Finding the "Network" Behind Neural Networks

Gabriel Darnell
ECE, UT Austin
gdarnell@utexas.edu

Thomas Leonard
ECE, UT Austin
thomasleonard@utexas.edu

Abstract

Neural networks are able to make sense of unstructured information such as images, but this process is often thought of as a "black box". In order to understand this process, we have started with a simple multilayer perceptron and analyzed the graphical structure and network science metrics. We have found a relationship between network modularity and model accuracy. Then LeNet was trained on MNIST and an initial network was built around the convolutional feature maps utilized. Nodes were selected as the highest value pixels from the feature map and connected with edges if the pixels were adjacent. This method showed higher average degree over an untrained network.

1. Introduction and Motivation

The main idea of this project is to understand the inner mechanism of neural networks and pull back the curtain on artificial intelligence. Convolutional neural networks (CNNs) are difficult to visualize due to hidden layers, which obscure the inner workings of the network. Representing CNN architectures with networks allows for a deeper understanding of the data flow of these models. This allows us to be able to analyze and predict the performance of different CNNs by looking at their underlying graph structure. Our approach analyzes the structure of feature maps within the CNN and builds networks based on topology. We selected the brightest pixels within the feature map and represented them with nodes. Then we connected the nodes in two different ways, spatial adjacency and value adjacency. We performed network analysis as a function of training to find relationships between the model accuracy and the network metrics. This visualization will help determine CNN efficacy beyond standard benchmarking.

2. Prior Work

There have been several investigations into the network science of convolutional neural networks. The work of the community has inspired the methodology presented here.

Filan et al. [1] found an intuitive way to characterize Multilayer Perceptron (MLP) neural networks by their clusterability. Groups of nodes (neurons) with high internal connectivity but low external connectivity create clusters. They found that trained neural networks are more clusterable than they were before training. Additionally, pruning and dropout were utilized and found to be effective at increasing the clusterability since the pressure to minimize connections increases each connection's value to the network. After investigating MLPs the same method was used for CNNs, using the channels within the hidden layers as the nodes. The results showed similar behavior for both the MLP and CNN. This method is very effective for visualizing the learned weights of trained neural networks [1].

You et al. [2] formulated an approach of generating relational graphs based on the communication between nodes in the neural network within a single layer. This method creates a relationship between the network science metrics, such as average path length (L) and clustering coefficient (C), and the performance of the CNN. The key to their approach is to represent an input and an output channel of a neural network as a single node of a graph, then represent communication between the nodes within the layer as an edge. Then a relaxed version of the Watts-Strogratz (WS) model without the requirement that all nodes have equal degree before rewiring is used to generate the graph. This method allows for nearly complete freedom along the parameter space C vs L. This work showed direct relation between the graph structure and the neural network performance for both MLP and CNN [2].

Given the success of these two teams, we formulated an approach for investigating the networks behind neural networks. We start with MLP networks and investigated the network science metrics that define their underlying graphical structure. We then correlate the metrics to performance

by investigating what metrics change the most moving from a randomly initialized MLP to a trained, highly accurate MLP. Modularity is dependent on edge weights [1] so we expect this metric to be dependent on training.

For the case of CNNs, drastic changes to the way the networks are built have to be made. For example, Zhao and Zhang utilized the feature topology within the CNN to build network models [3]. As a kernel moves along an input image, a feature map is produced based on the dot product between the kernel and the underlying pixels in the image. The value of the dot product determines the value for that pixel and by extension the "brightness". Several of these feature maps are produced and sampled within the CNN and highlight certain spatial arrangements of important regions within the input image that are resistant to shifting and rescaling, making them ideal for large datasets of similar images. The highest values within the feature map can be represented as nodes in a graph. Nodes are connected by edges if one feature is a subset of another, ie a single connection is made between nodes if one is a subset of the other, and more complicated edges are formed for a large group of nodes. This notation showed a relationship between kernel structure of the CNN and entropy of the resulting graphs. They found that if the kernel was setup for edge detection (0 on one side and 1 on the other) then the resulting graph was much more ordered than if the kernel started out with a random arrangement of values [3].

