Probing a Deep Neural Network

Chapter · January 2020

DOI: 10.1007/978-981-13-8950-4_19

5 authors, including:

Francesco Palmieri
Università degli Studi della Campania "Luigi Vanvitelli
178 PUBLICATIONS **1,867** CITATIONS

SEE PROFILE

Amedeo Buonanno
Università degli Studi della Campania "Luigi Vanvitelli
17 PUBLICATIONS **30** CITATIONS

SEE PROFILE

Probing a Deep Neural Network

Francesco A. N. Palmieri, Mario Baldi, Amedeo Buonanno,
Giovanni Di Gennaro and Francesco Ospedale

Università della Campania "Luigi Vanvitelli" (ex SUN)
Dipartimento di Ingegneria Industriale e dell'Informazione,
via Roma 29, 81031 Aversa (CE) - Italy
{amedeo.buonanno,giovanni.digennaro,francesco.palmieri}@unicampania.it,
{mario.baldi,francesco.ospedale}@studenti.unicampania.it

Abstract. We report a number of experiments on a deep convolutional network in order to understand the transformations that emerge from learning at the various layers. We analyze the backward flow and the reconstructed images, with various algorithms. We focus on the field of view of specific neurons, also using random parameters, in order to reveal the role of the activations that emerges in the multi-layer structure.

Keywords: Deep Learning, Neural Networks, Unsupervised Learning

1 Introduction

Deep Convolutional Networks (DCN) have recently shown outstanding performances in pattern recognition on images and signals. The results are attributed mainly to the multi-layer convolutional structure of the network and to a number of "tricks" in the learning algorithms (dropout, unsupervised initialization, etc.)[1]. However many open questions remain on the reasons why some structures perform so well in supervised learning and why the choice of parameters, such as number of layers, connectivity, stride in the convolutions, dimensionality of the feature spaces, pooling dimensions, type of non linearity, etc, may drastically affect the recognition performances. Most of the successful results have been reached after many lengthy and computationally expensive trials on configurations and learning algorithms, which reveals that there is still a fundamental lack of tools for understanding the basic functioning of the network, even if probes into the network activities[2], and partial analyses[3][4] have been presented in the literature.

Even if more complicated architecture such as LSTM [5], or Wavenets [6], that use computational units formed by decision nodes combined with linear units have been proposed, we limit ourselves in this work to a structure with standard linear combiners and rectifiers (RELUs), leaving the study of other paradigms to future works.

In the architecture that we analyze in this paper the peculiar role is played by the RELUs that provide an unfolding of the input space with the activations being the essence of how the information is stored at the various layers. We

demonstrate that the representation power of a multi-layer network is not so much in the specific parameters, but mostly in the configurations of activations that emerge at the various levels. It has been demonstrated that the hyperplane arrangements at the various stages correspond to a number of linear regions that grows exponentially with the number of layers [7][8]. This is in contrast to single-layer networks that, even though they have universal approximation capabilities [9], would require a very large number of units.

Visualization on actual images can provides many clues, specially when specific activations are propagated backward onto the input image [2], in an attempt to understand the specific features related to various regions of the network. In this paper we follow this idea and report some experiments on the backward propagation of the activations at specific neurons in the last layers, using various strategies for the backward flow. We focus on a specific pre-trained network, the VGG-16 [10] and modify the so-called *Deconvnet* process [3], using also random parametrizations. We demonstrate that the information stored in the network is mostly related to the activation configurations, rather than to specific parameters. By looking at the field of view (FOV), we show that the local linearities involved into specific activations, extract gradient-like statistics on image regions, rather then representing the storage of sub-images, as it would be if we had a designed a bank of matched filters. This seems to confirm why in computer vision SIFT-like algorithms [11] that use Histogram of Gradients (HOGs) were the state-of-the-art before being overcome by adaptive deep networks. The deep networks seem to learn through massive training similar kinds of image statistics.

In Section 2 we review the multi-layer architecture and in Section 3 we describe the experiments with the backward flow. In Sections 4 and 5 we compute best input selections and compute some activation statistics. Section 6 reports comments and conclusions.

