On the Geometry of Rectifier Convolutional Neural Networks

Matteo Gamba mgamba@kth.se
Hossein Azizpour azizpour@kth.se
Stefan Carlsson stefanc@kth.se
Mårten Björkman celle@kth.se

KTH Royal Institute of Technology
Stockholm, Sweden

Abstract

While recent studies have shed light on the expressivity, complexity and compositionality of convolutional networks, the real inductive bias of the family of functions reachable by gradient descent on natural data is still unknown. By exploiting symmetries in the preactivation space of convolutional layers, we present preliminary empirical evidence of regularities in the preimage of trained rectifier networks, in terms of arrangements of polytopes, and relate it to the nonlinear transformations applied by the network to its input[1].

1. Introduction

The foundational understanding of deep learning is an open problem that has attracted significant attention from the research community [12]. While a unified theory capable of fully accounting for the great empirical success of deep networks still eludes researchers, many disciplines have found successful application in explaining particular aspects of this class of parametric models [4, 8, 16].

Any comprehensive theory aiming at understanding the generalization capability of deep networks must consider three main factors, namely, the network architecture, the training algorithm and the dataset representing a population. Indeed, modern networks are highly engineered for performance on public benchmark datasets, with architectures and activation functions designed to ensure gradient flow, so these factors are deeply entwined [10].

One crucial open question on the generalization power of deep architectures is the search for inductive biases originating from the above three factors [18, 19], and specifically whether the hypothesis space of deep networks (and the parameter configurations reachable by gradient descent) are biased towards a particular family of functions that are posited to generalize well on natural datasets [24].

A prominent field in the study of generalization is geometry, which allows to characterize the expressivity of the hypothesis space of deep networks [5], study the trainability of the parameter space [6] and explain individual predictions of a trained classifier [15]. In particular, preimage studies are concerned with interpreting a given model in terms of its decision boundaries, the resulting classification regions and their complexity [2, 7].

One promising direction for discovering the implicit bias of a classifier is looking for regularities in the preimage of its layers, as a proxy for bias in the learned decision function and its corresponding classification boundaries. In the present work, we exploit symmetries in the preactivation space of convolutional layers of rectifier Convolutional Neural Networks (CNNs) to draw a connection with the non-linear activation of the layers. We lever this observation to define novel statistics on the weights of convolutional layers and study their distribution. Finally, we present preliminary empirical evidence of regularity in the preimage of convolutional layers, which we hypothesize to reflect inductive bias in the function learned by the model.

2. Related work

Geometric studies on the expressivity and complexity of deep networks describe the family of functions computable by a network of fixed depth [17, 20] and provide bounds on the number of linear regions of deep classifiers [21], by studying piece-wise linear activation functions. Theoretical studies on rectifier networks investigate the reachable parameter configurations and their optimality [1]. On the opposite extreme, the study of activations of a trained classifier allows to explain its decisions in terms of its input[15], by inverting individual activations for a given data sample.

Our work is placed in between the above two tracks. We propose a formal analysis of the properties of a CNN based on its weights and carry out a data-driven study on trained networks, to account for actual configurations reached by the optimizer. Recent related work describes the preimage of rectifier CNNs in terms of hyperplane arrangements, and

[1] Source code available at https://github.com/magamba/cones.

argues that specific arrangements of hyperplanes associated with successive convolutional layers can be characterized as nested cones [2]. Our work builds on the same symmetries, to provide empirical evidence of their connection to the transformation computed on the input space by a rectifier CNN. Finally, a related recent paper introduces statistics on the margin of neural classifiers with respect to a given data point and shows that they correlate well with the generalization gap of the model on computer vision datasets [11].

3. Method

We recall the definition of convolutional layer with Rectifier Linear Units (ReLUs) in section 3.1 and use the construction to describe its preimage in section 3.2. In section 3.3, we draw a connection between non-linearity of a ReLU activation and the corresponding preimage. In section 3.4, we introduce continuous statistics over the parameters of a convolutional layer, which are discretized into four finite states and are used to characterize the preimage.

