Exploring Interpretability
by Alan H. Teague
AI Safety Fundamentals (March 2024)

Version 1 (June 10, 2024)

During the Interpretability section of the AI Safety Fundamentals class, I was struck by the similarity of circuits to directed acyclic graphs (DAGs). The question that popped in my head was whether we can start with the output nodes of a model and work backwards to find circuits. Using this question, I identified a viable labeled data set, created a model and a set of data sets for analyzing the model based on the aggregation of how it activated for inputs which generate the same output. This roots the analysis from the output node and expands backwards through the model. The hypothesis is that as the DAG or activation vector moves from the output layer backwards to the input layer, they will increase how much the overlap with those associated with other outputs. By analyzing this overlap and divergence, I may be able to detect features and circuits.

While I planned to create DAGs rooted at the output nodes and expanding going backwards based on the activations, I switched to using vector comparisons. Both DAG and vector comparison are viable methods for this project. DAGs are possibly better at extracting the circuits as they capture the actual flow through the model, but they are computationally and developmentally expensive. Vectors are good at a whole model view of the activations and require significantly less computational and development resources but are not as good at extracting human interpretable features (See Appendex for a detailed discussion of DAG vs Vector Comparison from claude.ai).

First, some information about DAGs, features, and circuits.

DAGs can be used for Deep Neural Networks (DNN) such that the starting nodes are the input layer, the hidden layers are intermediate nodes, the output layers are the terminal nodes, and the weights of the model define the connections between the nodes. While these graphs can be used for a neural network (DAGNN), this project originally planned to use them as a visualization of parts of the neural network.

A “feature” can be defined as the activation of a neuron or a group of neurons in a layer (Neuron and node are used synonymously in this paper). Features capture specific patterns or aspects of the input data. Neurons that are consistently activated when a specific input leads to a particular output are likely contributing to features responsible for that output.

A “circuit” in a neural network is a computational subgraph that represents the flow of information through a specific set of neurons and their connections. It represents a coherent, interpretable computation or concept that the network has learned. It consists of a set of features and the weighted edges that connect them in the original network.

This project uses sets of inputs which all yield the same output to identify the neurons in each layer that are significantly activated and contribute to the final output. These neurons are considered to be features. The pathways formed by these activated neurons from one layer to the next are circuits.

It is worth noting, features and circuits are more complex than this basic description. Neurons can participate in multiple, distinct circuits (polysemanticity) as well as encode multiple concepts

(superposition). The field of interpretability is researching these and many other aspects of how models function.

Experimental Process

This exploration project uses the MNIST image dataset as the basis due to the limited timeframe for the project, the minimal computational requirements for DNN trained using this dataset, and a clear set of labeled inputs. First step was to create a DNN for classifying digits which has a high accuracy and low loss. Table 1 shows the details of the model. The model was implemented Keras-based model. The 28x28 image is flattened to a 784 vector. There are three hidden layers decreasing in size: 128-, 64-, and 32-node. These use the 'ReLU' activation and also used random dropouts during training. Note that dropout is not used during the prediction phase. The final 10-node layer uses the 'softmax' activation.

	Shapes of vectors and matrixes			
Model	Output	Weights	Biases	Data
L0	(784)	(784,128)	(128)	(784)
L1	(128)	(128,64)	(64)	(128)
L3	(64)	(64,32)	(32)	(64)
L5	(32)	(32,10)	(10)	(32)

Table 1: Description of Model (L2 and L4 are dropout layers) activated per level per digit

Next, the evaluation image set was segmented by digit and then filtered to only include those images which are clearly predicted. As can be seen in Table 2, this results in input/output data with a 100% accurate prediction of individual digits.

	Base Data Set		Filtered Data Set	
	Acc	Loss	Acc	Loss
All Digits	0.980	0.108		
0	0.992	0.055	1.000	0.0001
1	0.996	0.028	1.000	0.0005
2	0.971	0.133	1.000	0.0010
3	0.980	0.105	1.000	0.0015
4	0.975	0.155	1.000	0.0007
5	0.976	0.092	1.000	0.0002
6	0.985	0.122	1.000	0.0000
7	0.983	0.066	1.000	0.0000
8	0.973	0.162	1.000	0.0003
9	0.968	0.169	1.000	0.0004

Table 2: Model accuracy and loss values

These clean datasets were then run through the model collecting data on the output values for each neuron in a layer. I used this by-digit activation information to extract various vectors by layer for a given digit for comparing with similar vectors for other digits.