2 Multi-Layer Convolutional Networks

A DCN for supervised learning on images is composed by a first part of convolutional layers, intermingled with pooling sections, and by a final stage of a few (at least two) fully-connected layers for final classification. Figure 1 shows the popular network VGG-16 [10]. The first part, that is sometimes trained using unsupervised algorithms, and subsequently refined with supervised information backpropagated from the last layers, provides the mapping of the image to a feature space that is more prone to be used by the supervised part. It is well known that a classifier directly connected to the image would perform poorly and that parameter extraction is the crucial component of any pattern recognition system. The understanding of this transformation is crucial for identifying the real peculiarity of the DCN, also in comparison to the many other parameter extraction criteria presented in decades of literature in machine vision. In a DCN at each layer i we have a 3D array of size $N^i \times M^i \times D^i$, $i = 0, ..., L$. To fix ideas and numbering, we focus on the VGG-16 in Figure 1 where we have layers

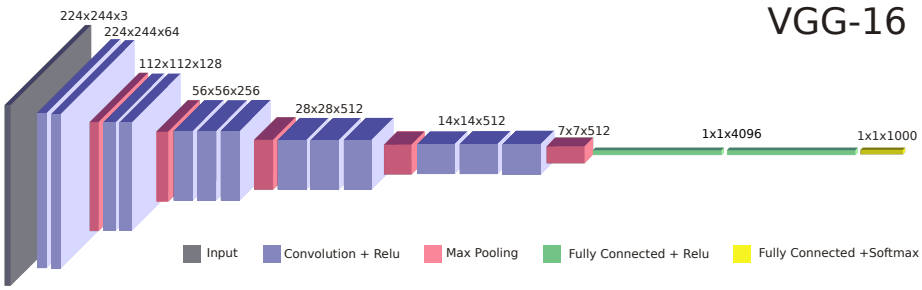

Fig. 1. The typical VGG-16 multi-layer DCN structure

0, 1, 2, (P3), 4, (P5), 6, 7, (P8), 9, 10, (P11), 12, 13, (P), 14, 15, (16S), where (P) indicates pooling, (Pi) pooling + linear+ RELU and (iS) linear+ softmax. In the first 13 layers we have progressively smaller images and larger depths $(D^0, ..., D^{13}) = (3, 64, 64, 128, 128, 256, 256, 256, 512, ..., 512)$. Max pooling is applied after groups of layers on 2x2 non overlapping masks, causing each time a reduction of the image size to a half (the max is taken on each feature slice separately). At the end there are three fully connected layers that loose the spatial dimension $(N^{14}, N^{15}, N^{16}) = (M^{14}, M^{15}, M^{16}) = (1, 1, 1)$ and have depths $(D^{14}, D^{15}, D^{16}) = (4096, 4096, 1000)$. All the linear layers in the first part are convolutional and use patches of size 3x3 and stride (overlap) one. Convolutions are applied in such a way that the image size is preserved by padding the outside with zeros at the boundaries. After each convolution a rectifier (RELU) is applied.

To write compact equations for the network, we reduce the 3D arrays to vectors x_i of sizes $n_i = N^i M^i D^i$, $i = 0, ..., L$ (here $L = 16$). Therefore a layer without pooling is described by the equation

$$x_i = r_i \odot (W_i x_{i-1} + b_i), \tag{1}$$

where $\odot$ is the Hadamard product (element-by-element), W_i is an $n_i \times n_{i-1}$ real matrix and b_i is and n_i-dimensional bias vectors. RELU rectification $R(x) = max(0, x)$ is described by an n_i-dimensional binary vector r_i with zeros and ones. Clearly r_i is data-dependent and represents the activation mask for every instance. A layer with pooling is described instead by the equation

$$x_i = r_i \odot (W_i P_i x_{i-1} + b_i), \tag{2}$$

where W_i is an $n_i \times n_i$ real matrix and the pooling operation is represented by a $n_i \times n_{i-1}$ sub-permutation matrix P_i (a sub-permutation matrix is such that each entry is either 1 or 0, each row contains only one 1 and each column contains at most one 1). Matrix P_i is clearly data-dependent and contains information on the position of all the local maxima. The last layer is described by the equation

$$x_L = S(W_L x_{L-1} + b_L), \tag{3}$$

where $S(y)$ is the softmax operation applied to the vector y.

