Circuit engineering

Authors:

- Kyrylo Shyvam Kumar (2021101080)
- Bhargav Srinivas (2021101065)
- Vansh Garg (2021111006)

Problem statement:

- We train transformers for simple algorithmic tasks (modular addition of 2 numbers, maximum, median and sorting of 5 numbers) and try to interpret them. We train models long after overfitting, causing them to grok - generalize long after overfitting. We are able to replicate results of completely re-engineering modular addition, and are able to extend some of this analysis to max operation. We also explore median and sorting operations in this work.
- Our dataset consists of 5,000 pairs of numbers and their modular sum as output. Dataset is manually constructed. For MAX/MEDIAN/SORT we use sets of 5 numbers.
- We use single layer decoder style transformer for all our experiments.

Related works

Grokking.

Grokking as first reported in [1], which trained 2L transformer, on several simple algorithmic tasks, and found that despite fast overfitting, validation accuracies often increase sharply long after achieving perfect train accuracies.

Interpreting grokking.

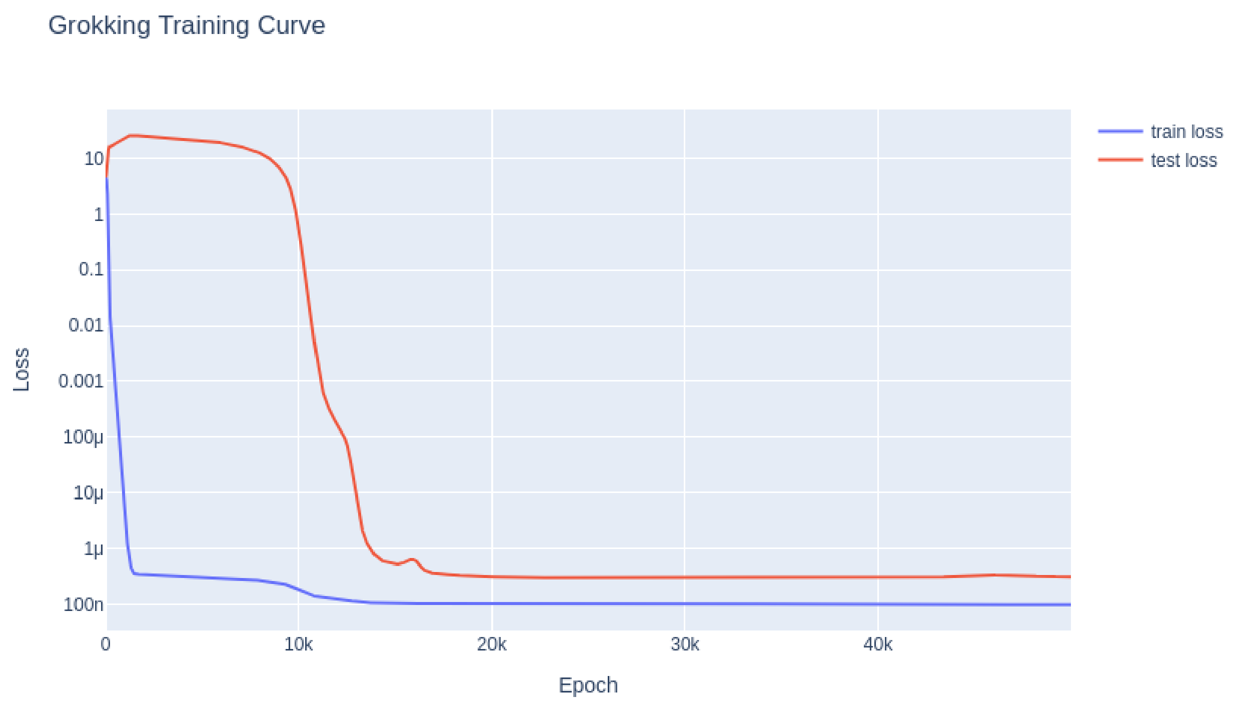
For a long time it was considered by works like [6] that grokking works because "SGD being a random walk on the optimal manifold". However later works show that there is continuous progress into achieving this sharp decline of validation loss. [2] forms the basis of our work, and successfully reverse-engineers the algorithm a grokked transformer has learnt. They also produce the hypothesis on grokking, calling grokking a continuous process , where generalisation arises due to regularisation removing circuits related to memorisation, but keeping circuits for general algorithm.

Interpretation of grokked non-transfomer models.

[2] uses 1L grokked transformer, and argues that multi headed attention can be easily interpreted. However, works like [4], perform similar analysis on MLP with single hidden layer.

Interpreting addition

- **Losses**



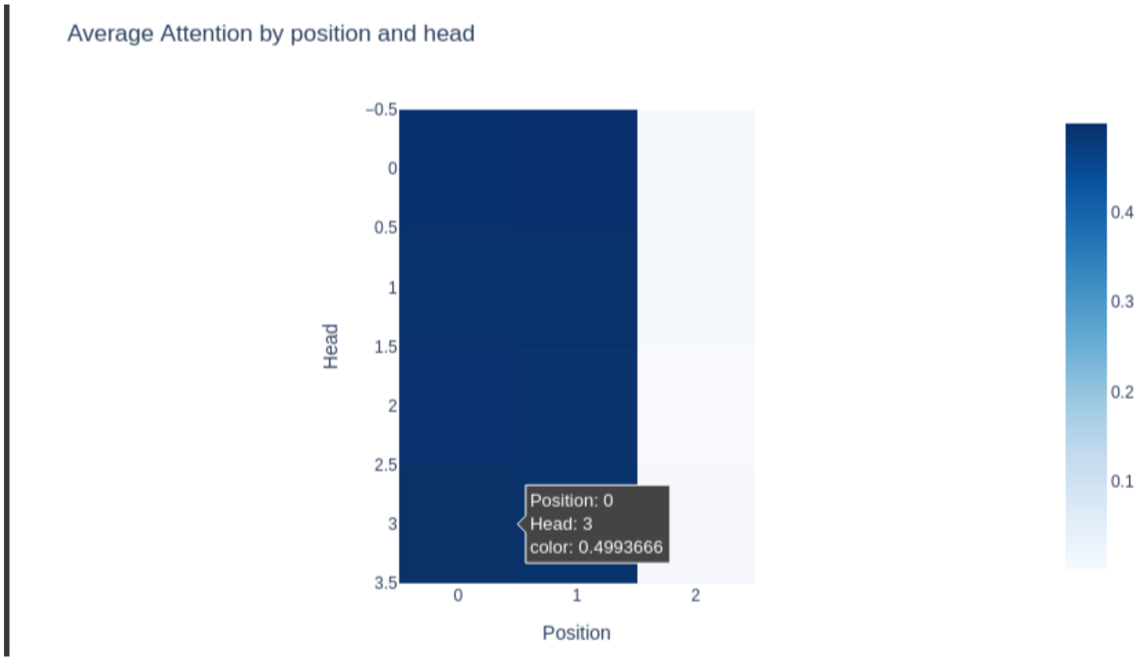
Grokking begins here around epoch 10k and ends around 15k.

- Closeness of positional encodings: The cosine similarity shows some but not complete closeness in positional encodings. Any possible difference is eliminated at the MLP (after attention block). So final prediction is position invariant to inputs.

```
Positions 0 and 1 are symmetric
Difference in position embeddings 0.7655194401741028
Cosine sim of position embeddings 0.6173164248466492
Difference in neuron activations for (x,y) and (y,x) 9.627481631468982e-06
```

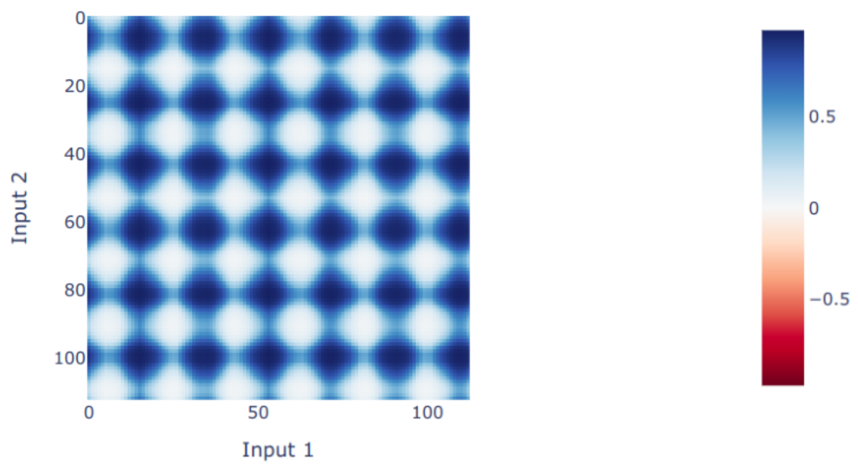
- How much attention does "=" pay to position 0 ("x") for each head?

We also show the CLS token pays equal attention to both of the positions (attention ~ 0.5).