Results and Analysis

This experiment collected the output layer values for each of the node in each of the 4 productive layers (dropout is only used during training and simply passes through the results during prediction). The mean, variance, and count of these values were calculated for all of the sample images for a given digit.

A key question is which nodes are actually contributing to the resulting classification. ‘ReLU’ and ‘softmax’ activation functions both filter out any negative values for determining a node’s output value. Any output value combined with a negative weight would push the model towards not activating the connected node. As these do not contribute to the specific classification, these activation data values are zeroed out prior to comparisons. Currently, the project drops all data points which would be used with a negative weight. While this approach is accurate for the output to the final classification, it becomes increasingly less accurate for the prior layers. Future work should only zero out the activations which feed into the final classification, and then only those that feed into the remaining activations, repeating this backwards through the model layers.

As can be seen in Table 3, the nodes contributing to the classification of a given digit are less than half of the nodes of a given layer. L0 has 784 nodes, L1 has 128 nodes, L3 has 64 nodes, and L5 has 32 nodes.

		Contributing Nodes			
Digit	Images	L0	L1	L3	L5
0	746	193	48	23	12
1	746	173	51	11	13
2	745	272	61	23	13
3	744	234	45	28	15
4	745	241	50	20	14
5	746	258	62	14	13
6	746	208	57	24	13
7	745	240	60	26	13
8	746	204	52	26	17
9	745	186	51	23	15

Table 3: Cleanly classifiable input images and resulting nodes

For the mean dataset, the ranges for the means are similar for the initial layer (L0) and get progressively different for deeper layers. See Table 4 for details on the ranges by digit and level.

Digit	L0	L1	L3	L5
0	3.2e-05 - 0.8	2.5e-04 - 5.8	3.5e-04 - 9.3	5.3 - 15.9
1	3.7e-05 - 1.0	0.5e-04 - 1.9	5.7e-04 - 7.3	4.9 - 12.1
2	3.7e-05 - 0.7	7.1e-04 - 3.2	1.3e-03 - 9.3	0.0 - 11.8
3	6.9e-05 - 0.8	8.9e-04 - 2.5	2.5e-04 - 8.4	2.5 - 10.2
4	5.3e-06 - 0.8	2.1e-04 - 2.9	1.3e-04 - 7.3	4.0 - 9.7
5	1.1e-05 - 0.7	2.0e-04 - 3.0	6.5e-04 - 8.4	5.5 - 10.8
6	2.6e-05 - 0.8	1.1e-04 - 4.4	1.7e-04 - 8.1	0.0 - 16.0
7	4.7e-05 - 0.8	6.2e-04 - 2.9	0.4e-04 - 5.7	4.5 - 10.2
8	2.6e-05 - 0.9	5.6e-04 - 4.0	1.7e-04 - 3.8	1.6 - 5.3
9	3.2e-05 - 0.9	3.2e-04 - 3.4	0.3e-04 - 4.3	2.1 - 8.3

Table 4: Ranges for mean activation values by digit at the given level.

For the activation count dataset, the ranges are similar for the most part with two digits (2 and 6) in L5 showing significantly different minimums – 5 and 8 compared to 697-746. Future work should look at these distributions and potentially drop the outliers implied by their means being 689. For L0, L1 and L3, the standard deviations in comparison to the means points to future work exploring these distributions and may be related to the method use for determining contributing nodes. See Table 5 for these details.

