A look at the topology of convolutional neural networks

Rickard Brüel Gabrielsson
Department of Computer Science
Stanford University
Stanford, CA 94305
rbg@cs.stanford.edu

Gunnar Carlsson
Department of Mathematics
Stanford University
Stanford, CA 94305
carlsson@stanford.edu

Abstract

Convolutional neural networks (CNN's) are powerful and widely used tools. However, their interpretability is far from ideal. In this paper we use topological data analysis to investigate what various CNN's learn. We show that the weights of convolutional layers at depths from 1 through 13 learn simple global structures. We also demonstrate the change of the simple structures over the course of training. In particular, we define and analyze the spaces of spatial filters of convolutional layers and show the recurrence, among all networks, depths, and during training, of a simple circle consisting of rotating edges, as well as a less recurring unanticipated complex circle that combines lines, edges, and non-linear patterns. We train over a thousand CNN's on MNIST and CIFAR-10, as well as use VGG-networks pretrained on ImageNet.

1 Introduction

The problem of understanding how convolutional neural nets (CNN's) work and learn is one of the fundamental problems in machine learning. It is important to study both the weights and the activations, as these roughly constitute the "coefficients" and the outputs in the computational model. To date, work in this area [1, 17, 18, 19, 20] has involved direct human inspection of features constructed in the network and has produced very interesting qualitative results. The goal of the present paper is to demonstrate that data sets constructed out of the weights are organized in simple *topological models*, which are strongly reminiscent of the results obtained in the topological analysis of data sets of local patches in natural images [2]. Such topological models yield insight by effectively summarizing the global structure of the spaces of weight configurations, and permit the exploration of density in the data set. The key point here is that the study of the function of neural nets is a problem in *data analysis*, since the density of particular features is clearly relevant, and since we clearly find the presence of anomalous and spurious elements. It is important to model the most strongly occurring motifs in a simple and understandable way.

The topological models we work with are part of *topological data analysis* (TDA) [3, 9, 15, 16], which in addition to the construction of the models provide invariants of the shape of the data set (persistent homology), that confirm that the shape of the data is as expressed in the model. We apply methods of TDA to data sets of *spatial filters* of the convolutional layers. In the i-th convolutional layer, an activation map is constructed by sliding a filter (a set of weights) along the spatial dimensions of all activation maps in the $(i-1)$-th layer. A filter thus has dimensions $w \times h \times c$, where w and h are the width and height of the spatial receptive field of the filter while c is the number of activation maps in the $(i-1)$-th layer. We define a *spatial filter* as one set of $w \times h$ weights with a fixed c-dimension. One single filter give c spatial filters and a convolutional layer with d number of activation maps give $d \times c$ spacial filters of dimension $w \times h$.

Preprint. Work in progress.

We perform analyses of CNN's trained on the MNIST [5], CIFAR-10 [6], and ImageNet [7] data sets. We find that in some cases, the models recapitulate the topological structures that occurred in [2], namely the *primary* and *secondary* circles (see Figure 1), but that in other situations different phenomena occur. This paper constitutes an exploratory analysis of the spatial filters described above. We have chosen, due to space considerations, to present the findings as they are, without a thorough discussion of what hypotheses we might construct based on them. We will return to the hypotheses and theoretical analysis in future work.

