GEOMETRY OF DEEP CONVOLUTIONAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

We give a formal procedure for computing preimages of convolutional network outputs using the dual basis defined from the set of hyperplanes associated with the layers of the network. We point out the special symmetry associated with arrangements of hyperplanes of convolutional networks that take the form of regular multidimensional polyhedral cones. We discuss the efficiency of large number of layers of nested cones that result from incremental small size convolutions in order to give a good compromise between efficient contraction of data to low dimensions and shaping of preimage manifolds. We demonstrate how a specific network flattens a non linear input manifold to an affine output manifold and discuss its relevance to understanding classification properties of deep networks.

1 INTRODUCTION

Deep convolutional networks for classification map input data domains to output domains that ideally correspond to various classes. The ability of deep networks to construct various mappings has been the subject of several studies over the years (1; 3; 10) and in general resulted in various estimates of capacity given a network structure. The actual mappings that are learnt by training a specific network however, often raise a set of questions such as why are increasingly deeper networks advantageous (13; 14) ? What are the mechanisms responsible for the successful generalisation properties of deep networks ? Also the basic question why deep learning over large datasets is so much more effective than earlier machine learning approaches is still essentially open, (7). These questions are not in general answered by studies of capacity. A more direct approach based on actual trained networks and the mappings they are efficiently able to produce seems needed in order to answer these questions. It seems ever more likely e.g that the ability of deep networks to generalize is connected with some sort of restriction of mappings that they theoretically can produce and that these mappings are ideally adapted to the problem for which deep learning has proven successful, Due to the complexity of deep networks the actual computation of how input domains are mapped to output classifiers has been considered prohibitively difficult. From general considerations of networks with rectifier (ReLU) non linearities we know that these functions must be piecewise linear (10) but the relation between network parameters such has convolutional filter weights and fully connected layer parameters and the actual functions remains largely obscure. In general, work has therefore been concentrated on empirical studies of actual trained networks (6; 8; 9)

Recently however there have been attempts to understand the relation between networks and their mapping properties from a more general and theoretical point of view. This has included specific procedures for generating preimages of network outputs (4) and more systematic studies of the nature of piecewise linear functions and mappings involved in deep networks, (2; 11; 15).

In this work we will make the assertion that understanding the geometry of deep networks and the manifolds of data they process is an effective way to understand the comparative success of deep networks. We will consider convolutional networks with ReLU non linearities. These can be completely characterised by the corresponding hyperplanes associated with individual convolutional kernels . We will demonstrate that the individual arrangement of hyperplanes inside a layer and the relative arrangement between layers is crucial to the understanding the success of various deep network structures and how they map data from input domains to output classifiers.

We will consider only the convolutional part of a deep network with a single channel. We will assume no subsampling or max pooling. This will allow us to get a clear understanding of the role of the convolutional part. A more complete analysis involving multiple channels and fully connected layers is possible but more complex and will be left to future work.

The focus of our study is to analyse how domains of input data are mapped through a deep network. A complete understanding of this mapping and its inverse or preimage will give a detailed description of the workings of the network. Since we are not considering the final fully connected layers we will demonstrate how to compute in detail the structure of input data manifold that can be mapped to a specified reduced dimensionality affine manifold in the activity space of the final convolutional output layer. This flattening of input data is often considered as a necessary preprocessing step for efficient classification.

The understanding of mappings between layers will be based on the specific understanding of how to compute preimages for networks activities. We will recapitulate and extend the work in (4) based on the construction of a dual basis from an arrangement of hyperplanes. By specialising to convolutional networks we will demonstrate that the arrangement of hyperplanes associated with a specific layer can be effectively described by a regular multidimensional polyhedral cone oriented in the identity direction in the input space of the layer. Cones associated with successive layers are then in general partly nested inside their predecessor. This leads to efficient contraction and shaping of the input domain data manifold. In general however contraction and shaping are in conflict in the sense that efficient contraction implies less efficient shaping. We will argue that this' conflict is resolved by extending the number of layers of the network with small incremental updates of filters at each layer.