	Activation Counts											
Digit	L0	avg	std	L1	avg	std	L3	avg	std	L5	avg	std
0	1-672	333	240	1-713	248	264	1-744	101	231	743-746	745	1
1	1-746	151	221	1-716	152	182	1-743	184	301	743-746	745	1
2	1-666	259	222	1-516	110	146	1-744	345	327	5-745	686	196
3	1-704	217	221	1-598	110	147	1-739	283	335	716-744	740	7
4	1-719	208	222	1-645	172	195	1-744	288	339	728-745	742	4
5	1-669	226	197	2-597	209	187	1-741	161	258	742-746	745	1
6	1-716	278	238	1-688	101	180	1-744	197	280	8-746	689	196
7	1-693	207	215	1-598	141	191	1-735	78	192	728-745	742	4
8	1-718	251	244	1-665	74	129	1-740	101	225	697-746	741	11
9	1-731	290	248	1-707	120	167	1-745	319	341	736-745	743	2

Table 5: Statistics on the number of times each neuron was activated for the specified digit at the given level.

There are multiple methods for comparing vectors: Euclidean Distance, Manhattan Distance, Chebyshev Distance, Cosine Similarity, and Jaccard Similarity. Euclidean is used for continuous data with similar scales. Manhattan is also used for continuous data but is better for different scales. Chebyshev focuses on the most significant differences. Cosine looks at similarities without considering magnitude differences. Jaccard is used for binary data to determine how much one overlaps with the other.

For this project, Euclidean and Jaccard comparison will be used. The values are continuous and the ranges are similar which makes Euclidean a reasonable comparison. For Jaccard, the activation vectors are converted to indicate activation. The similarity result is the percentage of the nodes in a given layer which are common across pairs of digits. Only nodes which are used in at least one of the digits are used for this calculation. Future work should look into the other methods to see what information may be extracted.

Using Euclidean comparison, comparing the means between digits show consistent divergence as the activations proceed down the layers. This is what was expected. For L0, there are no obvious similarities between digits. For L1, 0 diverges significantly from the other digits. This divergence is not apparent in L0 and L3, though L5 does show some divergence. This may indicate a feature distinguishing 0. The other comparisons are not providing any clear pair differences. See Table 6 for the specific values.

	Euclidean Comparison of Mean Activation Vectors									
L0	0	1	2	3	4	5	6	7	8	9
0	0	6.49	6.54	6.27	5.83	5.79	6.22	5.85	6.33	6.2
1	0	0	5.07	5.31	4.99	5.03	5.28	5	5.27	5.19
2	0	0	0	5.65	5.67	5.45	5.68	5.93	5.89	5.74

3	0	0	0	0	5.33	4.64	5.48	5.33	5.04	5.72
4	0	0	0	0	0	4.78	4.72	5.31	5.47	4.56
5	0	0	0	0	0	0	5.1	5.45	5.09	5.6
6	0	0	0	0	0	0	0	5.78	5.58	5.34
7	0	0	0	0	0	0	0	0	5.08	5.13
8	0	0	0	0	0	0	0	0	0	5.92
9	0	0	0	0	0	0	0	0	0	0
L1										
0	0	15.18	15.68	15.36	15.76	16.59	15.66	15.61	14.5	15.48
1	0	0	6.7	4.96	7.88	10.82	8.63	6.48	5.8	5.77
2	0	0	0	7.59	8.86	11.86	9.19	8.02	7.56	7.76
3	0	0	0	0	8.01	10.89	8.94	7.1	6.3	5.74
4	0	0	0	0	0	12.4	9.3	9.15	8.19	8.05
5	0	0	0	0	0	0	13.2	11.45	10.8	11.22
6	0	0	0	0	0	0	0	9.99	9.19	9.24
7	0	0	0	0	0	0	0	0	6.89	7.69
8	0	0	0	0	0	0	0	0	0	6.37
9	0	0	0	0	0	0	0	0	0	0
L3										
0	0	13.95	24.25	22.63	18.86	16	17.41	14.55	11.37	15.68
1	0	0	22.91	21.18	14.77	13.88	15.67	12.26	10.39	13.51
2	0	0	0	26.59	25.38	23.35	24.01	22.42	21.45	23.12
3	0	0	0	0	23.84	17.84	22.7	20.6	19.6	20.67
4	0	0	0	0	0	17.69	19	16.44	14.93	15.15
5	0	0	0	0	0	0	16	13.06	11.34	13.04
6	0	0	0	0	0	0	0	14.94	13.45	15.97
7	0	0	0	0	0	0	0	0	9.28	12.66
8	0	0	0	0	0	0	0	0	0	10.31
9	0	0	0	0	0	0	0	0	0	0
L5										
0	0	49.29	42.94	45.65	41.16	47.48	42.11	45.1	39.76	39.97
1	0	0	32.93	31.2	28.19	36.93	40.63	31.27	28.93	30.08
2	0	0	0	27.79	30.95	36.93	41.06	28.53	26.49	29.99
3	0	0	0	0	29.1	26.36	43.8	28.35	24.28	25.07
4	0	0	0	0	0	33.58	36.12	27.56	23.45	20.71
5	0	0	0	0	0	0	39.01	33.16	26.72	27.71
6	0	0	0	0	0	0	0	42.78	34.63	39.55
7	0	0	0	0	0	0	0	0	27.29	22.48
8	0	0	0	0	0	0	0	0	0	18.91
9	0	0	0	0	0	0	0	0	0	0

