A Multi-Planar Graph Visualization of Transformer Multi-Head Attention

Charles Ison[*]
Oregon State University

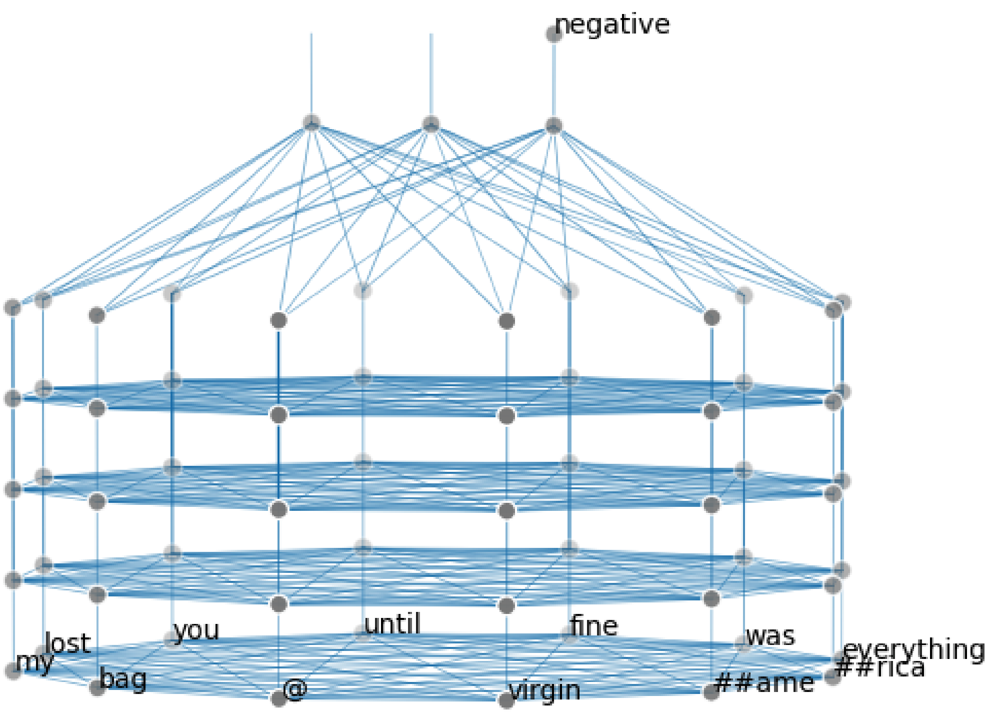

Figure 1: Visualization of multi-head attention is a 6 layer transition encoder trained to do sentiment analysis on tweets directed at major airlines.

1 Introduction

Multi-head attention is the mechanism by which Transformer neural networks quantify the relationships between input elements. The architecture was first proposed in 2017 by Vaswani et al. and has since become one of the most popular architectures for a wide variety of machine learning tasks tasks [Vaswani et al. 2017], [Dosovitskiy et al. 2020], [Arnab et al. 2021]. Because of the success Transformers have seen, there has been interest in using visualizations of multi-head attention blocks as a tool for attempting to explain Transformer's performance [Vig 2019] [Clark et al. 2019] (see Figure 2 for an example). In the original paper, "Attention Is All You Need," Vaswani et al. even included several attention visualizations and argued they could help improve interpretability. Although these visualizations are interesting and can help explain Transformer's inductive biases, they lack context of what roll the specific layer and attention head plays within the overall neural network architecture and eventual output. As an alternative approach, multi-head attention can be visualized as an interactive, 3D fully-connected graphs that are connected across layers by their linear transformations.

2 Background

Prior to the introduction of the Transformer architecture and multi-head attention, the state of the art machine learning models for sequential and time-series data were recurrent neural networks (RNNs). RNNs are a type of neural network where the input to each node is a single token from the input sequence along with the output from the previous node in layer (see Figure 3 for an example). Although this proved to be very successful, the reliance on stepping through each input tokens chronologically was identified as a limitation. For example, the further apart two tokens are in the sequence, the harder it became to identify a relationship between the two tokens. Long Short Term Memory networks (LSTMs) and early attention mechanisms helped mitigate some of these problems, but Vaswani et al. were the first to propose fully removing recurrent steps and focusing purely on attention in their paper "Attention Is All You Need." The intuition behind their idea was that by representing tokens and their relationships using the dot product of matrices, the relationships between each token could be calculated efficiently all at once with no diminishing signal over time.