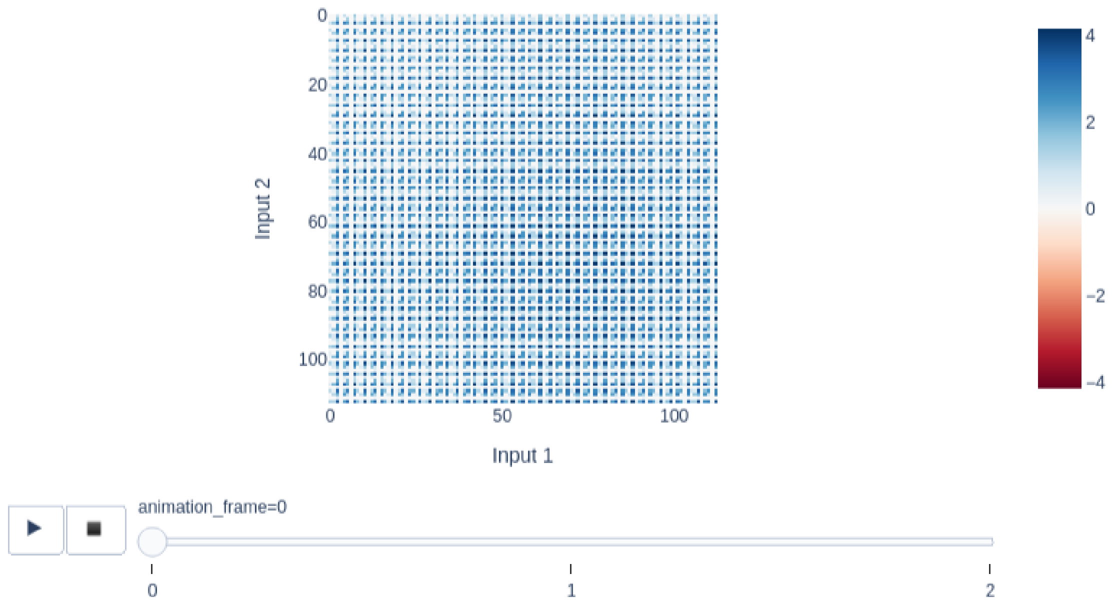
- Attention score for heads at position 0: We observe periodic patterns wrt to inputs.

Attention score for heads at position 0



- Activations of neurons just after up-project in the FFN layer.

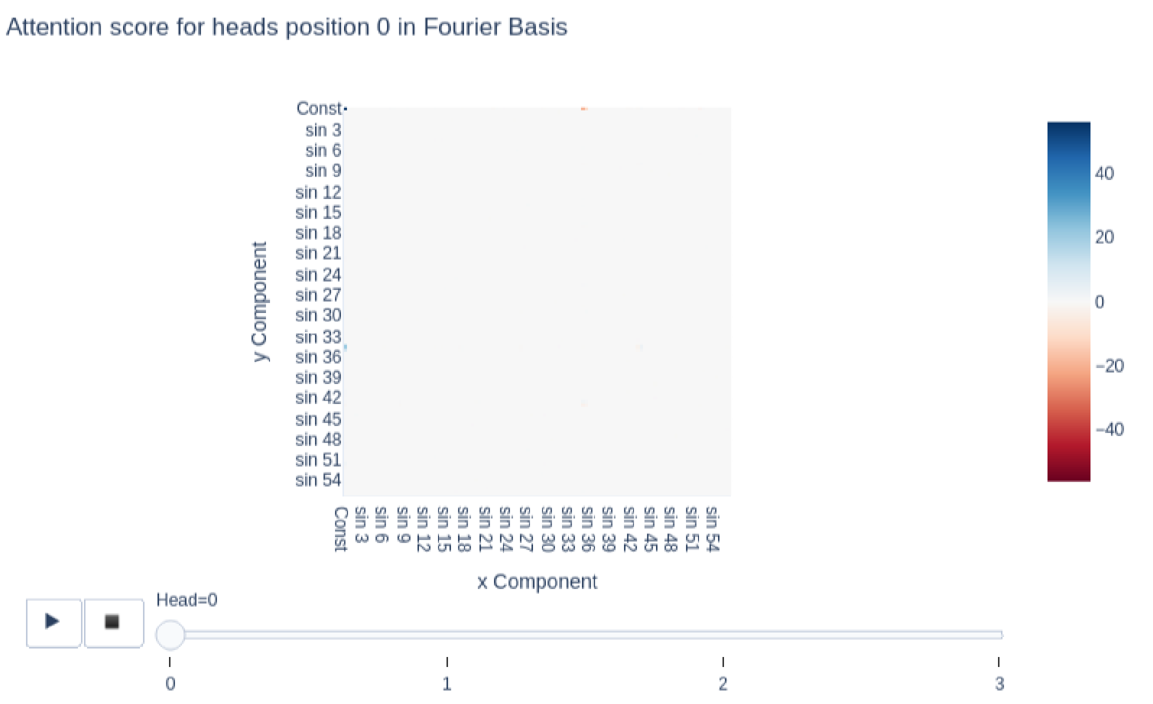
Activations for first 3 neurons

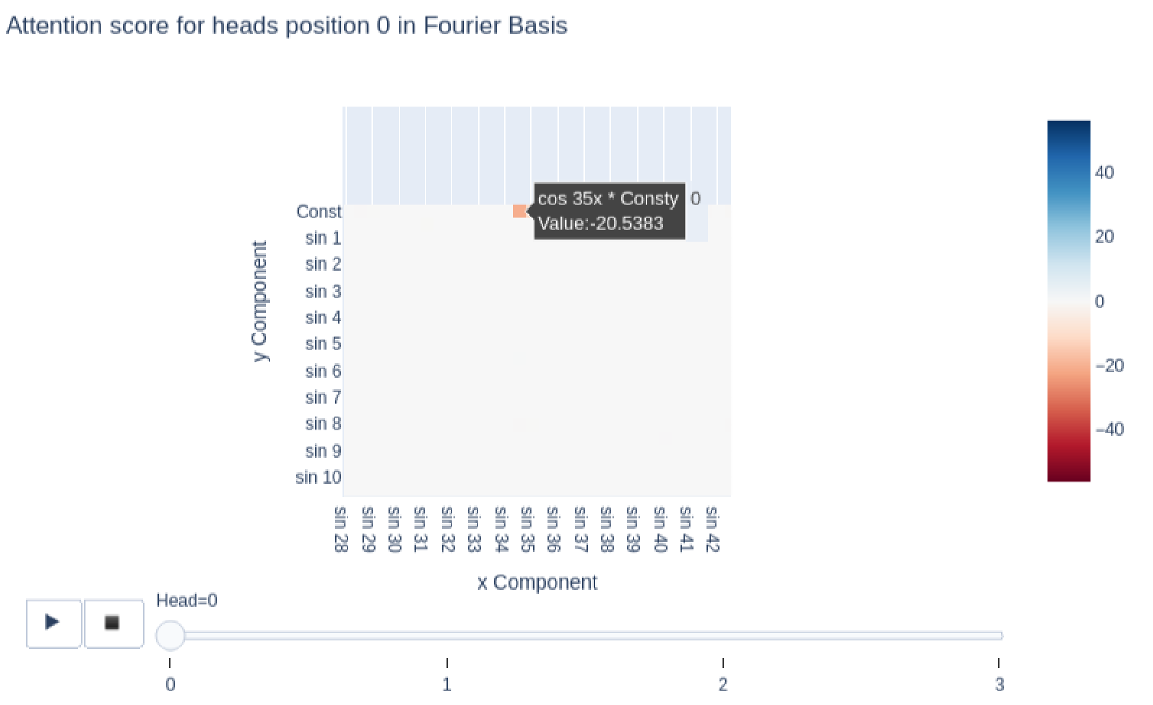


Periodicity in activations implies that there are few important frequencies, and we can perform Fourier decomposition to get them. Also activations will be sparse in Fourier basis.

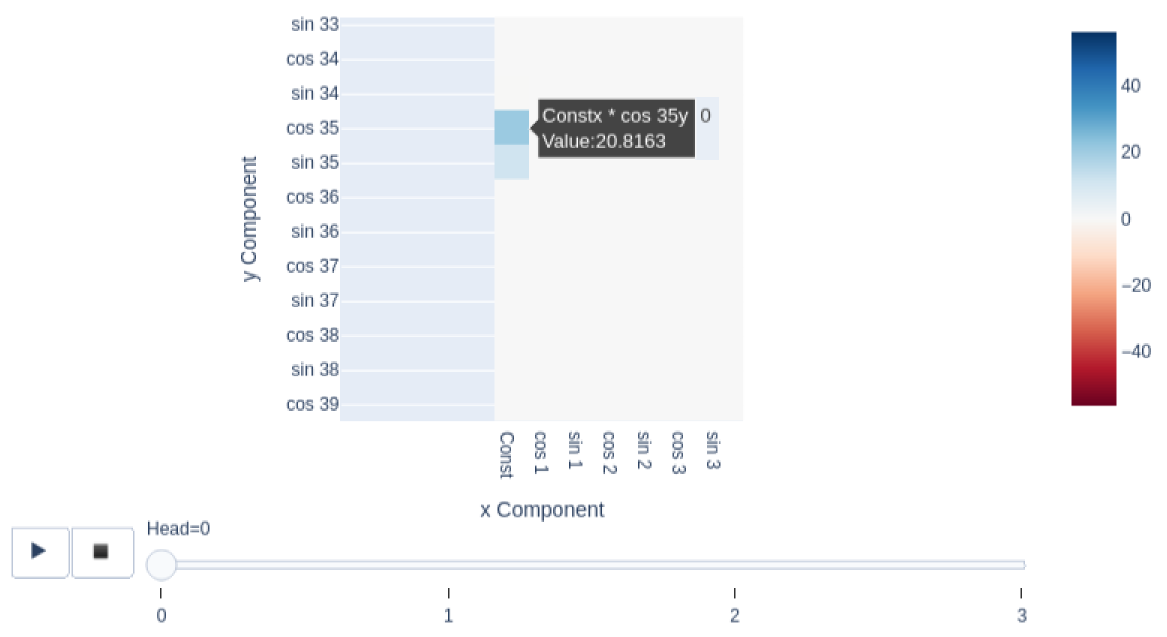
- Attention score for heads position 0 in Fourier Basis:

We do 2D Fouried decomposition (since we have 2 inputs) and we find frequncies below. Note, that it is symmetric for (x,y)





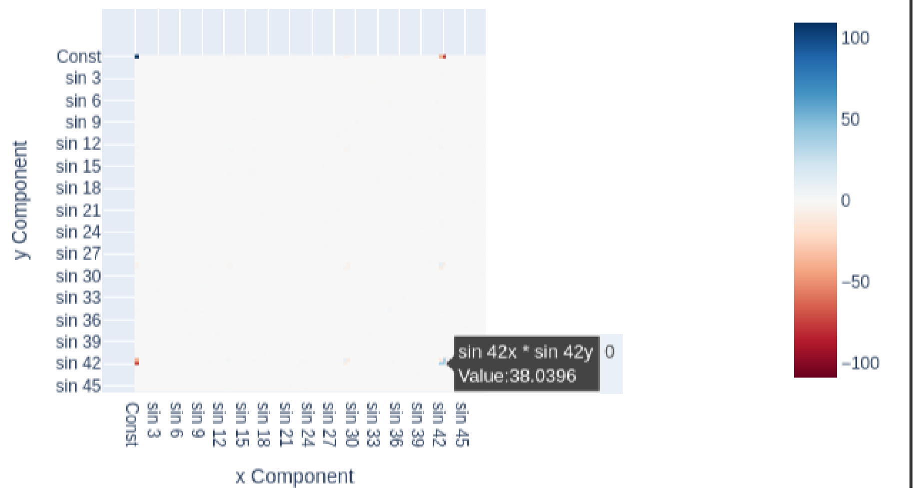
Attention score for heads position 0 in Fourier Basis



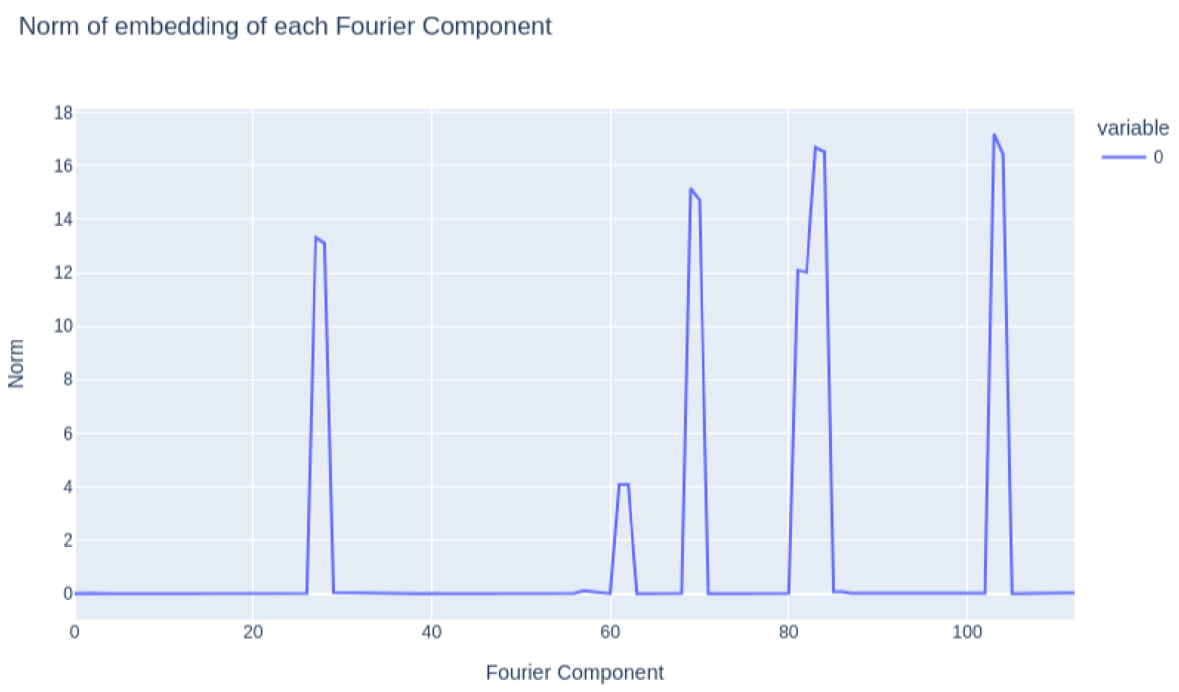
- Activations for first {top_k} neurons in 2D Fourier Basis:

Similarly it follows a different frequency at activations of MLP.

Activations for first 3 neurons in 2D Fourier Basis



- W_E in fourier basis: Embeddings can be projected to Fourier basis, showing that there only 5 important frequencies learnt.



Problems in achieving grokking with MAX/Median/SORT

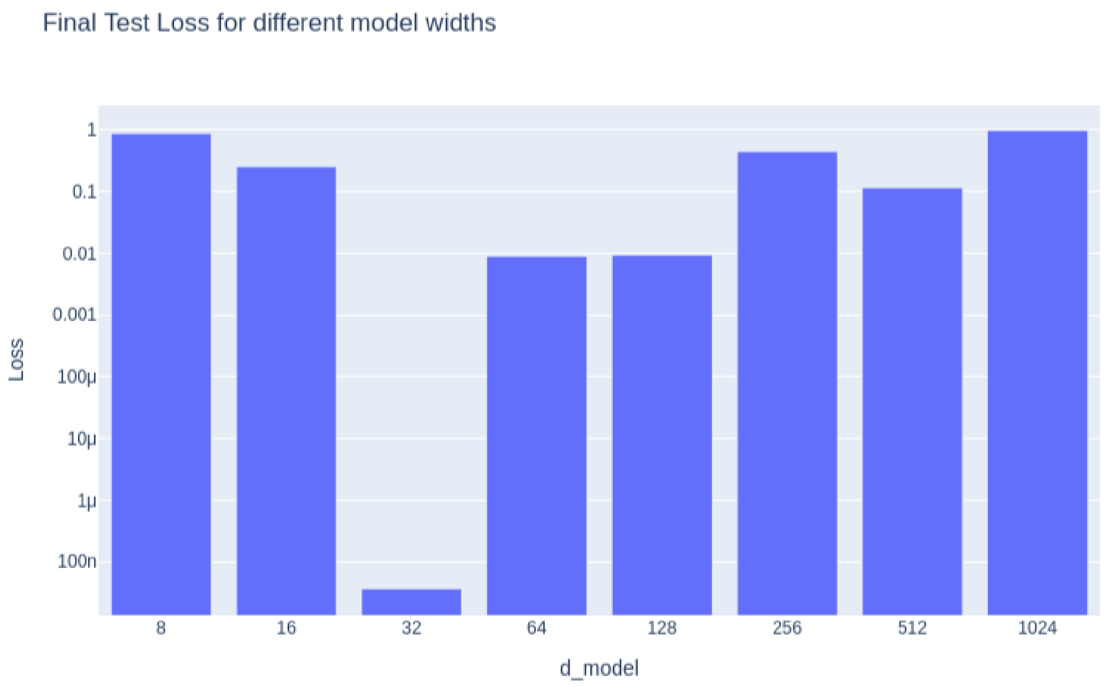
- Explain tasks: We decided to extend interpretation to other tasks. Since transformers can be grokked for any algorithmic tasks, we can derive algorithm for each of them.
    - Note: In sorting task we auto-regressively generate 5 tokens.
- Size of dataset: We faced problems in grokking for both max (too easy) and sorting (too difficult).
    - Max of 5 numbers involves output consisting of any 5 inputs. So model needs only to learn copying of information properly.
    - Sorting is a complicated task. Despite it requiring copying of information, the algorithm itself is non-trivial.
    - So, the max operation directly starts generalizing without overfitting first required for grokking. The sorting operation never generalizes properly. We mitigated this by increasing train set by a magnitude. We argue this is

    reasonable as we model relationship of 5 numbers at once, which has significanly more permutations than pairs of 2.

- Distribution of dataset: Addition with modulo is a circular operation. In other words it is an operation defined over cyclic group, leading to uniform distribution of output in general.
    - Operations like max/median output biased distribution for random input. Some numbers (those bigger in magnitude) have a higher probability to occur in output. In fact, when we randomly sampled 40k tuples of 5, only 10 had max below 20. So, there existed some outputs for which no backpropagation happened.
    - We had to augment dataset with samples which had all numbers small and thus had max < 20. This increased performance by 10% from ~85 to ~95% on augmented test set.