Table 6: Euclidean Distance between pairs of digits at a given layer based on means

Euclidean comparison of the activation counts show a divergence progressing from L0 through L1 to L3 but then converges at L5. This contradicts the expectation and probably confirms that the method for dropping nodes is not accomplishing the goal of focusing on the contributing nodes. The different activation count ranges for digits 2 and 6 in L5 identified above are not seen in the Euclidean comparisons. This would agree with the hypothesis that there are only minimal outliers. See Table 7 for the specific values.

	Euclidean Comparison of Activation Count Vectors									
L0	0	1	2	3	4	5	6	7	8	9
0	0	8.55	8.65	8.27	7.56	7.7	8.19	7.75	8.46	8.02
1	0	0	6.77	7.1	6.75	6.81	6.96	6.69	7.11	6.81
2	0	0	0	7.61	7.73	7.44	7.71	8.03	8.07	7.78
3	0	0	0	0	7.23	6.42	7.44	7.23	6.87	7.7
4	0	0	0	0	0	6.51	6.35	7.21	7.52	6.26
5	0	0	0	0	0	0	6.91	7.51	7.05	7.67
6	0	0	0	0	0	0	0	7.72	7.64	7.18
7	0	0	0	0	0	0	0	0	6.99	7.03
8	0	0	0	0	0	0	0	0	0	8.13
9	0	0	0	0	0	0	0	0	0	0
L1										
0	0	4.05	3.64	3.67	3.74	3.97	3.53	3.96	3.42	3.85
1	0	0	2.67	2.28	3.32	3.24	3	2.81	2.67	2.79
2	0	0	0	2.46	2.81	3.22	2.4	2.69	2.24	2.62
3	0	0	0	0	2.69	3.07	2.52	2.6	2.03	2.07
4	0	0	0	0	0	3.6	2.77	3.18	2.58	2.69
5	0	0	0	0	0	0	3.59	3.37	3.05	3.19
6	0	0	0	0	0	0	0	3.11	2.5	2.82
7	0	0	0	0	0	0	0	0	2.46	2.97
8	0	0	0	0	0	0	0	0	0	2.2
9	0	0	0	0	0	0	0	0	0	0
L3										
0	0	1.93	3.47	3.51	3.12	2.22	2.56	2.08	1.91	3.39
1	0	0	3.44	3.49	2.76	2.18	2.74	2.12	2.29	3.39
2	0	0	0	4	4.04	3.42	3.62	3.38	3.49	4.28
3	0	0	0	0	4.1	2.72	3.73	3.37	3.54	4.12
4	0	0	0	0	0	3.08	3.32	3.02	2.86	3.26
5	0	0	0	0	0	0	2.58	2.06	2.27	3.1
6	0	0	0	0	0	0	0	2.63	2.75	3.71
7	0	0	0	0	0	0	0	0	2.2	3.31
8	0	0	0	0	0	0	0	0	0	2.98
9	0	0	0	0	0	0	0	0	0	0
L5										
0	0	4.58	4	4.35	3.74	4.12	3.46	4.35	4.12	4.12
1	0	0	3.6	3.73	3.86	3.99	3.87	3.46	3.98	3.99
2	0	0	0	3.85	4.23	4.57	3.99	3.6	3.86	4.57
3	0	0	0	0	4.34	3.46	4.56	3.73	3.97	3.74
4	0	0	0	0	0	4.12	3.73	3.86	4.34	3.6
5	0	0	0	0	0	0	3.6	3.99	3.98	3.46
6	0	0	0	0	0	0	0	4.35	3.87	4.35
7	0	0	0	0	0	0	0	0	4.45	3.72
8	0	0	0	0	0	0	0	0	0	4.22
9	0	0	0	0	0	0	0	0	0	0