2 Persistent Homology

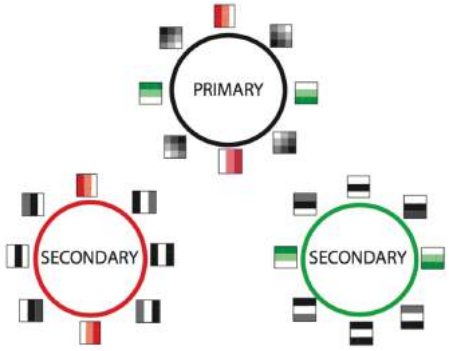

Figure 1: Primary and Secondary Circles

Within the domain of topology, *homology* refers to a collection of signatures that perform a sophisticated counting task for features, such as connected components, loops, spheres, etc. to obtain invariants of topological spaces. Its extension to point clouds is called *persistent homology*, which has been undergoing rapid development over the last 15 years. For each dimension k, the output of persistent homology is a *barcode*, i.e. an unordered collection of intervals on the real line, where a long bar indicates the presence of a feature that lives over a large range of values and is hence regarded as "real", and short bars are often attributed to noise. The barcode is a multiscale summary analogous to the dendrograms that arise in hierarchical clustering. For example, a long bar in the 1-dimensional bar code reflects the presence of a loop in the data. These invariants have been used in many different situations. One such is the analysis of local image patches performed in [2], which was motivated by the idea of understanding the tuning of neurons in the primary visual cortex. One of the outcomes of that paper is illustrated above (Figure 1), where we see that the data (suitably thresholded by density) is organized around three circles, which overlap to a degree, and which reflect the tuning of neurons to edge and line detectors. The idea of this paper is to perform this same analysis in the context of neural nets rather than the visual pathway. We computed persistent homologies and their respective barcodes by using the Plex software [13]. We used *LazyWitnessStream* and *MaxMinSelector* with 100 landmarks as well as a max division of 1000 for all our persistence computations.

3 Mapper

The topological modeling method ("Mapper", see [10] for details) we use starts with one, two, or three real valued functions on the data, which we refer to as *lenses*, as well as with a metric on the data set. By choosing overlapping coverings of the real line by intervals of the same length and overlap, we obtain coverings of $\mathbb{R}$, $\mathbb{R}^2$, or $\mathbb{R}^3$, which allow us to bin the data into bins, one for each set in the cover. We then perform a clustering step (single linkage clustering with a fixed heuristic for the choice of threshold, specified in [12]) based on the metric to generate a set of clusters. Because the intervals overlap, it is possible for clusters attached to one bin to overlap with clusters attached to another bin, and we define a graph whose node set is the collection of clusters we have defined, and where there is an edge connecting a pair of clusters if the two clusters share at least one data point. The topological version of this construction is well known, and comes with guarantees concerning the degree to which the construction approximates the original space. Such guarantees are not yet available for Mapper, although work in this direction is being done [11].

For the clustering step in the Mapper method we use the Variance Normalized Euclidean (VNE) metric. The VNE metric is a variant of standard Euclidean distance that first normalizes each column of the data set by dividing by its variance. For lenses we use PCA 1 and 2, which means that the point cloud is projected onto its two principal components before choosing overlapping coverings. Our results generalize to other lenses such as Ayasdi's Neighborhood Lens 1 and 2 [14] which capture more non-linear features of the data. However, since PCA lenses often gave the best-looking graphs and for sake of consistency and simplicity we only present results acquired by use of the PCA lenses. We use the implementation of Mapper found in the Ayasdi software [12]. In Ayasdi, *resolution* specifies the number of bins and *gain* determines the overlap as follows:

Table 1: *M*(*X*, *Y*, *Z*) CNN-architecture

Conv Layer 1	Conv Layer 2	FC layer	Readout
3×3×*X* filters	3×3×*Y* filters	*Z* nodes	10 nodes
ReLU	ReLU	ReLU	Softmax, Cross Entropy
2×2 max-pooling	2×2 max-pooling		Dropout 0.5, ADAM

percent overlap $= 1 - (1/gain)$. We specify Mapper by notation $Mapper(resolution, gain)$. In addition, the color of the nodes is determined by the number of points that the corresponding cluster contains, with red being the largest and blue the lowest. This number is a rough proxy for density.

4 Density Filtration

To determine the core subset of a point cloud X we perform a density filtration of the points based on a nearest neighbor estimate of the local density. For each $x \in X$ and $k > 0$ we calculate its distance to its k-th nearest neighbor, this distance being inversely correlated to the density at x. Then we take the top $p, 0 < p \leq 1$, fraction of the densest points. We can thus denote a density filtration with parameters k and p applied to X by $\rho(k, p, X)$.

5 Experiments

Our experiments were conducted on networks trained on the MNIST [5], CIFAR-10 [6], and ImageNet [7] datasets. MNIST consists of gray scale images of digits, CIFAR-10 consists of natural color images of 10 classes including airplanes, cats, dogs, and ships, and ImageNet consists of natural color images of a wide variety of classes. CNN's have achieved high accuracy all these data sets, suggesting that CNN's are able to learn structures present among the images in the data sets.