Gabrielsson and Carlsson [4] also use persistent homology, which are signature features that appear in most images within a class and can be used to sort inputs. They created nodes using an algorithm to perform hierarchical clustering based on predetermined metrics. Then edges were formed between clusters that overlapped. This method was utilized for a variety of kernels to create intricate networks that utilized a large parameter space. Both of these approaches harnessed the topology of the CNN and the clustering of information into nodes [4, 5].

3. Approach

The beginning of this project is an investigation into the networks behind multilayer perceptron (MLP) neural networks since they are the simplest to visualize. A three level MLP is constructed with randomly initialized weights. A network is built where the nodes are the neurons of the MLP and the edges are the links between neurons. Then the model is trained and another network is built to compare with the untrained network. The network science metrics are analyzed and related to neural network performance. After building a foundation with the MLP, we move on to LeNet. Based on our literature review we investigate the topology and build a network based on feature clustering.

The first network we built using feature maps in the two convolutional layers within LeNet. An untrained kernel can be considered a matrix of weights (3x3) that has not been optimized. The scalar product of this matrix of weights and a subsection of the input image is used to generate a new value. Rasterization of the kernel over the input image produces a new image from the values of the dot products. This is the feature map and it shows points of interest within the image that is passed through the model. The image and the feature map are arrays of values depending on the brightness of the pixel. As the model learns, the weights within the kernel will update and the kernel will be more efficient at recognizing images. This leads to new feature maps as the model becomes more accurate. We generate graphs based on these feature maps and compare the network metrics.

There are several ways to enact this method. We decided to pursue two methods creating nodes and two methods for connecting nodes with edges. The nodes are selected as the highest or lowest values in the feature map. The top 10% brightest pixels or the bottom 90% darkest pixels will be represented as nodes in their respective networks. We will focus our explanations around the brightest pixels approach.

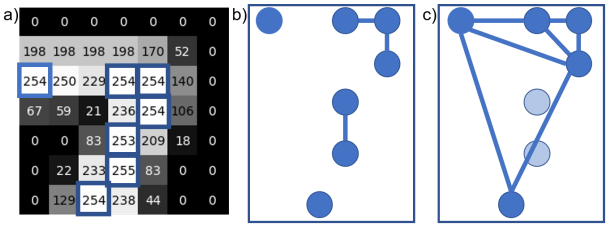

Figure 1. a) Section of an example feature map with the top 1% highest pixel values which will become nodes boxed. b) In the spatial adjacency model edges are assigned if the pixels are touching. c) In the value adjacency model edges are assigned if their pixel values are similar.

Edges are assigned in two different ways. The first method is based on spatial adjacency. If the pixels, for example the brightest 10%, are directly next to each other (excluding diagonals) in the feature map the nodes that represent them are connected with an edge. Figure 1a,b exemplifies this method. On the left is a subsection of a feature map for "7" with boxes highlighting the brightest 1% of pixels, values 253-255, that will become nodes. To the right is a toy model network for this subsection with edges connecting the nodes if they are directly adjacent. This method is the simplest and therefore we expect the simplest data from it, we should see higher average degree and less connected components after training since no digit has discontinuous features. This method will generate edges only if nodes are orthogonal, so it ignores diagonal adjacency. This will pre-

vent clustering from occurring since edges will form in a grid-pattern and no triangles will be formed. We chose 10% as the cutoff for spatial adjacency since it allowed for the clearest network evolution.

The second method is more abstract. Each node has a discrete value based on the brightness of that pixel. If a pixel has a value within a certain range of another pixel's value then an edge will connect their nodes. Figure 1a,c exemplifies this method, all nodes connect to every other node with the same value in the toy model. Pixels with values other than 254 are no longer connected. This will lead to a clustered network where we can analyze the clustering coefficient and average path length.

The main assumptions in our spatial adjacency approach is that the important pixels in the feature map becomes more connected as the CNN accuracy increases. For value adjacency we assume that the pixels become more homogeneous as a function of training and the value difference between the brightest and darkest pixels will increase. In other words we expect a gray-scale feature map to become more black and white.