When an input is presented to the network (layer zero), the information is propagated forward, and only the linear combinations that produce a positive output for the RELUs influence the following stages. Max-pooling produces a reduction in size letting through only one out of four activations. The code that emerges at the various layers is very sparse as shown in Figure 2 where, after the color input image, we show slices at various depths for layers 1, 4, 7, 9 and 13. The first layer seems to detect edges and elementary local configurations, while in the following stages progressive matching of the input space at larger scales is obtained. The question is: what is the image feature detected at a specific location in layer 13? To provide a partial answer to this question, we observe

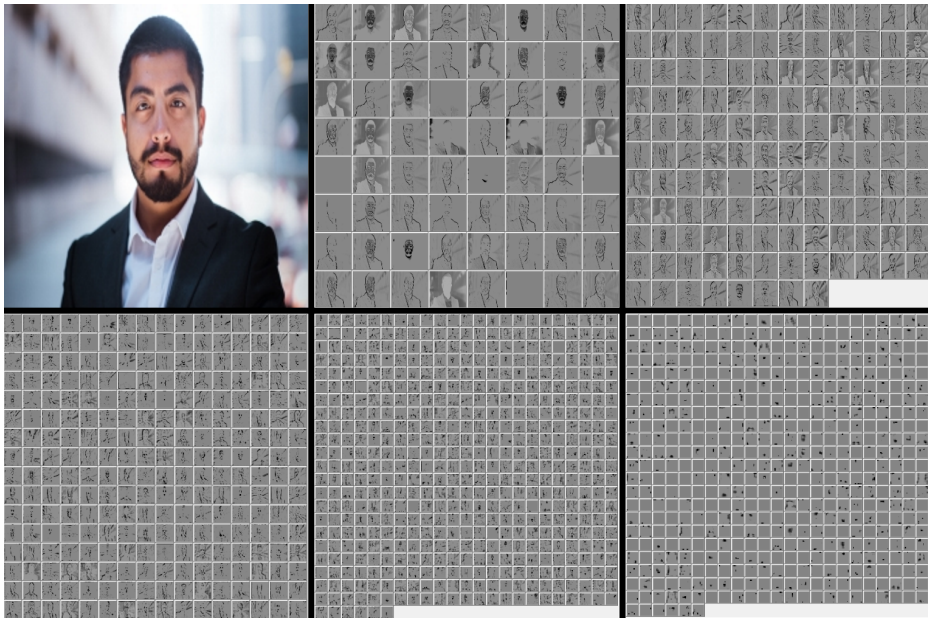

Fig. 2. Activations in the VGG-16 for an image presented at the input. The images represent activations at layers 1, 4 , 7, 9 and 13 respectively

that when a pixel in the last layer (13) is active, there is a complex tree of activations that causes that pixel to become alive. An activation tree is shown in figure Figure 3 on a simplified network where the connections that potentially can contribute to a specific output in the last layer are colored. They identify the Field Of View (FOV) of that unit. However for each input, a pattern of activations emerges and only a specific subset of these connections play a role (circled in Figure 3). They identify a locally linear (affine) transformation that represents the specific filter for the pattern present in that FOV of the image. Observe that there may be many different configurations within that FOV tree that can cause the final unit to become active. Therefore the unit at the end is

a sort of logic OR of many intermediate configurations. The network performs a hierachical coding in activating different sets of units in the intermediate layers.