We specify the architecture of our CNN's as in Table 1 and 2, where *X, Y, Z* corresponds to the depth of the first convolutional layer, the depth of the second convolutional layer, and the number of nodes in the fully connected layer respectively. If any of *X, Y*, or *Z* is 0 it means that that whole column or block is removed from the network. E.g. $M(64, 32, 64)$ is a network of type found in Table 1 with a first-convolutional-layer-depth of 64, a second-convolutional-layer-depth of 32, followed by a fully connected layer with 64 nodes. For notational efficiency we use superscripts to specify the convolutional layer from which the spatial filters were extracted and subscripts to specify the number of batch iterations the network was trained on. Further, preceding this notation by '$N\times$' means that N trained networks were used as the source of the spatial filters. Thus, with previously developed notation we can write $Mapper(30, 3)$ of $\rho(200, 0.2, 100 \times M^1_{100K}(64, 32, 64))$ to denote Mapper with resolution 30 and gain 3 applied to a point cloud generated by a k-nearest-neighbor filtration with $k = 200$, $p = 0.2$ of the mean-centered and normalized 1st convolutional layers' spatial filters of 100 networks of type $M(64, 32, 64)$ trained for 100,000 batch iterations. Throughout this work we treat each spatial filter of a convolutional layer as a point, i.e. each point is (*width*×*height*)-dimensional. We *always* mean-center and normalize each point, which is done before any density filtration. In addition, the padding on the convolutional layers preserves spacial dimensionality and a batch size of 124 was used throughout the experiments. We used Tensorflow [8] for implementation.

5.1 MNIST

MNIST was divided into 60,000 training examples and 10,000 test examples. We train 100 CNN's of type *M*(64,32,64) (Table 1) for 40,000 batch iterations with a batch size of 128 to a test accuracy of about 99.0%. These 100 trained CNN's give us $64 \times 100 = 6400$ 9-dimensional points (first layer spatial filters) which we mean-center and normalize. We then use k-nearest-neighbor density filtration with $k = 200$ and $p = 0.3$ to get 1920 points. To this point cloud (equivalent to $\rho(200, 0.3, 100 \times M^1_{40K}(64, 32, 64))$) we apply Mapper ($resolution = 30$, $gain = 3$) with Variance Normalized Euclidean Norm and two PCA lenses. The resulting graph can be seen in Figure 2. We also put, next to the graph, the mean of adjacent points to represent the spatial filters at that position in the graph. Recall that color codes for the size of the collection represented by the nodes, increasing from blue to red.

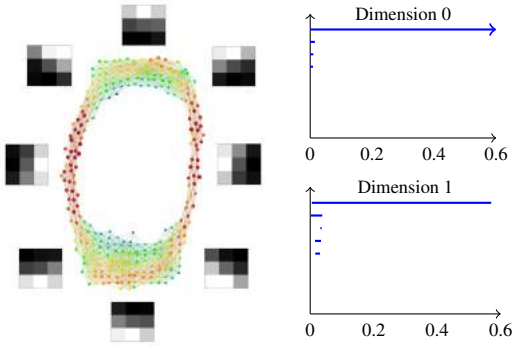

Figure 2: $Mapper(30, 3)$ and barcodes of $\rho(200, 0.3, 100 \times M^1_{40K}(64, 32, 64))$

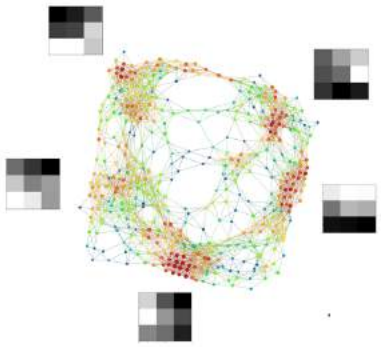

Figure 3: $Mapper(30, 3)$ of $\rho(10, 0.1, M^2_{40K}(64, 32, 64))$.

Table 2: *C*(*X*, *Y*, *Z*) CNN-architecture

Conv Layer 1	Conv Layer 2	FC layer	Readout
3×3×*X* filters	3×3×*Y* filters	*Z* nodes	10 nodes
ReLU	ReLU	ReLU	Softmax
3×3 max-pooling, stride: 2	2×2 max-pooling, stride: 1		Cross Entropy
Local response normalization[1]	Local response normalization[1]		L^2 loss, SGD

From this graph we see how the learned spatial filters are well approximated by the primary circle (Figure 1). The circle is further supported by the corresponding barcodes (Figure 2), which show one persistent loop or circle and one persistent connected component. We obtain almost identical results as in Figure 2 with $Mapper(30, 3)$ and barcodes of $\rho(200, 0.3, 100 \times M^1_{40K}(64, 0, 64))$, i.e. only having one convolutional layer. The results were also robust to other network configurations; the primary circle was found in the first layer spatial filters of trained networks of types $M(64, 8, 512)$, $M(64, 16, 512)$, and $M(256, 32, 512)$.