[*]e-mail: isonc@oregonstate.edu

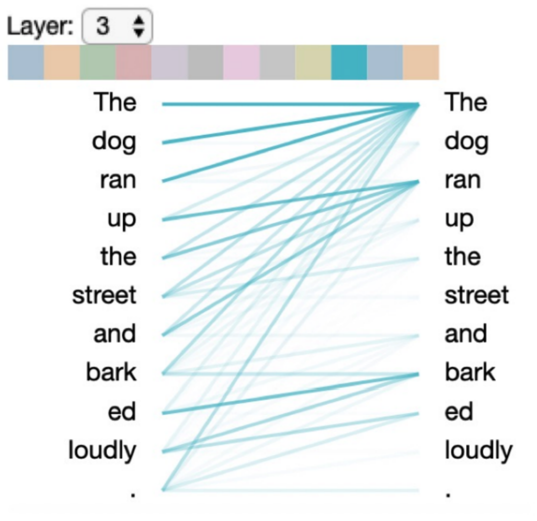

Figure 2: Visualization of attention heads from GPT-2 [Vig 2019].

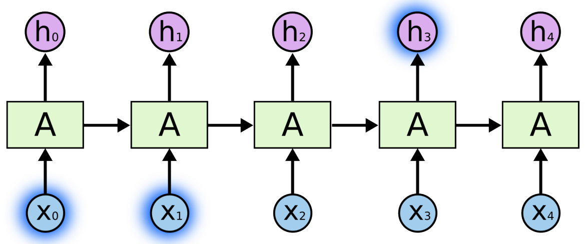

Figure 3: Example RNN layer where each x_i represents a sequential input token and h_i a sequential output token.[Olah 2015].

3 Multi-head Attention Definition

For completeness, a brief mathematical definition of multi-head attention will be given below. For a more formal explanation of the concept, please see [Phuong and Hutter 2022] which goes more in-depth than the original "Attention Is All You Need" paper.

Before starting multi-head attention, the input to the model needs to be converted into matrix form. For example, if we are attending to text, each word in the sentence could be converted to an embedding vector of dimension d. For the entire sentence of length s, this would yield a matrix $X \in R^{s \times d}$ (converting each word to an embedding vector is done outside of the multi-head attention component and is beyond the scope of this explanation).

Now that the sentence is in matrix form, the first step in multi-head attention is a series of linear transformations on the input X to create three separate new query, key, and values matrices:

$$Q = W^Q X, \quad K = W^K X, \quad V = W^V X \tag{1}$$

where $Q, K, V \in R^{s \times d}$ and the learned weight matrices $W^Q, W^Q, W^V \in R^{d \times d}$. Next attention is calculated using the formula:

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d}})V \tag{2}$$

Breaking down the attention equation, first the dot product of the query and the transpose of the key is taken to give a square matrix of shape $s \times s$. This dot product could be thought of as representing the relationship between each token in the original input matrix X (each token would represent one word in our hypothetical sentence of text input). Then these relationships are scaled by square-root of the token embedding dimension d and run through a softmax. The softmax takes the scaled relationships between each input token and assigns them to probability distribution such that:

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{s} e^{x_j}} \quad for\ i = 1, 2, \ldots, s \tag{3}$$

Lastly, after the softmax, the result is scaled by the value matrix to get the final attention calculation for a single head $H \in R^{s \times d}$. In this formulation, using the entire embedding dimension d, there will only be one attention head. With multi-head attention though, users can segment the original input matrix $X \in R^{s \times d}$ into multiple input matrices:

$$X_1,\ X_2,\ \ldots,\ X_i \in R^{s \times d_j} \text{ where } d = \sum_{j=1}^{i} d_j \tag{4}$$

These new input matrices can then be run through the same attention calculation to give:

$$Q_i = W^{Q_i} X_i, \quad K_i = W^{K_i} X_i, \quad V_i = W^{V_i} X_i \tag{5}$$

$$H_i = Attention(Q_i, K_i, V_i) \tag{6}$$

And then finally joined back together through concatenation:

$$MultiHeadAttention(X) = concat(H_1, ..., H_i) W^o \tag{7}$$

where $MultiHeadAttention(X) \in R^{s \times d}$ and the learned output weights $W^o \in R^{d \times d}$.