3.1. Convolutional layers

Convolutional layers implement the cross-correlation operation between an input tensor $\mathcal{X} \in \mathbb{R}^{C \times H \times W}$ and one or more kernels (or filters), as described in equation 1. The learned parameters are represented by a 4-way tensor $\mathcal{W} \in \mathbb{R}^{n_{\text{out}} \times n_{\text{in}} \times k \times k}$, where n_{out} represents the number of kernels learned by the layer, n_{in} is the number of input channels and $k \times k$ is the spatial size of each kernel. For input $\mathcal{X}$ of square spatial size $H = W$, the number of local receptive fields is r^2, where $r = (H - k + 2p)/s + 1$, where p represents the amount of zero-padding used and s is the stride with which each kernel is convolved over the input.

For each filter $\mathcal{W}_o := \mathcal{W}[o, :, :, :\ ]$, $o = 1, \ldots, n_{\text{out}}$, the convolutional layer computes

$$\begin{aligned} &\tilde{\mathcal{O}}(o,i,j) = \\ &b_o + \sum_{c=0}^{n_{\text{in}}-1} \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} \mathcal{X}(c, i+m, j+n) \cdot \mathcal{W}_o(c,m,n) \\ &\quad \text{for} \quad i = 0, \ldots, r-1 \quad \text{and} \quad j = 0, \ldots, r-1 \end{aligned} \tag{1}$$

where the cross-channel correlation (given by the outer summation) is computed over the full depth and the spatial cross-correlation is limited to the receptive field. Furthermore, the linear transformation computed by the convolutional filter is typically augmented to an affine transformation, by adding a bias term $b_o \in \mathbb{R}$. Finally, the output of all the n_{out} filters is stacked to produce a tensor $\tilde{\mathcal{O}} \in \mathbb{R}^{n_{\text{out}} \times r \times r}$, corresponding to the preactivation of the layer.

Convolutional layers are typically implemented by vectorizing the tensors $\mathcal{X}$ and $\mathcal{W}$ to exploit fast general matrix multiplication algorithms and then reshaping the product to match the size of the output tensor $\tilde{\mathcal{O}}$ [23].

For a single-kernel single-channel convolution, the input $\mathcal{X} \in \mathbb{R}^{1 \times H \times W}$ is flattened to a column vector $\mathbf{x} := \text{vec}\,(\mathcal{X}) \in \mathbb{R}^{HW}$ and $\mathcal{W} \in \mathbb{R}^{1 \times 1 \times k \times k}$ is reshaped to a matrix $W \in \mathbb{R}^{r^2 \times HW}$, where r^2 denotes the number of local receptive fields in $\mathcal{X}$. Each row of W has at most k^2 nonzero entries, corresponding to the parameters of $\mathcal{W}$.

If $\mathcal{W}$ is convolved with stride $s = 1$, then each row in W is obtained by shifting the components of the previous row one step to the right and W is a Toepliz matrix. Similarly, for a single-kernel multi-channel convolution, $\text{vec}\,(\mathcal{X}) \in \mathbb{R}^{n_{\text{in}}HW}$ and $\mathcal{W}$ is reshaped to a Toepliz matrix $W \in \mathbb{R}^{r^2 \times n_{\text{in}}HW}$.

The preactivation $\tilde{O}_q$ of the convolutional filter on the q-th receptive field, for $q = 0, \ldots, r^2 - 1$, is obtained by multiplying the q-th row W_q^T of W with $\mathbf{x}$

$$W_q^T \mathbf{x} + b =: \tilde{O}_q \in \mathbb{R} \tag{2}$$

Finally, for multi-kernel multi-channel convolutions, the weight matrix W is obtained by stacking the individual weight matrices corresponding to each filter $\mathcal{W}_o$, and the resulting $W \in \mathbb{R}^{n_{\text{out}} r^2 \times n_{\text{in}} HW}$ is block-circulant.

3.2. ReLU activations

To take advantage of the compositional topology of neural networks, a non-linear activation function φ is applied element-wise to $\tilde{\mathcal{O}}$ to produce an activation $\mathcal{O} \in \mathbb{R}^{n_{\text{out}} \times r \times r}$.