The main contribution of the paper is the exploitation of the properties of nested cones in order to explain how non linear manifolds can be shaped and contracted in order to comply with the distribution of actual class manifolds and to enable efficient preprocessing for the final classifier stages of the network. We will specifically demonstrate the capability of the convolutional part of the network to flatten non linear input manifolds which has previously been suggested as an important preprocessing step in object recognition, (5; 12)

2 Layer mappings in ReLU networks and their preimage

Transformations between layers in a network with ReLU as nonlinear elements can be written as

$$y \;=\; [\mathbf{W}x + b\,]_+ \tag{1}$$

Where $[\;]_+$ denotes the ReLU function $max(0, x_i)$. applied component wise to the elements of the input vector x which will be confined to the positive orthant of the d-dimensional Euclidean input space. It divides the components of the output vector y into two classes depending on the location of the input x:

$$\begin{aligned} \{j: \quad w_j^T x + b_j > 0\} \quad &\rightarrow \quad y_j = w_j^T x + b_j \\ \{i: \quad w_i^T x + b_i \leq 0\} \quad &\rightarrow \quad y_i = 0 \end{aligned} \tag{2}$$

In order to analyse the way domains are mapped through the network we will be interested in the set of inputs x that can generate a specific output y.

$$P(y) = \{x : y = [\mathbf{W}x + b\,]_+\} \tag{3}$$

This set, known as the preimage of y can be empty, contain a unique element x or consist of a whole domain of the input space. This last case is quite obvious by considering the ReLU nonlinearity that maps whole half spaces of the input domain to 0 components of the output y.

The preimage will depend on the location of the input relative to the arrangement of the hyperplanes defined by the affine part of the mapping:

$$\Pi_i = \quad \{x : w_i^T x + b_i = 0\} \qquad i = 1, 2, \ldots d \tag{4}$$

These hyperplanes divides the input space into a maximum of 2^d number of different cells with the maximum attained if all hyperplanes cut through the input space which we take as the non negative orthant of the d-dimensional Euclidean space R_+^d. Understanding the arrangement of these hyperplanes in general and especially in the case of convolutional mappings will be central to our understanding of how input domains are contracted and collapsed through the network.

The preimage problem can be treated geometrically using these hyperplanes as well as the constraint input domains defined by these. For a given output y we can denote the components where $y_j > 0$ as $y_{j_1}, y_{j_1} \dots y_{j_q}$ and the complementary index set where $y_i = 0$ as $i_1, i_1 \dots i_p$ With each positive component of y we can associate a hyperplane:

$$\Pi_j^* = \{x : y_j = w_j^T x + b_j\} \qquad j = j_1, j_2, \dots j_q \tag{5}$$

which is just the hyperplane Π_j translated with the output y_j For the 0-components of y we can define the half spaces

$$X_i^- \ = \{x : w_i^T x + b_i \leq 0\} \qquad i = i_1, i_2, \dots i_p \tag{6}$$

I.e the half space cut out by the negative side of the plane Π_i. These planes and half spaces together with the general input domain constraint of being inside R_d^+ define the preimage constraints given the output y.
If we define the affine intersection subspace:

$$\Pi^* \ = \Pi_{j_1}^* \cap \Pi_{j_2}^* \cap \dots \cap \Pi_{j_q}^* \tag{7}$$

and the intersection of half spaces:

$$X^- \ = X_{i_1}^- \cap X_{i_2}^- \cap \dots \cap X_{i_p}^- \tag{8}$$

the preimage of y can be defined as:

$$P(y) \ = \Pi^* \cap X^- \cap R_+^d \tag{9}$$

The constraint sets and the preimage set is illustrated in figure 1 for the case of $d = 3$ and various outputs y with different number of 0-components.

For fully connected networks, computing the preimage set amounts to finding the intersection of an affine subspace with a polytope in $d-$ dimensional space. This problem is known to be exponential in d and therefore intractable. However, we will see that this situation is changed substantially when we consider convolutional instead of fully connected networks.

3 The dual basis for expressing the preimage

In order to get more insight into the nature of preimages we will devise a general method of computing that highlights the nature of the arrangement of hyperplanes. The set of hyperplanes $\Pi_i, \ i = 1 \quad \dots d$ will be assumed to be in general position, i.e. no two planes are parallel. The intersection of all hyperplanes excluding plane i:

$$S_i \ = \ \Pi_1 \cap \Pi_2 \cap \dots \cap \Pi_{i-1} \cap \Pi_{i+1} \dots \cap \Pi_d \tag{10}$$

is then a one-dimensional affine subspace S_i that is contained in all hyperplanes Π_j excluding j = i. For all i we can define vectors e_i in R^d parallel to S_i. The general position of the hyperplanes then guarantees that the set e_i is complete in R^d. By translating all vectors e_i to the point in R^d which is the mutual intersection of all planes Π_i

$$\Pi_1 \cap \Pi_2 \cap \dots \cap \Pi_d \tag{11}$$

they can therefore be used as a basis that spans R^d. This construction also has the property that the intersection of the subset of hyperplanes:

$$\Pi_{j_1} \cap \Pi_{j_2} \cap \dots \cap \Pi j_q \tag{12}$$

is spanned by the complementary dual basis set

$$e_{i_1}, e_{i_2} \dots e_{i_p} \tag{13}$$

The dual basis can now be used to express the solution to the preimage problem. The affine intersection subspace P^* associated with the positive components $j_1 j_2 \dots j_q$ of the output y is spanned by the complementary vectors associated with the negative components $i_1 i_2 \dots i_p$. These indices also define the hyperplanes $\Pi_1, \Pi_2 \dots \Pi_p$ that constrain the preimage to lie in the intersections of half spaces associated with the negative sides.