Table 7: Euclidean Distance between pairs of digits at a given layer based on activation counts

The Jaccard similarity for the mean activation vectors has several interesting data point. L0 does not show any significant differences of similarities, which is expected as this is close to the raw image. In L1, 7 vs 1, 7 vs 5, 8 vs 2, and 9 vs 8, show significant overlaps. In L3, 7 vs 0, 7 vs 3, and 8 vs 6, show significant overlaps. In L4, 5 vs 3 show significant overlap. In L3, 2 vs 1 show a significant divergence. These may point to potential feature discriminators at these points in the model. There is no value in running Jaccard on the activation counts as the conversion to binary will result in identical vectors for comparison. See Table 8 for details on the Jaccard Similarities.

Jaccard Similarities using Mean Activation Vectors

L0	0	1	2	3	4	5	6	7	8	9
0	100%	20%	27%	23%	29%	27%	29%	26%	23%	32%
1		100%	22%	21%	22%	26%	30%	23%	20%	29%
2			100%	28%	26%	30%	23%	19%	22%	25%
3				100%	27%	30%	25%	23%	26%	27%
4					100%	35%	29%	25%	26%	28%
5						100%	28%	21%	28%	21%
6							100%	21%	21%	28%
7								100%	23%	25%
8									100%	15%
9										100%
L1										
0	100%	16%	20%	19%	38%	21%	21%	20%	27%	21%
1		100%	24%	16%	11%	55%	27%	**41%**	16%	20%
2			100%	25%	31%	34%	23%	32%	**45%**	33%
3				100%	32%	19%	31%	27%	45%	39%
4					100%	17%	24%	25%	34%	36%
5						100%	23%	**47%**	20%	22%
6							100%	27%	22%	32%
7								100%	19%	29%
8									100%	**43%**
9										100%
L3										
0	100%	21%	18%	24%	30%	16%	17%	**40%**	23%	21%
1		100%	**3%**	22%	19%	14%	25%	23%	19%	13%
2			100%	13%	26%	23%	17%	14%	14%	24%
3				100%	17%	14%	33%	**42%**	23%	24%
4					100%	13%	19%	18%	15%	26%
5						100%	15%	14%	11%	19%
6							100%	28%	**43%**	24%
7								100%	21%	26%
8									100%	20%
9										100%
L5										
0	100%	9%	19%	17%	30%	19%	39%	14%	26%	23%
1		100%	30%	33%	29%	24%	24%	37%	30%	27%
2			100%	33%	17%	8%	24%	30%	36%	17%
3				100%	21%	**40%**	17%	33%	33%	36%
4					100%	23%	35%	29%	24%	38%

5	100%	30%	24%	30%	40%
6		100%	13%	30%	22%
7			100%	20%	33%
8				100%	28%
9					100%

Table 8: Jaccard Similarity between pairs of digits at a given layer based on activation means

While the Euclidean Distance results show promise on there being extractable features and circuits within the model, the method used for focusing on contributing nodes needs to be corrected. Even with this deficiency, the Jaccard Similarities shows a lot of promise on being able to identify features.

The main takeaways from this exploration are that using DAG as an analysis technique should be pursued as it conveys the intra-layer interactions which would be used for identifying circuits, Jaccard Similarity proved a useful analysis method, and the selection of which nodes are contributing is critical.