General Observations

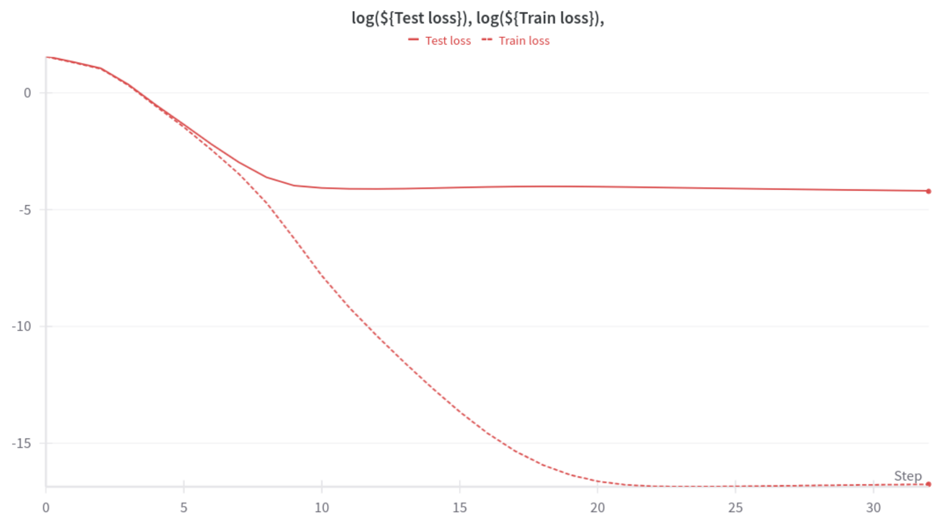
- **Sorting is better learned than median operation :** We observe 10x better loss, and +5% increase in percent points wrt same dataset of sorting vs median operations. The reason we propose is sorting actually predicts five numbers, which acts as extra training signal (or CoT) helping model to train.
- B**igger dimension size is better for tasks like median:** We observe that optimal dimension for addition is 32 (best observed loss), and the whole distribution follows bias variance tradeoff. The best loss for median task occurs at dim=512, showing that this task has higher intrinsic dimension.



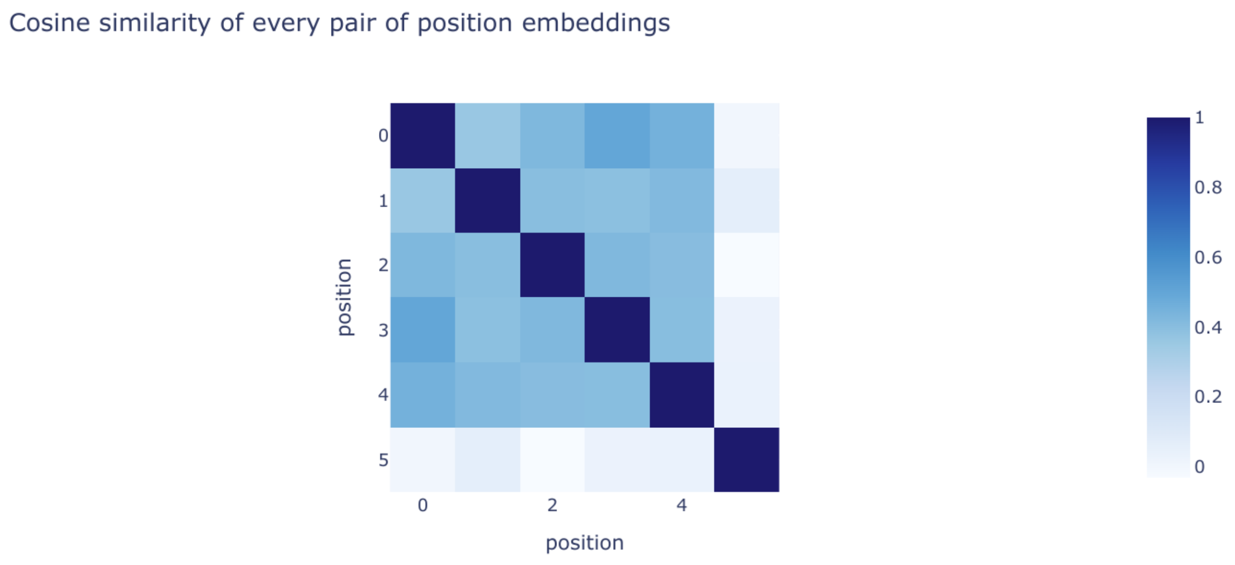
- Adding a special dataset containing the edge cases missed in the random dataset only improves performance marginally.

Max Transformer:

- **Training loss plots:**

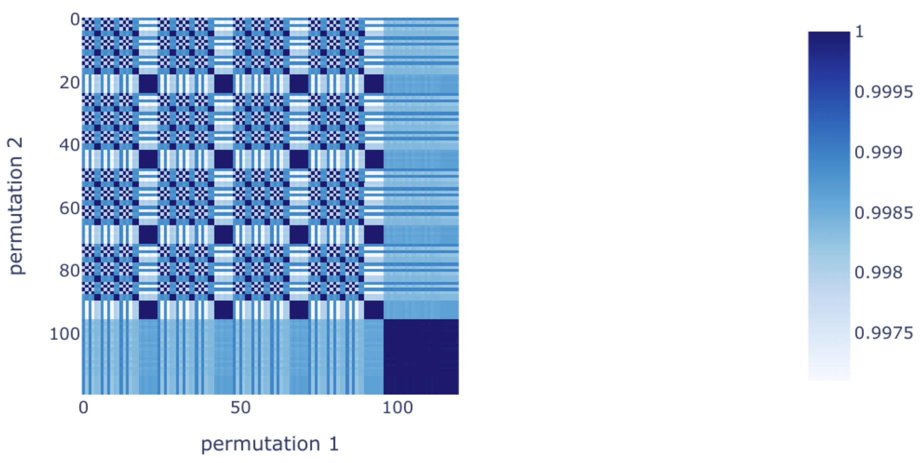


- **Position embedding comparison across permutations:** Positional embeddings are close to each other, and perpendicular to cls token. The max operation is independent of the order in which the inputs are given. We wanted to verify if the transformer learns this positional independence. This is similar to verifying the commutativity in the case of addition transformer.



- **Activations comparison across permutations:** To achieve this we picked 5 random numbers and calculated all the activations w.r.t to each one of the 5! (=120) permutations of the input.

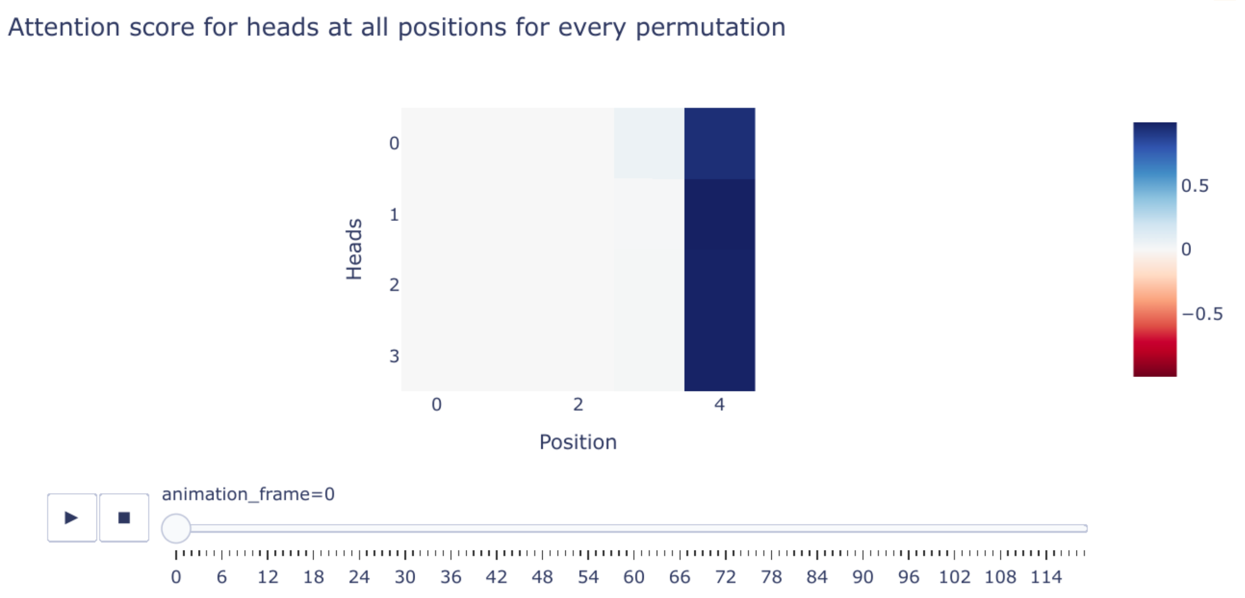
Cosine similarity of neuron activations pre-ReLU corresponding to two permuatations