4. Experimental Setup and Results

We developed our MLP model and trained on the MNIST handwritten digit dataset and tested the network science metrics before and after training.

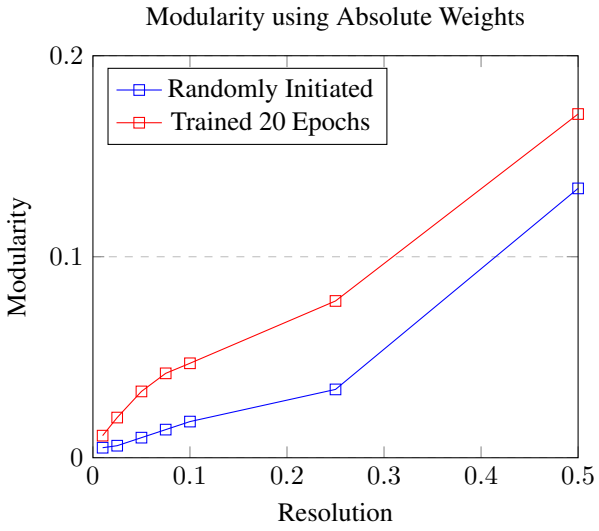

Figure 2. MLP modularity increases after training for all resolutions, indicating stronger communities after training

Our model consists of 5 total layers. The input layer has 784 nodes, one for each pixel of the input image. The hidden layer is comprised of 3 layers with 50 neurons each and followed by a dropout layer p=0.2 to prevent overfitting. Finally there are 10 output nodes, one for each digit.

Before training a randomly initialized graph was generated. The model was trained for 20 epochs by separating the training data into batches. During training we collected the average validation loss and accuracy after each epoch. A final accuracy of 97% and a validation loss of 0.11 was achieved. After the model was trained, another graph was generated in the same fashion for comparison.

After training, network science metrics that did not consider weight were unchanged since only the weights were altered by training. Clustering coefficient remained 0, network diameter remained 4, and average path length remained 2.08. However, modularity was an exception given its dependence on edge weight. The trained network always had a higher modularity, and a lower number of communities, than before training. This is shown in figure 2 for a range of resolutions. This follows what Filan et al. discovered where accuracy was related to clusterability [1]. Also , using the absolute value of the edge weights, we found that the average weighted degree increased after training from 1.15 to 2.25. This relationship indicates that the network has more clustered communities and stronger connections after training, and these characteristics allow for a more accurate neural network.

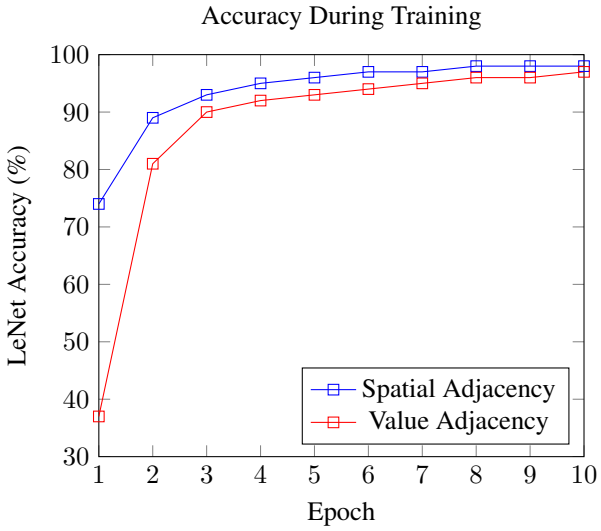

Figure 3. Accuracy during training for spatial and value adjacency methods using brightest 10% of pixels

