in \mathbb{R} : g(g(a,b),c) = g(a, g(b,c))$ and $\forall a, b \in \mathbb{R} : g(a,b) = g(b,a)$, the *generalized tensor product* $\otimes_g$, an operator intaking tensors $\mathcal{A} \in \mathbb{R}^{M_1 \times \cdots \times M_P}, \mathcal{B} \in \mathbb{R}^{M_{P+1} \times \cdots \times M_{P+Q}}$ and returning tensor $\mathcal{A} \otimes_g \mathcal{B} \in \mathbb{R}^{M_1 \times \cdots \times M_{P+Q}}$, is defined as follows:

$$(\mathcal{A} \otimes_g \mathcal{B})_{d_1,\ldots,d_{P+Q}} = g(\mathcal{A}_{d_1,\ldots,d_P}, \mathcal{B}_{d_{P+1},\ldots,d_{P+Q}}) \quad (2)$$

Generalized tensor decompositions are simply obtained by plugging in the generalized tensor product $\otimes_g$ in place of the standard tensor product $\otimes$.

4. From Networks to Tensors

The ConvNet architecture analyzed in this paper is presented in fig. 1. The input to a network, denoted X, is composed of N *patches* $\mathbf{x}_1 \ldots \mathbf{x}_N \in \mathbb{R}^s$. For example, X could represent a 32-by-32 RGB image through 5×5 regions crossing the three color bands, in which case, assuming a patch is taken for every pixel (boundaries padded), we have $N = 1024$ and $s = 75$. The first layer in a network is referred to as *representation*, and may be thought of as a generalized convolution. Namely, it consists of applying M *representation functions* $f_{\theta_1} \ldots f_{\theta_M} : \mathbb{R}^s \to \mathbb{R}$ to all patches of the input, thereby creating M feature maps. In the case where the representation functions are standard neurons, *i.e.* $f_{\theta_d}(\mathbf{x}) = \sigma(\mathbf{w}_d^\top \mathbf{x} + b_d)$ for parameters $\theta_d = (\mathbf{w}_d, b_d) \in \mathbb{R}^s \times \mathbb{R}$ and some chosen activation $\sigma(\cdot)$, we obtain a conventional convolutional layer. More elaborate settings are also possible, for example modeling the representation as a cascade of convolutional layers with pooling in-between.

Following the representation, a network includes L hidden layers indexed by $l = 0 \ldots L-1$. Each hidden layer l begins with a 1×1 *conv* operator, which is simply a 3D convolution with r_l channels and receptive field 1×1 followed by point-wise activation $\sigma(\cdot)$. We allow the convolution to operate without weight sharing, in which case 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 layer (see [29]). We refer to it as the *unshared* case, in contrast to the *shared* case that gives rise to a standard 1×1 convolution. The second (last) operator in a hidden layer is spatial pooling. Feature maps generated by 1×1 conv are decimated, by applying the pooling operator $P(\cdot)$ (*e.g.* max or average) to non-overlapping 2D windows that cover the spatial extent. The last of the L hidden layers ($l = L-1$) reduces feature maps to singletons (its pooling operator is global), creating a vector of dimension r_{L-1}. This vector is mapped into Y network outputs through a final dense linear layer.

Altogether, the architectural parameters of a ConvNet are the type of representation functions (f_{θ_d}), the pooling window sizes (which in turn determine the number of hidden layers L), the setting of conv weights as shared or unshared, the number of channels in each layer (M for representation, $r_0 \ldots r_{L-1}$ for hidden layers, Y for output), and the

[2] The definitions we give are actually concrete special cases of more abstract algebraic definitions as given in [11]. We limit the discussion to these special cases since they suffice for our needs and are easier to grasp.

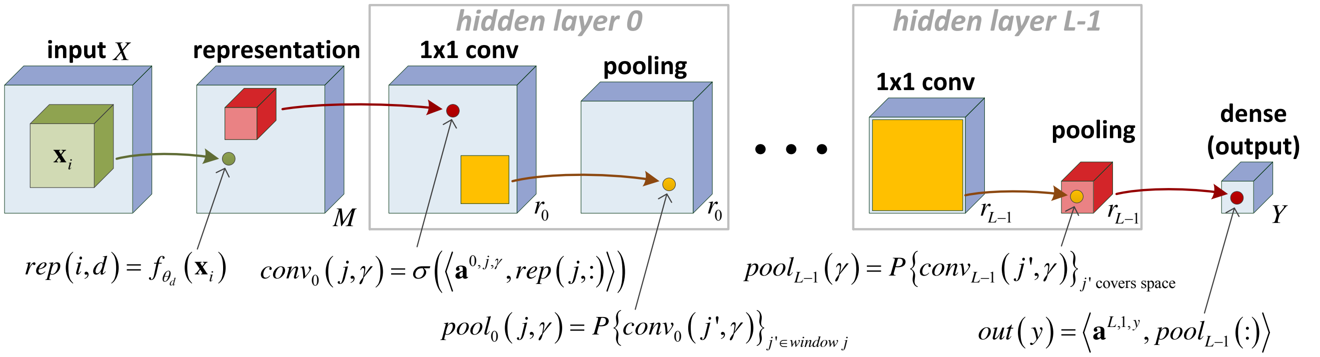
hidden layer 0
hidden layer L-1
input X
representation
1x1 conv
pooling
$\mathbf{x}_i$
M
r_0
r_0
• • •
1x1 conv
pooling
dense (output)
r_{L-1}
r_{L-1}
Y
$rep(i,d) = f_{\theta_d}(\mathbf{x}_i)$
$conv_0(j,\gamma) = \sigma(\langle \mathbf{a}^{0,j,\gamma}, rep(j,:)\rangle)$
$pool_0(j,\gamma) = P\{conv_0(j',\gamma)\}_{j' \in window\ j}$
$pool_{L-1}(\gamma) = P\{conv_{L-1}(j',\gamma)\}_{j'\text{ covers space}}$
$out(y) = \langle \mathbf{a}^{L,1,y}, pool_{L-1}(:)\rangle$

Figure 1. ConvNet architecture analyzed in this paper. The representation convolves functions $f_{\theta_d}(\cdot)$ across input patches (a standard convolutional layer is obtained by setting $f_{\theta_d}(\mathbf{x}) = \sigma(\mathbf{w}_d^\top \mathbf{x} + b_d)$). L hidden layers follow, each comprising 1×1 convolution (optionally without spatial weight sharing) followed by activation $\sigma(\cdot)$ and pooling $P(\cdot)$. The last hidden layer reduces feature maps to singletons, and these are mapped to network outputs through a dense linear layer. Convolutional arithmetic circuits as analyzed in [5] correspond to linear activation ($\sigma(z) = z$) and product pooling ($P\{c_j\} = \prod c_j$). Convolutional rectifier networks are obtained by setting ReLU activation ($\sigma(z) = \max\{0, z\}$) and max or average pooling ($P\{c_j\} = \max\{c_j\}$ or $P\{c_j\} = \text{mean}\{c_j\}$ respectively). Best viewed in color.

choice of activation and pooling operators ($\sigma(\cdot)$ and $P(\cdot)$ respectively). Given these architectural parameters, the learnable parameters of a network are the representation weights (θ_d), the conv weights ($\mathbf{a}^{l,j,\gamma}$ for hidden layer l, location j and channel γ in the unshared case; $\mathbf{a}^{l,\gamma}$ for hidden layer l and channel γ in the shared case), and the output weights ($\mathbf{a}^{L,1,y}$).

The choice of activation and pooling operators determines the type of network we arrive at. For linear activation ($\sigma(z) = z$) and product pooling ($P\{c_j\} = \prod c_j$) we get a convolutional arithmetic circuit as analyzed in [5]. For ReLU activation ($\sigma(z) = \max\{0, z\}$) and max or average pooling ($P\{c_j\} = \max\{c_j\}$ or $P\{c_j\} = \text{mean}\{c_j\}$ respectively) we get the commonly used convolutional rectifier networks, on which we focus in this paper.

In terms of pooling window sizes and network depth, we direct our attention to two special cases representing the extremes. The first is a shallow network that includes global pooling in its single hidden layer – see illustration in fig. 2. The second is the deepest possible network, in which all pooling windows cover only two entries, resulting in $L = \log_2 N$ hidden layers. These ConvNets, which we refer to as *shallow* and *deep* respectively, will be shown to correspond to canonical tensor decompositions. It is for this reason, and for simplicity of presentation, that we focus on these special cases. One may just as well consider networks of intermediate depths with different pooling window sizes, and that would correspond to other, non-standard, tensor decompositions. The analysis carried out in sec. 5 can easily be adapted to such networks.

In a classification setting, the Y outputs of a network correspond to different categories, and prediction follows the output with highest activation. Specifically, if we denote by $h_y(\cdot)$ the mapping from network input to output y, the predicted label for the instance $X = (\mathbf{x}_1, \ldots, \mathbf{x}_N) \in (\mathbb{R}^s)^N$ is determined by the following classification rule:

$$\hat{y} = \operatorname*{argmax}_{y \in [Y]} h_y(X)$$

We refer to h_y as the *score function* of category y. Score functions are studied in this paper through the notion of *grid tensors*. Given fixed vectors $\mathbf{x}^{(1)} \ldots \mathbf{x}^{(M)} \in \mathbb{R}^s$, referred to as *templates*, the grid tensor of h_y, denoted $\mathcal{A}(h_y)$, is defined to be the tensor of order N and dimension M in each mode whose entries are given by:

$$\mathcal{A}(h_y)_{d_1 \ldots d_N} = h_y(\mathbf{x}^{(d_1)}, \ldots, \mathbf{x}^{(d_N)}) \tag{3}$$

That is to say, the grid tensor of a score function under M templates $\mathbf{x}^{(1)} \ldots \mathbf{x}^{(M)}$, is a tensor of order N and dimension M in each mode, holding score values on the exponentially large grid of instances $\{X_{d_1 \ldots d_N} := (\mathbf{x}^{(d_1)}, \ldots, \mathbf{x}^{(d_N)}) : d_1 \ldots d_N \in [M]\}$. Before heading on to our analysis of grid tensors generated by ConvNets, to simplify notation, we define $F \in \mathbb{R}^{M \times M}$ to be the matrix holding the values taken by the representation functions $f_{\theta_1} \ldots f_{\theta_M} : \mathbb{R}^s \to \mathbb{R}$ on the selected templates $\mathbf{x}^{(1)} \ldots \mathbf{x}^{(M)} \in \mathbb{R}^s$:

$$F := \begin{bmatrix} f_{\theta_1}(\mathbf{x}^{(1)}) & \cdots & f_{\theta_M}(\mathbf{x}^{(1)}) \\ \vdots & \ddots & \vdots \\ f_{\theta_1}(\mathbf{x}^{(M)}) & \cdots & f_{\theta_M}(\mathbf{x}^{(M)}) \end{bmatrix} \tag{4}$$

To express the grid tensor of a ConvNet's score function using generalized tensor decompositions (see sec. 3), we set the underlying function $g : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ to be the *activation-pooling* operator defined by:

$$g(a, b) = P(\sigma(a), \sigma(b)) \tag{5}$$

where $\sigma(\cdot)$ and $P(\cdot)$ are the network's activation and pooling functions, respectively. Notice that the activation-pooling operator meets the associativity and commutativity requirements under product pooling with linear activation ($g(a,b) = a{\cdot}b$), and under max pooling with ReLU activation ($g(a,b) = \max\{a,b,0\}$). To account for the case of average pooling with ReLU activation, which a-priori leads to a non-associative activation-pooling operator, we simply replace average by sum, *i.e.* we analyze sum pooling with ReLU activation ($g(a,b) = \max\{a,0\} + \max\{b,0\}$), which from the point of view of expressiveness is completely equivalent to average pooling with ReLU activation (scaling factors can always blend in to linear weights that follow pooling).

With the activation-pooling operator g in place, it is straightforward to see that the grid tensor of h_y^S – a score function generated by the shallow ConvNet (fig. 2), is given by the following generalized tensor decomposition:

$$\mathcal{A}\left(h_y^S\right) = \sum_{z=1}^{Z} a_z^y \cdot \left(F\mathbf{a}^{z,1}\right) \otimes_g \cdots \otimes_g \left(F\mathbf{a}^{z,N}\right) \qquad (6)$$

Z here is the number of channels in the network's single hidden layer, $\{\mathbf{a}^{z,i} \in \mathbb{R}^M\}_{z\in[Z],i\in[N]}$ are the weights in the hidden conv, and $\mathbf{a}^y \in \mathbb{R}^Z$ are the weights of output y. The factorization in eq. 6 generalizes the classic CP (CANDECOMP/PARAFAC) decomposition (see [14] for a historic survey), and we accordingly refer to it as the *generalized CP decomposition*.

Turning to the deep ConvNet (fig. 1 with size-2 pooling windows and $L = \log_2 N$ hidden layers), the grid tensor of its score function h_y^D is given by the hierarchical generalized tensor decomposition below:

$$\begin{aligned}
\phi^{1,j,\gamma} &= \sum_{\alpha=1}^{r_0} a_\alpha^{1,j,\gamma} \left(F\mathbf{a}^{0,2j-1,\alpha}\right) \otimes_g \left(F\mathbf{a}^{0,2j,\alpha}\right) \\
&\cdots \\
\phi^{l,j,\gamma} &= \sum_{\alpha=1}^{r_{l-1}} a_\alpha^{l,j,\gamma} \underbrace{\phi^{l-1,2j-1,\alpha}}_{\text{order } 2^{l-1}} \otimes_g \underbrace{\phi^{l-1,2j,\alpha}}_{\text{order } 2^{l-1}} \\
&\cdots \\
\phi^{L-1,j,\gamma} &= \sum_{\alpha=1}^{r_{L-2}} a_\alpha^{L-1,j,\gamma} \underbrace{\phi^{L-2,2j-1,\alpha}}_{\text{order } \frac{N}{4}} \otimes_g \underbrace{\phi^{L-2,2j,\alpha}}_{\text{order } \frac{N}{4}} \\
\mathcal{A}\left(h_y^D\right) &= \sum_{\alpha=1}^{r_{L-1}} a_\alpha^{L,1,y} \underbrace{\phi^{L-1,1,\alpha}}_{\text{order } \frac{N}{2}} \otimes_g \underbrace{\phi^{L-1,2,\alpha}}_{\text{order } \frac{N}{2}} \qquad (7)
\end{aligned}$$

$r_0 \ldots r_{L-1} \in \mathbb{N}$ here are the number of channels in the network's hidden layers, $\{\mathbf{a}^{0,j,\gamma} \in \mathbb{R}^M\}_{j\in[N],\gamma\in[r_0]}$ are the weights in the first hidden conv, $\{\mathbf{a}^{l,j,\gamma} \in \mathbb{R}^{r_{l-1}}\}_{l\in[L-1],j\in[N/2^l],\gamma\in[r_l]}$ are the weights in the following hidden convs, and $\mathbf{a}^{L,1,y} \in \mathbb{R}^{r_{L-1}}$ are the weights of output y. The factorization in eq. 7 generalizes the Hierarchical Tucker decomposition introduced in [10], and is accordingly referred to as the *generalized HT decomposition*.