For the same networks of type $M(64, 32, 64)$ used to generate Figure 2 we also obtain $64 \times 32 \times 100 = 204800$ 9-dimensional second layer spatial filters. After strong density filtration ($p = 0.1$, $k = 10$) we find a very weak primary circle. In Figure 3 we display an example of Mapper applied to the spatial filters learned by a *single* network (64×32 filters) with filtration $p = 0.32$, $k = 220$.

5.2 CIFAR-10

CIFAR-10 was divided into 50,000 training examples and 10,000 test examples. The input was preprocessed by taking a random 24×24 crop of the image, applying a random left-right flip, mean-centering, and normalizing.

5.2.1 Grayscaled

The input was grayscaled using the weights $(0.2989, 0.5870, 0.1140)$ for red, green, and blue respectively. We train 100 CNN's of configuration $C(64, 32, 64)$ for 70,000 batch iterations (test accuracy of about 77.0%) to obtain 6,400 first-layer spatial filters and 204,800 second-layer spatial filters. The result of $(p = 0.5, k = 200)$ density filtration and Mapper on the first-layer spatial filters can be seen in Figure 5. We also train 48 CNN's of configuration $C(64, 0, 64)$ for 70,000 batch iterations (test accuracy of about 69.2%) to obtain 3,072 first-layer spatial filters; the result of Mapper on these first-layer filters can be see in Figure 4. Notice that that in both Figure 4 and 5 we find five cluster structures but that the clusters differ between the two figures. In Figure 5 we find clusters around horizontal and vertical lines while this is not the case in Figure 4. In neither of the 'well-trained' networks were we able to find a significant primary circle.

[1]With depth radius of 4

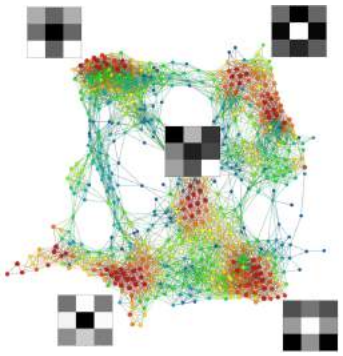

Figure 4: $Mapper(30, 3)$ of $48 \times C^1_{70K}(64, 0, 64)$.

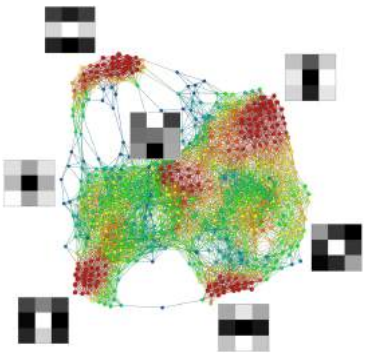

Figure 5: $Mapper(30, 3)$ of $\rho(200, 0.5, 100 \times C^1_{70K}(64, 32, 64))$

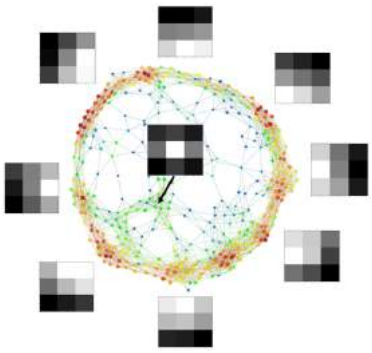

Figure 6: $Mapper(30, 3)$ of $\rho(75, 0.37, C^2_{60K}(64, 32, 64))$

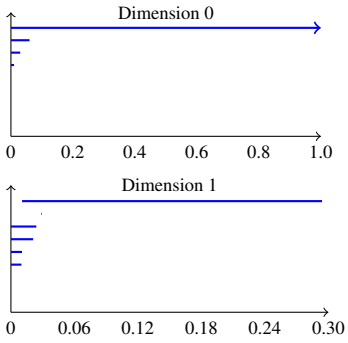

Figure 7: Barcodes of $\rho(15, 0.1, 100 \times C^2_{50K}(64, 32, 64))$

In Figure 7 we show the barcodes of the 204,800 second-layer spatial filters from the 100 CNN's of configuration $C(64, 32, 64)$ trained for 50,000 batch iterations (test accuracy of about 76.2%) and with density filtration $p = 0.1, k = 15$. In Figure 6 we show Mapper applied to the 2,048 second-layer spatial filters of a single CNN of configuration $C(64, 32, 64)$ trained for 60,000 batch iterations (test accuracy of about 77.1 %), and with density filtration $p = 0.37, k = 75$. Note that even though we needed more networks to get the clear barcodes showing the circle in Figure 7, Figure 6 demonstrates that the primary circle (with some other weaker structures) appears in the training of a single network.