The understanding of the structure of the activations is crucial to capture the

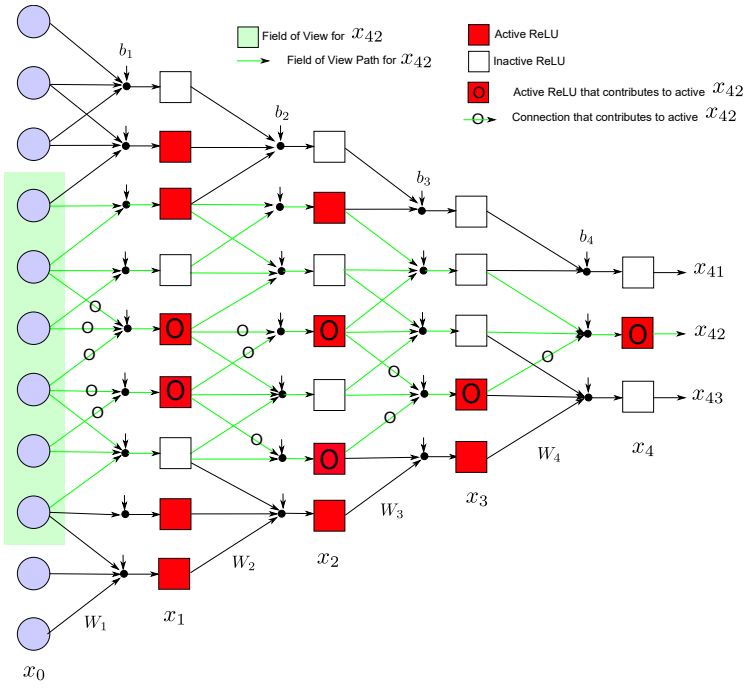

Fig. 3. Schematic network where for x_{42} is shown the FOV with its connection tree (green). On a specific instance of x_0 in which x_{42} becomes active, the pictures distinguishes among the active ReLUs and the ones that contribute to activate x_{42}. Also the contributing weights within the FOV are marked with a circle.

nature of the transformation implemented by the network, which is peculiar to the multi-layer architecture in contrast to a shallow one-layer bank of filters. To shed light on some of these issues we have performed a number of experiments in backpropagating the information from layer 13 backward to the image plane in the VGG-16.

3 Backward Reconstructions

Deconvnet In a first set of experiments, after having presented an image to the network and produced the activations in all layers, we have used *Deconvnet*, an algorithm proposed in [4], to obtain the image reconstructed form an activated output. We do not propagate back the whole activation, but we choose a specific location in layer 13 and a specific feature. From that element we propagate x_{13} backward after setting to zero all the others elements of the vector. Such a vector

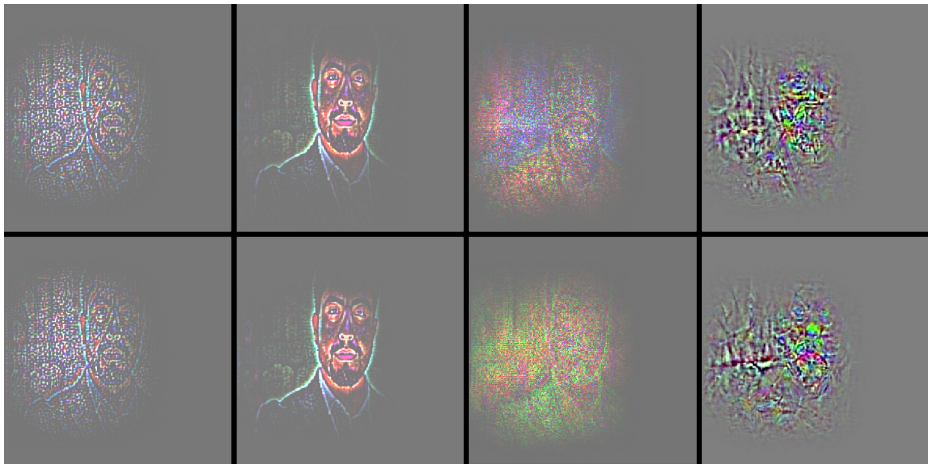

Fig. 4. Results on backward reconstruction from a specific neuron in layer 13 and two different features (upper and lower row). The image is obtained using (col. 1) Deconvnet; (col. 2) Deconvnet with binary masks; (col. 3) Deconvolution with random weights and masks. Column 4 shows the exact filters activated by the FOV.

is denoted with x^b_{13} and similarly x^b_i denote all the intermediate activations in the backward flow. The process at the layer where no pooling is present, can be compactly described by the equation

$$x^b_{i-1} = R(W_i^T x^b_i), \tag{4}$$

where R represents the RELU function applied during the backward propagation process, and in the layers with pooling by the equation

$$x^b_{i-1} = P_i^T R(W_i^T x^b_i). \tag{5}$$

All the activations are confined to the FOV tree of the chosen location. Note that the backward operation through the *Unpooling* (P_i^T) needs uses local information from the forward flow. Note that the RELU operations applied to the backward flow by no means guarantee that the same units are active in the backward flow in comparison to the forward flow. This is because the matrices are not orthonormal (the transpose is not the inverse). However a faded and stylized reconstruction appears anyhow in the image plane within the FOV as in Figure 4(col. 1), for one location and two different features.