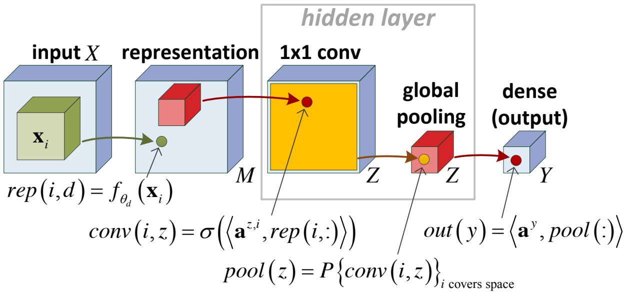
hidden layer
input X
representation
1x1 conv
global pooling
dense (output)
$\mathbf{x}_i$
M
Z
Z
Y
$rep(i,d) = f_{\theta_d}(\mathbf{x}_i)$
$conv(i,z) = \sigma\left(\left\langle \mathbf{a}^{z,i}, rep(i,:)\right\rangle\right)$
$out(y) = \left\langle \mathbf{a}^y, pool(:)\right\rangle$
$pool(z) = P\left\{conv(i,z)\right\}_{i \text{ covers space}}$

Figure 2. Shallow ConvNet with global pooling in its single hidden layer. Best viewed in color.

To conclude this section, we presented a ConvNet architecture (fig. 1) whose activation and pooling operators may be chosen to realize convolutional arithmetic circuits (linear activation, product pooling) or convolutional rectifier networks (ReLU activation, max/average pooling). We then defined the grid tensor of a network's score function as a tensor holding function values on an exponentially large grid whose points are sequences with elements chosen from a finite set of templates. Then, we saw that the grid tensor of a shallow ConvNet (fig. 2) is given by the generalized CP decomposition (eq. 6), and a grid tensor of a deep ConvNet (fig. 1 with $L = \log_2 N$) is given by the generalized HT decomposition (eq. 7). In the next section we utilize the connection between ConvNets and generalized tensor decompositions for an analysis of the expressive power and depth efficiency of convolutional rectifier networks.

5. Capacity Analysis

In this section we analyze score functions expressible by the shallow and deep ConvNets (fig. 2, and fig. 1 with $L = \log_2 N$, respectively) under ReLU activation with max or average pooling (convolutional rectifier networks), comparing these settings against linear activation with product pooling (convolutional arithmetic circuits). Score functions are analyzed through grid tensors (eq. 3), represented by the generalized tensor decompositions established in the previous section: the generalized CP decomposition (eq. 6) corresponding to the shallow network, and the generalized HT decomposition (eq. 7) corresponding to the deep network. The analysis is organized as follows. In sec. 5.1 we present preliminary material required in order to follow our proofs. Sec. 5.2 discusses templates and representation functions, which form the bridge between score functions and generalized tensor decompositions. Sec. 5.3 presents matricization – a technical tool that facilitates the use of matrix theory for analyzing generalized tensor decompositions. The actual analysis begins in sec. 5.4, where we address the question of universality, *i.e.* of the ability of networks to realize any score function when their size is unlimited. This is followed by sec. 5.5 which studies depth efficiency, namely, situations where functions efficiently computable by deep networks require shallow networks to have super-polynomial size. Finally, sec. 5.6 analyzes the case of coefficient sharing, in which the conv operators of our networks are standard convolutions (as opposed to the more general locally-connected layers).

5.1. Preliminaries

For evaluating the completeness of depth efficiency, and for other purposes as well, we are often interested in the "volume" of sets in a Euclidean space, or more formally, in their Lebesgue measure. While an introduction to Lebesgue measure theory is beyond the scope of this paper (the interested reader is referred to [13]), we restate here several concepts and results our proofs will rely upon. A zero measure set can intuitively be thought of as having zero volume. A union of countably many zero measure sets is itself a zero measure set. If we randomize a point in space by some continuous distribution, the probability of hitting a zero measure set is always zero. A useful fact (proven in [1] for example) is that the zero set of a polynomial, *i.e.* the set of points on which a polynomial vanishes, is either the entire space (when the polynomial in question is the zero polynomial), or it must have measure zero. An open set always has positive measure, and when a point in space is drawn by a continuous distribution with non-vanishing continuous probability density function, the probability of hitting such a set is positive.

Apart from measure theory, we will also be using tools from the field of tensor analysis. Here too, a full introduction to the topic is beyond our scope (we refer the interested reader to [11]), and we only list some concepts and results that will be used. First, a fact that relates to abstract tensor products over function spaces is the following. If $f_{\theta_1}\ldots f_{\theta_M} : \mathbb{R}^s \to \mathbb{R}$ are linearly independent functions, then the product functions $\{(\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(M)}) \mapsto \prod_{i=1}^{M} f_{\theta_{d_i}}(\mathbf{x}^{(i)})\}_{d_1\ldots d_M\in[M]}$ from $(\mathbb{R}^s)^M$ to $\mathbb{R}$ are linearly independent as well. Back to tensors as we have defined them (multi-dimensional arrays), a very important concept is that of *rank*, which for order-2 tensors reduces to the standard notion of matrix rank. A tensor is said to have rank 1 if it may be written as a tensor product between non-zero vectors ($\mathcal{A} = \mathbf{v}^1 \otimes \cdots \otimes \mathbf{v}^N$). The rank of a general tensor is defined to be the minimal number of rank-1 tensors that may be summed up to produce it. A useful fact is that the rank of an order-N tensor with dimension M_i in each mode $i \in [N]$, is no greater than $\prod_i M_i / \max_i M_i$. On the other hand, all such tensors, besides a zero measure set, have rank equal to at least $\min\{\prod_{i\ even} M_i, \prod_{i\ odd} M_i\}$. As in the special case of matrices, the rank is sub-additive, *i.e.* $rank(\mathcal{A}+\mathcal{B}) \leq rank(\mathcal{A}) + rank(\mathcal{B})$ for any tensors $\mathcal{A},\mathcal{B}$ of matching dimensions. The rank is sub-multiplicative w.r.t. the tensor product, *i.e.* $rank(\mathcal{A}\otimes\mathcal{B}) \leq rank(\mathcal{A}) \cdot rank(\mathcal{B})$ for any tensors $\mathcal{A},\mathcal{B}$. Finally, we use the fact that permuting the modes of a tensor does not alter its rank.

5.2. Templates and Representation Functions

The expressiveness of our ConvNets obviously depends on the possible forms that may be taken by the representation functions $f_{\theta_1}\ldots f_{\theta_M} : \mathbb{R}^s \to \mathbb{R}$. For example, if representation functions are limited to be constant, the ConvNets can only realize constant score functions. We denote by $\mathcal{F} := \{f_\theta : \mathbb{R}^s \to \mathbb{R} : \theta \in \Theta\}$ the parametric family from which representation functions are chosen, and make two mild assumptions on this family:

- **Continuity**: $f_\theta(\mathbf{x})$ is continuous w.r.t. both θ and $\mathbf{x}$.
- **Non-degeneracy**: For any $\mathbf{x}^{(1)}\ldots\mathbf{x}^{(M)} \in \mathbb{R}^s$ such that $\mathbf{x}_i \neq \mathbf{x}_j\ \forall i{\neq}j$, there exist $f_{\theta_1}\ldots f_{\theta_M} \in \mathcal{F}$ for which the matrix F defined in eq. 4 is non-singular.

Both of the assumptions above are met for most reasonable choices of $\mathcal{F}$. In particular, non-degeneracy holds when representation functions are standard neurons:

Claim 1. *The parametric family:*

$$\mathcal{F} = \left\{f_\theta(\mathbf{x}) = \sigma(\mathbf{w}^\top\mathbf{x} + b) : \theta = (\mathbf{w}, b) \in \mathbb{R}^s \times \mathbb{R}\right\} \tag{8}$$

where $\sigma(\cdot)$ is any sigmoidal activation [3] *or the ReLU activation, meets the non-degeneracy condition (*i.e. *for any distinct $\mathbf{x}^{(1)}\ldots\mathbf{x}^{(M)} \in \mathbb{R}^s$ there exist $f_{\theta_1}\ldots f_{\theta_M} \in \mathcal{F}$ such that the matrix F defined in eq. 4 is non-singular).*

Proof. We first show that given distinct $\mathbf{x}^{(1)}\ldots\mathbf{x}^{(M)} \in \mathbb{R}^s$, there exists a vector $\mathbf{w} \in \mathbb{R}^s$ such that $\mathbf{w}^\top\mathbf{x}^{(i)} \neq \mathbf{w}^\top\mathbf{x}^{(j)}$ for all $1{\leq}i < j{\leq}M$. $\mathbf{w}$ satisfies this condition if it is not perpendicular to any of the finitely many non-zero vectors $\{\mathbf{x}^{(i)} - \mathbf{x}^{(j)} : 1{\leq}i < j{\leq}M\}$. If for every $1{\leq}i < j{\leq}M$ we denote by $P^{(i,j)} \subset \mathbb{R}^s$ the set of points perpendicular to $\mathbf{x}^{(i)} - \mathbf{x}^{(j)}$, we obtain that $\mathbf{w}$ satisfies the desired condition if it does not lie in the union $\bigcup_{1\leq i<j\leq M} P^{(i,j)}$. Each $P^{(i,j)}$ is the zero set of a non-zero polynomial, and in particular has measure zero. The finite union $\bigcup_{1\leq i<j\leq M} P^{(i,j)}$ thus has measure zero as well, and accordingly cannot cover the entire space. This implies that $\mathbf{w} \in \mathbb{R}^s \setminus \bigcup_{1\leq i<j\leq M} P^{(i,j)}$ indeed exists.

Assume without loss of generality $\mathbf{w}^\top\mathbf{x}^{(1)} < \ldots < \mathbf{w}^\top\mathbf{x}^{(M)}$. We may then choose $b_1\ldots b_M \in \mathbb{R}$ such that $-\mathbf{w}^\top\mathbf{x}^{(M)} < b_M < \ldots < -\mathbf{w}^\top\mathbf{x}^{(1)} < b_1$. For $i,j \in [M]$, $\mathbf{w}^\top\mathbf{x}^{(i)} + b_j$ is positive when $j{\leq}i$ and negative when $j > i$.

[3] $\sigma(\cdot)$ is sigmoidal if it is monotonic with $\lim_{z\to-\infty}\sigma(z) = c$ and $\lim_{z\to+\infty}\sigma(z) = C$ for some $c{\neq}C$ in $\mathbb{R}$.

Therefore, if $\sigma(\cdot)$ is chosen as the ReLU activation, defining $f_{\theta_j}(\mathbf{x}) = \sigma(\mathbf{w}^\top\mathbf{x} + b_j)$ for every $j \in [M]$ gives rise to a matrix F (eq. 4) that is lower triangular with non-zero values on its diagonal. This proves the desired result for the case of ReLU activation.

Consider now the case of sigmoidal activation, where $\sigma(\cdot)$ is monotonic with $\lim_{z\to-\infty}\sigma(z) = c$ and $\lim_{z\to+\infty}\sigma(z) = C$ for some $c{\neq}C$ in $\mathbb{R}$. Letting $\mathbf{w} \in \mathbb{R}^s$ and $b_1 \ldots b_M \in \mathbb{R}$ be as above, we introduce a scaling factor $\alpha > 0$, and define $f_{\theta_j}(\mathbf{x}) = \sigma(\alpha\mathbf{w}^\top\mathbf{x} + \alpha b_j)$ for every $j \in [M]$. It is not difficult to see that as $\alpha \to +\infty$, the matrix F tends closer and closer to a matrix holding C on and below its diagonal, and c elsewhere. The latter matrix is non-singular, and in particular has non-zero determinant $d \neq 0$. The determinant of F converges to d as $\alpha \to +\infty$, so for large enough α, F is non-singular. □

Non-degeneracy means that given distinct templates, one may choose representation functions for which F is non-singular. We may as well consider the opposite situation, where we are given representation functions, and would like to choose templates leading to non-singular F. Apparently, so long as the representation functions are linearly independent, this is always possible:

Claim 2. *Let $f_{\theta_1} \ldots f_{\theta_M} : \mathbb{R}^s \to \mathbb{R}$ be any linearly independent continuous functions. Then, there exist $\mathbf{x}^{(1)} \ldots \mathbf{x}^{(M)} \in \mathbb{R}^s$ such that F (eq. 4) is non-singular.*

Proof. We may view the determinant of F (eq. 4) as a function of $(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)})$:

$$\det F(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)}) = \sum_{\delta\in S_M} sign(\delta) \prod_{i=1}^{M} f_{\theta_{\delta(i)}}(\mathbf{x}^{(i)})$$

where S_M stands for the permutation group on $[M]$, and $sign(\delta) \in \{\pm1\}$ is the sign of the permutation δ. This in particular shows that $\det F(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)})$ is a non-zero linear combination of the product functions $\{(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)}) \mapsto \prod_{i=1}^{M} f_{\theta_{d_i}}(\mathbf{x}^{(i)})\}_{d_1 \ldots d_M \in [M]}$. Since these product functions are linearly independent (see sec. 5.1), $\det F(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)})$ cannot be the zero function. That is to say, there exist $\mathbf{x}^{(1)} \ldots \mathbf{x}^{(M)} \in \mathbb{R}^s$ such that $\det F(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)}) \neq 0$. □

As stated previously, the analysis carried out in this paper studies score functions expressible by ConvNets through the notion of grid tensors. The translation of score functions into grid tensors is facilitated by the choice of templates $\mathbf{x}^{(1)} \ldots \mathbf{x}^{(M)} \in \mathbb{R}^s$ (eq. 3). For general templates, the correspondence between score functions and grid tensors is not injective – a score function corresponds to a single grid tensor, but a grid tensor may correspond to more than one score function. We use the term *covering* to refer to templates leading to an injective correspondence, *i.e.* to a situation where two score functions associated with the same grid tensor are effectively identical. In other words, the templates $\mathbf{x}^{(1)} \ldots \mathbf{x}^{(M)}$ are covering if the value of score functions outside the exponentially large grid $\left\{X_{d_1 \ldots d_N} := (\mathbf{x}^{(d_1)}, \ldots, \mathbf{x}^{(d_N)}) : d_1 \ldots d_N \in [M]\right\}$ is irrelevant for classification. Some of the claims in our analysis will assume existence of covering templates (it will be stated explicitly when so). We argue in app. A that for structured compositional data (*e.g.* natural images), $M \in \Omega(100)$ suffices in order for this assumption to hold.