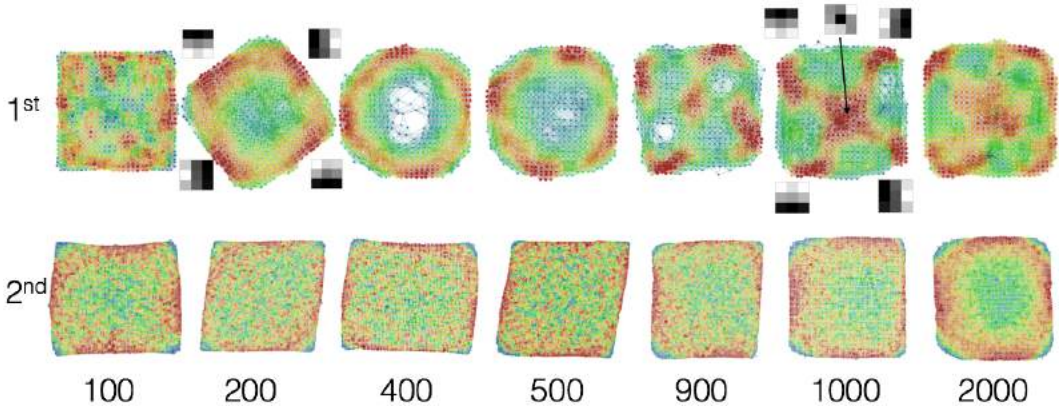

Figure 8: $Mapper(30, 3)$ of $100 \times C^1(64, 32, 64)$ and $Mapper(70, 2)$ of $\rho(15, 0.5, 100 \times C^2(64, 32, 64))$ from 100-2000 batch iterations. Best viewed in color.

Next we look at the spatial filters of the first and second convolutional layers of 100 CNN's of configuration $C(64, 32, 64)$ at batch iterations 100 to 2000. In Figure 8 we see Mapper applied to both these point clouds. The vertical axis specifies the index of the convolutional layer (1st or 2nd) and the horizontal axis specifies the number of batch iterations. For the 2nd layer spatial filters a density filtration of $p = 0.5, k = 15$ was applied, while no density filtration was applied to the first layer. We find that in the first layer the primary circle reveals itself at 400 batch iterations, breaks apart at 500 batch iterations, and then starts to reappear in the second layer at 2000 batch iterations. Note that the four edges in the first layer shown at 200 and 1000 iterations appear relatively stable over many batch iterations.

5.2.2 Color

We train 60 CNN's of configuration $C(64, 32, 64)$ for 100,000 batch iterations (test accuracy of about 81.2%). This gives us 11,520 first-layer spatial filters and 204,800 second-layer spatial filters. In Figure 9 we show Mapper applied to the 11,520 first layer spatial filters at 100,000 batch iterations and density filtration $p = 0.14, k = 200$. In Figure 10 we show Mapper applied to the 2,048 second layer spatial filters of a single network at 50,000 batch iterations (test accuracy of about 79.9%) and density filtration $p = 0.32, k = 10$.

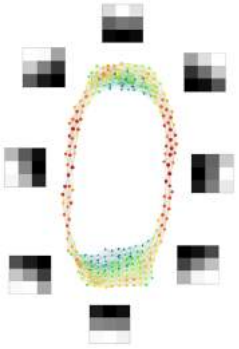

Figure 9: $Mapper(30, 3)$ of $\rho(200, 0.14, 60 \times C^1_{100K}(64, 32, 64))$

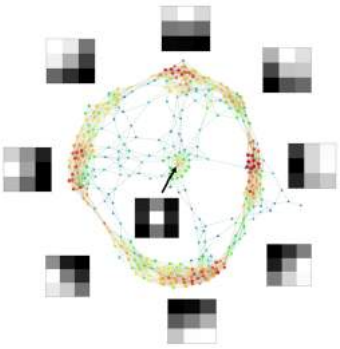

Figure 10: $Mapper(30, 3)$ of $\rho(10, 0.32, C^2_{50K}(64, 32, 64))$

We compute the barcodes of the point cloud used to generate Figure 9 and find an equally persistent circle and connected component as in the barcodes of Figure 2. We also compute the barcodes of all the 204,800 second-layer spatial filters at 100,000 batch iterations and density filtration $p = 0.1, k = 15$ and find similar support for the circle as found in the gray scaled case of Figure 7. In addition, we look at the first layer spatial filters for each input channel, i.e. red, green, and blue, independently and find the primary circle in each one.