Future Improvements

This work would benefit from the following:
- Add a section for related research
- Add visualizations
    The exploration has generated images showing heatmaps of the activated nodes by level and digit. These are not included in this version of the paper. Some are included in the Appendix
    Visualizations of the DAGs
- Add error analysis
- Expand on the node selection and see how the results vary based on changing parameter to the selection process

APPENDIX:

<u>Comparing Vectors:</u>

1. Euclidean Distance
    euclidean = np.linalg.norm(A - B)
    Square root of the sum of the square of the differences

    Used for continuous data, similar scales
    Sensitive to outliers and assumes all features contribute equally

2. Manhattan Distance
    manhattan = np.sum(np.abs(A - B))
    Sum of the distances between elements

    Used for continuous data, different scales and potential outliers
    Ignores correlations between features

3. Chebyshev Distance
    from scipy.spacial import distance
    chebyshev = distance.chebyshev(A, B)

Maximum distance between elements

Focus on the most significant differences
Ignores other feature differences

4. Cosine Similarity

dot_prod = np.dot(A, B)
norm_A = np.linalg.norm(A)
norm_B = np.linalg.norm(B)
cosine = dot_prod / (norm_A * norm_B)
Sum of the dot product divided by the dot product of the square roots of the sums of squares

Used for document similarities and collaborative filtering
Does not consider magnitude differences

5. Jaccard Similarity

intersection = len(A.intersection(B))
union = len(A.union(B))
jaccard = float(intersection) / union
Percentage which are common

Used for binary data (e.g., true false). Focuses on shared elements.
Ignores magnitude and order

Need to convert vectors to binary based on > 0
Do both full vector and using only where both are not zero
Need to modify to compare based on order

Analyzing using DAGs vs Vectors:

From claude.ai:

Prompt:
For an image classification DNN, I am recording the activations at each layer. I run multiple inputs which all strongly classify to the same value and collect each output layer's values. I have two possible ways to compare the activations between two different images: 1) Create and then compare two directed acyclic graphs rooted at the classification following the positive activations backwards to the input layer; 2) Compare the activation vectors at each output layer for each classification.
Why would I want to choose one vs the other?

Great question! This touches on an interesting aspect of interpreting and comparing the internal representations of deep neural networks (DNNs) for image classification. Let's dive into the two approaches you've mentioned and discuss the pros and cons of each.

1. Comparing Directed Acyclic Graphs (DAGs) of Activations:
- Pros:

a. Captures the causal flow of information: By following positive activations backwards, you trace the path of features that most strongly influenced the final classification. This can reveal the "reasoning" behind the network's decision.

b. Highlights salient features: The paths in the DAG might correspond to the most discriminative parts of the image for that class, potentially aligning with human-interpretable features.

c. Sparsity and efficiency: If the network learns sparse representations, the DAG might be simpler than comparing full activation vectors, making comparisons more efficient.

d. Insight into network structure: It can reveal which neurons or filters are most critical for a particular classification across different images.

- Cons:

a. Complexity: Constructing and comparing DAGs can be computationally expensive, especially for large networks.

b. Thresholding issues: Deciding what constitutes a "positive" activation (e.g., ReLU > 0, or some other threshold) can be arbitrary and impact the resulting DAGs.

c. Ignores negative influences: By only following positive activations, you might miss important negative activations that inhibit other classes.

d. Less suited for densely connected layers: In layers like fully connected ones, most activations might be positive, making the DAG less informative.

2. Comparing Activation Vectors at Each Layer:

- Pros:

a. Full information: You consider all activations, both positive and negative, capturing the full representation at each layer.

b. Easier computation: Comparing vectors (e.g., using cosine similarity, Euclidean distance) is generally faster and more straightforward than comparing graphs.

c. Statistical analysis: You can apply various statistical techniques (PCA, t-SNE) to visualize and understand the distribution of activations across images.

d. Invariance and equivariance: Similarities in activation vectors can reveal invariances (features that don't change with certain input transformations) or equivariances (features that change predictably).