For rectifier CNNs, the use of ReLU activations $\varphi(x) := \max(0, x)$ induces affine hyperplanes in the preactivation space of vec $(\mathcal{O})$. In fact, for each component $\mathcal{O}_i$ of vec$(\mathcal{O})$, the ReLU identifies two affine halfspaces $X_q^+ = \{\mathbf{x} \in \mathbb{R}^{n_{\text{in}}HW} \mid W_q\mathbf{x} + b \geq 0\}$ and $X_q^- = \{\mathbf{x} \in \mathbb{R}^{n_{\text{in}}HW} \mid W_q\mathbf{x} + b < 0\}$ delimited by the hyperplane with normal vector W_q^T, translated by b. If $\mathcal{O}_i > 0$, then its preimage corresponds to one point in the preactivation space, lying in the halfspace X_q^+. If instead $\mathcal{O}_i \leq 0$, the preimage is mapped to the entire halfspace X_q^-. Hence, the activation $\mathcal{O}$ can be described by the position of $\mathbf{x}$ with respect to the hyperplanes identified by each W_q^T, $q = 0, \ldots, r^2 - 1$ [2].

In particular, for a multi-channel convolution $\mathcal{W}$ with stride $s = 1$, the corresponding Toepliz matrix W identifies r^2 distinct hyperplanes in $\mathbb{R}^D$, with $D = n_{\text{in}}HW$ such that:

$$\begin{cases} w_0x_0 + w_1x_1 + \ldots + w_{D-1}x_{D-1} & + b \geq 0 \\ w_{D-1}x_0 + w_0x_1 + \ldots + w_{D-2}x_{D-1} & + b \geq 0 \\ \vdots & \vdots \\ w_{D-r^2+1}x_0 + \ldots + w_{D-r^2}x_{D-1} & + b \geq 0 \end{cases} \tag{3}$$

The high sparsity of each row, arising from the small spatial size of $k \ll H$, guarantees that the Toepliz property holds in practice for equation 3 for valid convolutions.

Thus, starting from the normal vector W_0^T to the first hyperplane, the m-th hyperplane is obtained by cyclically shifting the components of W_0^T by $m-1$ positions to the right, resulting in a set of intersecting hyperplanes that are arranged symmetrically around the identity line of $\mathbb{R}^D$.

For the general case of multi-channel convolutions, the resulting hyperplane arrangement describes convex polytopes in the preactivation space. Furthermore, for single-channel kernels, if appropriate zero padding is used to preserve the spatial dimensionality of the input, i.e. $r^2 = HW$, the resulting matrix W is square and the corresponding hyperplanes are arranged as a polyhedral cone in the input space. The intersection of all faces of the cone gives its apex $\mathbf{v} = (-\frac{b}{a}, \ldots, -\frac{b}{a})$ lying on the identity line of $\mathbb{R}^D$, with $a = \sum_{i=0}^{HW} W_{qi}$.

3.3. Arrangement of polytopes and non-linearity of activations

For a fixed weight assignment $\mathcal{W}^l$ of a convolutional layer l and an input $\mathbf{x}$, the ith channel $\mathcal{O}_i^l$ of the activation $\mathcal{O}^l$ is determined by the relative position of $\mathbf{x}$ with respect to the faces of the polytope corresponding to the ith kernel of $\mathcal{W}^l$. As highlighted in the previous section, if $\mathbf{x}$ belongs to the intersection of the positive halfspaces induced by the faces of the polytope, then each of its components is transformed linearly by the ReLU. Each component of $\mathbf{x}$ lying in a negative halfspace is instead contracted to 0.

Let l be a convolutional layer with n_1 filters and $l+1$ be another convolutional layer learning n_2 filters. When stacking l and $l+1$, the i-th channel $\mathcal{O}_i^{l+1}$ of $\mathcal{O}^{l+1}$ depends on the relative position of $\mathbf{x}$ with respect to the n_1 polytopes $\mathcal{C}_1^l, \ldots, \mathcal{C}_{n_1}^l$ of layer l and the polytope $\mathcal{C}_i^{l+1}$ of layer $l+1$. If the spatial dimensionality is preserved by using zero-padding, then all the polytopes can be represented in the input space of $\mathbf{x}$, each with its apex on the identity line.