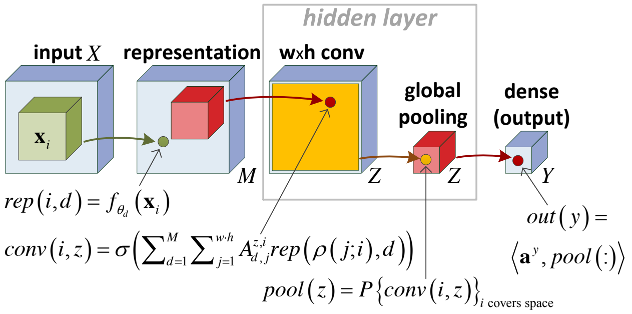
hidden layer
input X
representation
w×h conv
global pooling
dense (output)
$\mathbf{x}_i$
M
Z
Z
Y
$rep(i,d) = f_{\theta_d}(\mathbf{x}_i)$
$conv(i,z) = \sigma\left(\sum_{d=1}^{M}\sum_{j=1}^{w\cdot h} A_{d,j}^{z,i} rep(\rho(j;i),d)\right)$
$pool(z) = P\{conv(i,z)\}_{i \text{ covers space}}$
$out(y) = \langle\mathbf{a}^y, pool(:)\rangle$

Figure 3. Shallow ConvNet with conv receptive field expanded from 1×1 to $w{\times}h$. The weight vectors $\mathbf{a}^{i,z} \in \mathbb{R}^M$ have been replaced by matrices $A^{i,z} \in \mathbb{R}^{M\times w\cdot h}$, and we denote by $\rho(j;i)$ the spatial location of element j in the $w{\times}h$ window revolving around i. Best viewed in color.

5.3. Matricization

When analyzing grid tensors, we will often consider their arrangement as matrices. The *matricization* of a tensor $\mathcal{A}$, denoted $[\mathcal{A}]$, is its arrangement as a matrix with rows corresponding to odd modes and columns corresponding to even modes. Specifically, if $\mathcal{A} \in \mathbb{R}^{M_1\times\cdots\times M_N}$, and assuming for simplicity that the order N is even, the matricization $[\mathcal{A}] \in \mathbb{R}^{(M_1\cdot M_3\cdot\ldots\cdot M_{N-1})\times(M_2\cdot M_4\cdot\ldots\cdot M_N)}$ holds $\mathcal{A}_{d_1,\ldots,d_N}$ 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 $1 + \sum_{i=1}^{N/2}(d_{2i} - 1)\prod_{j=i+1}^{N/2} M_{2j}$.

The matrix analogy of the tensor product $\otimes$ (eq. 1) is called the *Kronecker product*, and is denoted by $\odot$. For $A \in \mathbb{R}^{M_1\times M_2}$ and $B \in \mathbb{R}^{N_1\times N_2}$, $A{\odot}B$ is the matrix in $\mathbb{R}^{M_1N_1\times M_2N_2}$ holding $A_{ij}B_{kl}$ in row index $(i-1)N_1 + k$ and column index $(j-1)N_2 + l$. The relation $[\mathcal{A} \otimes \mathcal{B}] = [\mathcal{A}] \odot [\mathcal{B}]$, where $\mathcal{A}$ and $\mathcal{B}$ are arbitrary tensors of even order, implies that the tensor and Kronecker products are indeed analogous, *i.e.* they represent the same operation under tensor and matrix viewpoints, respectively. We generalize the Kronecker product analogously to our generalization of the tensor product (eq. 2). For an associative and commutative binary operator $g(\cdot,\cdot)$, the *generalized Kronecker product* $\odot_g$, is an operator that intakes matrices $A \in \mathbb{R}^{M_1\times M_2}$ and $B \in \mathbb{R}^{N_1\times N_2}$, and returns a matrix

$A \odot_g B \in \mathbb{R}^{M_1 N_1 \times M_2 N_2}$ holding $g(A_{ij}, B_{kl})$ in row index $(i-1)N_1 + k$ and column index $(j-1)N_2 + l$. The relation between the tensor and Kronecker products holds for their generalized versions as well, *i.e.* $[\mathcal{A} \otimes_g \mathcal{B}] = [\mathcal{A}] \odot_g [\mathcal{B}]$ for arbitrary tensors $\mathcal{A}, \mathcal{B}$ of even order.

Equipped with the matricization operator $[\cdot]$ and the generalized Kronecker product $\odot_g$, we are now in a position to translate the generalized HT decomposition (eq. 7) to an expression for the matricization of a grid tensor generated by the deep ConvNet:

$$
\begin{aligned}
\phi^{1,j,\gamma} &= \sum_{\alpha=1}^{r_0} a_\alpha^{1,j,\gamma} (F\mathbf{a}^{0,2j-1,\alpha}) \otimes_g (F\mathbf{a}^{0,2j,\alpha}) \qquad (9)\\
&\cdots \\
\left[\phi^{l,j,\gamma}\right] &= \sum_{\alpha=1}^{r_{l-1}} a_\alpha^{l,j,\gamma} \underbrace{\left[\phi^{l-1,2j-1,\alpha}\right]}_{M^{2^{l-2}}\text{-by-}M^{2^{l-2}}} \odot_g \underbrace{\left[\phi^{l-1,2j,\alpha}\right]}_{M^{2^{l-2}}\text{-by-}M^{2^{l-2}}} \\
&\cdots \\
\left[\phi^{L-1,j,\gamma}\right] &= \sum_{\alpha=1}^{r_{L-2}} a_\alpha^{L-1,j,\gamma} \underbrace{\left[\phi^{L-2,2j-1,\alpha}\right]}_{M^{N/8}\text{-by-}M^{N/8}} \odot_g \underbrace{\left[\phi^{L-2,2j,\alpha}\right]}_{M^{N/8}\text{-by-}M^{N/8}} \\
\left[\mathcal{A}\left(h_y^D\right)\right] &= \sum_{\alpha=1}^{r_{L-1}} a_\alpha^{L,1,y} \underbrace{\left[\phi^{L-1,1,\alpha}\right]}_{M^{N/4}\text{-by-}M^{N/4}} \odot_g \underbrace{\left[\phi^{L-1,2,\alpha}\right]}_{M^{N/4}\text{-by-}M^{N/4}}
\end{aligned}
$$

We refer to this factorization as the *matricized generalized HT decomposition*. Notice that the expression above for $\phi^{1,j,\gamma}$ is the same as in the original generalized HT decomposition, as order-2 tensors need not be matricized.

For the matricization of a grid tensor generated by the shallow ConvNet, we translate the generalized CP decomposition (eq. 6) into the *matricized generalized CP decomposition*:

$$
\begin{gathered}
\left[\mathcal{A}\left(h_y^S\right)\right] = \qquad (10)\\
\sum_{z=1}^{Z} a_z^y \cdot \left((F\mathbf{a}^{z,1}) \odot_g (F\mathbf{a}^{z,3}) \odot_g \cdots \odot_g (F\mathbf{a}^{z,N-1}) \right) \odot_g \\
\left((F\mathbf{a}^{z,2}) \odot_g (F\mathbf{a}^{z,4}) \odot_g \cdots \odot_g (F\mathbf{a}^{z,N}) \right)^\top
\end{gathered}
$$

The matricized generalized CP and HT decompositions (eq. 10 and 9 respectively) will be used throughout our proofs to establish depth efficiency. This is generally done by providing a lower bound on $rank[\mathcal{A}(h_y^D)]$ – the rank of the deep ConvNet's matricized grid tensor, and an upper bound on $rank[\mathcal{A}(h_y^S)]$ – the rank of the shallow ConvNet's matricized grid tensor. The upper bound on $rank[\mathcal{A}(h_y^S)]$ will be linear in Z, and so requiring $\mathcal{A}(h_y^S) = \mathcal{A}(h_y^D)$, and in particular $rank[\mathcal{A}(h_y^S)] = rank[\mathcal{A}(h_y^D)]$, will give us a lower bound on Z. That is to say, we obtain a lower bound on the number of hidden channels in the shallow ConvNet, that must be met in order for this network to replicate a grid tensor generated by the deep ConvNet. Our analysis of depth efficiency is given in sec. 5.5. As a prerequisite, we first head on to sec. 5.4 to analyze universality.

5.4. Universality

Universality refers to the ability of a network to realize (or approximate) any function of choice when no restrictions are imposed on its size. It is well-known that fully-connected neural networks are universal under all types of non-linear activations typically used in practice, even if the number of hidden layers is restricted to one ([6, 12, 18]). To the best of our knowledge universality has never been studied in the context of convolutional rectifier networks. This is the purpose of the current subsection. Specifically, we analyze the universality of our shallow and deep ConvNets (fig. 2, and fig. 1 with $L = \log_2 N$, respectively) under ReLU activation and max or average pooling.

We begin by stating a result similar to that given in [5], according to which convolutional arithmetic circuits are universal:

Claim 3. *Assuming covering templates exist, with linear activation and product pooling the shallow ConvNet is universal (hence so is the deep).*

Proof. Let $\mathbf{x}^{(1)} \ldots \mathbf{x}^{(M)} \in \mathbb{R}^s$ be distinct covering templates, and $f_{\theta_1} \ldots f_{\theta_M}$ be representation functions for which F is invertible (non-degeneracy implies that such functions exist). With linear activation and product pooling the generalized CP decomposition (eq. 6) reduces to its standard version, which is known to be able to express any tensor when size is large enough (*e.g.* $Z \geq M^N$ suffices). The shallow ConvNet can thus realize any grid tensor on covering templates, precisely meaning that it is universal. As for the deep ConvNet, setting $r_0 = \cdots = r_{L-1} = Z$ and $a_\alpha^{l,j,\gamma} = \mathbb{1}[\alpha = \gamma]$, where $l \in [L-1]$ and $\mathbb{1}[\cdot]$ is the indicator function, reduces its decomposition (eq. 7) to that of the shallow ConvNet (eq. 6). This implies that all grid tensors realizable by the shallow ConvNet are also realizable by the deep ConvNet. □

Heading on to convolutional rectifier networks, the following claim tells us that max pooling leads to universality:

Claim 4. *Assuming covering templates exist, with ReLU activation and max pooling the shallow ConvNet is universal (hence so is the deep).*

Proof. The proof follows the same line as that of claim 3, except we cannot rely on the ability of the standard CP decomposition to realize any tensor of choice. Instead, we need to show that the generalized CP decomposition (eq. 6) with $g(a, b) = \max\{a, b, 0\}$ can realize any tensor, so long as Z is large enough. We will show that $Z \geq 2 \cdot M^N$ suffices.

For that, it is enough to consider an arbitrary indicator tensor, *i.e.* a tensor holding 1 in some entry and 0 in all other entries, and show that it can be expressed with $Z = 2$.

Let $\mathcal{A}$ be an indicator tensor of order N and dimension M in each mode, its active entry being $(d_1, \ldots, d_N)$. Denote by $\mathbf{1} \in \mathbb{R}^M$ the vector holding 1 in all entries, and for every $i \in [N]$, let $\bar{\mathbf{e}}_{d_i} \in \mathbb{R}^M$ be the vector holding 0 in entry d_i and 1 elsewhere. With the following weight settings, a generalized CP decomposition (eq. 6) with $g(a,b) = \max\{a,b,0\}$ and $Z = 2$ produces $\mathcal{A}$, as required:

- $a_1^y = 1, a_2^y = -1$
- $\mathbf{a}^{1,1} = \cdots = \mathbf{a}^{1,N} = \mathbf{1}$
- $\forall i \in [N] : \mathbf{a}^{2,i} = \bar{\mathbf{e}}_{d_i}$

□

At this point we encounter the first somewhat surprising result, according to which convolutional rectifier networks are not universal with average pooling:

Claim 5. *With ReLU activation and average pooling, both the shallow and deep ConvNets are not universal.*

Proof. Let $\mathbf{x}^{(1)} \ldots \mathbf{x}^{(M)} \in \mathbb{R}^s$ be any templates of choice, and consider grid tensors produced by the generalized CP and HT decompositions (eq. 6 and 7 respectively) with $g(a,b) = \max\{a,0\} + \max\{b,0\}$ (this corresponds to *sum* pooling and ReLU activation, but as stated in sec. 4, sum and average pooling are equivalent in terms of expressiveness). We will show that such grid tensors, when arranged as matrices, necessarily have low rank. This obviously implies that they cannot take on any value. Moreover, since the set of low rank matrices has zero measure in the space of all matrices (see sec. 5.1), the set of values that can be taken by the grid tensors has zero measure in the space of tensors with order N and dimension M in each mode.

In accordance with the above, we complete our proof by showing that with $g(a,b) = \max\{a,0\} + \max\{b,0\}$, the matricized generalized CP and HT decompositions (eq. 10 and 9 respectively) give rise to low-rank matrices. For the matricized generalized CP decomposition (eq. 10), corresponding to the shallow ConvNet, we have with $g(a,b) = \max\{a,0\} + \max\{b,0\}$:

$$\left[\mathcal{A}\left(h_y^S\right)\right] = \mathbf{v}\mathbf{1}^\top + \mathbf{1}\mathbf{u}^\top$$

where $\mathbf{1}$ is the vector in $\mathbb{R}^{M^{N/2}}$ holding 1 in all entries, and $\mathbf{v}, \mathbf{u} \in \mathbb{R}^{M^{N/2}}$ are defined as follows:

$$\begin{aligned}
\mathbf{v} &:= \sum_{z=1}^{Z} a_z^y \cdot \max\left\{(F\mathbf{a}^{z,1}) \odot_g \cdots \odot_g (F\mathbf{a}^{z,N-1}), 0\right\} \\
\mathbf{u} &:= \sum_{z=1}^{Z} a_z^y \cdot \max\left\{(F\mathbf{a}^{z,2}) \odot_g \cdots \odot_g (F\mathbf{a}^{z,N}), 0\right\}
\end{aligned}$$

Obviously the matrix $\left[\mathcal{A}\left(h_y^S\right)\right] \in \mathbb{R}^{M^{N/2} \times M^{N/2}}$ has rank 2 or less.