The CNN was not as straightforward to translate into a graph as the MLP. First a LeNet model was built and we adjusted the forward function to return the feature maps generated from both convolutional layers. Afterwards MNIST

testing images were passed through the model and the untrained model returned the feature maps for those input images. We then collect the individual feature maps in a list and average them all to have a single composite feature map representing the entire testing dataset. We then trained the CNN in the standard way with the MNIST training dataset. This training progress is shown in figure 3 for 10 epochs, achieving an accuracy of 98%. We elected to use 5 epochs, or 95% model accuracy, to build our networks. Then the testing images were passed through the model and the trained kernel created feature maps again which were passed through the average function to obtain the final composite feature map again. We then sorted the pixels of the averaged feature map by numerical value, otherwise known as brightness, and selected the top 10% of pixels to become nodes. Then we created an edge connecting nodes if they were spatially adjacent. We then separated the testing dataset by digit to allow us to analyze the feature maps for a specific digit. We used the same methods for averaging a list of feature maps to allow us to generate a composite feature map for an entire digit class just like we did for the untrained and trained feature maps.

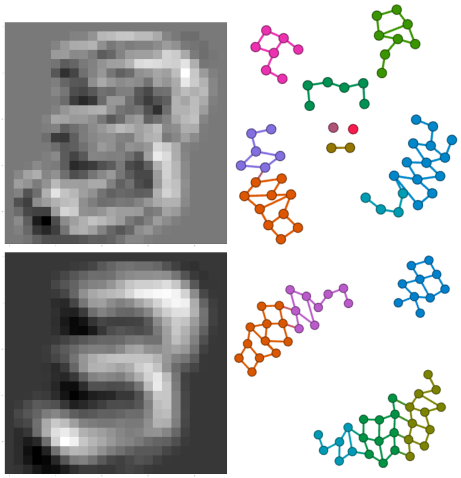

Figure 4. Feature map and associated network generated by the untrained model above and trained model below showing a more connected network after training

Figure 4 exemplifies this process. The top left image shows the feature map for an image in class "3" that was generated by the untrained model. The bottom left image shows a more defined feature map generated by the trained model. To the right of these feature maps are their respective networks. The untrained network has several components and unconnected nodes. The trained network has only 3 components. The average degree rose after training from 2.1 to 2.7, which is indicative of the increased uniformity and connectivity of the network after training.

Network metrics were performed on the spatial adjacency networks generated from the feature maps for each class 0-9 before and after training. Network diameter and average path length increased after training, but only since the number of components decreased. There was zero clustering in any network as hypothesized since diagonal edges were not formed. Therefore the number of components and average degree were investigated over other metrics. Figure 5 highlights the average degree before and after training for all classes 0-9, showing all networks have significantly higher average degree after training.

To build on this result we analyzed the evolution of the network throughout training. We simulated training 10 different times that all reached different accuracy. We generated networks for each and analyzed the average degree. This result is plotted in figure 6 with representative networks inset. There are three regions we observe. The first region represented by the least connected network has several small components and lone nodes. There is a steady increase in network average degree as these lone nodes are incorporated in until around 70% LeNet accuracy. Then we observe a jump in degree as the three main components emerge, shown in the center inset. This is followed by another linear region where the three main components become highly connected, shown in the right most inset.

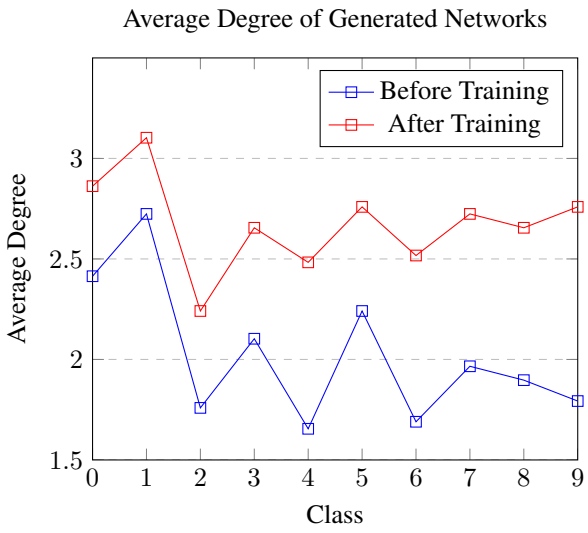

Figure 5. Average degree for spatial adjacency method networks increases significantly after training for all classes

The spatial adjacency model was also used on the darkest 90% of nodes but this showed less improvement as a function of training. Since this method uses the "dark" regions around the digit it was not as sensitive to the training.