Deconvnet With Masks In an attempt to better separate the roles of weights and activations we have run the Deconvnet algorithm storing also the activations in the forward flow, i.e. non relaying only on the RELUs applied to the backward flow. More specifically if we conserve the binary masks r_i and the max-pooling sub-permutations P_i for every layer, the backward flow at each layer with no pooling can be described by the equation

$$x^b_{i-1} = R(W_i^T (r_i \odot x^b_i)), \tag{6}$$

where $\odot$ denote the Hadamard product (element-by-element), and in the layers with pooling by the equation

$$x_{i-1}^b = P_i^T R(W_i^T(r_i \odot x_i^b)). \tag{7}$$

Figure 4(col. 2) shows the reconstructions revealing that the activations masks influence greatly this attempted inversion. Vestiges of the image appear more clearly as information makes it better through the "holes" determined by the forward activations. This seems to show that the essence of the backward reconstruction should be heavily based on the activation patterns, rather than on the specifics of the matrix inversions.

Deconvnet with masks and random parameters To investigate further the issue we have randomized the weights in the backward flow and used only the activation masks. The equations for the reconstructions are again (6) and (7). To keep values in a reasonable range, weights and biases have been randomly chosen according to a gaussian distribution with means and variances that match the learned network weights. Typical results are shown in Figure 4(col. 3). It is quite striking to see that an image in the FOV still appers confirming that the patterns of activations may be the most important means by which the image is coded in the network structure! Note that in this case no values are used, but only random numbers going though the masks.

3.1 Filter Reconstruction

A network made of linear combiners and RELUs implements functions that are piecewise linear (actually affine because of the biases). Every time an input is presented to the network a region of linearity is involved in the very high-dimensional input space. We know that the number of regions that characterize a multi-layer network grows exponentially with the number of layers [7]. If we focus on one activation in the last layer it must be then the composition of many linear regions. Now if linear region represents a feature revealed at the end of the chain on that neuron the natural the question is: how do these linear regions look like in the input space of the FOV of that neuron? Do they resemble image patterns, or they provide a statistical account of that region?

In ou attempt to shed some light on this issue we have performed a number of reconstruction experiments in a way that is different from the above deconvnet algorithm and its variations. More specifically we have taken a delta activation in the last layer (1 for one location and one feature only, and zero else) and using the masks memorized in the forward flow for a specific image we have used the following equations in the backward process. At each layer with no pooling we use the equation

$$x_{i-1}^b = (W_i^T(r_i \odot x_i^b)) \tag{8}$$

and in the layers with pooling the equation

$$x_{i-1}^b = P_i^T(W_i^T(r_i \odot x_i^b)). \tag{9}$$

The information goes backward through the activations and computes exactly the local linear combination activated in that specific instance. The results for two different features from the 13th layer are shown in Figure 4(col. 4). It is very interesting to look at the results because the activated filter is nothing like an image patch, rather a very alternating pattern that seems to compute local statistics. This is particularly reveling because just as in some the feature extractors used in the computer vision such as the SIFT algorithm [11], the filter seems to be computing local gradients. Extensive training of the VGG-16 network has automatically discovered this feature to be relevant for classification.

4 Best Input Selection

We have already pointed out that the activation of a specific neuron can be seen as the logic OR of possibly very many different image configurations. In order to reveal the kind of patterns that can activate a specific neuron, we have used an optimization algorithm to get a partial answer. More specifically, as suggested in [2], for the pre-trained VGG-16 we search for the image that maximally excite a specific neuron x_{13}^i in the last layer. More specifically

$$x_0^* = \text{argmax}_{x_0} x_{13}^i. \tag{10}$$

The algorithm starts form a random image and it is updated according to a simple gradient ascent criterion. The value of x_{13}^i is backpropagated, just as in standard backpropagation, through the network from the 13th layer to the input. Every time the algorithm is started form a different random image, it converges to a different configuration. Figure 6 shows the results of three runs for the same neuron. The pictures show clearly within the FOV of that neuron a filter that does not resemble the image at all, but a rather complex linear transformation. This should be seen as a different way of obtaining a filter as in Subsection 3.1 without reference to a specific image.