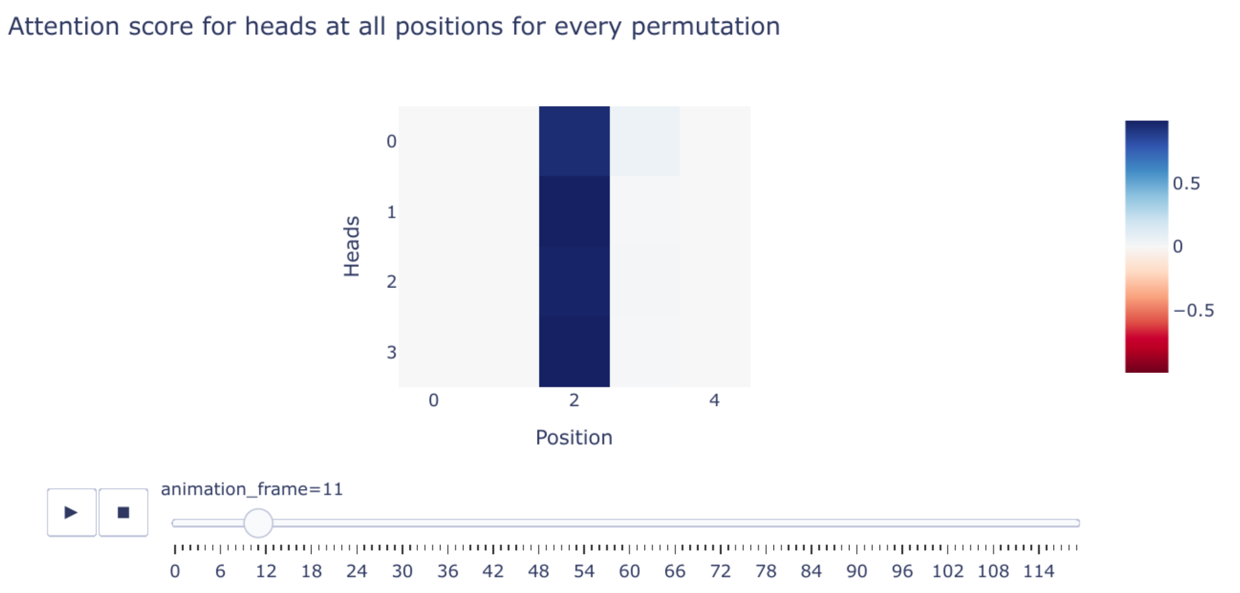


If we look at the scale of the plot, we see that almost all the values are close to 1. Which means the neuron activations are very close to each other. We see a pattern which can be explained by the periodicity of the permutations itself.

From the above experiments we conclude that the order in which the input numbers are given does not influence the output.

- **Attention paid by cls token to each of the input numbers for each head across all permutations:** All 4 attention heads pay exclusive extension to the largest element. In 1st figure it is at the end. In 2nd figure it is at position 2.
    - This leads us to speculation that attention patterns (Q-K circuit) chooses the position at which maximum element occurs.
    - And the rest of weights (OV circuit + MLP + Unembedding), convert the position of token to the actual number corresponding to it. We verify this in following experiments.

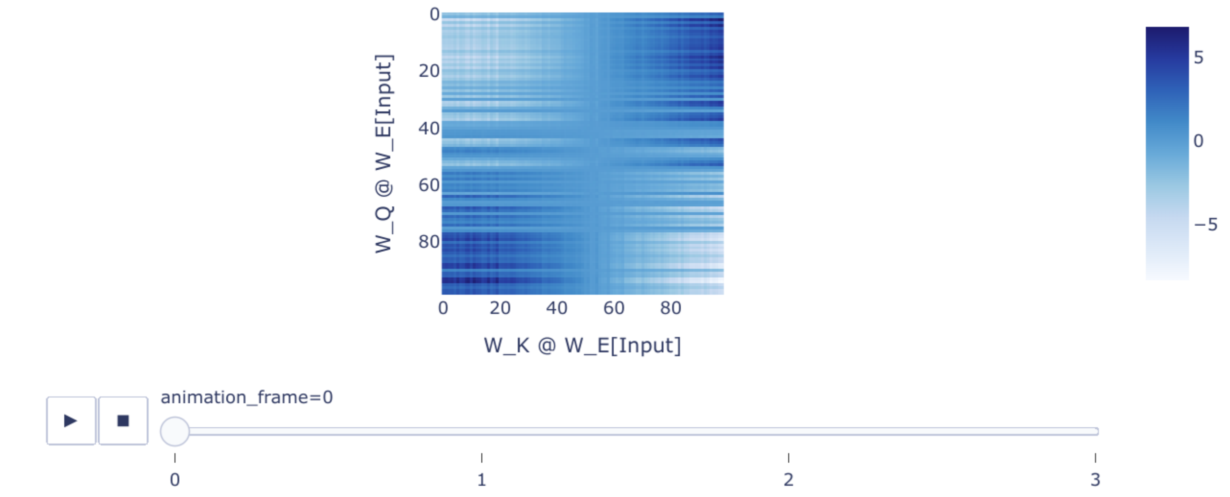




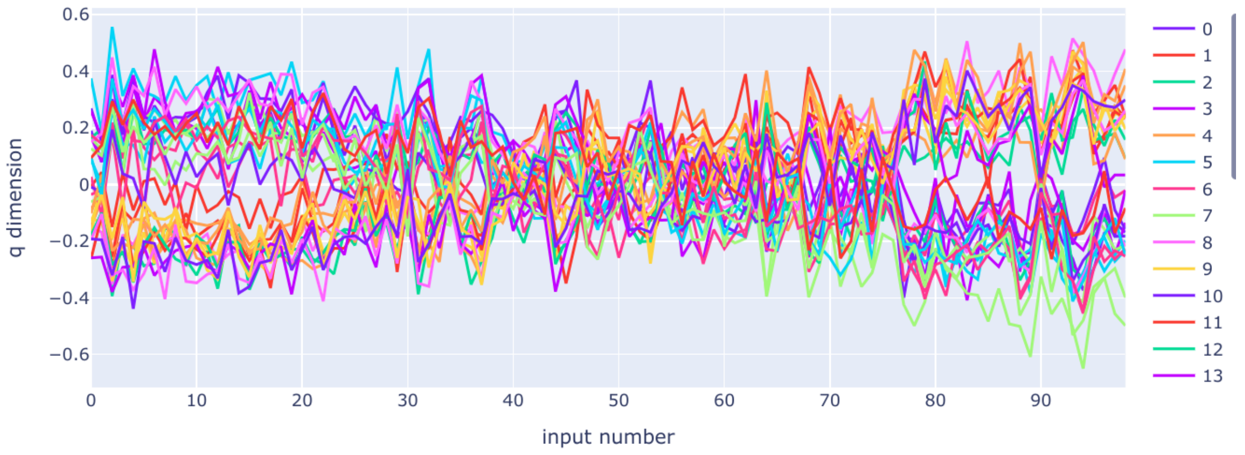
Circuit Analysis

- **Attention score for every pair of inputs:** We plot the un-normalised attention values for all possible pairs of queries and keys. We can see that elements closer to diagonal have smaller absolute difference and evaluate to smaller attention.

Attention score for query of every number with key of every number



- Here we plot all 32 dimensions of a single head, and specifically how it (value in ith dimension) varies with input. Note, that we only need CLS token as we have single layer transformer.

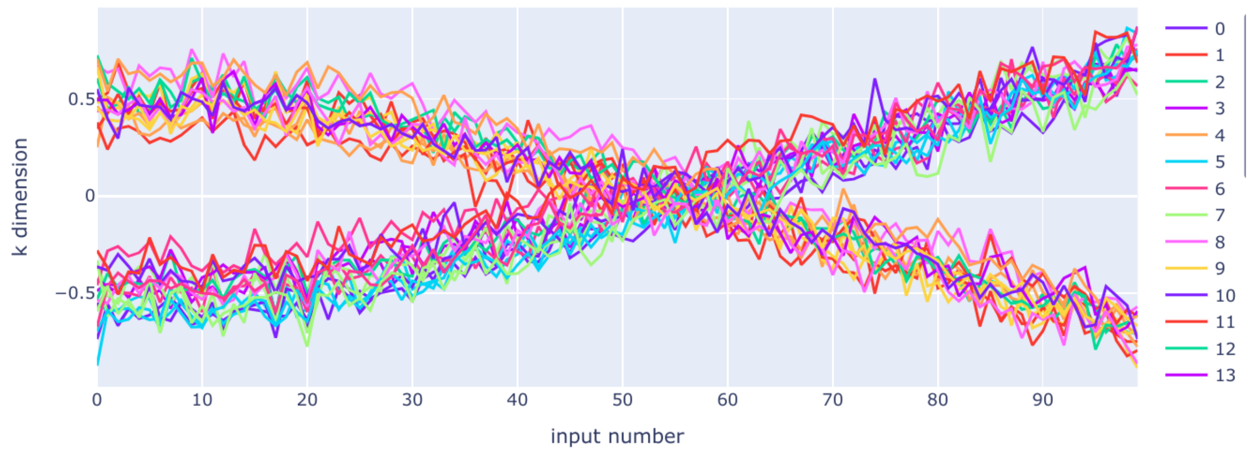


- **k vector for each input:** The graph shows same thing but for keys. We can see that all dimensions are closely correlated to each other and represent essentially a straight line. Note: numbers 0-20 are under-represented in dataset. It explains why there is no slope in that segment.

- We theorize, that the whole QK circuit can be represented in much simpler terms in the form a linear equation. The linear equation just represents the difference bw input and constant, which is than softmaxed.
- Thus we can replace each of these lines with a line : y= mx+c, and still expect them to work properly.
- Do you see the pattern?