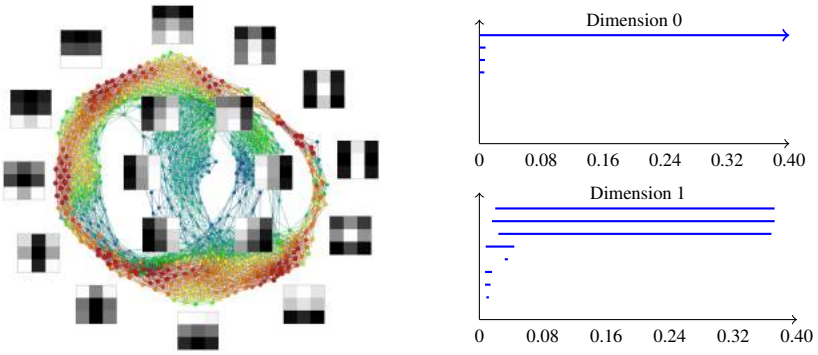

Figure 11: $Mapper(30, 3)$ and barcodes of $\rho(200, 0.32, 82 \times C^{*1}_{30K}(48, 0, 64))$. *: Without max-pooling

Next we train 82 CNN's of configuration $C(48, 0, 64)$ but *without* max-pooling and find among the 11,808 first layer spacial filters at 30,000 batch iterations (test accuracy of about 71.8%) and filtration

$p = 0.32, k = 200$ the two-circle model showed in Figure 11. We see that the circles intersect at two points and that one of the circles (the weaker) is the primary circle while the other (the stronger) is a strange circle we have not seen before. Two circles intersecting at two points have three loops and one connected component, which can be seen among the barcodes in Figure 11.

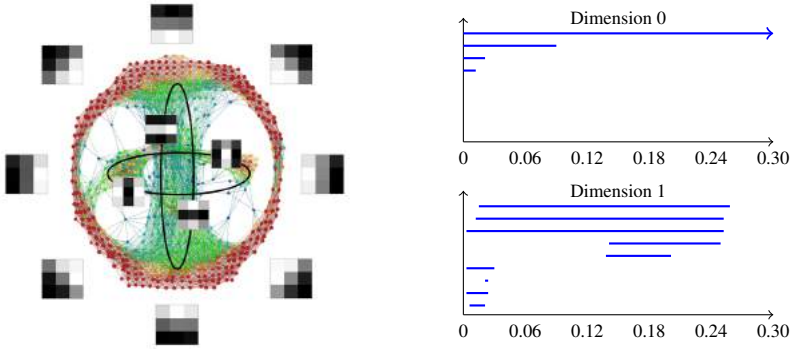

Figure 12: $Mapper(30, 3)$ and barcodes of $\rho(100, 0.35, 100 \times C^1_{100K}(64, 32, 64))$.

A closer examination of the 11,520 first-layer spatial filters of the configuration $C(64, 32, 64)$, trained for 100,000 batch iterations at filtration $p = 0.35, k = 100$, shows that the three circle model found in the image patch data [2] appears. The barcodes and Mapper applied to this point cloud can be seen in Figure 12. Note the stronger outer primary circle and the two weaker secondary circles; each of the secondary circles intersect the primary circle twice but they do not intersect each other.

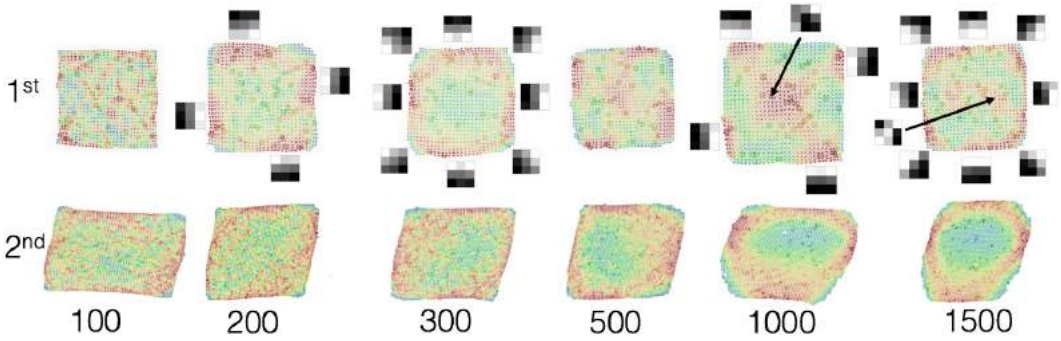

Figure 13: $Mapper(33, 2)$ of $60 \times C^1(64, 32, 64)$ and $Mapper(60, 2)$ of $\rho(100, 0.3, 60 \times C^2(64, 32, 64))$ from 100-1500 batch iterations. Best viewed in color.