5 Activation Statistics

In this last section we have investigated how the various features are activated at various layers in the VGG16 for a typical image. This analysis has the objective of understanding the approximate cardinality of the coding that emerges in a multi-layer convolutional network. In the first convolutional layer we see that about a half of the 64 features are activated. The number grows in the second convolutional layer and remains approximately the same in percentage after pooling to return to about a half in the subsequent layer. After the second pooling we see that the fraction remain about the same with a tendency to go towards a sparse code towards the end.

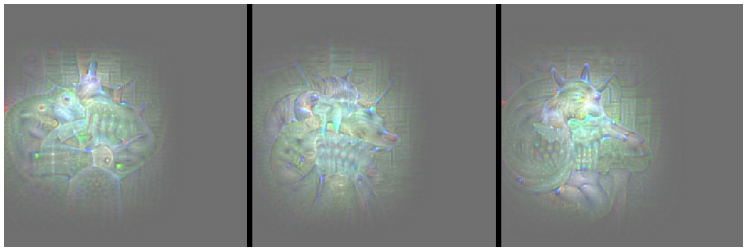

Fig. 5. Images resulting from the maximization of a specific neuron in the 13th layer.

6 Comments and Conclusions

In this work we have conducted a number of experiments on a deep convolutional neural network with the objective of obtaining a better understanding of the inner transformations that are learned through extensive training. The motivation is in the striking performances that these system provide in image recognition. We have focused on a specific pre-trained network, the VGG-16, and focused on the activations of specific neurons in layer 13, that are located at the end of the feature extraction part. The experiments carried out in propagating backward the information through the network in various ways, have revealed a number of interesting characteristics that should be common to most deep convolutional networks. The crucial role of the activations and no so much the parameters has been evidences in experiments also with randomly chosen parameters. Also the filters that become activated seem to compute statistics on image patches rather memorize specific patterns. Much still needs to learned about the working of these complex architectures in our current effort of obtaining a more comprehensive quantitative account for analysis and design.

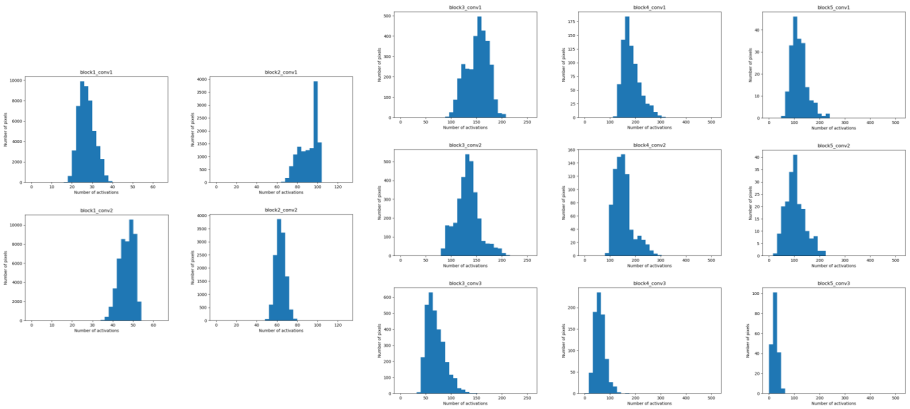

Fig. 6. Histograms on the number of features activated at the various layer in the VGG-16 for a typical image

References

1. T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao, "Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review," *International Journal of Automation and Computing*, vol. 14, pp. 503–519, Oct 2017.
2. J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding Neural Networks Through Deep Visualization," *ArXiv e-prints*, June 2015.
3. M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), (Cham), pp. 818–833, Springer International Publishing, 2014.
4. M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *2011 International Conference on Computer Vision*, pp. 2018–2025, Nov 2011.
5. S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
6. A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," *ArXiv e-prints*, Sept. 2016.
7. G. Montúfar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, (Cambridge, MA, USA), pp. 2924–2932, MIT Press, 2014.
8. G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 2924–2932, Curran Associates, Inc., 2014.
9. G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, pp. 303–314, Dec 1989.
10. K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *ArXiv e-prints*, Sept. 2014.
11. D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, pp. 1150–1157 vol.2, 1999.