  If you look carefully, each dimension of key vector and query vector is either roughly increasing or decreasing (handwavy) w.r.t input. Interestingly, if a dimension in query vectors is increasing, the same dimension is decreasing in the key vectors. This actually leads to interesting properties.

$$q_{ij} = m_j x_i + c_j$$
$$k_{ij} = m_j^{'} x_i + c_j^{'}$$
$$m_j^{'} m_j < 0$$



- **Linear regression and then show the coefficients:** So, we replace each neuron with equation y = mx + c, by performing linear regression over all inputs (for each neuron separately).

So, for some single neuron, y = -0.012x + 0.7 (check out keys 2nd diagram).

Now we can derive the general expression for attention.

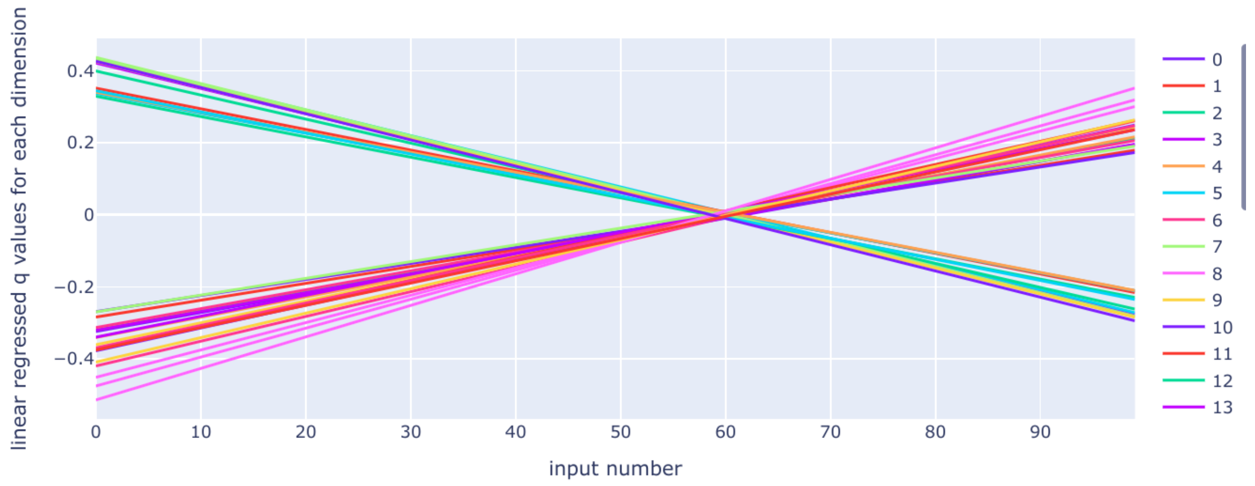
Attention (unnormalised) a = $\sum(mx_i + c) * (mx_j + c)$

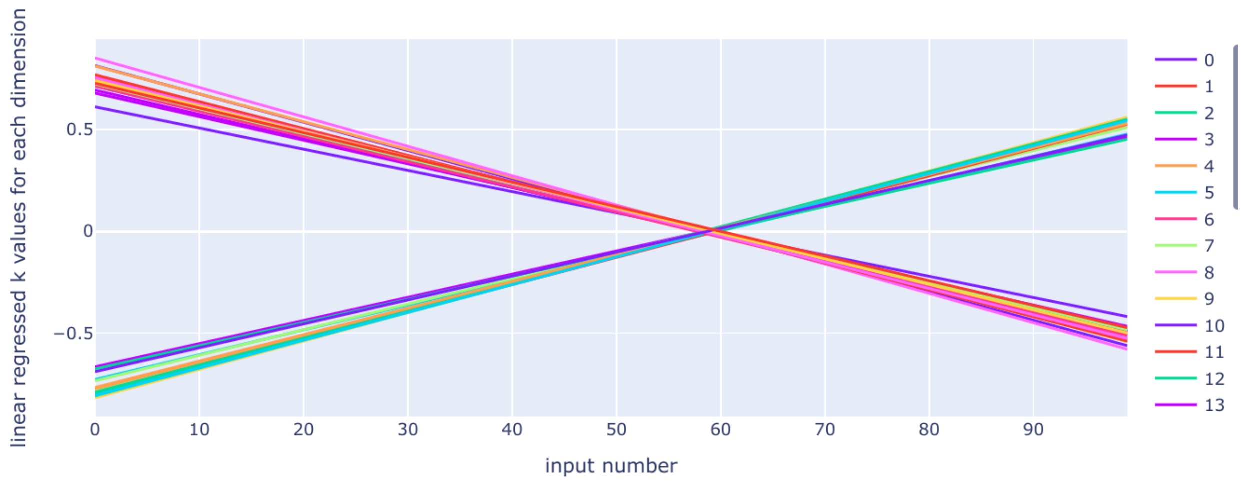
= 0.15x - 8.93 [Since we always use same query]

= 0.15(x - 59.53)

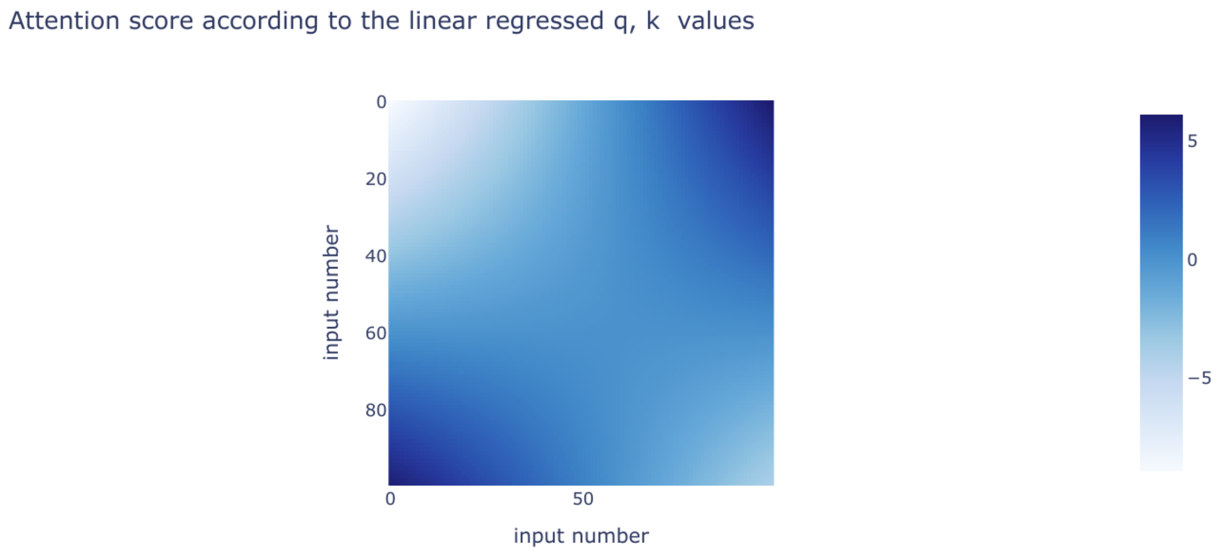
So actual attention = softmax(0.15(x - 59.53)) for some input x.

Note: 0.15 here works as temperature. 59.53~60 does not change any value of attention, however interestingly it is effective centre of our data (we have effective data from 20-100, outputs <20 are very rare).



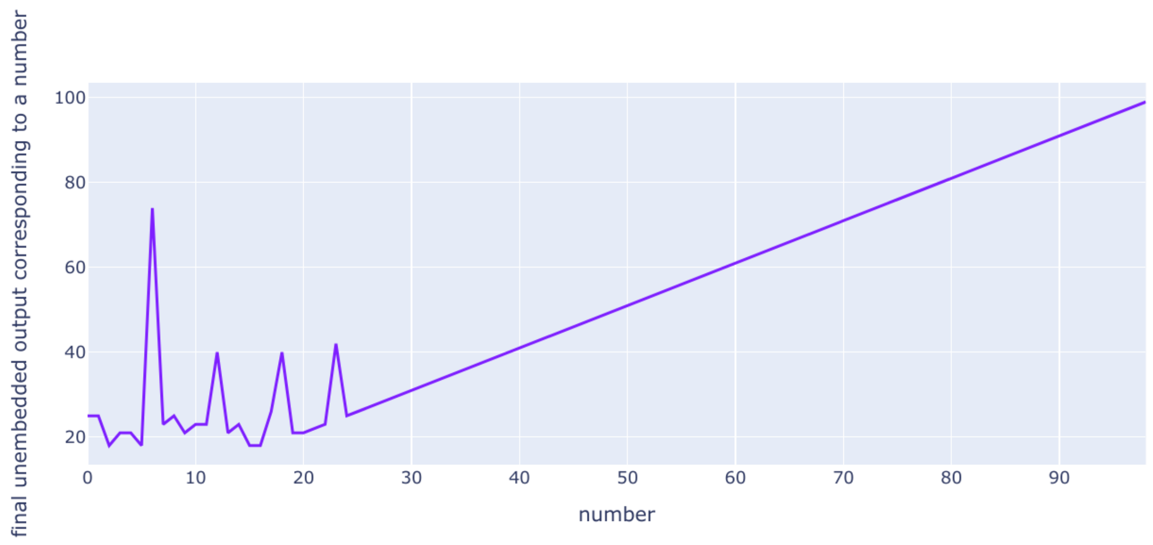


- **Un-normalized attention score for every pair of inputs according to linear regression:** The final attention scores we get from regressed values is as following. It is very similar to original plot which is less smooth. This is strong evidence that our interpretation is correct.