Ultimately, the result of multi-head attention is just a matrix of the same shape as the original input X. Because of this, multi-head attention could be thought of as a transformation step that needs to be done before eventual further processing is completed to generate some output. Figure 4. shows the original Transformer that was proposed for translation tasks and how multi-head attention is combined with traditional fully connected layers and an encoder-decoder architecture to generate results. One disclaimer about multi-head attention, is that it is not fundamentally a sequence learning architecture, but rather treats the inputs as a set. In order for multi-head attention to operate on sequences, pre-processing steps such as positional encoding are required, which more details can be found about in [Phuong and Hutter 2022] [Vaswani et al. 2017].

4 Related Work

There are two main subsections of related work for the proposed interactive multi-head attention visualization tool. First, there are other interactive visualization tools that have been developed for more traditional neural network architectures such as convolutional neural networks (CNNs) and fully connected neural networks. Two examples of these systems are the TensorFlow Playground [Smilkov et al. 2017] and the TensorSpace Playground [TensorSpace Team 2023]. The TensorFlow Playground provides

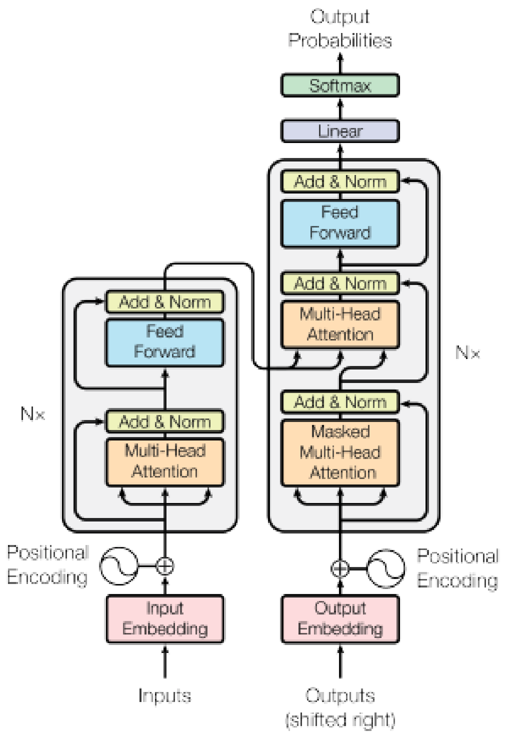

Figure 4: Transformer architecture [Vaswani et al. 2017].

an 2-dimensional network graph of a fully connected neural network where users can change hyperparameters and watch the model train based on these new settings. The TensorSpace Playground provides a 3-dimensional visualization of pre-trained CNNs that users can pass inputs into and watch how different layers impact the final result.

The other direction for related work is researchers who have attempted to use multi-head attention visualizations as a tool for explaining Transformer neural networks. Clark et al. inspected the popular Transformer architecture BERT that is used for natural language processing and claim to find attention heads that attend to specific linguistic concepts [Clark et al. 2019]. Then Jesse Vig introduced a visual analytic system that users can interact with to view specific attention heads at specific layers in a Transformer (see Figure 2 for an example) [Vig 2019].