We now define the *positive* direction of the vector e_i as that associated with the *negative* side of the plane Π_i. If we consider the intersection of the subspace P^* and the subspace generated by the intersections of the hyperplanes Π_i associated with the negative components of y we get:

$$\Pi^*_{j_1} \cap \Pi^*_{j_2} \cap \ldots \cap \Pi^*_{j_q} \cap \Pi_{i_1} \cap \ldots \cap \Pi_{i_p} \tag{14}$$

Due to complementarity of the positive and negative indices, this is a unique element $x^* \in R^d$ (marked "output" in figure 1 which lies in the affine subspace of the positive output components P^* as well as on the intersection of the boundary hyperplanes Π_i that make up the half space intersection constraint X^- for the preimage. if we take the subset of the dual basis vectors with $e_{i_1}, e_{i_2} \ldots e_{i_p}$ and move them to this intersection element, they will span the part of the negative constraint region X^- associated with the preimage. I.e. the preimage of the output y is given by:

$$P(y) = \{x \in R^d_+ : \quad x = x^* + \sum_{i=1}^{i=i_p} \alpha_i e_i \quad \alpha_i \geq 0\} \tag{15}$$

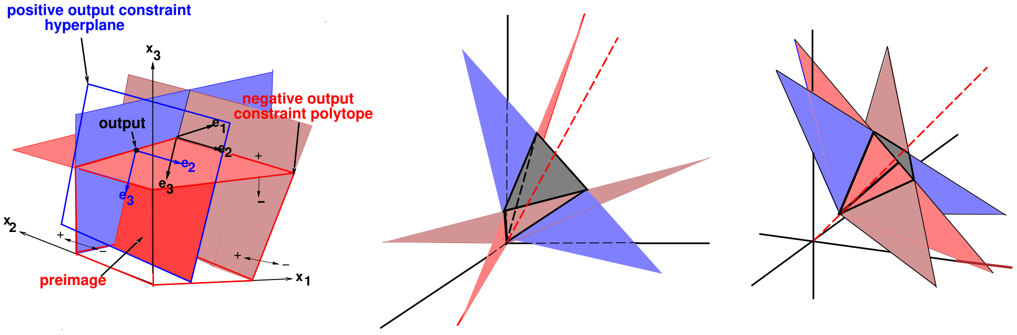
positive output constraint hyperplane
x3
negative output constraint polytope
output
e1
e2
e2
e3
e3
+
−
x2
+
−
preimage
+
−
x1

Figure 1: **Left***: 3 planes in general position and the preimage (red) of the output (black dot) e_2 and e_3 components of dual basis used to generate preimage.* **Right***: Polyhedral cone of 3 planes from a circulant layer transformation matrix. (two different views) The nesting property of the cone will refer to it's ability to "grip" the coordinate axis of the input space.*

4 Arrangements of hyperplanes for convolutional layers form regular polyhedral cones

We will now specialise to the standard case of convolutional networks. In order to emphasize the basic role of geometric properties we will consider only a single channel with no subsampling. Most of what we state will generalize to the more general case of multiple channels with different convolutional kernels but needs a more careful analysis we will exploit the fact that convolutional matrices are in most respects asymptotically equivalent to those of circulant matrices where each new row is a one element cyclic shift of the previous. For any convolution matrix we will consider the corresponding circulant matrix that appends rows at the end to make it square and circulant. Especially when the support of the convolution is small relative to the dimension d, typically the order of 10 in relation to 1000, this approximation will be negligible. Except for special cases the corresponding circulant will be full rank, which means that properties about dual basis etc. derived previously will apply also here. As is standard we will assume that the bias b is the same for all applications of the convolution kernels.

The first thing to note about hyperplanes associated with circulant matrices is that they all intersect on the identity line going through the origin and the point $(1, 1, \ldots 1)$. Denote the circulant matrix as C with elements $c_{i,j}$. The circulant property implies $c_{i+1,j} = c_{i,j-1}, \;\; i = 1 \ldots d-1, \;\; j = 2 \ldots d$ and $c_{i+1,1} = c_{i,d}$. Each row is shifted one step cyclically relative to the previous. For the hyperplane corresponding to row i we have:

$$\sum_{j=1}^{j=d} c_{i,j} x_j \; + b = 0 \tag{16}$$

It is easy to see that the circulant property implies that the sum of all elements along a row is the same for all rows. Let the sum of the row be a. We then get: $x_j = -b/a$ for $j = 1 \dots d$ as a solution for this system of equations which is a point on the identity line in R^d.