- Cons:

a. Less intuitive: Activation vectors, especially in deeper layers, may not have clear human-interpretable meanings.

b. Curse of dimensionality: In high-dimensional spaces (many neurons), distances can become less meaningful, and you might need to use more advanced similarity measures.

c. Redundancy: Some dimensions of the activation vector might be correlated or redundant, adding noise to the comparison.

d. Missing causal structure: Unlike the DAG approach, you don't directly capture the causal flow of information through the network.

Which to Choose?

1. Choose DAGs if:

- You're more interested in the causal "story" of why an image was classified a certain way.
- Your network learns sparse, interpretable features.
- You want to identify critical neurons or paths for a class across images.
- Computational complexity isn't a major concern.

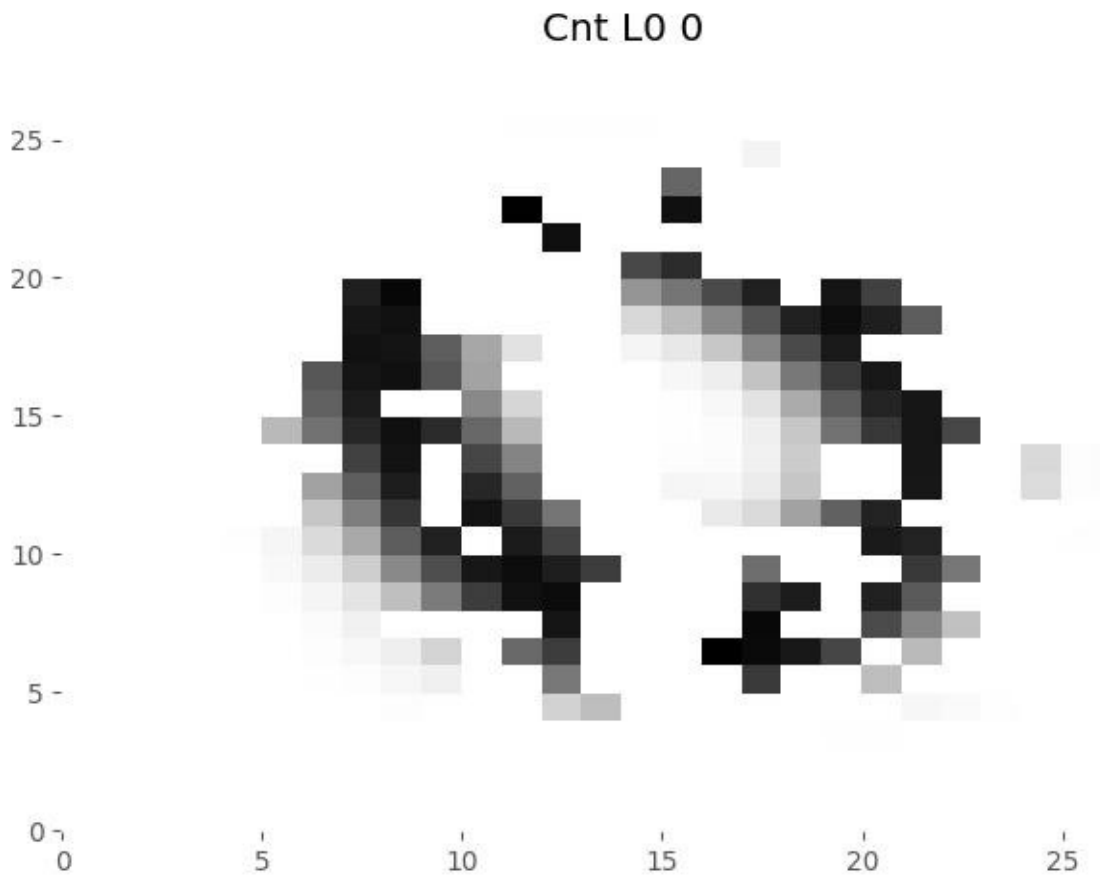
2. Choose Activation Vectors if:
   - You want a holistic view of the network's internal representations.
   - Speed and simplicity of comparison are important.
   - You plan to use statistical techniques to analyze the representations.
   - Your network has many dense connections where most activations are non-zero.