Turning to the matricized generalized HT decomposition (eq. 9), which corresponds to the deep ConvNet, we have with $g(a,b) = \max\{a,0\} + \max\{b,0\}$:

$$\left[\mathcal{A}\left(h_y^D\right)\right] = V \odot O + O \odot U$$

where $\odot$ is the standard Kronecker product (see definition in sec. 5.3), $O \in \mathbb{R}^{M^{N/4} \times M^{N/4}}$ is a matrix holding 1 in all entries, and the matrices $V, U \in \mathbb{R}^{M^{N/4} \times M^{N/4}}$ are given by:

$$\begin{aligned}
V &:= \sum_{\alpha=1}^{r_{L-1}} a_\alpha^{L,1,y} \max\left\{\left[\phi^{L-1,1,\alpha}\right], 0\right\} \\
U &:= \sum_{\alpha=1}^{r_{L-1}} a_\alpha^{L,1,y} \max\left\{\left[\phi^{L-1,2,\alpha}\right], 0\right\}
\end{aligned}$$

The rank of O is obviously 1, and since the Kronecker product multiplies ranks, *i.e.* $rank(A \odot B) = rank(A) \cdot rank(B)$ for any matrices A and B, we have that the rank of $\left[\mathcal{A}\left(h_y^D\right)\right] \in \mathbb{R}^{M^{N/2} \times M^{N/2}}$ is at most $2 \cdot M^{N/4}$. In particular, $\left[\mathcal{A}\left(h_y^D\right)\right]$ cannot have full rank. □

One may wonder if perhaps the non-universality of ReLU activation and average pooling is merely an artifact of the conv operator in our ConvNets having 1×1 receptive field. Apparently, as the following claim shows, expanding the receptive field does not remedy the situation, and indeed non-universality is an inherent property of convolutional rectifier networks with average pooling:

Claim 6. *Consider the network illustrated in fig. 3, obtained by expanding the conv receptive field in the shallow ConvNet from 1×1 to $w \times h$, where $w \cdot h < {}^N\!/_2 + 1 - \log_M N$ (conv windows cover less than half the feature maps that precede them). Such a network, when equipped with ReLU activation and average pooling, is not universal.*

Proof. Compare the original shallow ConvNet (fig. 2) to the shallow ConvNet with expanded receptive field that we consider in this claim (fig. 3). The original shallow ConvNet has 1×1 receptive field, with conv entry in location $i \in [N]$ and channel $z \in [Z]$ assigned through a cross-channel linear combination of the representation entries in the same location, the combination weights being $\mathbf{a}^{z,i} \in \mathbb{R}^M$. In the shallow ConvNet with receptive field expanded to $w \times h$, linear combinations span multiple locations. In particular, conv entry in location i and channel z is now assigned through a linear combination of the representation entries at all channels that lie inside a spatial window revolving around i. We denote by $\{\rho(j;i)\}_{j \in [w \cdot h]}$ the locations comprised by this window. More specifically, $\rho(j;i)$ is the j'th location in the window, and the linear weights that correspond to it are held in the j'th column of the weight matrix $A^{z,i} \in \mathbb{R}^{M \times w \cdot h}$. We assume for simplicity that conv windows stepping out of bounds encounter zero padding [4], and adhere to the convention under which indexing the row of a matrix with $d_{\rho(j;i)}$ produces zero when location j of window i steps out of bounds.

We are interested in the case of ReLU activation ($\sigma(z) = \max\{0, z\}$) and average pooling ($P\{c_j\} = \text{mean}\{c_j\}$). Under this setting, for any selected templates $\mathbf{x}^{(1)} \ldots \mathbf{x}^{(M)} \in \mathbb{R}^s$, the grid tensor of $h_y^{S(w \times h)}$ – network's y'th score function, is given by:

$$\mathcal{A}(h_y^{S(w \times h)})_{d_1,\ldots,d_N} = \sum_{i=1}^{N} \mathcal{B}^i_{d_{\rho(1;i)},\ldots,d_{\rho(w \cdot h;i)}}$$

where for every $i \in [N]$, $\mathcal{B}^i$ is a tensor of order $w{\cdot}h$ and dimension M in each mode, defined by:

$$\mathcal{B}^i_{c_1,\ldots,c_{w \cdot h}} = \sum_{z=1}^{Z} \frac{a_z^y}{N} \max\left\{ \sum_{j=1}^{w \cdot h} (FA^{z,i})_{c_j, j}, 0 \right\}$$

Let $\mathcal{O}$ be a tensor of order $N - w{\cdot}h$ and dimension M in each mode, holding 1 in all entries. We may write:

$$\mathcal{A}(h_y^{S(w \times h)}) = \sum_{i=1}^{N} p_i(\mathcal{B}^i \otimes \mathcal{O}) \qquad (11)$$

where for every $i \in [N]$, $p_i(\cdot)$ is an appropriately chosen operator that permutes the modes of an order-N tensor.

We now make use of some known facts related to tensor rank (see sec. 5.1), in order to show that eq. 11 is not universal, *i.e.* that there are many tensors which cannot be realized by $\mathcal{A}(h_y^{S(w \times h)})$. Being tensors of order $w{\cdot}h$ and dimension M in each mode, the ranks of $\mathcal{B}^1 \ldots \mathcal{B}^N$ are bounded above by $M^{w \cdot h - 1}$. Since $\mathcal{O}$ is an all-1 tensor, and since permuting modes does not alter rank, we have: $rank(p_i(\mathcal{B}^i \otimes \mathcal{O})) {\leq} M^{w \cdot h - 1}\ \forall i \in [N]$. Finally, from sub-additivity of the rank we get: $rank(\mathcal{A}(h_y^{S(w \times h)})) {\leq} N {\cdot} M^{w \cdot h - 1}$. Now, we know by assumption that $w{\cdot}h < {}^{N}\!/_{2} + 1 - \log_M N$, and this implies: $rank(\mathcal{A}(h_y^{S(w \times h)})) < M^{N/2}$. Since there exist tensors of order N and dimension M in each mode having rank at least $M^{N/2}$ (actually only a negligible set of tensors do not meet this), eq. 11 is indeed not universal. That is to say, the shallow ConvNet with conv receptive field expanded to $w{\times}h$ (fig. 3) cannot realize all grid tensors on the templates $\mathbf{x}^{(1)} \ldots \mathbf{x}^{(M)}$. □

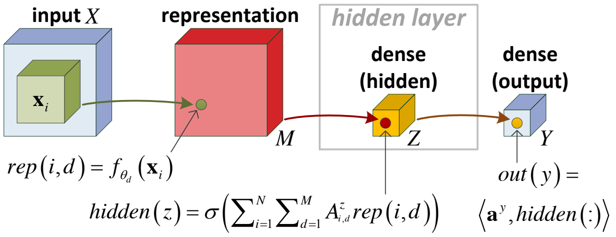
input X
representation
hidden layer
dense (hidden)
dense (output)
$\mathbf{x}_i$
M
Z
Y
$rep(i,d) = f_{\theta_d}(\mathbf{x}_i)$
$hidden(z) = \sigma\left(\sum_{i=1}^{N} \sum_{d=1}^{M} A^z_{i,d}\, rep(i,d)\right)$
$out(y) =$
$\langle \mathbf{a}^y, hidden(:) \rangle$

Figure 4. Shallow fully-connected network obtained by expanding the conv receptive field in the shallow ConvNet to cover the entire spatial extent. The hidden layer consists of a Z-channel dense linear operator weighted by $\{A^z \in \mathbb{R}^{N \times M}\}_{z \in [Z]}$, and followed by point-wise activation $\sigma(\cdot)$. The resulting Z-dimensional vector is mapped to Y network outputs through a dense linear operator weighted by $\{\mathbf{a}^y \in \mathbb{R}^Z\}_{y \in [Y]}$. Best viewed in color.

We conclude this subsection by noting that the non-universality result in claim 6 does *not* contradict the known universality of shallow (single hidden layer) fully-connected neural networks. Indeed, a shallow fully-connected network corresponds to the ConvNet illustrated in fig. 3, with conv receptive field covering the entire spatial extent ($w{\cdot}h = N$), thereby effectively removing the pooling operator (assuming the latter realizes the identity on singletons). In claim 7 below we show that such a network, when equipped with ReLU activation, is universal. On the other hand, in claim 6 we assumed that the ConvNet's receptive field covers less than half the spatial extent ($w{\cdot}h < {}^{N}\!/_{2} + 1 - \log_M N$), and have shown that with ReLU activation and average pooling, this leads to non-universality. Loosely speaking, our findings imply that for networks with ReLU activation, which are known to be universal when fully-connected, introducing locality disrupts universality with average pooling (and maintains it with max pooling).

Claim 7. *Assume that there exist covering templates* $\mathbf{x}^{(1)} \ldots \mathbf{x}^{(M)}$*, and corresponding representation functions* $f_{\theta_1} \ldots f_{\theta_M}$ *leading to a matrix* F *(eq. 4) that has non-recurring rows and a constant non-zero column* [5]*. Consider the fully-connected network illustrated in fig. 4, obtained by expanding the conv receptive field in the shallow ConvNet to cover the entire spatial extent. Such a network, when equipped with ReLU activation, is universal.*

[4] Modifying our proof to account for different padding schemes (such as duplication or no padding at all) is trivial – we choose to work with zero padding merely for notational convenience.

[5] The assumption that such representation functions exist differs from our usual non-degeneracy assumption. The latter requires F to be non-singular, whereas here we pose the weaker requirement of F having non-recurring rows. On the other hand, here we also demand that F have a constant non-zero column, *i.e.* that there be a representation function f_{θ_d} such that $f_{\theta_d}(\mathbf{x}^{(1)}) = \cdots = f_{\theta_d}(\mathbf{x}^{(M)}) = c \neq 0$. In claim 1 we showed that standard neurons meet the non-degeneracy assumption. A slight modification to its proof shows that they also meet the assumption made here. Namely, if we modify the constructions for the cases of ReLU activation and sigmoidal activation by setting $f_{\theta_1}(\mathbf{x}) = \sigma(\mathbf{0}^\top \mathbf{x} + 1)$ and $f_{\theta_1}(\mathbf{x}) = \sigma(\mathbf{0}^\top \mathbf{x} + \alpha)$ respectively, we get matrices F that are not only non-singular, but also have a constant non-zero column.

Proof. Let $h_y^{S(fc)}$ be the y'th score function of our shallow fully-connected network (fig. 4) when equipped with ReLU activation ($\sigma(z) = \max\{0, z\}$). We would like to show that $\mathcal{A}(h_y^{S(fc)})$ – the grid tensor of $h_y^{S(fc)}$ w.r.t. the covering templates $\mathbf{x}^{(1)} \ldots \mathbf{x}^{(M)}$, may take on any value when hidden and output weights ($\{A^z\}_{z\in[Z]}$ and $\mathbf{a}^y$ respectively) are chosen appropriately.

For any $d_1 \ldots d_N \in [M]$, define the following matrix:

$$F^{(d_1 \ldots d_N)} := \begin{bmatrix} f_{\theta_1}(\mathbf{x}^{(d_1)}) & \cdots & f_{\theta_M}(\mathbf{x}^{(d_1)}) \\ \vdots & \ddots & \vdots \\ f_{\theta_1}(\mathbf{x}^{(d_N)}) & \cdots & f_{\theta_M}(\mathbf{x}^{(d_N)}) \end{bmatrix} \in \mathbb{R}^{N \times M}$$

In words, $F^{(d_1 \ldots d_N)}$ is the matrix obtained by taking rows $d_1 \ldots d_N$ from F (recurrence allowed), and stacking them one on top of the other. It holds that:

$$\mathcal{A}(h_y^{S(fc)})_{d_1 \ldots d_N} = \sum_{z=1}^{Z} a_z^y \max\left\{0, \left\langle F^{(d_1 \ldots d_N)}, A^z \right\rangle\right\}$$

where $\langle \cdot, \cdot \rangle$ stands for the inner-product operator, *i.e.* $\left\langle F^{(d_1 \ldots d_N)}, A^z \right\rangle := \sum_{i=1}^{N} \sum_{d=1}^{M} F_{i,d}^{(d_1 \ldots d_N)} A_{i,d}^z$.

By assumption F has a constant non-zero column. This implies that there exist $j \in [M], c \neq 0$ such that for any $d_1 \ldots d_N \in [M]$, all entries in column j of $F^{(d_1 \ldots d_N)}$ are equal to c. For every $d_1 \ldots d_N \in [M]$ and $z \in [Z]$, denote by $\tilde{F}^{(d_1 \ldots d_N)}$ and $\tilde{A}^z$ the matrices obtained by removing the j'th column from $F^{(d_1 \ldots d_N)}$ and A^z respectively. Defining $\mathbf{b} \in \mathbb{R}^Z$ to be the vector whose z'th entry is given by $b_z = c \cdot \sum_{i=1}^{N} A_{i,j}^z$, we may write:

$$\mathcal{A}(h_y^{S(fc)})_{d_1 \ldots d_N} = \sum_{z=1}^{Z} a_z^y \max\left\{0, \left\langle \tilde{F}^{(d_1 \ldots d_N)}, \tilde{A}^z \right\rangle + b_z\right\}$$

noting that for every $z \in [Z]$, $\tilde{A}^z$ and b_z may take on any values with proper choice of A^z. Since by assumption F has non-recurring rows, and since all rows hold the same value (c) in their j'th entry, we have that $\tilde{F}^{(d_1 \ldots d_N)} \neq \tilde{F}^{(d'_1 \ldots d'_N)}$ for $(d_1 \ldots d_N) \neq (d'_1 \ldots d'_N)$. An application of lemma 1 now shows that when $Z \geq M^N$, any value for the grid tensor $\mathcal{A}(h_y^{S(fc)})$ may be realized with proper assignment of $\{\tilde{A}^z\}_{z\in[Z]}$, $\mathbf{b}$ and $\mathbf{a}^y$. Since $\{\tilde{A}^z\}_{z\in[Z]}$ and $\mathbf{b}$ may be set arbitrarily through $\{A^z\}_{z\in[Z]}$, we get that with proper choice of hidden and output weights ($\{A^z\}_{z\in[Z]}$ and $\mathbf{a}^y$ respectively), the grid tensor of our network w.r.t. the covering templates may take on any value, precisely meaning that universality holds. □

Lemma 1. *Let $\mathbf{v}_1 \ldots \mathbf{v}_k \in \mathbb{R}^D$ be distinct vectors ($\mathbf{v}_i \neq \mathbf{v}_j$ for $i \neq j$), and $c_1 \ldots c_k \in \mathbb{R}$ be any scalars. Then, there exist $\mathbf{w}_1 \ldots \mathbf{w}_k \in \mathbb{R}^D$, $\mathbf{b} \in \mathbb{R}^k$ and $\mathbf{a} \in \mathbb{R}^k$ such that $\forall i \in [k]$:*

$$\sum_{j=1}^{k} a_j \max\{0, \mathbf{w}_j^\top \mathbf{v}_i + b_j\} = c_i \tag{12}$$

Proof. As shown in the proof of claim 1, for distinct $\mathbf{v}_1 \ldots \mathbf{v}_k \in \mathbb{R}^D$ there exists a vector $\mathbf{u} \in \mathbb{R}^D$ such that $\mathbf{u}^\top \mathbf{v}_i \neq \mathbf{u}^\top \mathbf{v}_j$ for all $1 \leq i < j \leq k$. We assume without loss of generality that $\mathbf{u}^\top \mathbf{v}_1 < \ldots < \mathbf{u}^\top \mathbf{v}_k$, and set $\mathbf{w}_1 \ldots \mathbf{w}_k$, $\mathbf{b}$ and $\mathbf{a}$ as follows:

- $\mathbf{w}_1 = \cdots = \mathbf{w}_k = \mathbf{u}$
- $b_1 = -\mathbf{u}^\top \mathbf{v}_1 + 1$
- $b_j = -\mathbf{u}^\top \mathbf{v}_{j-1}$ for $j = 2 \ldots k$
- $a_1 = c_1$
- $a_j = \frac{c_j - c_{j-1}}{\mathbf{u}^\top \mathbf{v}_j - \mathbf{u}^\top \mathbf{v}_{j-1}} - \sum_{t=1}^{j-1} a_t$ for $j = 2 \ldots k$

To complete the proof, we show below that this assignment meets the condition in eq. 12 for $i = 1 \ldots k$.