The arrangement of the set of hyperplanes:

$$w_i^T x + b = 0 \qquad i = 1 \dots d \tag{17}$$

with w_i^T the i:th row of the circulant augmented convolutional matrix W, will be highly regular. Consider a cyclic permutation Px of the components of the input x described by the single shift matrix P i.e x_i is mapped to x_{i+1} for $i = 1 \dots d-1$ and x_d is mapped to x_1. We then get:

$$\begin{aligned} w_i^T Px + b &= w_{i+1}^T x + b &= 0 \qquad i = 1 \dots d-1 \\ w_d^T Px + b &= w_1^T x + b &= 0 \end{aligned} \tag{18}$$

which states that points on the hyperplane associated with weights w_i are mapped to hyperplane associated with weights w_{i+1}. The hyperplanes associated with the weights w_i $i = 1 \dots d$ therefore form a regular multidimensional polyhedral cone in R^d around the identity line, with the apex located at $x^T = (-b/a, -b/a \dots - b/a)$ controlled by the bias b and the sum of filter weights a. Geometrically, the cone is determined by the apex location, the angle of the planes to the central identity line and its rotation in d-dimensional space. Apex location and angle are two parameters which leaves $d-2$ parameters for the multidimensional rotation in R^d. This maximum degree of freedom is however attained only for unrestricted circulant transformations. The finite support of the convolution weights in CNN:s will heavily restrict rotations of the cone. The implications of this will be discussed later.

5 Nested cones efficiently contract input data

Any transformation between two layers in a convolutional network can now be considered as a mapping between two regular multidimensional polyhedral cones that are symmetric around the identity line in R^d. The coordinate planes of the input space R^d_+ can be modelled as such a cone as well as the output space given by the convolution. The strong regularity of these cones will of course impose strong regularities on the geometric description of the mapping between layers. Just as in the general case, this transformation will be broken down to transformations between intersection subspaces of the two cones.

In order to get an idea of this we will start with a simple multi layer network with two dimensional input and output and a circulant transformation:

$$\begin{aligned} x_1^{(l+1)} &= [a^{(l)} x_1^{(l)} + b^{(l)} x_2^{(l)} + c^{(l)}]_+ \\ x_2^{(l+1)} &= [b^{(l)} x_1^{(l)} + a^{(l)} x_2^{(l)} + c^{(l)}]_+ \end{aligned} \tag{19}$$

Figure 2 illustrates the mapping of data from the input space (x_1, x_2) to the output space (y_1, y_2) for two networks with 3 and 6 layers respectively. The dashed lines represent successive preimages of data that maps to a specific location at a layer. By connecting them we get domains of input data mapped to the same output at the final layer, i.e they are contraction flows depicting how data is moved through the network. Note that in both layers the major part of the input domain is mapped to output $(0, 0)$. This is illustrated for the first trivial bias only layer with $a = 1, b = 0$. The domain of the input that is mapped to output domain is just quite trivial planar manifolds.

The second network with more varied weights illustrates how input domain manifolds with more structure can be created. It also demonstrates the importance of the concept of “nested cones” and how this affects the input data manifolds. The red lines represent data that is associated with layer cones that are completely nested inside its predecessors, while the black lines represent data where the succeeding cone has a wider angle than its predecessor. When this happens, the hyperplanes associated with the output cone will intersect the hyperplanes of the input cone and input data beyond this intersection is just transformed linearly. Since all data in figure 2 is remapped to the input space this has the effect that data is not transformed at all. This has no effect at all at the shaping of the input manifold. One could say that these layers are “wasted” beyond the location of the intersection as far as network properties are concerned since they neither contribute to the shaping or the contraction of