In practice, both methods offer valuable insights, and the best choice depends on your specific goals:
- For interpretability and understanding decision paths, DAGs might be more illuminating.
- For quantitative comparisons, statistical analysis, or when working with very large networks, activation vectors are often more practical.

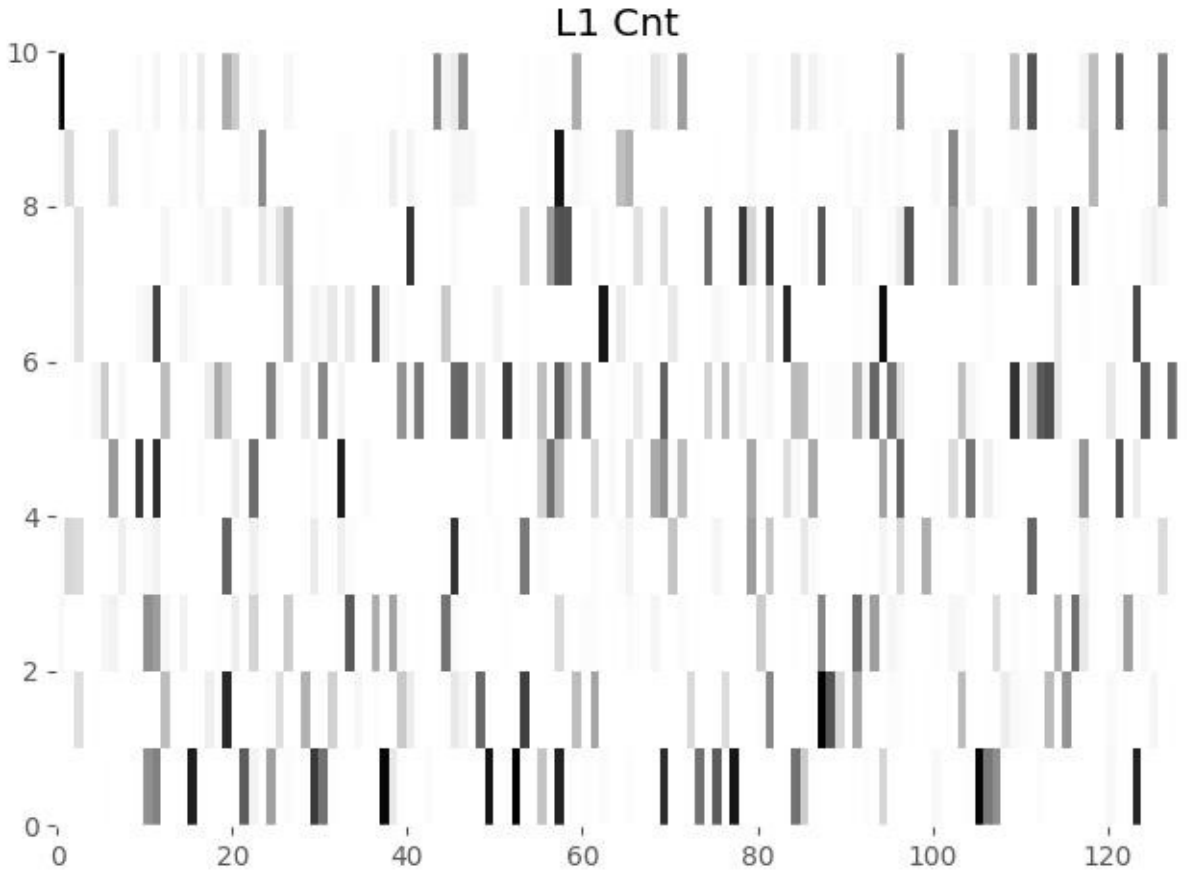
A hybrid approach could also be valuable. For example, use activation vector similarities to quickly find similar representations, then analyze the most interesting cases with DAGs to understand why they're similar.

In summary, DAGs offer a causal, feature-centric view but can be complex, while activation vectors provide a holistic, statistically-friendly view but may be less interpretable. Your choice should align with whether you prioritize interpretability of decision paths or efficient, holistic comparison of internal representations.

Visualizing Activations



L0 Activation Counts for 0



L1 Activation Counts by Digit