The fact that:

$$\mathbf{w}_j^\top \mathbf{v}_1 + b_j = \begin{cases} \mathbf{u}^\top \mathbf{v}_1 - \mathbf{u}^\top \mathbf{v}_1 + 1 = 1 & , j = 1 \\ \mathbf{u}^\top \mathbf{v}_1 - \mathbf{u}^\top \mathbf{v}_{j-1} \leq 0 & , 2 \leq j \leq k \end{cases}$$

implies that the condition in eq. 12 indeed holds for $i = 1$:

$$\sum_{j=1}^{k} a_j \max\{0, \mathbf{w}_j^\top \mathbf{v}_1 + b_j\} = a_1 \cdot 1 + \sum_{j=1}^{k} a_j \cdot 0 = a_1 = c_1$$

For $i > 1$ we have:

$$\mathbf{w}_j^\top \mathbf{v}_i + b_j = \begin{cases} \mathbf{u}^\top \mathbf{v}_i - \mathbf{u}^\top \mathbf{v}_1 + 1 > 0 & , j = 1 \\ \mathbf{u}^\top \mathbf{v}_i - \mathbf{u}^\top \mathbf{v}_{j-1} > 0 & , 2 \leq j \leq i \\ \mathbf{u}^\top \mathbf{v}_i - \mathbf{u}^\top \mathbf{v}_{j-1} \leq 0 & , i < j \leq k \end{cases}$$

which implies:

$$\sum_{j=1}^{k} a_j \max\{0, \mathbf{w}_j^\top \mathbf{v}_i + b_j\} = \\ a_1(\mathbf{u}^\top \mathbf{v}_i - \mathbf{u}^\top \mathbf{v}_1 + 1) + \sum_{j=2}^{i} a_j(\mathbf{u}^\top \mathbf{v}_i - \mathbf{u}^\top \mathbf{v}_{j-1})$$

Comparing this to the same expression with i replaced by $i-1$ we obtain:

$$\sum_{j=1}^{k} a_j \max\{0, \mathbf{w}_j^\top \mathbf{v}_i + b_j\} = \\ \sum_{j=1}^{k} a_j \max\{0, \mathbf{w}_j^\top \mathbf{v}_{i-1} + b_j\} + \\ (\mathbf{u}^\top \mathbf{v}_i - \mathbf{u}^\top \mathbf{v}_{i-1}) \sum_{j=1}^{i} a_j$$

Now, if we follow an inductive argument and assume that the condition in eq. 12 holds for $i-1$, *i.e.* that $\sum_{j=1}^{k} a_j \max\{0, \mathbf{w}_j^\top \mathbf{v}_{i-1} + b_j\} = c_{i-1}$, we get:

$$\sum_{j=1}^{k} a_j \max\{0, \mathbf{w}_j^\top \mathbf{v}_i + b_j\} = \\ c_{i-1} + (\mathbf{u}^\top \mathbf{v}_i - \mathbf{u}^\top \mathbf{v}_{i-1}) \sum_{j=1}^{i} a_j$$

Plugging in the definition $a_i = \frac{c_i - c_{i-1}}{\mathbf{u}^\top \mathbf{v}_i - \mathbf{u}^\top \mathbf{v}_{i-1}} - \sum_{j=1}^{i-1} a_j$ gives:

$$\begin{gathered}\textstyle\sum_{j=1}^{k} a_j \max\{0, \mathbf{w}_j^\top \mathbf{v}_i + b_j\} = \\ c_{i-1} + (\mathbf{u}^\top \mathbf{v}_i - \mathbf{u}^\top \mathbf{v}_{i-1}) \frac{c_i - c_{i-1}}{\mathbf{u}^\top \mathbf{v}_i - \mathbf{u}^\top \mathbf{v}_{i-1}} = c_i\end{gathered}$$

Thus the condition in eq. 12 holds for i as well. We have therefore shown by induction that our assignment of $\mathbf{w}_1 \ldots \mathbf{w}_k$, $\mathbf{b}$ and $\mathbf{a}$ meets the lemma's requirement. □

5.5. Depth Efficiency

The driving force behind deep learning is the expressive power that comes with depth. It is generally believed that deep networks with non-linear layers efficiently express functions that cannot be efficiently expressed by shallow networks, *i.e.* that would require the latter to have super-polynomial size. We refer to such scenario as *depth efficiency*. Being concerned with the minimal size required by a shallow network in order to realize (or approximate) a given function, the question of depth efficiency implicitly assumes universality, *i.e.* that there exists some (possibly exponential) size with which the shallow network is capable of expressing the target function. [6]

To the best of our knowledge, at the time of this writing the only work to formally analyze depth efficiency in the context of ConvNets is [5]. This work focused on convolutional arithmetic circuits, showing that with such networks depth efficiency is *complete*, *i.e.* besides a negligible set, all functions realizable by a deep network enjoy depth efficiency. We frame this result in our setup:

Claim 8 (adaptation of theorem 1 in [5])**.** *Let $f_{\theta_1} \ldots f_{\theta_M}$ be any set of linearly independent representation functions for a deep ConvNet (fig. 1 with $L = \log_2 N$) with linear activation and product pooling. Suppose we randomize the linear weights ($\mathbf{a}^{l,j,\gamma}$) of the network by some continuous distribution. Then, with probability 1, we obtain score functions that cannot be realized by a shallow ConvNet (fig. 2) with linear activation and product pooling if the number of hidden channels in the latter (Z) is less than $\min\{r_0, M\}^{N/2}$.*

Proof. Let $\mathbf{x}^{(1)} \ldots \mathbf{x}^{(M)} \in \mathbb{R}^s$ be templates such that F is invertible (existence follows from claim 2). The deep network generates grid tensors on $\mathbf{x}^{(1)} \ldots \mathbf{x}^{(M)}$ through the standard HT decomposition (eq. 7 with $g(a, b) = a{\cdot}b$). The proof of theorem 1 in [5] shows that when arranged as matrices, such tensors have rank at least $\min\{r_0, M\}^{N/2}$ almost always, *i.e.* for all weight ($\mathbf{a}^{l,j,\gamma}$) settings but a set of (Lebesgue) measure zero. On the other hand, the shallow network generates grid tensors on $\mathbf{x}^{(1)} \ldots \mathbf{x}^{(M)}$ through the standard CP decomposition (eq. 6 with $g(a, b) = a{\cdot}b$), possibly with a different matrix F (representation functions need not be the same). Such tensors, when arranged as matrices, are shown in the proof of theorem 1 in [5] to have rank at most Z. Therefore, for them to realize the grid tensors generated by the deep network, we almost always must have $Z \geq \min\{r_0, M\}^{N/2}$. □

We now turn to convolutional rectifier networks, for which depth efficiency has yet to be analyzed. In sec. 5.4 we saw that convolutional rectifier networks are universal with max pooling, and non-universal with average pooling. Since depth efficiency is only applicable to universal architectures, we focus on the former setting. The following claim establishes existence of depth efficiency for ConvNets with ReLU activation and max pooling:

Claim 9. *There exist weight settings for a deep ConvNet with ReLU activation and max pooling, giving rise to score functions that cannot be realized by a shallow ConvNet with ReLU activation and max pooling if the number of hidden channels in the latter (Z) is less than $\min\{r_0, M\}^{N/2} \cdot \frac{2}{M \cdot N}$.*

Proof. The proof traverses along the following path. Letting $\mathbf{x}^{(1)} \ldots \mathbf{x}^{(M)} \in \mathbb{R}^s$ be any distinct templates, we show that when arranged as matrices, grid tensors on $\mathbf{x}^{(1)} \ldots \mathbf{x}^{(M)}$ generated by the shallow network have rank at most $Z \cdot \frac{M \cdot N}{2}$. Then, defining $f_{\theta_1} \ldots f_{\theta_M}$ to be representation functions for the deep network giving rise to an invertible F (non-degeneracy implies that such functions exist), we show explicit linear weight ($\mathbf{a}^{l,j,\gamma}$) settings under which the grid tensors on $\mathbf{x}^{(1)} \ldots \mathbf{x}^{(M)}$ generated by the deep network, when arranged as matrices, have rank at least $\min\{r_0, M\}^{N/2}$.

In light of the above, the proof boils down to showing that with $g(a, b) = \max\{a, b, 0\}$:

- The matricized generalized CP decomposition (eq. 10) produces matrices with rank at most $Z \cdot \frac{M \cdot N}{2}$.
- For an invertible F, there exists a weight ($\mathbf{a}^{l,j,\gamma}$) setting under which the matricized generalized HT decomposition (eq. 9) produces a matrix with rank at least $\min\{r_0, M\}^{N/2}$.

We begin with the first point, showing that for every $\mathbf{v}_1, \ldots, \mathbf{v}_{N/2} \in \mathbb{R}^M$ and $\mathbf{u}_1, \ldots, \mathbf{u}_{N/2} \in \mathbb{R}^M$:

$$rank \left(\mathbf{v}_1 \odot_g \cdots \odot_g \mathbf{v}_{\frac{N}{2}} \right) \odot_g \left(\mathbf{u}_1 \odot_g \cdots \odot_g \mathbf{u}_{\frac{N}{2}} \right)^\top \leq \frac{M \cdot N}{2} \tag{13}$$

This would imply that every summand in the matricized generalized CP decomposition (eq. 10) has rank at most $\frac{M \cdot N}{2}$, and the desired result readily follows. To prove

[6] While technically it is possible to consider depth efficiency with a non-universal shallow network, in the majority of the cases, particularly in our framework, the shallow network would simply not be able to express a function generated by a deep network, no matter how large we allow it to be. Arguably, this provides little insight into the complexity of functions brought forth by depth.

eq. 13, note that each of the vectors $\bar{\mathbf{v}} := \mathbf{v}_1 \odot_g \cdots \odot_g \mathbf{v}_{\frac{N}{2}}$ and $\bar{\mathbf{u}} := \mathbf{u}_1 \odot_g \cdots \odot_g \mathbf{u}_{\frac{N}{2}}$ are of dimension $M^{N/2}$, but have only up to $\frac{M \cdot N}{2}$ unique values. Let $\delta_{\mathbf{v}}, \delta_{\mathbf{u}} : [M^{N/2}] \to [M^{N/2}]$ be permutations that arrange the entries of $\bar{\mathbf{v}}$ and $\bar{\mathbf{u}}$ in descending order. Permuting the rows of the matrix $\bar{\mathbf{v}} \odot_g \bar{\mathbf{u}}^\top$ via $\delta_{\mathbf{v}}$, and the columns via $\delta_{\mathbf{u}}$, obviously does not change its rank. On the other hand, we get a $M^{N/2} \times M^{N/2}$ matrix with a $\frac{M \cdot N}{2} \times \frac{M \cdot N}{2}$ block structure, each block being constant (*i.e.* all entries of a block hold the same value). This implies that the rank of $\bar{\mathbf{v}} \odot_g \bar{\mathbf{u}}^\top$ is at most $\frac{M \cdot N}{2}$, which is what we set out to prove.

Moving on to the matricized generalized HT decomposition (eq. 9), for an invertible F we define the following weight setting ($\mathbf{0}$ and $\mathbf{1}$ here denote the all-0 and all-1 vectors, respectively):

- $\mathbf{a}^{0,j,\gamma} = \begin{cases} F^{-1}\bar{\mathbf{e}}_\gamma & ,\gamma \leq M \\ \mathbf{0} & ,\gamma > M \end{cases}$, where $\bar{\mathbf{e}}_\gamma \in \mathbb{R}^M$ is defined to be the vector holding 0 in entry γ and 1 in all other entries.
- $\mathbf{a}^{l,j,\gamma} = \begin{cases} \mathbf{1} & ,\gamma = 1 \, , \, l \in [L-1] \\ \mathbf{0} & ,\gamma > 1 \, , \, l \in [L-1] \end{cases}$
- $\mathbf{a}^{L,1,y} = \mathbf{1}$

Under this setting, the produced matrix $\left[\mathcal{A}\left(h_y^D\right)\right]$ holds $\min\{r_0, M\}$ everywhere besides $\min\{r_0, M\}^{N/2}$ entries on its diagonal, where it holds $\min\{r_0, M\} - 1$. The rank of this matrix is at least $\min\{r_0, M\}^{N/2}$. □

Nearly all results in the literature that relate to depth efficiency merely show its existence, and claim 9 is no different in that respect. From a practical perspective, the implications of such results are slight, as a-priori, it may be that only a small fraction of the functions realizable by a deep network enjoy depth efficiency, and for all the rest shallow networks suffice. In sec. 5.5.2 we extend claim 9, arguing that with ReLU activation and max pooling, depth efficiency becomes more and more prevalent as the number of hidden channels in the deep ConvNet grows. However, no matter how large the deep ConvNet is, with ReLU activation and max pooling depth efficiency is *never complete* – there is always positive measure to the set of weight configurations that lead the deep ConvNet to generate score functions efficiently realizable by the shallow ConvNet:

Claim 10. *Suppose we randomize the weights of a deep ConvNet with ReLU activation and max pooling by some continuous distribution with non-vanishing continuous probability density function. Then, assuming covering templates exist, with positive probability, we obtain score functions that can be realized by a shallow ConvNet with ReLU activation and max pooling having only a single hidden channel ($Z = 1$).*

Proof. Let $\mathbf{x}^{(1)} \ldots \mathbf{x}^{(M)} \in \mathbb{R}^s$ be covering templates, and $f_{\theta_1} \ldots f_{\theta_M}$ be representation functions for the deep network under which F is invertible (non-degeneracy implies that such functions exist). We will show that there exists a linear weight ($\mathbf{a}^{l,j,\gamma}$) setting for the deep network with which it generates a grid tensor that is realizable by a shallow network with a single hidden channel ($Z = 1$). Moreover, we show that when the representation parameters (θ_d) and linear weights ($\mathbf{a}^{l,j,\gamma}$) are subject to small perturbations, the deep network's grid tensor can still be realized by a shallow network with a single hidden channel. Since templates are covering grid tensors fully define score functions. This, along with the fact that open sets in Lebesgue measure spaces always have positive measure (see sec. 5.1), imply that there is positive measure to the set of weight configurations leading the deep network to generate score functions realizable by a shallow network with $Z = 1$. Translating the latter statement from measure theoretical to probabilistic terms readily proves the result we seek after.

In light of the above, the proof boils down to the following claim, framed in terms of our generalized tensor decompositions. Fixing $g(a,b) = \max\{a, b, 0\}$, per arbitrary invertible F there exists a weight ($\mathbf{a}^{l,j,\gamma}$) setting for the generalized HT decomposition (eq. 7), such that the produced tensor may be realized by the generalized CP decomposition (eq. 6) with $Z = 1$, and this holds even if the weights $\mathbf{a}^{l,j,\gamma}$ and matrix F are subject to small perturbations [7].