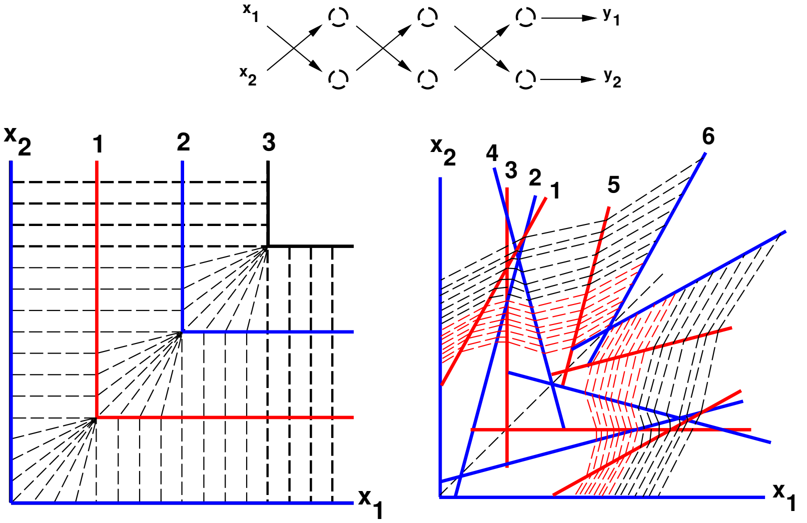

Figure 2: *Illustration of how data in 2d input space (x_1, x_2) contracts by successive layers in multi layer 2 node networks with circulant transformations.* **Left***: Transformation $(a, b) = (1, 0)$. Only the bias differs between layers. Alternating red and blue frames show successive layers remapped to the input space.* **Right***: Arbitrary circulant transformations. Note how the nesting property insures a good variation in the generated manifolds. When nesting becomes less pronounced for higher values of input the variation of the manifolds diminishes.*

input data manifolds. The effect of this on the input manifold can be seen as a less diverse variation of its shape (black) compared to the previous part associated with the completely nested part of the layer cones.

In higher dimensions the effects of nested vs. partially nested cones appear in the same way but more elaborate. In addition to the 2d case we also have to consider rotations of the cone, which as was pointed out earlier, has $d-2$ degrees of freedom for cones in d dimensional space. The effects of contraction of data from higher to lower dimensions also become more intricate as the number of different subspace dimensionalities increases. Most of these effects can be illustrated with 3 dimensional input and output spaces. For $d = 3$ the generic circulant matrix can be used to define a layer transformation:

$$\begin{aligned} y_1 &= [ax_1 + bx_2 + cx_3 + d]_+ \\ y_2 &= [cx_1 + ax_2 + bx_3 + d]_+ \\ y_3 &= [bx_1 + cx_2 + ax_3 + d]_+ \end{aligned} \tag{20}$$

The transformation properties of this network are most easily illustrated if we start with the pure bias case with transformation $W = I$, i.e $a = 1, b = 0, c = 0$. A specific element in input space is mapped according to its position relative to the hyperplanes. If we use the dual basis to define the coordinates of the output data, the mapping for the input element will be the same in input cells with the same relation to all hyperplanes. In d dimensions, the hyperplanes divide the input space into 2^d cells where elements are mapped to a specific associated intersection subspace in the output domain.

In figure 3, the grey shaded boxes indicate two cells with different numbers of negative constraints 1 and 2 respectively. The content of the upper one with one negative constraint including all its bounding faces and their intersections is mapped to a specific $2d$ hyperplane in the output domain while the content of the lower one with two negative constraints is mapped to the 1d intersection of two hyperplanes. This illustrates the most important property of the nesting of the cones associated with the input and output layer: For a range of transformations in the vicinity of the identity mapping, the input space, properly normalised in range, is divided into cells where the elements of the cells including their bounding faces an their intersections are mapped to output intersection subspaces with equal or lower dimension. This means that the content of the cell is irreversibly contracted to lower dimensions.