For a set $A \subseteq \mathbb{R}^D$, let $A^\complement$ denote the complement of A in $\mathbb{R}^D$. Given two polytopes $\mathcal{C}_j^l$ and $\mathcal{C}_i^{l+1}$, we distinguish four cases:

1. If $\mathcal{C}_i^{l+1}$ is fully contained in $\mathcal{C}_j^l$, then the region of input space $(\mathcal{C}_i^{l+1})^\complement \setminus (\mathcal{C}_j^l)^\complement$, which is linearly transformed by $\mathcal{C}_j^l$, is now contracted by $\mathcal{C}_i^{l+1}$. Hence, the convolutional kernel of $\mathcal{C}_i^{l+1}$ is effectively contributing non-linearity to the transformation computed by the network. We denote this state OUT_FULL_IN.

2. If $\mathcal{C}_j^l$ is instead fully contained in $\mathcal{C}_i^{l+1}$, then the latter kernel is not contributing additional non-linearity to the network. We call this state IN_FULL_OUT.

3. If $\mathcal{C}_j^l$ and $\mathcal{C}_i^{l+1}$ have partial non-empty intersection and the apex of $\mathcal{C}_i^{l+1}$ is inside the convex hull of $\mathcal{C}_j^l$, then each dimension of the region $(\mathcal{C}_i^{l+1})^\complement \setminus (\mathcal{C}_j^l)^\complement$ is contracted to 0. We call this OUT_PARTIAL_IN.

4. If $\mathcal{C}_j^l$ and $\mathcal{C}_i^{l+1}$ have partial non-empty intersection, but the apex of $\mathcal{C}_i^{l+1}$ is this time outside the convex hull of $\mathcal{C}_j^l$, then additional non-linearity is contributed only in the regions $(\mathcal{C}_i^{l+1})^\complement \cap \mathcal{C}_j^l$. This state is called IN_PARTIAL_OUT.

In the next section, we introduce continuous statistics that allow to detect the above four states for polyhedral cones.

Finally, it is worth noting that hyperplane arrangements can also be used to characterize the preimage of fully connected layers, but their computational complexity grows exponentially with the depth of the network [3].

3.4. Statistics over convolutional weights

In this section, we introduce three continuous statistics over the weights of a pair of stacked convolutional layers, that can be used to describe the mutual position of two polyhedral cones $\mathcal{C}_j^l$ and $\mathcal{C}_i^{l+1}$, in terms of the affine alignment of their convex hulls, expressed by a translation component and two planar angles.

- Let $\mathbf{v}_i$ and $\mathbf{v}_j$ resp. denote the apices of $\mathcal{C}_i^{l+1}$ and $\mathcal{C}_j^l$ in $\mathbb{R}^D$. The absolute distance $||\mathbf{v}_j - \mathbf{v}_i||_2$, denotes the translation along the identity line needed to move one vertex onto the other. To allow for comparisons that are independent from the input space dimensionality D, the distance is scaled by $\frac{1}{D}$. In the experiments, the maximum distance over all pairs of kernels is used to normalize the absolute distance to $[0, 1]$.

- The inclusion and intersection of the two cones depends on the opening angle α_i (resp. α_j) of each cone, which we compute as the angle between the identity line and any face of the cone. Since the angle is bound in $[0, \frac{\pi}{2}]$, we express the offset between the two cones as $\frac{2}{\pi}|\alpha_i - \alpha_j|$.

- The alignment of two cones depends on the rotation angles of each cone around its axis. In D dimensions, there are $D-2$ possible angles for each cone. Given the hyperplane corresponding to any face of $\mathcal{C}_i^{l+1}$, we aggregate the rotation angle by computing the minimum angle of rotation that transforms the hyperplane into a face of $\mathcal{C}_j^l$. The result is then normalized by π.

We observe that the relative arrangement of cones is fully determined by the weights of corresponding kernels.

Thanks to the continuous measures introduced above, for each pair $(\mathcal{C}^l, \mathcal{C}^{l+1})$, the alignment of the two polytopes can be described by the four discrete states introduced in section 3.3. In section 4, we empirically study the distribution of the four states for two families of CNN architectures.