We found highly connected networks with only one component before and after training, and only a small increase in the average degree after training. Therefore we focus on the brightest pixels in our analysis.

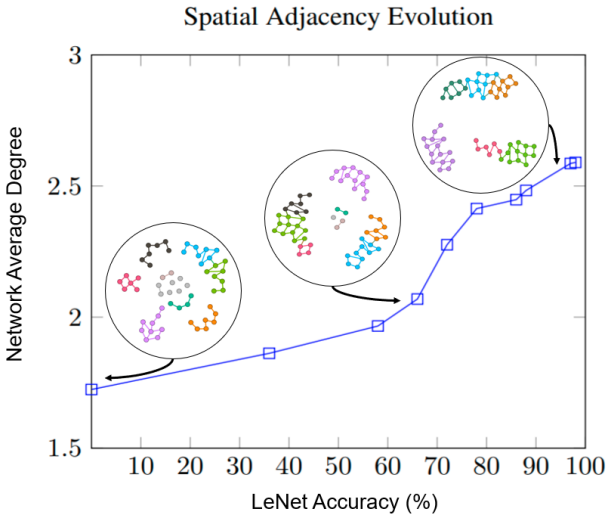

Figure 6. Spatial adjacency evolution with top 10% brightest pixels with three regions highlighted showing network evolution

We also tested the evolution of the network based on the average feature map of the testing dataset. We achieved this by averaging all feature maps returned by the model during testing for each epoch and then creating a graph from this averaged feature map. For this approach we used value adjacency for the top 10% brightest pixels. The value adjacency model was normalized by using a dynamic offset value based on the difference between the brightest and darkest pixels from the averaged feature map. Then an edge was formed if the value difference between two pixels was less than this offset value, and the weight of the edge was determined by the offset divided by the value difference. This increased the efficiency of edge creation. We found a similar strong correlation between training and network metrics. The accuracy was 37% after the first epoch and 97% for the last epoch, shown in figure 3. The average weighted degree went from 32 to 94, showing a strong increase in connectedness in figure 7. Additionally, the clustering coefficient and average path length were analyzed. The clustering coefficient remained high and steady throughout training, while the average path length decreased significantly after a couple of epochs. The normalized relationship is plotted in figure 8 and indicates the emergence of a small world phenomenon.

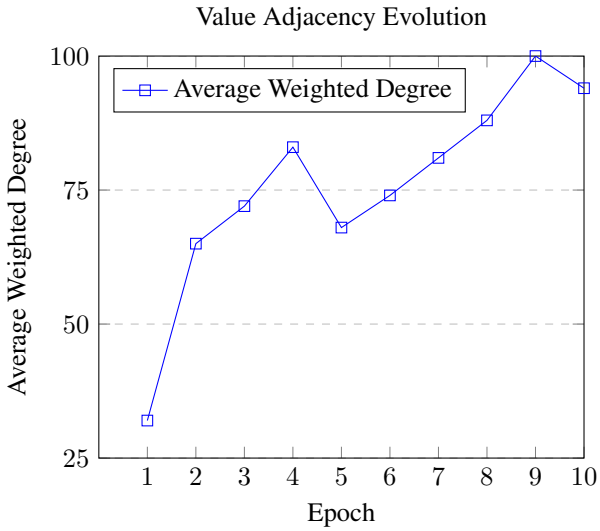

Figure 7. A strong correlation between training level and average weighted degree was observed in the value adjacency evolution

Analyzing only a single class, digit 3, we found similar results before and after training. While the average degree was unchanged, the average weighted degree increased significantly from 16.7 to 25.2. The average path length also decreased by roughly 50% while the clustering coefficient remained unchanged. This indicates that our value adjacency model is scalable since we observed nearly identical results from one digit as from the entire dataset.