We will now show that the following weight setting meets our requirement ($\mathbf{0}$ and $\mathbf{1}$ here denote the all-0 and all-1 vectors, respectively):

- $\mathbf{a}^{0,j,\gamma} = \begin{cases} F^{-1}\mathbf{1} & ,j \text{ odd} \\ \mathbf{0} & ,j \text{ even} \end{cases}$
- $\mathbf{a}^{l,j,\gamma} = \begin{cases} \mathbf{1} & ,j \text{ odd} \, , \, l \in [L-1] \\ \mathbf{0} & ,j \text{ even} \, , \, l \in [L-1] \end{cases}$
- $\mathbf{a}^{L,1,y} = \mathbf{1}$

Let $\mathcal{E}^F$ be an additive noise matrix applied to F, and $\{\boldsymbol{\epsilon}^{l,j,\gamma}\}_{l,j,\gamma}$ be additive noise vectors applied to $\{\mathbf{a}^{l,j,\gamma}\}_{l,j,\gamma}$. We use the notation $\mathbf{o}(\epsilon)$ to refer to vectors that tend to $\mathbf{0}$ as $\mathcal{E}^F \to 0$ and $\boldsymbol{\epsilon}^{l,j,\gamma} \to \mathbf{0}$, with the dimension of a vector to be understood by context. Plugging in the noisy variables into the generalized HT decomposition

[7] Recall that by assumption representation functions are continuous w.r.t. their parameters ($f_\theta(\mathbf{x})$ is continuous w.r.t. θ), and so small perturbations on representation parameters (θ_d) translate into small perturbations on the matrix F (eq. 4).

(eq. 7), we get for every $j \in [N/2]$ and $\alpha \in [r_0]$:

$$\begin{aligned} &((F + \mathcal{E}^F)(\mathbf{a}^{0,2j-1,\alpha} + \boldsymbol{\epsilon}^{0,2j-1,\alpha})) \\ &\otimes_g ((F + \mathcal{E}^F)(\mathbf{a}^{0,2j,\alpha} + \boldsymbol{\epsilon}^{0,2j,\alpha})) \\ &= ((F + \mathcal{E}^F)(F^{-1}\mathbf{1} + \boldsymbol{\epsilon}^{0,2j-1,\alpha})) \\ &\otimes_g ((F + \mathcal{E}^F)(\mathbf{0} + \boldsymbol{\epsilon}^{0,2j,\alpha})) \\ &= (\mathbf{1} + \mathbf{o}(\epsilon)) \otimes_g \mathbf{o}(\epsilon) \end{aligned}$$

If the applied noise $(\mathcal{E}^F, \boldsymbol{\epsilon}^{l,j,\gamma})$ is small enough this is equal to $(\mathbf{1} + \mathbf{o}(\epsilon)) \otimes \mathbf{1}$ (recall that $\otimes$ stands for the *standard* tensor product), and we in turn get for every $j \in [N/4]$ and $\gamma \in [r_1]$:

$$\begin{aligned} &\phi^{1,2j-1,\gamma} \otimes_g \phi^{1,2j,\gamma} \\ &= \left(\textstyle\sum_{\alpha=1}^{r_0} a_\alpha^{1,2j-1,\gamma}(\mathbf{1} + \mathbf{o}(\epsilon)) \otimes \mathbf{1}\right) \\ &\otimes_g \left(\textstyle\sum_{\alpha=1}^{r_0} a_\alpha^{1,2j,\gamma}(\mathbf{1} + \mathbf{o}(\epsilon)) \otimes \mathbf{1}\right) \\ &= \left(\textstyle\sum_{\alpha=1}^{r_0}(1 + \epsilon_\alpha^{1,2j-1,\gamma})(\mathbf{1} + \mathbf{o}(\epsilon)) \otimes \mathbf{1}\right) \\ &\otimes_g \left(\textstyle\sum_{\alpha=1}^{r_0} \epsilon_\alpha^{1,2j,\gamma}(\mathbf{1} + \mathbf{o}(\epsilon)) \otimes \mathbf{1}\right) \\ &= ((r_0\mathbf{1} + \mathbf{o}(\epsilon)) \otimes \mathbf{1}) \otimes_g (\mathbf{o}(\epsilon) \otimes \mathbf{1}) \end{aligned}$$

With the applied noise $(\mathcal{E}^F, \boldsymbol{\epsilon}^{l,j,\gamma})$ small enough this becomes $(r_0\mathbf{1} + \mathbf{o}(\epsilon) \otimes \mathbf{1} \otimes \mathbf{1} \otimes \mathbf{1}$. Continuing in this fashion over the levels of the decomposition, we get that with small enough noise, for every $l \in [L-1]$, $j \in [N/2^{l+1}]$ and $\gamma \in [r_l]$:

$$\phi^{l,2j-1,\gamma} \otimes_g \phi^{l,2j,\gamma} = \left(\prod_{l'=0}^{l-1} r_{l'} \cdot \mathbf{1} + \mathbf{o}(\epsilon)\right) \otimes \left(\otimes_{i=1}^{2^{l+1}-1} \mathbf{1}\right)$$

where $\otimes_{i=1}^{2^{l+1}-1} \mathbf{1}$ stands for the tensor product of the vector $\mathbf{1}$ with itself $2^{l+1} - 1$ times. We readily conclude from this that with small enough noise, the tensor produced by the decomposition may be written as follows:

$$\mathcal{A}\left(h_y^D\right) = \left(\prod_{l=0}^{L-1} r_l \cdot \mathbf{1} + \mathbf{o}(\epsilon)\right) \otimes \left(\otimes_{i=1}^{N-1} \mathbf{1}\right) \quad (14)$$

To finish our proof, it remains to show that a tensor as in eq. 14 may be realized by the generalized CP decomposition (eq. 6) with $Z = 1$ (and $g(a, b) = \max\{a, b, 0\}$). Indeed, we may assume that the latter's F, which we denote by $\tilde{F}$ to distinguish from the matrix in the generalized HT decomposition (eq. 7), is invertible (non-degeneracy ensures that this may be achieved with proper choice of representation functions for the shallow ConvNet). Setting the weights of the generalized CP decomposition (eq. 6) through:

- $a_1^y = 1$
- $\mathbf{a}^{1,i} = \begin{cases} \tilde{F}^{-1}\left(\prod_{l=0}^{L-1} r_l \cdot \mathbf{1} + \mathbf{o}(\epsilon)\right) & , i = 1 \\ \mathbf{0} & , i > 1 \end{cases}$

leads to $\mathcal{A}\left(h_y^S\right) = \mathcal{A}\left(h_y^D\right)$, as required. □

Comparing claims 8 and 10, we see that depth efficiency is complete under linear activation with product pooling, and incomplete under ReLU activation with max pooling. We interpret this as indicating that ***convolutional arithmetic circuits benefit from the expressive power of depth more than convolutional rectifier networks do***. This result is rather surprising, especially given the fact that convolutional rectifier networks are much more commonly used in practice. We attribute the discrepancy primarily to historical reasons, and conjecture that developing effective methods for training convolutional arithmetic circuits, thereby fulfilling their expressive potential, may give rise to a deep learning architecture that is provably superior to convolutional rectifier networks but has so far been overlooked by practitioners.

Loosely speaking, we have shown that the gap in expressive power between the shallow and deep ConvNets is greater with linear activation and product pooling than it is with ReLU activation and max pooling. One may wonder at this point if it is plausible to deduce from this which architectural setting is more expressive, as a-priori, altering the shallow *vs.* deep ConvNet comparisons such that one network has linear activation with product pooling and the other has ReLU activation with max pooling, may change the expressive gaps in favor of the latter. Claims 11 and 12 below show that this is not the case. Specifically, they show that the depth efficiency of the deep ConvNet with linear activation and product pooling remains complete when the shallow ConvNet has ReLU activation and max pooling (claim 11), and on the other hand, the depth efficiency of the deep ConvNet with ReLU activation and max pooling remains incomplete when the shallow ConvNet has linear activation and product pooling (claim 12). This affirms our stand regarding the expressive advantage of convolutional arithmetic circuits over convolutional rectifier networks.

Claim 11. *Let $f_{\theta_1} \ldots f_{\theta_M}$ be any set of linearly independent representation functions for a deep ConvNet with linear activation and product pooling. Suppose we randomize the weights of the network by some continuous distribution. Then, with probability 1, we obtain score functions that cannot be realized by a shallow ConvNet with ReLU activation and max pooling if the number of hidden channels in the latter (Z) is less than* $\min\{r_0, M\}^{N/2} \cdot \frac{2}{M \cdot N}$.

Proof. The proof here follows readily from those of claims 8 and 9. Namely, in the proof of claim 8 we state that for templates $\mathbf{x}^{(1)} \ldots \mathbf{x}^{(M)} \in \mathbb{R}^s$ chosen such that F is invertible (these exist according to claim 2), a grid tensor produced by the deep ConvNet with linear activation and product pooling, when arranged as a matrix, has rank at least $\min\{r_0, M\}^{N/2}$ for all linear weight $(\mathbf{a}^{l,j,\gamma})$ settings but a set of measure zero. That is to say, a matrix produced by the matricized generalized HT decomposition (eq. 9) with $g(a,b) = a{\cdot}b$, has rank at least $\min\{r_0, M\}^{N/2}$ for all weight $(\mathbf{a}^{l,j,\gamma})$ settings but a set of measure zero. On the other hand, we have shown in the proof of claim 9 that a shallow ConvNet with ReLU activation and max pooling generates grid tensors that when arranged as matrices, have rank at most $Z{\cdot}\frac{M\cdot N}{2}$. More specifically, we have shown that the matricized generalized CP decomposition (eq. 10) with $g(a,b) = \max\{a,b,0\}$ produces matrices with rank at most $Z{\cdot}\frac{M\cdot N}{2}$. This implies that under almost all linear weight $(\mathbf{a}^{l,j,\gamma})$ settings for a deep ConvNet with linear activation and product pooling, the generated grid tensor cannot be replicated by a shallow ConvNet with ReLU activation and max pooling if the latter has less than $Z = \min\{r_0, M\}^{N/2} \cdot \frac{2}{M\cdot N}$ hidden channels. □

Claim 12. *Suppose we randomize the weights of a deep ConvNet with ReLU activation and max pooling by some continuous distribution with non-vanishing continuous probability density function. Then, assuming covering templates exist, with positive probability, we obtain score functions that can be realized by a shallow ConvNet with linear activation and product pooling having only a single hidden channel ($Z = 1$).*

Proof. The proof here is almost identical to that of claim 10. The only difference is where we show that a tensor as in eq. 14 may be realized by the generalized CP decomposition (eq. 6) with $Z = 1$. In the proof of claim 10 the underlying operation of the decomposition was $g(a,b) = \max\{a,b,0\}$ (corresponding to ReLU activation and max pooling), whereas here it is $g(a,b) = a{\cdot}b$ (corresponding to linear activation and product pooling). To account for this difference, we again assume that $\tilde{F}$ – the matrix F of the generalized CP decomposition, is invertible (non-degeneracy ensures that this may be achieved with proper choice of representation functions for the shallow ConvNet), and modify the decomposition's weight setting as follows:

- $a_1^y = 1$
- $\mathbf{a}^{1,i} = \begin{cases} \tilde{F}^{-1}\left(\prod_{l=0}^{L-1} r_l \cdot \mathbf{1} + \mathbf{o}(\epsilon)\right) & ,i=1 \\ \tilde{F}^{-1}\mathbf{1} & ,i>1 \end{cases}$

This leads to $\mathcal{A}\left(h_y^S\right) = \mathcal{A}\left(h_y^D\right)$, as required. □

5.5.1 Approximation

In their current form, the results in our analysis establishing depth efficiency (claims 8, 9, 11 and the analogous ones in sec. 5.6) relate to exact realization. Specifically, they provide a lower bound on the size of a shallow ConvNet required in order for it to *realize exactly* a grid tensor generated by a deep ConvNet. From a practical perspective, a more interesting question would be the size required by a shallow ConvNet in order to *approximate* the computation of a deep ConvNet. A-priori, it may be that although the size required for exact realization is exponential, the one required for approximation is only polynomial. As we briefly discuss below, this is not the case, and in fact all of the lower bounds we have provided apply not only to exact realization, but also to arbitrarily-well approximation.

When proving that a grid tensor generated by a shallow ConvNet beneath a certain size cannot be equal to a grid tensor generated by a deep ConvNet, we always rely on matricization rank. Namely, we arrange the grid tensors as matrices, and derive constants $R, r \in \mathbb{N}$, $R > r$, such that the matrix corresponding to the deep ConvNet has rank at least R, while that corresponding to the shallow ConvNet has rank at most r. While used in our proofs solely to show that the matrices are different, this actually entails information regarding the distance between them. Namely, if we denote the singular values of the matrix corresponding to the deep ConvNet by $\sigma_1 \geq \sigma_2 \geq \ldots \geq 0$, the squared Euclidean (Frobenius) distance between the matrices is at least $\sigma_{r+1}^2 + \cdots + \sigma_R^2$. Since the matrices are merely rearrangements of the grid tensors, we have a lower bound on the distance between the shallow ConvNet's grid tensor and the target grid tensor generated by the deep ConvNet, so in particular arbitrarily-well approximation is not possible.

5.5.2 On the Incidence of Depth Efficiency

In claim 8 we saw that depth efficiency is complete with linear activation and product pooling. That is to say, with linear activation and product pooling, besides a negligible set, all weight settings for the deep ConvNet (fig. 1 with size-2 pooling windows and $L = \log_2 N$ hidden layers) lead to score functions that cannot be realized by the shallow ConvNet (fig. 2) unless the latter has super-polynomial size. We have also seen (claims 9 and 10) that replacing the activation and pooling operators by ReLU and max respectively, makes depth efficiency incomplete. There are still weight settings leading the deep ConvNet to generate score functions that require the shallow ConvNet to have super-polynomial size, but these do not occupy the entire space. In other words, there is now positive measure to the set of deep ConvNet weight configurations leading to score functions efficiently realizable by the shallow ConvNet. A natural question would then be just how frequent depth efficiency is under ReLU activation and max pooling. More formally, we may consider a uniform distribution over a compact domain in the deep ConvNet's weight space, and ask the following. Assuming weights for the deep ConvNet are drawn from this distribution, what is the probability that generated score functions exhibit depth efficiency, *i.e.* require super-polynomial size from the shallow ConvNet? In the following we address this question, arguing that the probability tends to 1 as the number of channels in the hidden layers of the deep ConvNet grows. We do not prove this formally, but nonetheless provide a framework we believe may serve as a basis for establishing formal results concerning the incidence of depth efficiency. The framework is not limited to ReLU activation and max pooling – it may be used under different choices of activation and pooling operators as well.