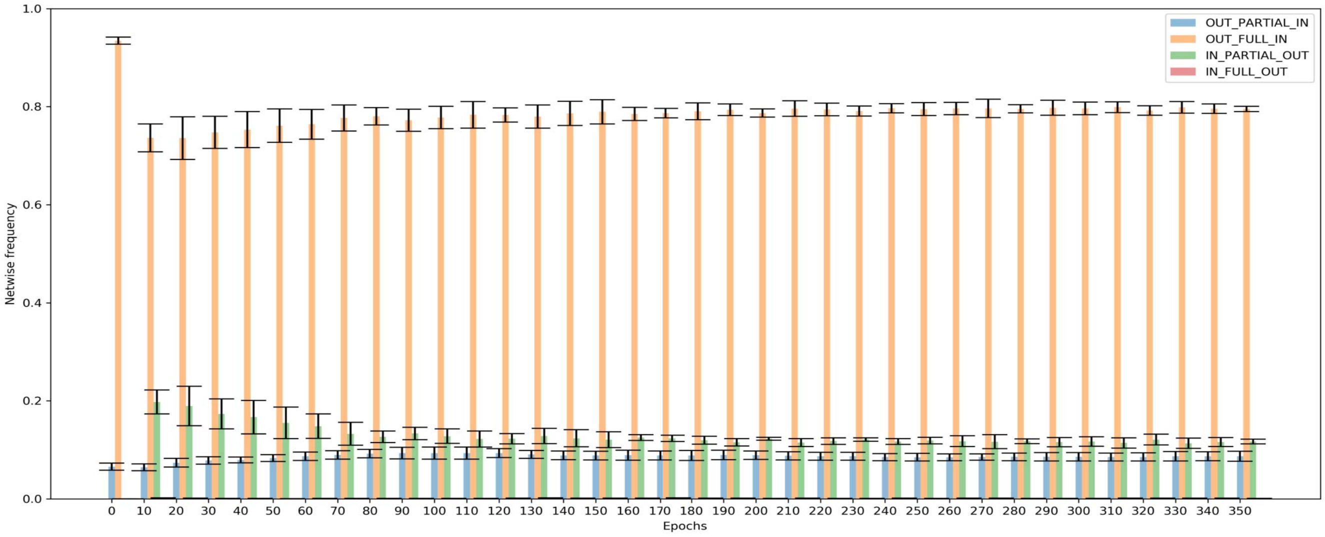

Figure 1: Discrete distribution for a 9-layer LeNet on CIFAR-10, every 10 epochs. The results are averaged over 5 runs.

4. Evaluation

To evaluate our methodology, we train CNNs based on LeNet [14] and VGG [22] on CIFAR-10 [13].

Given a pair of stacked convolutional layers l and $l+1$, equation 1 shows that the oth channel of each kernel of the second layer is convolved with the oth channel of $\mathcal{O}^l$. This in turn is the result of convolving the oth kernel of layer l, composed of n_l channels, with its input $\mathcal{O}^{l-1}$ over the full depth. Hence, then mutual arrangement of the polytope $\mathcal{C}_o^{l+1}$ is computed with respect to the n_l polytopes $\{C_p^l\}_{p=0}^{n_l-1}$ corresponding to the channels of the oth kernel of layer l.

For the discrete states, the arrangement of $\mathcal{C}_o^{l+1}$ is computed w.r.t. the conical combination of $\{C_p^l\}_{p=0}^{n_l-1}$. Finally, to restrict the study to the affine arrangement of polyhedral cones and make our theory as tight as possible, convolutional layers are implemented by swapping the order of cross-channel correlation and ReLU activation, according to equation 4. This allows to investigate for any bias arising from the optimization process in the arrangement of cones.

$$\mathcal{O}(o,i,j) = b_o + \sum_{c=0}^{n_{\text{in}}-1} \varphi\Big(\sum_{m=0}^{k-1}\sum_{n=0}^{k-1} \mathcal{X}(c,i+m,j+n)\cdot \mathcal{W}_o(c,m,n)\Big)$$
$$\text{for} \quad i=0,\dots,r-1 \quad \text{and} \quad j=0,\dots,r-1 \tag{4}$$

The distribution of the four discrete states is studied at initialization and throughout training of two networks, one based on VGG and one on LeNet, each with 9 total learned layers (described in section B of the supplemental material).