5 Design

Based on discussion with peers and instructors in the Artificial Intelligence department at Oregon State University, one of the biggest blockers preventing adoption of visualization tools for working with neural networks seems to be ease of use. In order to encourage more frequent use, a tool should be easily built into traditional machine learning development workflows and relatively lightweight. Because of this, the proposed mutlti-head attention visualization tool was developed as a Python library that can be wrapped around a Pytorch Transformer to generate the visualization at either training or testing time. Pytorch is a Python machine learning framework that is widely used in research and industry for the development of neural networks [Paszke et al. 2019]. Furthermore, within the Pytorch library, there is a predefined Transformer neural network module and multi-head attention module that developers can use as building blocks for more complicated architectures. The multi-head attention visualization tool is specifically designed to interact with these predefined modules. Users can simply wrap the module they wish to visualize and then pass a flag on the forward pass if they would like to intercept the input and generate the interactive graph, otherwise the Transformer neural network or multi-head attention block will operate exactly as normal. Below is some example code showing how this would work:

```
import torch.nn
import viz_tool

// Fetch data
data = fetch_data()

// Define default Pytorch Transformer
transformer = nn.Transformer()

// No visualization created
output = transformer.forward(data)

// Wrap default Pytorch Transformer
transformer = viz_tool.Wrapper(transformer)

// No visualization created
output = transformer.forward(data)

// Visualization created
output = transformer.forward(data, viz=True)
```

Within the tool two main outside libraries are used besides Pytorch: NetworkX [Hagberg et al.] to create the initial 3D graph layouts and then Matplotlib [Hunter 2007] to draw the 3D graphs and allow for user interaction. Users are able to zoom in and out of the graph, rotate the graph by clicking and dragging, and select specific nodes and edges on the graph for further inspection.

For the design of the actual graph, we will refer back to several concepts from the previously given definition of multi-head attention and the Transformer architecture graph from Figure 4. The first step is to think of the square matrix of probabilities given by the softmax from Equation 2 as a fully connected graph representing the relationships between each token in the input sequence. Here is where the first step has to be taken to abstract some inner workings from the user, the square matrix actually represents two fully connected graphs (one representing the forward relationship between each set of tokens and one representing the backwards relationship between each set of tokens). Rather than trying to visualize two fully connected graphs at each layer, the decision was made to simply sum the probabilities in order to create one fully connected graph that represents both the relationship in the forward and backwards direction. Next, this fully connected graph was drawn for each multi-head attention layer that was included in the Transformer (typically values for the number of layers can range from 6 all the way to 24 and up).

Then each layer's fully connected graph can be connected by applying the dot product from the value matrix in Equation 2 to the square softmax matrix represented by the fully connected graph. This step transforms the multi-headed attention back to the shape of the original input matrix and provides the starting point for the next layer in the graph. It should be noted, once again there is a small amount of information hidden from the user here. Referring to Figure 4 again, there is a fully connected layer after the multi-head attention that expands the input dimension to some larger embedding dimension d_{max} and then applies another linear transformation to shrink the embedding dimension back to d. This step could be easily visualized as a expansion and contraction of the graph, but it greatly clutters the visualization while providing little new value on the relationships between the tokens. Because of these reason, it has been left out of the visualization for now.

Finally, to visualize the multi-head attention weights in the graph, edge opacity was used. All of the weights first had their absolute value taken and then normalized between [0, 1]. Using this approach, the edge should gradually fade or disappear if two tokens are loosely related and appear normally if the relationship is strong. Alternatives to this approach could leverage distance between nodes and introduce edge colors.

6 Results

The code to run the visualization will be included in a zipped file along with this paper. Currently the code is sitting within a Jupyter Notebook to allow for easier exploration and tinkering, but could be easily ported into Python library when it is ready for release. One note on running the code, it should be run using JupyterLab to allow for the visualization to be interactive.

In order to test the code, I trained a 6-layer Transformer encoder to perform sentiment analysis on Twitter tweets directed at major airlines. The Transformer encoder can be seen on the left side of the architecture in Figure 4 and is a sub-component of the Transformer architecture that is typically used for classification tasks, whereas the decoder and entire encoder-decoder Transformers are typically used for generative tasks. The goal for using a simple Transformer encoder was to create a toy example that was easy to understand for earlier iterations of the visualization tool. On top of the multi-head attention visualization tool already proposed, a visualization was added for the final fully-connected classification layer that exists for encoder only Transformers. The goal here was to make it easier to see how the multi-head attention was contributing to the final output from the model. This is a custom visualization step though and the proposed library would not be able to automatically generate this step unless it knew the model was an encoder only Transformer performing classification.