The central tool used in the paper for proving depth efficiency is the rank of grid tensors when these are arranged as matrices. We establish upper bounds on the rank of matricized grid tensors produced by the shallow ConvNet through the matricized generalized CP decomposition (eq. 10). These upper bounds are typically linear in the size of the input (N) and the number of hidden channels in the network (Z). The challenge is then to derive a super-polynomial (in N) lower bound on the rank of matricized grid tensors produced by the deep ConvNet through the matricized generalized HT decomposition (eq. 9). In the case of linear activation and product pooling ($g(a,b)=a{\cdot}b$), the generalized Kronecker product $\odot_g$ reduces to the standard Kronecker product $\odot$, and the rank-multiplicative property of the latter ($rank(A{\odot}B) = rank(A){\cdot}rank(B)$) can be used to show (see [5]) that besides in negligible (zero measure) cases, rank grows rapidly through the levels of the matricized generalized HT decomposition (eq. 9), to the point where the final produced matrix has exponential rank. This situation does not persist when the activation and pooling operators are replaced by ReLU and max (respectively). Indeed, in the proof of claim 10 we explicitly presented a non-negligible (positive measure) case where the matricized generalized HT decomposition (eq. 9) produces a matrix of rank 1. To study the incidence of depth efficiency under ReLU activation and max pooling, we assume the weights ($\mathbf{a}^{l,j,\gamma}$) of the matricized generalized HT decomposition (eq. 9) are drawn independently and uniformly from a bounded interval (*e.g.* $[-1,1]$), and question the probability of the produced matrix $[\mathcal{A}\left(h_y^D\right)]$ having rank super-polynomial in N.

To study $rank[\mathcal{A}\left(h_y^D\right)]$, we sequentially traverse through the levels $l=1\ldots L$ of the matricized generalized HT decomposition (eq. 9), at each level going over all locations $j\in[N/2^l]$. When at location j of level l, for each $\alpha\in[r_{l-1}]$, we draw the weights $\mathbf{a}^{l-1,2j-1,\alpha}$ and $\mathbf{a}^{l-1,2j,\alpha}$ (independently of the previously drawn weights), and observe the random variable $R^{l,j,\alpha}$, defined as the rank of the matrix $[\phi^{l-1,2j-1,\alpha}]\odot_g[\phi^{l-1,2j,\alpha}]$. Given the weights drawn while traversing through the previous levels of the decomposition, the random variables $\{R^{l,j,\alpha}\in\mathbb{N}\}_{\alpha\in[r_{l-1}]}$ are independent and identically distributed. The random variable $R^{l,j}:=\max_{\alpha\in[r_{l-1}]}\{R^{l,j,\alpha}\}$ thus tends to concentrate on higher and higher values as r_{l-1} (number of channels in hidden layer $l{-}1$ of the deep ConvNet) grows. When the next level ($l+1$) of the decomposition will be traversed, the weights $\{\mathbf{a}^{l,j,\gamma}\}_{\gamma\in[r_l]}$ will be drawn, and the matrices $\{[\phi^{l,j,\gamma}]\}_{\gamma\in[r_l]}$ will be generated. According to claim 13 below, with probability 1, all of these matrices will have rank equal to at least $R^{l,j}$. We conclude that, assuming the generalized Kronecker product $\odot_g$ has the potential of increasing the rank of its operands, ranks will generally ascend across the levels of the matricized generalized HT decomposition (eq. 9), with steeper ascends being more and more probable as the number of channels in the hidden layers of the deep ConvNet ($r_0\ldots r_{L-1}$) grows.

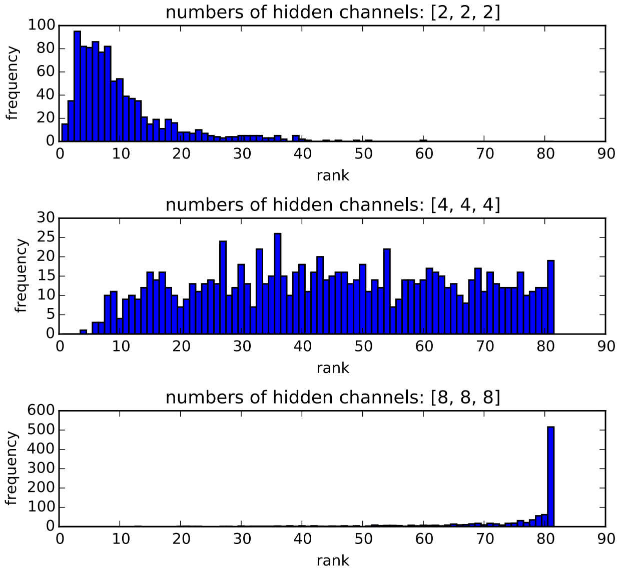

Figure 5. Simulation results demonstrating that under ReLU activation and max pooling, the incidence of depth efficiency increases as the number of channels in the hidden layers of the deep ConvNet ($r_0\ldots r_{L-1}$) grows. The plots show histograms of the ranks produced by the matricized generalized HT decomposition (eq. 9) with $g(a,b)=\max\{a,b,0\}$. The number of levels in the decomposition was set to $L=3$ (implying input size of $N=2^L=8$). The size of the representation matrix F was set through $M=3$, and the matrix itself was fixed to the identity. Weights ($\mathbf{a}^{l,j,\gamma}$) were drawn at random independently and uniformly from the interval $[-1,1]$. Three channel-width configurations were tried: (i) $r_0=r_1=r_2=2$ (ii) $r_0=r_1=r_2=4$ (ii) $r_0=r_1=r_2=8$. For each configuration 1000 random tests were run, creating the histograms presented in the figure (each test produced a single matrix $[\mathcal{A}(h_y^D)]$, accounting for a single entry in a histogram). As can be seen, the distribution of the produced rank ($rank[\mathcal{A}(h_y^D)]$) tends towards the maximum ($M^{N/2}=81$) as the numbers of hidden channels grow.

The main piece that is missing in order to complete the sketch we have outlined above into a formal proof, is the behavior of rank under the generalized Kronecker product $\odot_g$. This obviously depends on the choice of underlying operator g. In the case of linear activation and product pooling $g(a,b) = a{\cdot}b$, the generalized Kronecker product $\odot_g$ reduces to the standard Kronecker product $\odot$, and ranks always increase multiplicatively, *i.e.* $rank(A{\odot}B) = rank(A){\cdot}rank(B)$ for any matrices A and B. The fact that there is a simple law governing the behavior of ranks makes this case relatively simple to analyze, and we indeed have a full characterization (claim 8). In the case of *linear* activation and max pooling the underlying operator is given by $g(a,b) = \max\{a,b\}$, and it is not difficult to see that $\odot_g$ does not decrease rank, *i.e.* $rank(A{\odot_g}B){\geq}\min\{rank(A), rank(B)\}$ for any matrices A and B [8]. For ReLU activation and max pooling, corresponding to the choice $g(a,b) = \max\{a,b,0\}$, there is no simple rule depicting the behavior of ranks under $\odot_g$, and in fact, for matrices A and B holding negative values, the rank of $rank(A{\odot_g}B)$ necessarily drops to zero. Nonetheless, it seems reasonable to assume that at least in some cases, a non-linear operation such as $\odot_g$ does increase rank, and as we have seen, benefiting from these cases is more probable when the hidden layers of the deep ConvNet include many channels. To this end, we provide in fig. 5 simulation results for the case of ReLU activation and max pooling ($g(a,b) = \max\{a,b,0\}$), demonstrating that indeed ranks produced by the matricized generalized HT decomposition (eq. 9) tend to be higher as $r_0{\ldots}r_{L-1}$ grow. We leave a complete formal analysis of this phenomenon to future work.

Claim 13. *Let $A_1{\ldots}A_m$ be given matrices of the same size, having ranks $r_1{\ldots}r_m$ respectively. For every weight vector $\boldsymbol{\alpha} \in \mathbb{R}^m$ define the matrix $A(\boldsymbol{\alpha}) := \sum_{i=1}^m \alpha_i A_i$, and suppose we randomize $\boldsymbol{\alpha}$ by some continuous distribution. Then, with probability 1, we obtain a matrix $A(\boldsymbol{\alpha})$ having rank at least $\max_{i\in[m]} r_i$.*

Proof. Our proof relies on concepts and results from Lebesgue measure theory (see sec. 5.1 for a brief discussion). The result to prove is equivalent to stating that there is measure zero to the set of weight vectors $\boldsymbol{\alpha}$ for which $rank(A(\boldsymbol{\alpha})) < \max_{i\in[m]} r_i$.

Assume without loss of generality that $\max_{i\in[m]} r_i$ is equal to r_1, and that the top-left $r_1{\times}r_1$ block of A_1 is non-singular. For every $\boldsymbol{\alpha}$ define $p(\boldsymbol{\alpha}) := \det(A(\boldsymbol{\alpha})_{1:r_1,1:r_1})$, *i.e.* $p(\boldsymbol{\alpha})$ is the determinant of the $r_1{\times}r_1$ top-left block of the matrix $A(\boldsymbol{\alpha})$. $p(\boldsymbol{\alpha})$ is obviously a polynomial in the entries of $\boldsymbol{\alpha}$, and by assumption $p(\mathbf{e}_1) \neq 0$, where $\mathbf{e}_1 \in \mathbb{R}^m$ is the vector holding 1 in its first entry and 0 elsewhere. Since a non-zero polynomial vanishes only on a set of zero measure (see [1] for example), the set of weight vectors $\boldsymbol{\alpha}$ for which $p(\boldsymbol{\alpha}) = 0$ has measure zero. This implies that the top-left $r_1{\times}r_1$ block of $A(\boldsymbol{\alpha})$ is non-singular almost everywhere, and in particular $rank(A(\boldsymbol{\alpha})){\geq}r_1 = \max_{i\in[m]} r_i$ almost everywhere. □

5.6. Shared Coefficients for Convolution

To this end, our analysis has focused on the unshared setting, where the coefficients of the 1×1 conv filters in our networks (fig. 1) may vary across spatial locations. In practice, ConvNets typically enforce sharing, which in our framework implies that the coefficients of the 1×1 conv filter in channel γ of hidden layer l, are the same for all locations j. In this subsection we analyze the shared setting, following a line similar to that of our analysis for the unshared setting given in sec. 5.4 and 5.5. For brevity, we assume the reader is familiar with the latter, and do not repeat discussions given there.

In the shared setting, the shallow ConvNet (fig. 2) would have a single weight vector $\mathbf{a}^z$ for every hidden channel z, as opposed to the unshared setting where it had a weight vector $\mathbf{a}^{z,i}$ for every location i in every hidden channel z. Grid tensors produced by the shallow ConvNet in the shared setting are given by what we call the *shared generalized CP decomposition*:

$$\mathcal{A}\left(h_y^S\right) = \sum_{z=1}^{Z} a_z^y \cdot \underbrace{(F\mathbf{a}^z) \otimes_g \cdots \otimes_g (F\mathbf{a}^z)}_{N \text{ times}} \qquad (15)$$

As for the deep ConvNet (fig. 1 with size-2 pooling windows and $L = \log_2 N$ hidden layers), in the shared setting, instead of having a weight vector $\mathbf{a}^{l,j,\gamma}$ for every hidden layer l, channel γ and location j, there is a single weight vector $\mathbf{a}^{l,\gamma}$ for all locations of channel γ in hidden layer l. Produced grid tensors are then given by the *shared generalized HT decomposition*:

$$\begin{aligned}
\phi^{1,\gamma} &= \sum_{\alpha=1}^{r_0} a_\alpha^{1,\gamma} (F\mathbf{a}^{0,\alpha}) \otimes_g (F\mathbf{a}^{0,\alpha}) \\
&\cdots \\
\phi^{l,\gamma} &= \sum_{\alpha=1}^{r_{l-1}} a_\alpha^{l,\gamma} \underbrace{\phi^{l-1,\alpha}}_{\text{order } 2^{l-1}} \otimes_g \underbrace{\phi^{l-1,\alpha}}_{\text{order } 2^{l-1}} \\
&\cdots \\
\phi^{L-1,\gamma} &= \sum_{\alpha=1}^{r_{L-2}} a_\alpha^{L-1,\gamma} \underbrace{\phi^{L-2,\alpha}}_{\text{order } \frac{N}{4}} \otimes_g \underbrace{\phi^{L-2,\alpha}}_{\text{order } \frac{N}{4}} \\
\mathcal{A}\left(h_y^D\right) &= \sum_{\alpha=1}^{r_{L-1}} a_\alpha^{L,y} \underbrace{\phi^{L-1,\alpha}}_{\text{order } \frac{N}{2}} \otimes_g \underbrace{\phi^{L-1,\alpha}}_{\text{order } \frac{N}{2}} \qquad (16)
\end{aligned}$$

We now turn to analyze universality and depth efficiency in the shared setting.

[8] To see this, simply note that under the choice $g(a,b) = \max\{a,b\}$ there is either a sub-matrix of $A{\odot_g}B$ that is equal to A, or one that is equal to B.

5.6.1 Universality

In the unshared setting we saw (sec. 5.4) that linear activation with product pooling and ReLU activation with max pooling both lead to universality, whereas ReLU activation with average pooling does not. We will now see that in the shared setting, no matter how the activation and pooling operators are chosen, universality is never met.

A shallow ConvNet with shared weights produces grid tensors through the shared generalized CP decomposition (eq. 15). A tensor $\mathcal{A}$ generated by this decomposition is necessarily *symmetric*, *i.e.* for any permutation $\delta : [N] \to [N]$ and indexes $d_1 \ldots d_N$ it meets: $\mathcal{A}_{d_1 \ldots d_N} = \mathcal{A}_{\delta(d_1) \ldots \delta(d_N)}$. Obviously not all tensors share this property, so indeed a shallow ConvNet with weight sharing is not universal. A deep ConvNet with shared weights produces grid tensors through the shared generalized HT decomposition (eq. 16). For this decomposition, a generated tensor $\mathcal{A}$ is invariant to replacing the first and second halves of its modes, *i.e.* for any indexes $d_1 \ldots d_N$ it meets: $\mathcal{A}_{d_1,\ldots,d_N} = \mathcal{A}_{d_{N/2+1},\ldots,d_N,d_1,\ldots,d_{N/2}}$. Although this property is much less stringent than symmetry, it is still not met by most tensors, and so a deep ConvNet with weight sharing is not universal either.

5.6.2 Depth Efficiency

Depth efficiency deals with the computational complexity of replicating a deep network's function using a shallow network. In order for this question to be applicable, we require that the shallow network be a universal machine. If this is not the case, then it is generally likely that the deep network's function simply lies outside the reach of the shallow network, and we do not obtain a quantitative insight into the true power of depth. Since our shallow ConvNets are not universal with shared weights (sec. 5.6.1), we evaluate depth efficiency of deep ConvNets with shared weights against shallow ConvNets with *unshared* weights. Specifically, we do this for the activation-pooling choices leading shallow ConvNets with unshared weights to be universal: linear activation with product pooling, and ReLU activation with max pooling (see sec. 5.4).

For linear activation with product pooling, the following claim, which is essentially a derivative of theorem 1 in [5], tells us that in the shared setting, as in the unshared setting, depth efficiency holds completely:

Claim 14 (shared analogy of claim 8)**.** *Let $f_{\theta_1} \ldots f_{\theta_M}$ be any set of linearly independent representation functions for a deep ConvNet with linear activation, product pooling and weight sharing. Suppose we randomize the weights of the network by some continuous distribution. Then, with probability 1, we obtain score functions that cannot be realized by a shallow ConvNet with linear activation and product pooling (*not *limited by weight sharing), if the number of hidden channels in the latter (Z) is less than* $\min\{r_0, M\}^{N/2}$.