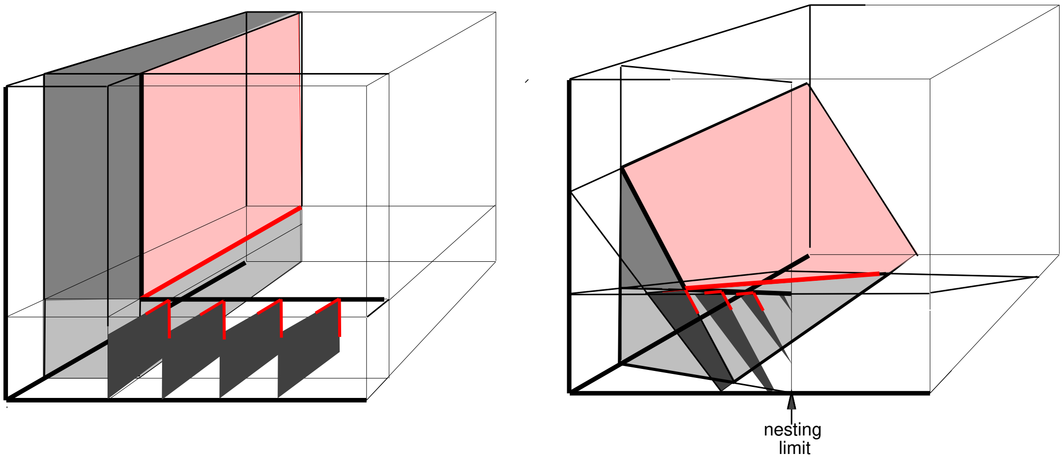

Figure 3: *Illustration of how data maps between intersection subspaces in two successive layers of 3 node networks with circulant transformations* **Left***: Identity transformation and bias only. An element in the input space maps according to it's cell location determined by the sign relative to the output hyperplanes. The figure illustrates how two different cells, dark grey and light grey are mapped to red 2d plane and red 1d line in the output layer. Note that both 3d volume, 2d faces and 1d edges maps to the same output intersection subspace. This means that dimension of subspace location for the element is non increasing which implies that data gets contracted. This is a consequence of the nesting properties of the output and input polyhedral cones.* **Right** *The transformation is now such that the angle of the cone is increased. The planes of the output layer now intersects the coordinate axis of the input space. Beyond the intersections the non increasing contraction property ceases. The dark gray areas indicate preimages of points located on the output 1-d coordinate axis. Note how they are decreases in size between the left and right examples.*

Figure 3 also contains examples of preimages to individual elements (dark shaded grey rectangles) and the components of the dual bases used to span these. Note that these are affected by changing the angle of the output cone. It introduces a limit of the nesting beyond which the mapping properties of the transformation are changed so that data no longer maps to a manifold of equal or lower dimensionality. I.e the contraction property is lost in those regions of the input space where nesting of cones ceases.

We will formally define this important property of nested cones as:

> Let R^d_+ be the non negative orthant of the Euclidean d-space. Let Π^0_i be the hyperplane defined as $x_i = 0$ for $(x_1 \dots x_d) \in R^d_+$. Consider a set of corresponding hyperplanes $\Pi_1 \dots \Pi_d$ in R^d_+ associated with a circulant matrix. Take a subset $i_1 \dots i_p$ of these hyperplanes and form the intersection subset: $M_{i_1 \dots i_p} = \Pi_{i_1} \cap \dots \cap \Pi_{i_p}$. If for each x in $M_{i_1 \dots i_p}$ the positive span $S_+(e_{j_1} \dots e_{j_q})$ of the associated dual basis contains an element in the corresponding intersection subset $\Pi^0_{i_1} \cap \dots \cap \Pi^0_{i_p}$ and thereby in each subset of planes with these indexes, but no other subset, we say that the cone formed by the hyperplanes Π is completely nested in the cone formed by the hyperplanes Π^0.

We see that this definition implies that the cone formed by planes Π is completely contained in that formed by the planes Π^0 but also that its relative rotation is restricted. We will have reason to relax the condition of inclusion of all elements $i_1 \dots i_p$ in the intersection subset and talk about cones with restricted nesting. Complete nesting implies contraction of data from one layer to the next which can be seen from the fact that all elements of the complete intersection subset $\Pi^0_{i_1} \cap \dots \cap \Pi^0_{i_p}$ and thereby in each subset are mapped to the intersection subset $M_{i_1 \dots i_p} = \Pi_{i_1} \cap \dots \cap \Pi_{i_p}$ with same dimensionality. In addition elements from intersection subsets formed by subsets of indexes $i_1 \dots i_p$ will also be mapped to this same intersection subset. The subsets associated with these indexes are however of higher dimension. Consequently, mapping of data between layers will be from intersection subsets to intersection subsets with equal or lower dimension. This is the crucial property connecting degree of nesting with degree of contraction.

By going further through the network to higher layers this contraction is iterated and data is increasingly concentrated on intersection subspaces with lower dimension which is reflected by the increased sparsity of the nodes of the network. The convolutional part of a deep network can therefore be seen as a component in a metric learning system where the purpose of the training will be to create domains in input space associated with different classes that are mapped to separate low dimensional outputs as a preprocessing for the final fully connected layers that will make possible efficient separation of classes.

There is therefore a conflict between the diversity of input manifolds that contract to low dimensional outputs and the degree of contraction that can be generated in a convolutional network. The efficient resolution of this conflict seems to lie in increasing the number of layers in the network in order to be able to shape diverse input manifolds but with small incremental convolution filters that retain the nesting property of the cone in order to preserve the proper degree of contraction. Empirically, this is exactly what has been demonstrated to be the most efficient way to increase performance of deep networks (13; 14).