A major critique of the results is the differences in edge opacity, while visible, do not convey enough information about the relationships. In future iterations of the tool, different normalization techniques, colors, and edge lengths should all be experimented with as alternatives. Another critique of the current mechanism is it only allows for the visualization of one headed attention. It should be trivial to extend the existing code to multi-head attention, but before this is done the opacity issue needs to be fixed to make the inclusion of more information in the visualization meaningful for users.

7 Future Work

The first step in future work should be attempting to fix the critiques discussed the results section. Once a mechanism for visualizing edge weights has been decided upon, some user studies with the model should be conducted to evaluate it's usability. One idea would be presenting several researchers with a Pytorch model that has a clear bug and seeing if they could debug the issue using the visualization.

Another direction for future work, could be investigating ways to prune the graph or perform dimensionality reduction so that modern Transformers which can contain billions of parameters can be visualized using the tool. If the tool was used as is with a modern Transformer architecture (for example a large language model like Chat-GPT) it would likely crash or simply present the user with an overwhelming amount of data.

One final research direction would be visualizing the topological properties of the graph at both training and testing time. Formation of cliques and cavities within traditional feed-forward neural networks has been shown to fluctuate during the training of neural networks [Corneanu et al. 2019] and visualizing this information during training time could provide users with an intuition about when their model is beginning to overfit the data. Furthermore, examining this information during testing time could help debug incorrect model performance. One outstanding problem in this research direction, is previous work on the topological characteristics of neural networks has focused on simpler fully connected networks. It is not immediately clear how those previous approaches could be leveraged on Transformer architectures, particularly when multi-headed attention could be thought of as a weighted clique.

References

ARNAB, A., DEHGHANI, M., HEIGOLD, G., SUN, C., LUCIC, M., AND SCHMID, C. 2021. Vivit: A video vision transformer. *CoRR abs/2103.15691*.

CLARK, K., KHANDELWAL, U., LEVY, O., AND MANNING, C. D., 2019. What does bert look at? an analysis of bert's attention.

CORNEANU, C. A., MADADI, M., ESCALERA, S., AND MARTINEZ, A. M. 2019. What does it mean to learn in deep networks? and, how does one detect adversarial attacks? In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4752–4761.

CORNEANU, C. A., MADADI, M., ESCALERA, S., AND MARTÍNEZ, A. M. 2020. Computing the testing error without a testing set. *CoRR abs/2005.00450*.

DOSOVITSKIY, A., BEYER, L., KOLESNIKOV, A., WEISSENBORN, D., ZHAI, X., UNTERTHINER, T., DEHGHANI, M., MINDERER, M., HEIGOLD, G., GELLY, S., USZKOREIT, J., AND HOULSBY, N. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR abs/2010.11929*.

HAGBERG, A., SWART, P., AND S CHULT, D. Exploring network structure, dynamics, and function using networkx.

HUNTER, J. D. 2007. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering 9*, 3, 90–95.

OLAH, C., 2015. Understanding lstm networks.

PASZKE, A., GROSS, S., MASSA, F., LERER, A., BRADBURY, J., CHANAN, G., KILLEEN, T., LIN, Z., GIMELSHEIN, N., ANTIGA, L., DESMAISON, A., KÖPF, A., YANG, E., DEVITO, Z., RAISON, M., TEJANI, A., CHILAMKURTHY, S., STEINER, B., FANG, L., BAI, J., AND CHINTALA, S., 2019. Pytorch: An imperative style, high-performance deep learning library.

PHUONG, M., AND HUTTER, M., 2022. Formal algorithms for transformers.

SMILKOV, D., CARTER, S., SCULLEY, D., VIÉGAS, F. B., AND WATTENBERG, M. 2017. Direct-manipulation visualization of deep networks. *CoRR abs/1708.03788*.

TENNEY, I., DAS, D., AND PAVLICK, E. 2019. BERT rediscovers the classical NLP pipeline. *CoRR abs/1905.05950*.

TENSORSPACE TEAM, 2023. Tensorspace.js. [Online; accessed 12-April-2023].

VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, L. U., AND POLOSUKHIN, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, Curran Associates, Inc., I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30.

Vig, J. 2019. A multiscale visualization of attention in the transformer model. *CoRR abs/1906.05714*.