Proof. The proof here is almost identical to that of claim 8. The only difference is that in the latter, we used the fact that the generalized HT decomposition (eq. 7), when equipped with $g(a, b) = a \cdot b$, almost always produces tensors whose matrix arrangements have rank at least $\min\{r_0, M\}^{N/2}$, whereas here, we require an analogous result for the *shared* generalized HT decomposition (eq. 16). Such result is provided by the proof of theorem 1 in [5]. □

Heading on to ReLU activation and max pooling, we will show that here too, the situation in the shared setting is the same as in the unshared setting. Specifically, depth efficiency holds, but not completely. We prove this via two claims, analogous to claims 9 and 10 in sec. 5.5:

Claim 15 (shared analogy of claim 9)**.** *There exist weight settings for a deep ConvNet with ReLU activation, max pooling and weight sharing, giving rise to score functions that cannot be realized by a shallow ConvNet with ReLU activation and max pooling (*not *limited by weight sharing), if the number of hidden channels in the latter (Z) is less than* $\min\{r_0, M\}^{N/2} \cdot \frac{2}{M \cdot N}$.

Proof. In the proof of claim 9 we have shown, for arbitrary distinct templates $\mathbf{x}^{(1)} \ldots \mathbf{x}^{(M)} \in \mathbb{R}^s$, an explicit weight setting for the deep ConvNet with ReLU activation and max pooling, leading the latter to produce a grid tensor that cannot be realized by a shallow ConvNet with ReLU activation and max pooling, if that has less than $\min\{r_0, M\}^{N/2} \cdot \frac{2}{M \cdot N}$ hidden channels. Since the given weight setting was location invariant, *i.e.* the assignment of $\mathbf{a}^{l,j,\gamma}$ did not depend on j, it applies as is to a deep ConvNet with weight sharing, and the desired result readily follows. □

Claim 16 (shared analogy of claim 10)**.** *Suppose we randomize the weights of a deep ConvNet with ReLU activation, max pooling and weight sharing by some continuous distribution with non-vanishing continuous probability density function. Then, assuming covering templates exist, with positive probability, we obtain score functions that can be realized by a shallow ConvNet with ReLU activation and max pooling having only a single hidden channel ($Z = 1$).*

Proof. The proof is similar in spirit to that of claim 10, which dealt with incompleteness of depth efficiency under ReLU activation and max pooling in the unshared setting. Our focus here is on the shared setting, or more specifically, on the case where the deep ConvNet is limited by weight sharing while the shallow ConvNet is not. Accordingly, we would like to show the following. Fixing $g(a, b) = \max\{a, b, 0\}$, per arbitrary invertible F there exists a weight ($\mathbf{a}^{l,\gamma}$) setting for the shared generalized HT decomposition (eq. 16), such that the produced tensor may be realized by the generalized CP decomposition (eq. 6) with $Z = 1$, and this holds even if the weights $\mathbf{a}^{l,\gamma}$ and matrix F are subject to small perturbations.

Before heading on to prove that a weight setting as above exists, we introduce a new definition that will greatly simplify our proof. We refer to a tensor $\mathcal{A}$ of order P and dimension M in each mode as *basic*, if there exists a vector $\mathbf{u} \in \mathbb{R}^M$ with non-decreasing entries ($u_1 \leq \ldots \leq u_M$), such that $\mathcal{A} = \mathbf{u} \otimes_g \cdots \otimes_g \mathbf{u}$ (*i.e.* $\mathcal{A}$ is equal to the generalized tensor product of $\mathbf{u}$ with itself P times, with underlying operation $g(a,b) = \max\{a,b,0\}$). A basic tensor can obviously be realized by the generalized CP decomposition (eq. 6) with $Z = 1$ (given that non-degeneracy is used to ensure the latter's representation matrix is non-singular), and so it suffices to find a weight ($\mathbf{a}^{l,\gamma}$) setting for the shared generalized HT decomposition (eq. 16) that gives rise to a basic tensor, and in addition, ensures that small perturbations on the weights $\mathbf{a}^{l,\gamma}$ and matrix F still yield basic tensors. Two trivial facts that relate to basic tensors and will be used in our proof are: (i) the generalized tensor product of a basic tensor with itself is basic, and (ii) a linear combination of basic tensors with non-negative weights is basic.

Turning to the main part of the proof, we now show that the following weight setting meets our requirement:

- $\mathbf{a}^{0,\gamma} = F^{-1}\mathbf{v}$
- $\mathbf{a}^{l,\gamma} = \mathbf{1},\ l \in [L-1]$
- $\mathbf{a}^{L,y} = \mathbf{1}$

$\mathbf{v}$ here stands for the vector $[1, 2, \ldots, M]^\top \in \mathbb{R}^M$, and $\mathbf{1}$ is an all-1 vector with dimension to be understood by context. Let $\mathcal{E}^F$ be an additive noise matrix applied to F, and $\{\boldsymbol{\epsilon}^{l,\gamma}\}_{l,\gamma}$ be additive noise vectors applied to $\{\mathbf{a}^{l,\gamma}\}_{l,\gamma}$. We would like to prove that under the weight setting above, when applied noise ($\mathcal{E}^F, \boldsymbol{\epsilon}^{l,\gamma}$) is small enough, the grid tensor produced by the shared generalized HT decomposition (eq. 16) is basic.

For convenience, we adopt the notation $\mathbf{o}(\epsilon)$ as referring to vectors that tend to $\mathbf{0}$ as $\mathcal{E}^F \to 0$ and $\boldsymbol{\epsilon}^{l,\gamma} \to \mathbf{0}$, with the dimension of a vector to be understood by context. Plugging in the noisy variables into the shared generalized HT decomposition (eq. 16), we get for every $\alpha \in [r_0]$:

$$\begin{aligned} &((F+\mathcal{E}^F)(\mathbf{a}^{0,\alpha}+\boldsymbol{\epsilon}^{0,\alpha})) \otimes_g ((F+\mathcal{E}^F)(\mathbf{a}^{0,\alpha}+\boldsymbol{\epsilon}^{0,\alpha})) \\ &= ((F+\mathcal{E}^F)(F^{-1}\mathbf{v}+\boldsymbol{\epsilon}^{0,\alpha})) \otimes_g ((F+\mathcal{E}^F)(F^{-1}\mathbf{v}+\boldsymbol{\epsilon}^{0,\alpha})) \\ &= \tilde{\mathbf{v}}^\alpha \otimes_g \tilde{\mathbf{v}}^\alpha \end{aligned}$$

where $\tilde{\mathbf{v}}^\alpha = \mathbf{v} + \mathbf{o}(\epsilon)$. If the applied noise ($\mathcal{E}^F, \boldsymbol{\epsilon}^{l,\gamma}$) is small enough the entries of $\tilde{\mathbf{v}}^\alpha$ are non-decreasing and $\tilde{\mathbf{v}}^\alpha \otimes_g \tilde{\mathbf{v}}^\alpha$ is a basic tensor (matrix). Moving to the next level of the decomposition, we have for every $\gamma \in [r_1]$:

$$\phi^{1,\gamma} = \sum_{\alpha=1}^{r_0} (a_\alpha^{1,\gamma} + \epsilon_\alpha^{1,\gamma}) \cdot \tilde{\mathbf{v}}^\alpha \otimes_g \tilde{\mathbf{v}}^\alpha$$

When applied noise ($\mathcal{E}^F, \boldsymbol{\epsilon}^{l,\gamma}$) is small enough the weights of this linear combination are non-negative, and together with the tensors (matrices) $\tilde{\mathbf{v}}^\alpha \otimes_g \tilde{\mathbf{v}}^\alpha$ being basic, this leads $\phi^{1,\gamma}$ to be basic as well. Continuing in this fashion over the levels of the decomposition, we get that with small enough noise, for every $l \in [L-1]$ and $\gamma \in [r_l]$, $\phi^{l,\gamma}$ is a basic tensor. A final step in this direction shows that under small noise, the produced grid tensor $\mathcal{A}\left(h_y^D\right)$ is basic as well. This is what we set out to prove. □

To recapitulate this subsection, we have shown that introducing weight sharing into the 1×1 conv operators of our networks, thereby limiting the general locally-connected linear mappings to be standard convolutions, disrupts universality, but leaves depth efficiency intact – it remains to hold completely under linear activation with product pooling, and incompletely under ReLU activation with max pooling.

6. Discussion

The contribution of this paper is twofold. First, we introduce a construction in the form of *generalized tensor decompositions*, that enables transforming convolutional arithmetic circuits into *convolutional rectifier networks* (ConvNets with ReLU activation and max or average pooling). This opens the door to various mathematical tools from the world of arithmetic circuits, now available for analyzing convolutional rectifier networks. As a second contribution, we make use of such tools to prove new results on the expressive properties that drive this important class of networks.

Our analysis shows that convolutional rectifier networks are universal with max pooling, but not with average pooling. This implies that if non-linearity originates solely from ReLU activation, increasing network size alone is not sufficient for expressing arbitrary functions. More interestingly, we analyze the behavior of convolutional rectifier networks in terms of *depth efficiency*, *i.e.* of cases where a function generated by a deep network of polynomial size requires shallow networks to have super-polynomial size. It is known that convolutional arithmetic circuits exhibit *complete depth efficiency*, meaning that besides a negligible (zero measure) set, all functions generated by deep networks of this type are depth efficient. We show that this is not the case with convolutional rectifier networks, for which depth efficiency exists, but is weaker in the sense that it is not complete (there is positive measure to the set of functions generated by a deep network that may be efficiently realized by shallow networks).

Depth efficiency is believed to be the key factor behind the success of deep learning. Our analysis indicates that from this perspective, the widely used convolutional rectifier networks are inferior to convolutional arithmetic circuits. This leads us to believe that convolutional arithmetic circuits bear the potential to improve the performance of deep learning beyond what is witnessed today. Of course, a practical machine learning model is measured not only by its expressive power, but also by our ability to train it. Over the years, massive amounts of research have been devoted to training convolutional rectifier networks. Convolutional arithmetic circuits on the other hand received far less attention, although they have been successfully trained in recent works on the SimNet architecture ([3, 4]), demonstrating how the enhanced expressive power can lead to state of the art performance in computationally limited settings.

We believe that developing effective methods for training convolutional arithmetic circuits, thereby fulfilling their expressive potential, may give rise to a deep learning architecture that is provably superior to convolutional rectifier networks but has so far been overlooked by practitioners.

Acknowledgments

This 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

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

[2] Christopher Clark and Amos Storkey. Teaching deep convolutional neural networks to play go. *arXiv preprint arXiv:1412.3409*, 2014.

[3] Nadav Cohen and Amnon Shashua. SimNets: A Generalization of Convolutional Networks. *NIPS Deep Learning and Representation Learning Workshop*, 2014.

[4] Nadav Cohen, Or Sharir, and Amnon Shashua. Deep SimNets. *arXiv.org*, June 2015.

[5] Nadav Cohen, Or Sharir, and Amnon Shashua. On the expressive power of deep learning: a tensor analysis. *arXiv preprint arXiv:1509.05009*, 2015.

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

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

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

[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. Book in preparation for MIT Press, 2016. URL `http://goodfeli.github.io/dlbook/`.

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

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

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

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

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

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, pages 1106–1114, 2012.

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

[17] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[18] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

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

[20] 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.

[21] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[22] 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.

[23] Tomaso Poggio, Fabio Anselmi, and Lorenzo Rosasco. I-theory on depth vs width: hierarchical function composition. Technical report, Center for Brains, Minds and Machines (CBMM), 2015.

[24] 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.

[25] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the companion publication of the 23rd international conference on World wide web companion*, pages 373–374. International World Wide Web Conferences Steering Committee, 2014.

[26] 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.

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

[28] 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.

[29] 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.

[30] Matus Telgarsky. Benefits of depth in neural networks. *arXiv preprint arXiv:1602.04485*, 2016.

[31] Izhar Wallach, Michael Dzamba, and Abraham Heifets. Atomnet: A deep convolutional neural network for bioactivity prediction in structure-based drug discovery. *arXiv preprint arXiv:1510.02855*, 2015.

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

A. Existence of Covering Templates

In this paper we analyze the expressiveness of networks, *i.e.* the functions they can realize, through the notion of *grid tensors*. Recall from sec. 4 that given *templates* $\mathbf{x}^{(1)}\ldots\mathbf{x}^{(M)}\in\mathbb{R}^s$, the grid tensor of a score function $h_y:(\mathbb{R}^s)^N\to\mathbb{R}$ realized by some network, is defined to be a tensor of order N and dimension M in each mode, denoted $\mathcal{A}(h_y)$, and given by eq. 3. In particular, it is a tensor holding the values of h_y on all instances $X=(\mathbf{x}_1,\ldots,\mathbf{x}_N)\in(\mathbb{R}^s)^N$ whose *patches* $\mathbf{x}_i$ are taken from the set of templates $\{\mathbf{x}^{(1)}\ldots\mathbf{x}^{(M)}\}$ (recurrence allowed). Some of the claims in our analysis (sec. 5) assumed that there exist templates for which grid tensors fully define score functions. That is to say, there exist templates such that score function values outside the exponentially large grid $\{X_{d_1\ldots d_N}:=(\mathbf{x}^{(d_1)},\ldots,\mathbf{x}^{(d_N)}):d_1\ldots d_N\in[M]\}$ are irrelevant for classification. Templates meeting this property were referred to as *covering* (see sec. 5.2). In this appendix we address the existence of covering templates.

If we allow M to grow arbitrarily large then obviously covering templates can be found. However, since in our construction M is tied to the number of channels in the first (representation) layer of a network (see fig. 1), such a trivial observation does not suffice, and in fact we would like to show that covering templates exist for values of M that correspond to practical network architectures, *i.e.* $M\in\Omega(100)$. For such an argument to hold, assumptions must be made on the distribution of input data. Given that ConvNets are used primarily for processing natural images, we assume here that data is governed by their statistics. Specifically, we assume that an instance $X=(\mathbf{x}_1,\ldots,\mathbf{x}_N)\in(\mathbb{R}^s)^N$ corresponds to a natural image, represented through N image patches around its pixels: $\mathbf{x}_1\ldots\mathbf{x}_N\in\mathbb{R}^s$.

If the dimension of image patches is small then it seems reasonable to believe that relatively few templates can indeed cover the possible appearances of a patch. For example, in the extreme case where each patch is simply a gray-scale pixel ($s=1$), having $M=256$ templates may provide the standard 8-bit resolution, leading grid tensors to fully define score functions by accounting for all possible images. However, since in our construction input patches correspond to the receptive field in the first layer of a ConvNet (see fig. 1), we would like to establish an argument for image patch sizes that more closely correlate to typical receptive fields, *e.g.* 5×5. For this we rely on various studies (*e.g.* [32]) characterizing the statistics of natural images, which have shown that for large ensembles of images, randomly cropped patches of size up to 16×16 may be relatively well captured by Gaussian Mixture Models with as few as 64 components. This complies with the common belief that there is a moderate number of appearances taken by the vast majority of local image patches (edges, Gabor filters *etc.*). That is to say, it complies with our assumption that covering templates exist with a moderate value of M. We refer the reader to [5] for a more formal argument on this line.