We look at the spatial filters of the first and second convolutional layers of 60 CNN's of configuration $C(64, 32, 64)$ at batch iterations 100 to 1500. In Figure 13 we see Mapper applied to both these point clouds. The vertical axis specifies the index of the convolutional layer (1st or 2nd) and the horizontal axis specifies the number of batch iterations. Note, in the first layer, that the primary circle appears at 300 batch iterations, breaks apart at 500 iterations, and then reappears at 1500 batch iterations with some inner secondary structures. The primary circle appears in the second convolutional layer at 1000 batch iterations.

5.3 ImageNet and VGG

We look at the spatial filters of a single pre-trained network VGG16 [4] trained on ImageNet. VGG16 contains 13 convolutional layers. The first layer only has $3 \times 64 = 192$ spatial filters which proved too few to locate a significant topological structure using Mapper or Plex. However, subsequent layers have many more spatial filters. In Figure 14 we include the Mapper output of the 12 convolutional layers following the first layer. For each layer we use $Mapper(30, 3)$ and for layer 3-13 we use $\rho(100, 0.3)$ while for layer 2 we use $\rho(100, 0.4)$.

In all but the last layer (layer 13) we find the primary circle as the dominant structure. We also find some patches that have no counterpart in the Klein bottle model in [2], notably in layers

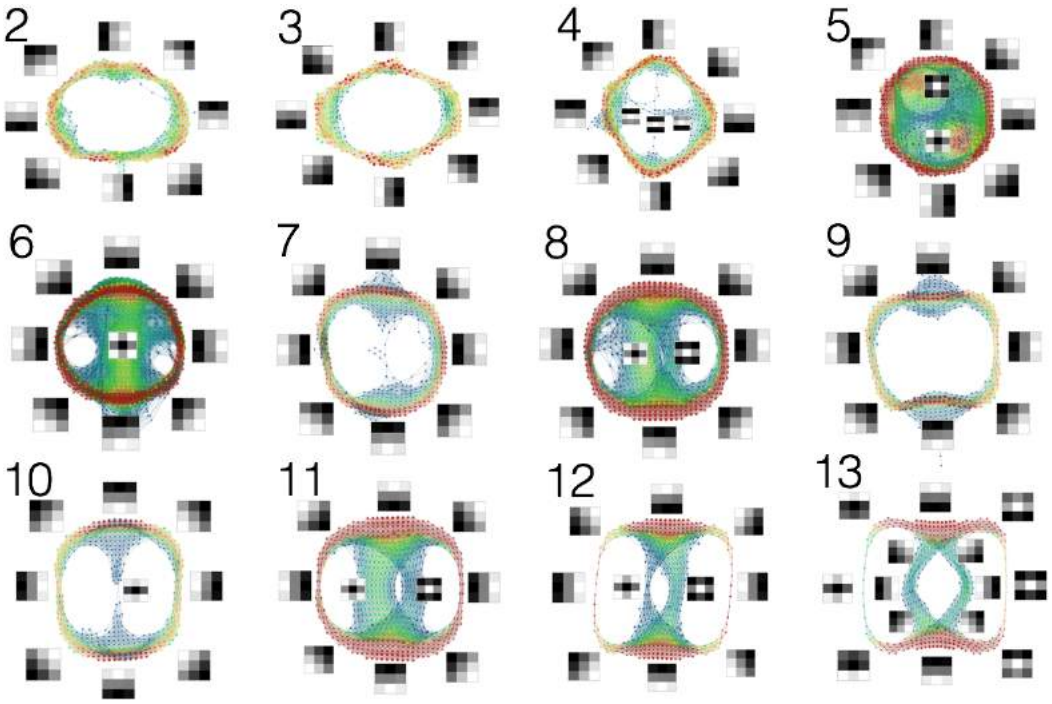

Figure 14: Mapper applied to the convolutional spatial filters of VGG16

5,6,8,11,12,and 13. Note that they appear in the higher layers and may reflect things detected in higher layers in the human visual pathway. We also look at a pre-trained network VGG19 [4] where we find other dominant structures at certain layers, for example already at layer 5 in VGG19 we find the dominant circle in layer 13 of VGG16, see Figure 15. Also note that this circle closely resembles that found in Figure 11.