First, we observe that the high frequency of state OUT_FULL_IN at initialization arises from the zero-initialization of the bias terms of each layer and the use of conical combinations of polytopes $\{C_p^l\}_{p=0}^{n_l-1}$. For this reason, we focus our evaluation on trained weights. Figure 1 shows that, as training progresses, the mutual arrangement of cones changes in the early epochs to stabilize towards full inclusion of polyhedral cones into the interior of those at the previous layer. Furthermore, around 10% of cones has partial non-empty intersection with their predecessors, while another 10% (case IN_PARTIAL_OUT) are less effective in contributing nonlinearity to the transformation computed by the network. The key insight is that optimization drives the convolutional parameters towards the case OUT_FULL_IN, in which each channel effectively contracts a new region of the input space, that was transformed linearly by kernels at the previous layer. Finally, the error bars show that the variance of the observed states across different runs is reduced as training progresses. Similar trends are observed for our VGG-like network and LeNet7 (see supplemental material, section D).

5. Conclusion

Based on symmetries in the preimage of convolutional layers, we propose discrete statistics to study the mutual arrangement of the hyperplanes that define ReLU activations. Our special CNNs allow to assess our methodology as tightly as possible. We show that training on natural image data changes the arrangement of the cones in a way that is biased towards configurations that efficiently contribute non-linearity to the function learned by the network. We are currently investigating to what extent our findings reflect the actual bias of state of the art CNNs.

Acknowledgement. This work is partially funded by the Swedish Research Council project 2017-04609 and Wallenberg AI, Autonomous Systems and Software Program (WASP).

References

[1] Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. *arXiv preprint arXiv:1611.01491*, 2016.

[2] Stefan Carlsson. Geometry of deep convolutional networks. *CoRR*, abs/1905.08922, 2019.

[3] Stefan Carlsson, Hossein Azizpour, and Ali Razavian. The preimage of rectifier network activities. 2016.

[4] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pages 192–204, 2015.

[5] Nadav Cohen, Or Sharir, and Amnon Shashua. On the expressive power of deep learning: A tensor analysis. In *Conference on Learning Theory*, pages 698–728, 2016.

[6] Nadav Cohen and Amnon Shashua. Convolutional rectifier networks as generalized tensor decompositions. In *International Conference on Machine Learning*, pages 955–963, 2016.

[7] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, Pascal Frossard, and Stefano Soatto. Classification regions of deep neural networks. *arXiv preprint arXiv:1705.09552*, 2017.

[8] Anna C Gilbert, Yi Zhang, Kibok Lee, Yuting Zhang, and Honglak Lee. Towards understanding the invertibility of convolutional neural networks. *arXiv preprint arXiv:1705.08664*, 2017.

[9] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[11] Yiding Jiang, Dilip Krishnan, Hossein Mobahi, and Samy Bengio. Predicting the generalization gap in deep networks with margin distributions. *arXiv preprint arXiv:1810.00113*, 2018.

[12] Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. Generalization in deep learning. *arXiv preprint arXiv:1710.05468*, 2017.

[13] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55, 2014.

[14] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[15] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015.

[16] Charles H Martin and Michael W Mahoney. Rethinking generalization requires revisiting old ideas: statistical mechanics approaches and complex learning behavior. *arXiv preprint arXiv:1710.09553*, 2017.

[17] 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.

[18] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 5947–5956, 2017.

[19] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*, 2014.

[20] Razvan Pascanu, Guido Montufar, and Yoshua Bengio. On the number of response regions of deep feed forward networks with piece-wise linear activations. *arXiv preprint arXiv:1312.6098*, 2013.

[21] Thiago Serra, Christian Tjandraatmadja, and Srikumar Ramalingam. Bounding and counting linear regions of deep neural networks. *arXiv preprint arXiv:1711.02114*, 2017.

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

[23] Aravind Vasudevan, Andrew Anderson, and David Gregg. Parallel multi channel convolution using general matrix multiplication. In *2017 IEEE 28th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 19–24. IEEE, 2017.

[24] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.