6 Mapping nonlinear input to affine output manifold

We are now in a position to give a general characterizaton of the preimage corresponding to a specified output domain at the final convolutional output layer assuming the property of nested layer cones. Ideally we would like to include the final fully connected layers in this analysis but it will require a special study since we cannot assume the nesting property to be valid for these. In the end the network should map different classes to linearly separable domains in order to enable efficient classification. It is generally suggested that the preprocessing part of a network corresponds to flattening nonlinear input manifolds in order to achieve this final separation at the output. In order to be able to draw as general conclusions as possible we shall demonstrate the exact structure of a nonlinear input manifold that maps to a prespecified affine manifold at the final convolutional layer. We denote this manifold M and the output at the final convolutional layer by $x^{(l)}$. The final layer can be characterised by the set of hyperplanes: $\Pi_1^{(l)} \ldots \Pi_d^{(l)}$. Let the zero components of the output $x^{(l)}$ be $i_1, i_2 \ldots i_q$. It can then be associated with the intersection of the output manifold and the corresponding hyperplanes

$$M \cap \Pi_{i_1}^{(l)} \cap \Pi_{i_2}^{(l)} \cap \ldots \cap \Pi_{i_q}^{(l)} \qquad (21)$$

The degree of intersection q will depend on the dimensionality of M. If M is a $d-1$ dimensional hyperplane in general position it will intersect any combination of hyperplanes at output level l. This is the maximum complexity situation that will generate a $d-1$ - dimensional input manifold. Reducing dimensionality of M means reducing the possible intersection with combinations of hyperplanes. Note that if M intersects the set the intersection of planes $i_1, i_2 \ldots i_p$ it also intersects the intersection of any subset of these. Intersecting M with each of these subsets will generate pieces of intersections linked together. These are affine subsets with different dimensionality and the preimage of each piece will be generated by complementary dual basis components. This is illustrated by figure 4 for the case of an affine plane in R_+^3 intersecting to give a triangular domain. In this case we have three points on the coordinate axis, and three lines connecting these. The three points will all span 2d planes bases on different pairs of complementary dual basis components. In addition to these, the points on the lines of the triangle generated by intersecting M with each of the three individual output planes will generate $1d$ lines that jointly will span a $2d$ plane. This plane will connect continuously with the planes spanned from the points on the axis to yield a piecewise planar input manifold to the final layer. Continuing through the network, this piecewise planar manifold will intersect with the planes of layer $l-1$ and the procedure is iterated until we reach the input layer.

This procedure generalises to arbitrary dimensions but the complexity of course grows with the increasing combinatorics. The basic principle of layer by layer recursively generating piecewise affine manifolds still holds. The complexity lies in the fact that each intersection of the manifold M with every subset of possible hyperplane intersections will generate a seeding hyperplane and and each of these will act as a new manifold M at the next layer. Note however that the nested cone property substantially reduces complexity compared to the general case of arbitrary hyperplanes.