6 Discussion

The purpose of this paper is to demonstrate that topological modeling can be used as an effective tool to obtain understanding of the functioning of CNN's. We have chosen not to draw conclusions in the present paper because we want to expose the results, and encourage others to formulate hypotheses from this kind of analysis. Many of the results we found were unexpected and non-trivial, and went beyond the results of the motivating paper [2].

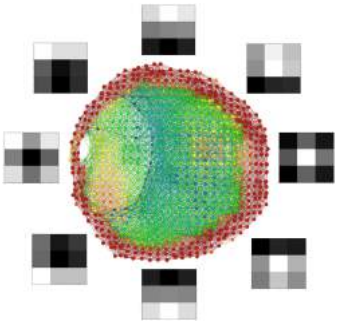

Figure 15: $Mapper(30, 3)$ of $\rho(100, 0.3)$ of the fifth convolutional layer in VGG19

We have shown that the spaces of spatial filters learn simple global structures. This is true not only for the first layer, but occurs at least up to layers at depth 13. We have also demonstrated the change of the simple structures over the course of training.

Topological data analysis is thus a useful framework for the analysis of CNN's. Most immediately, it can make precise observations that have been made on an examination of individual weight vectors, but we also feel that it can give new understanding and new findings that are not made clear by direct examination of data points. CNN's are clearly very powerful tools, and we hope to better understand what and how they learn so that we might use them as way to help increasing human understanding rather than replacing it.

References

[1] Zeiler, Matthew D. & Fergus, Rob. (2014) Visualizing and Understanding Convolutional Networks. *Computer Vision –ECCV 2014*, pp. 818–833. Springer International Publishing

[2] Carlsson, Gunnar and Ishkhanov, Tigran and de Silva, Vin and Zomorodian, Afra. (2008) On the Local Behavior of Spaces of Natural Images. *International Journal of Computer Vision* **76**(1)1-12

[3] Carlsson, Gunnar. (2009) Topology and Data. *Bulletin (New Series) of the American Mathematical Society* **46**(2) 255–308

[4] Karen Simonyan and Andrew Zisserman. (2014) Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR* 1409.1556

[5] Y. LeCun. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist.

[6] A. Krizhevsky. (2009) Learning multiple layers of features from tiny images. Technical report, University of Toronto

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. FeiFei. (2009) Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition.*

[8] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. (2015) TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[9] G. Carlsson. (2014) Topological pattern recognition for point cloud data. *Acta Numerica* **23** 289-368

[10] G. Singh, F. Memoli, and G. Carlsson. (2007) Topological methods for the analysis of high dimensional data sets and 3D object recognition. *Point Based Graphics*, Prague

[11] M. Carriere and S. Oudot. (2015) Structure and stability of the 1-Dimensional Mapper. arXiv:1511.05823.

[12] Ayasdi, TDA and machine learning (2016) https://www.ayasdi.com/wp-content/uploads/_downloads/wp-tda-and-machine-learning.pdf

[13] Tausz, Andrew and Vejdemo-Johansson, Mikael and Adams, Henry. (2014) JavaPlex: A research software package for persistent (co)homology. *Lecture Notes in Computer Science 8592*, pp. 129-136

[14] H. Sexton, J. Kloke. (2015) Systems and methods for capture of relationships within information. U.S. Patent. 14/639,954. Filed Mar. 5, 2015

[15] Tierny, Julien. (2017) Topological data analysis for scientific visualization. Mathematics and Visualization. *Springer, Cham.* ISBN: 978-3-319-71506-3; 978-3-319-71507-062-07

[16] Topological and statistical methods for complex data. Tackling large-scale, high-dimensional, and multivariate data spaces. Papers from the Workshop on the Analysis of Largescale, High-Dimensional, and Multi-Dimensional and Multi-Variate Data Using Topology and Statistics held in Le Barp, June 12–14, 2013. Edited by Janine Bennett, Fabien Vivodtzev and Valerio Pascucci. Mathematics and Visualization. Springer, Heidelberg, 2015. ISBN: 978-3-662-44899-1; 978-3-662-44900-4 94-06

[17] Karen Simonyan, Andrea Vedaldi, Andrew Zisserman. (2014) Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. arXiv:1312.6034

[18] Anh Nguyen, Jason Yosinski, Jeff Clune. (2016) Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned By Each Neuron in Deep Neural Networks. arXiv:1602.03616v2

[19] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, Jeff Clune. (2016) Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. arXiv:1605.09304

[20] A. Mahendran and A. Vedaldi. (2015) Understanding deep image representations by inverting them. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, 2015, pp. 5188-5196