The darkest 10% of pixels were also analyzed for this value adjacency method. The cutoff was significantly reduced since the number of edges would be too cumbersome if 90% of the nodes were utilized. We found very similar results for this method since the darkest 10% of nodes largely exist in the digit itself. The average weighted degree roughly doubled after training but the average path length actually increased after training, since some nodes are a part of the background and therefore skew the data. Therefore, we focused on the brightest pixels for our analysis.

Altogether we found that the average degree increased significantly after training for our spatial adjacency method. We observed a positive correlation between LeNet accuracy and average degree. For our value adjacency model the average weighted degree rose significantly after training and the average path length decreased while the clustering coefficient remained high. These relationships show our networks are more connected after training, suggesting these characteristics allow for a more accurate neural network.

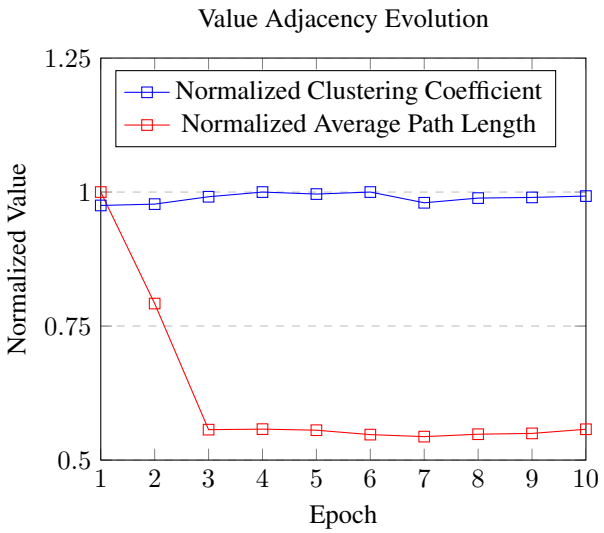

Figure 8. Normalized clustering coefficient and average path length for value adjacency method as a function of training. Small world behavior is observed as clustering coefficient remains high while average path length drops significantly.

5. Conclusion and Future Work

In conclusion, an MLP network was modeled, trained, and analyzed. We achieved high accuracy and showed the modularity and average weighted degree scaled with this accuracy. For the CNN we analyzed the topology of feature maps for each digit class before and after training using our novel spatial adjacency model. We focused on the brightest 10% of pixels which showed the most promising results. The network metrics showed that the average degree increased and the number of components decreased, suggesting the networks are more connected after training. We observed strong correlation between LeNet accuracy and average degree during our network evolution analysis. For the value adjacency model we saw similar relationships as the average weighted degree nearly tripled after training. We also observed a small world emerge during our network evolution analysis as the clustering coefficient remained at 0.8 while the average path length dropped from 4 to 2. This increased connectivity correlates to the increased accuracy of the CNN after training.

Continuation of this work would analyze other CNN architectures such as ResNet and AlexNet. We would also evaluate our methods efficacy against the corrupted MNIST and fashion MNIST datasets.

6. Contributions and Lessons Learned

Gabriel wrote the code that trained LeNet and constructed the networks using our methods. Thomas evaluated the network science metrics and analyzed results.

There was a steep learning curve for both team members. We both had limited machine learning experience so the whole project was new to us. Gabriel had limited Python experience and Thomas had no previous software experience. Both team members learned a lot about machine learning models in general and the correlation between network metrics and performance of multiple machine learning models.

References

[1] Filan Daniel et al. “Clusterability in Neural Networks”. In: *arXiv* (2021).

[2] J. You et al. “Graph structure of neural networks”. In: *PMLR* (2020).

[3] Yang Zhao and Hao Zhang. “A topological approach to exploring convolutional neural networks”. In: *PNAS* (2020).

[4] Rickard Gabrielsson and Gunnar Carlsson. “A look at the topology of convolutional neural networks”. In: *arXiv* (2018).

[5] Gurjeet Singh, Facundo Mémoli, and Gunnar Carlsson. “Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition”. In: *Eurographics Symposium on Point-Based Graphics* (2007).