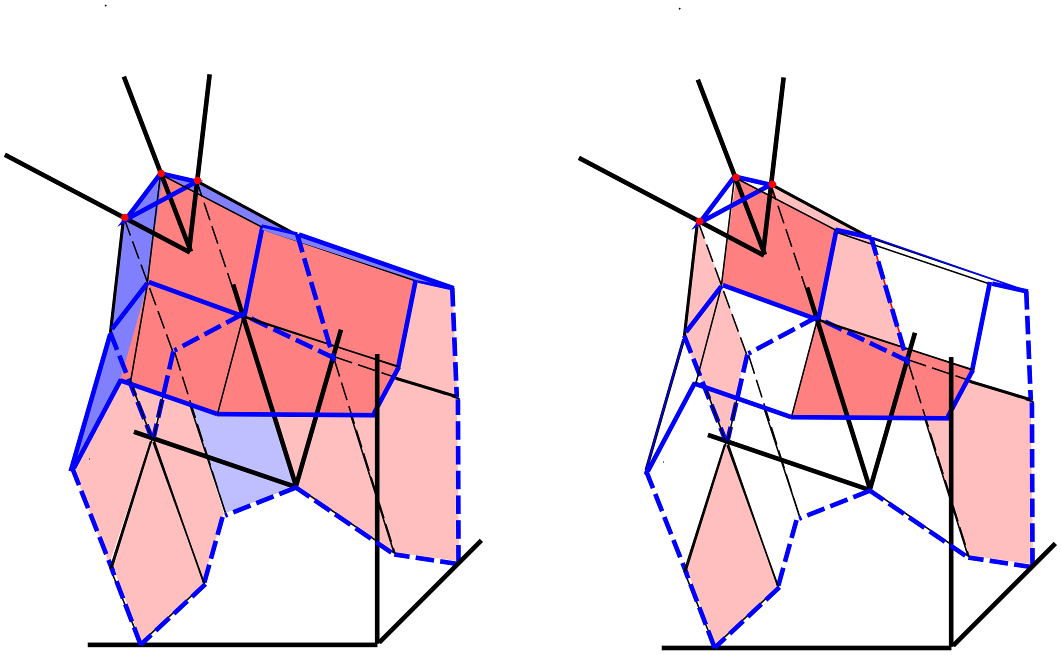

Figure 4: *Piecewise planar manifold in 3d input space that maps to affine manifold (blue triangle) at the final convolutional layer in a 3-node 3-layer network with circulant transformations. All data is remapped to the input space.* **Left***: Red patches are mapped to 0 dimensional red points at the three output coordinate axis Blue patches are mapped to 1d lines connecting the points. (dark red is outside light red is inside of the manifold* **Right***: Patches that are generated by selective components of the dual basis at each layer. The positive span generated by selective components of the dual basis emanating from the red output points on the triangle as well as from each intersection with coordinate lines in early layers, intersects with the arrangement of hyperplanes representing the preceding layer. The 1-d intersections are then used as seed points for new spans that intersect next preceding layer etc. The 2d intersections together with selective edges from the spans generate linking patches that ensures the continuity of the input manifold.*
as

It should be pointed out that these manifold do not necessarily correspond to actual class manifolds since we are not considering the complete network with fully connected layers. They can however be considered as more elaborate and specific building blocks in order to construct the actual class manifolds of a trained network.

7 Summary and conclusions

We have defined a formal procedure for computing preimages of deep linear transformation networks with ReLU non linearities using the dual basis extracted from the set of hyperplanes representing the transformation. Specialising to convolutional networks we demonstrate that the complexity and the symmetry of the arrangement of corresponding hyperplanes is substantially reduced and we show that these arrangements can be modelled closely with multidimensisional regular polyhedral cones around the identity line in input space. We point out the crucial property of nested cones which guarantees efficient contraction of data to lower dimensions and argue that this property could be relevant in the design of real networks. By increasing the number of layers to shape input manifolds in the form of preimages we can retain the nested cone property that most efficiently exploits network data in order to construct input manifolds that comply with manifolds corresponding to real classes and would explain the success of ever deeper networks for deep learning. The retaining of the nested cone property can be expressed as a limitation of the degrees of freedom of multidimensional rotation of the cones. Since convolutional networks essentially always have limited spatial support convolutions, this is to a high degree built in to existing systems. The desire to retain the property of nesting could however act as an extra constraint to further reduce the complexity of the convolutions. This of course means that the degrees of freedom are reduced for a network which could act as a regularization constraint and potentially explain the puzzling efficiency of generalisation of deep networks in spite of a high number of parameters.

We demonstrate that it is in principle possible to compute non linear input manifolds that map to affine output manifolds. This demonstrates the possibility of deep convolutional networks to achieve flattening of input data which is generally considered as an important preprocessing step for classification. Since we do not consider a complete network with fully connected layers at the end we cannot give details how classification is achieved. The explicit demonstration of non linear manifolds that map to affine outputs however indicates a possible basic structure of input manifolds for classes. It is easy to see that a parallel translation of the affine output manifold would result in two linearly separable manifolds that would be generated by essentially parallel translated non linear manifolds in the input space. This demonstrates that convolutional networks can be designed to exactly separate sufficiently “covariant “ classes. and that this could be the reason for the relative success of convolutional networks over previous machine learning approaches to classification and explain why using a large number of classes for training is advantageous since they all contribute to very similar individual manifolds.

Disregarding these speculations the fact remains that these manifolds will always exist since they are derived on purely formal grounds from the structure of the network. If they have no role in classification their presence will have to be explained in other ways.

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] Ronen Basri and David W. Jacobs. Efficient representation of low-dimensional manifolds using deep networks. *CoRR*, abs/1602.04723, 2016.

[3] Yoshua Bengio and Olivier Delalleau. On the expressive power of deep architectures. In *International Conference on Algorithmic Learning Theory*, pages 18–36. Springer, 2011.

[4] Stefan Carlsson, Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, and Kevin Smith. The preimage of rectifier network activities. In *International Conference on Learning Representations (workshop)*, 2017.

[5] James J DiCarlo and David D Cox. Untangling invariant object recognition. *Trends in cognitive sciences*, 11(8):333–341, 2007.

[6] Alexey Dosovitskiy and Thomas Brox. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4829–4837, 2016.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[8] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[9] Aravindh Mahendran and Andrea Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. *International Journal of Computer Vision (IJCV)*, 2016.

[10] 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.

[11] Richard Baraniuk Randall Balestriero. A spline theory of deep learning. *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[12] Sam T Rowes. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:232, 2000.

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

[14] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[15] Liwen Zhang, Gregory Naitzat, and Lek-Heng Lim. Tropical geometry of deep neural networks. *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.