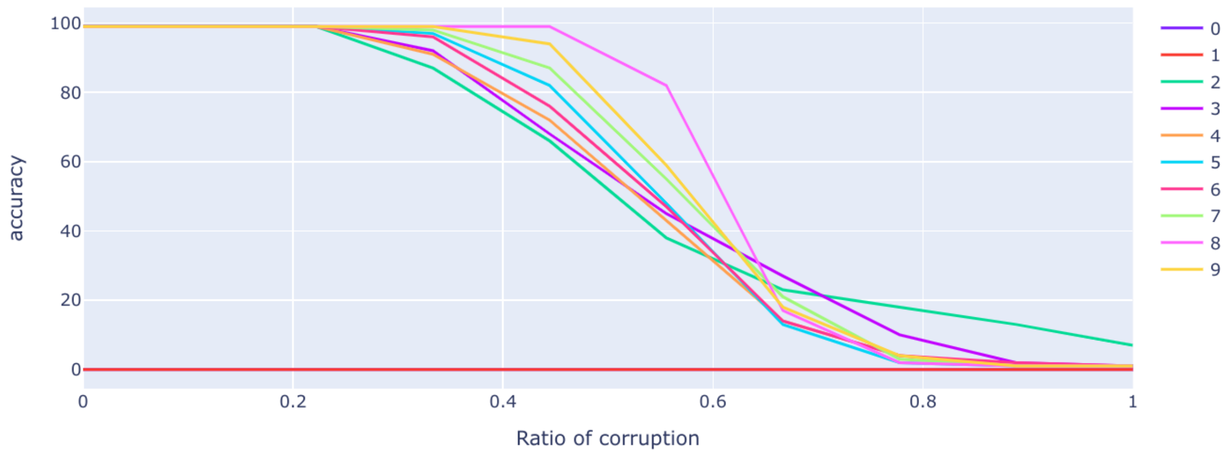


This is roughly the same as the original unnormalized attention scores.

- **o-v circuit role is copying:** Below graphs visulizes predictions by OV circuit if output of QK circuit is correct. If we ignore 0-20 range which is not well trained, we get almost perfect score while running the circuit.
    - This shows if attention mask selects correct position at which max term occurs, OV circuit will definitely predict the correct token.
    - Why is 0-20 bad here? Because there is close to zero probability of position containing number < 20 be selected by QK circuit, and thus that info never passes through OV circuit.



- **Robustness of copying:** x-axis shows extent (0% to 100%) of adding noise (other competing v-value) and its effect on accuracy of copying in OV circuit. We see that we can add 20-30% percent of other v-value before accuracy drops from 100%. Robustness of copying shows redundancy in OV-circuit.

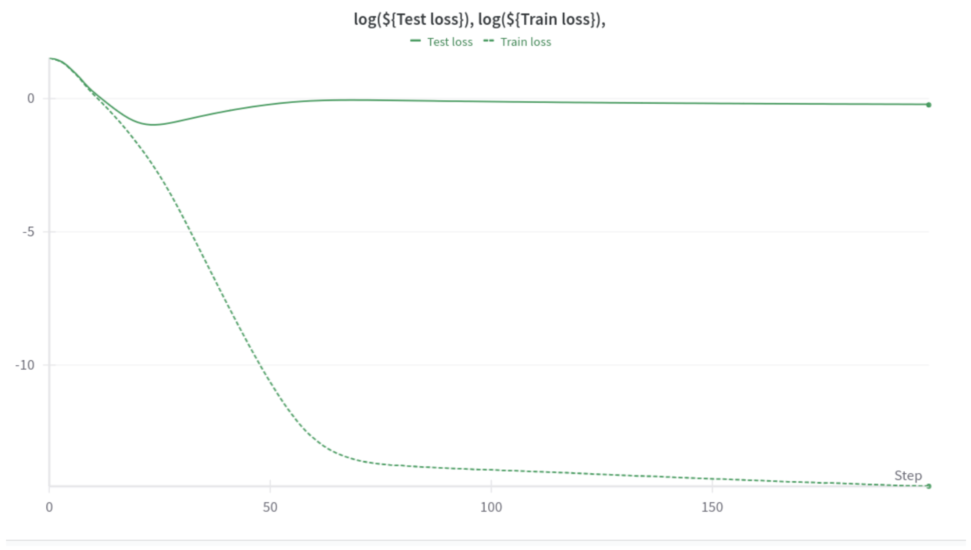


- **Final algorithm:**
  - **QK circuit (Embedding + W_query + W_key):** Convert one hot encodings into actual numbers x, to calculate y = mx + c → attention of cls token wrt all keys containing x as input. Note: We do not imply that we actually get x, only that attention scores are linear function of input.

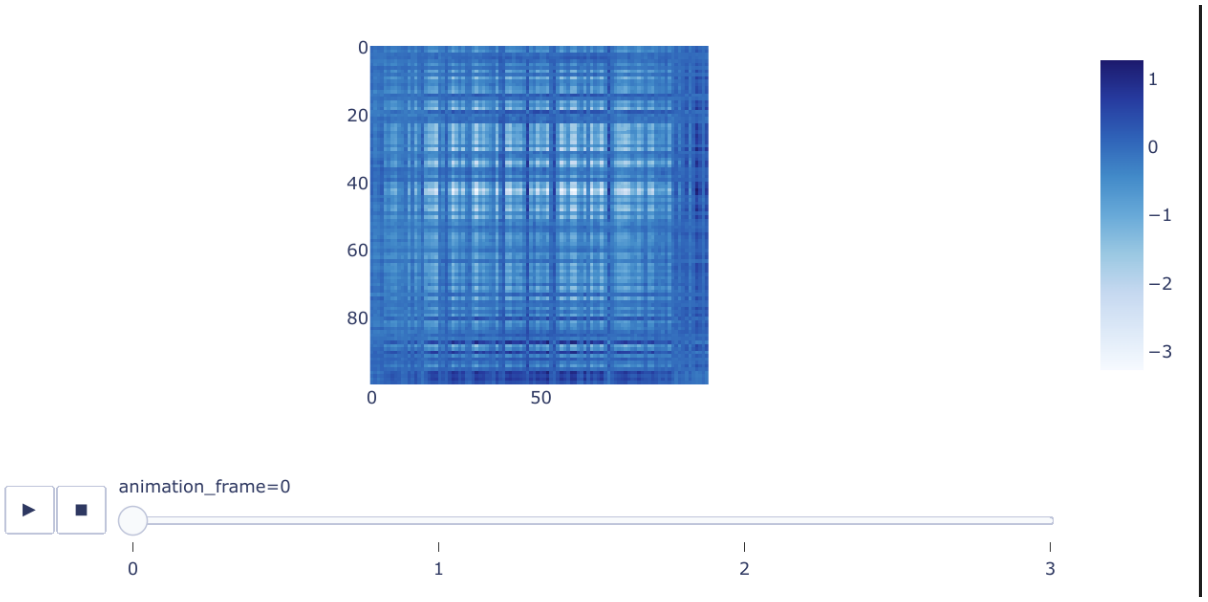
- **OV circuit (W_Value + W_O + MLP + Unembedding)**: Convert the v_value at the position to which CLS token attends to its actual value.

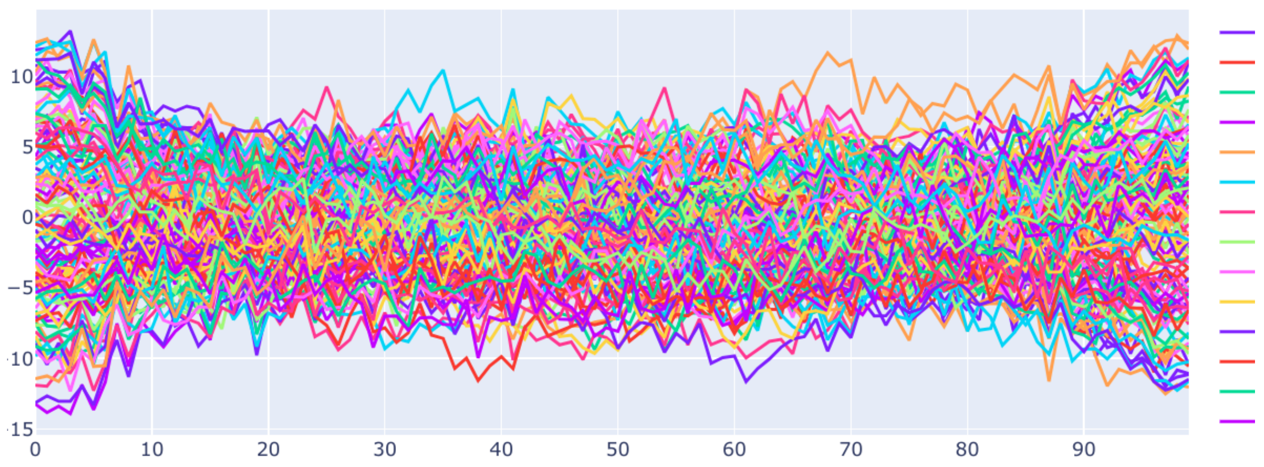
Median Transformer:

- **Loss plots:**



- Absolutely no fucking clue how to interpret

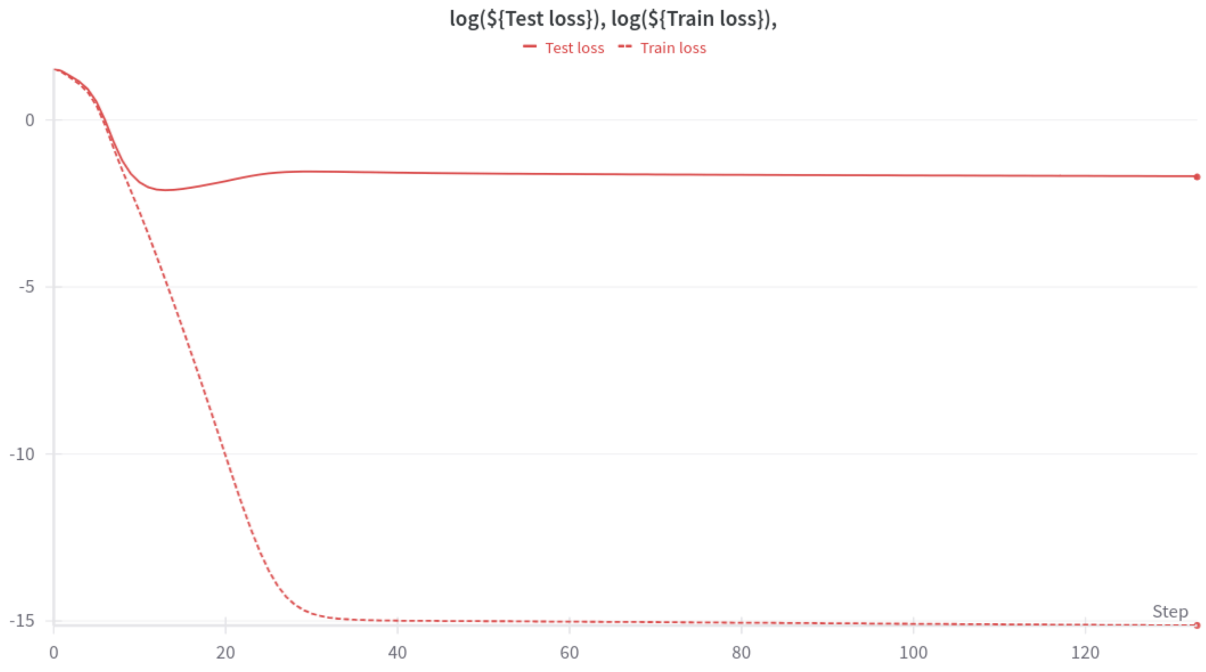




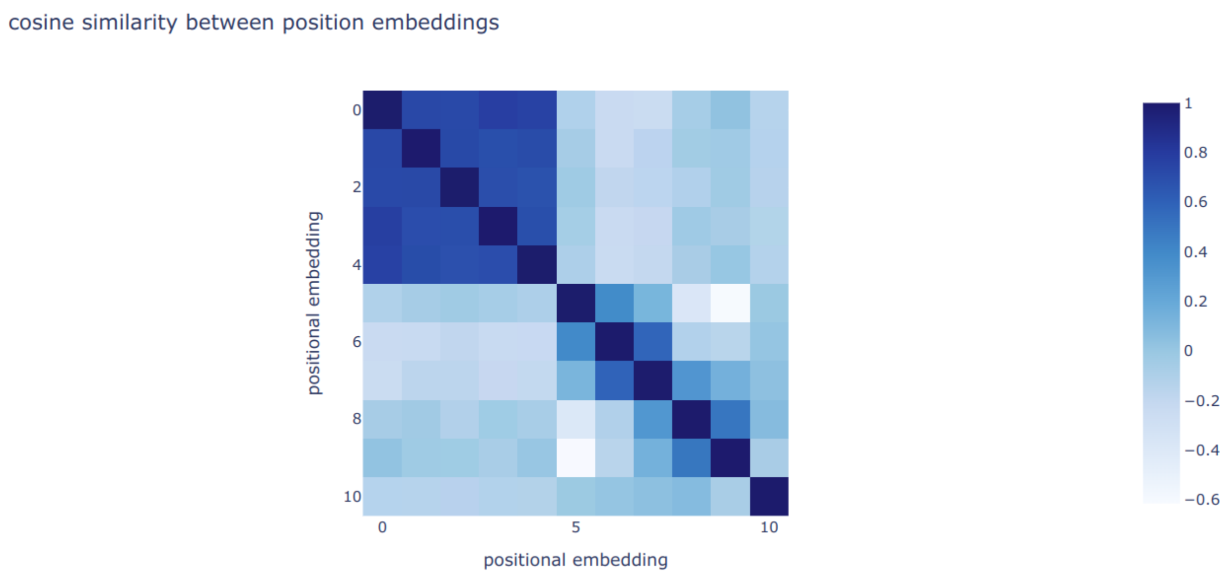
- O-V circuit still robust

SORT Transformer:

- **Train loss and acc plots:**



Position embedding analysis

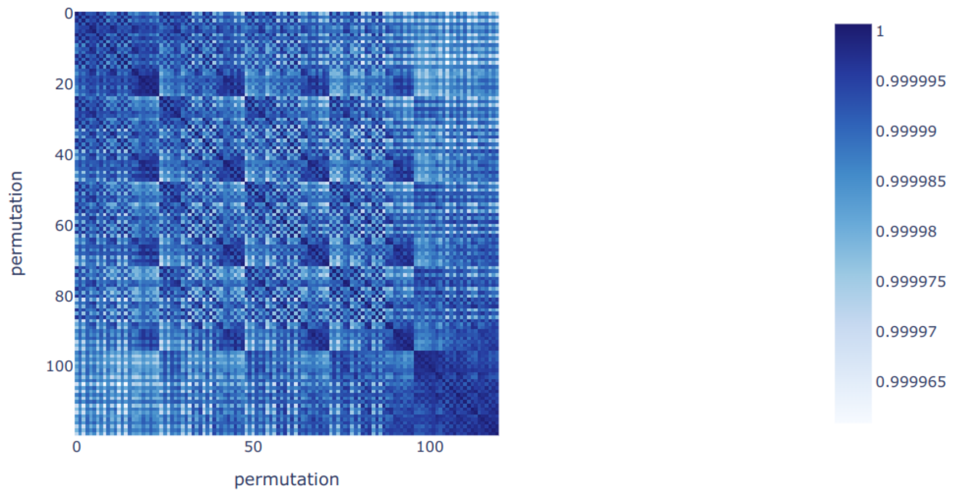


- Just as we observed with the max operation, the position encodings for the first five positions are very similar showing that the model learned positional invariance.

Activations across permutations

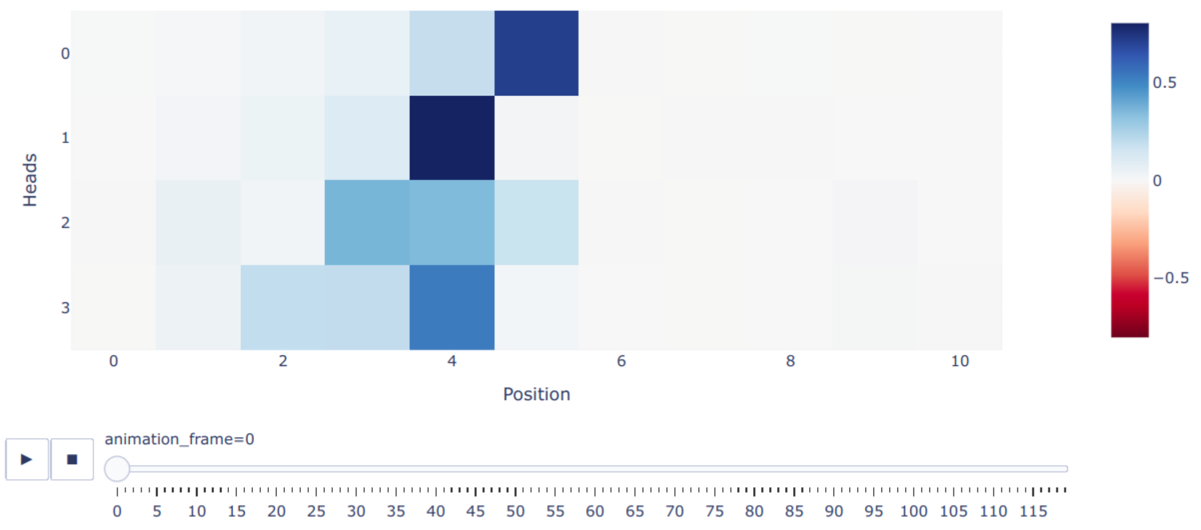
- To further prove positional invariance, we also plot the cosine similarity between the neuron activation for different permuations and again the neuron activations to be invariant to the permuation of the input.

Cosine similarity of Neuron activation for a pair of permutations.



Attention Plots for Different Heads:

Attention score for heads at all positions by 10th position for every permutation



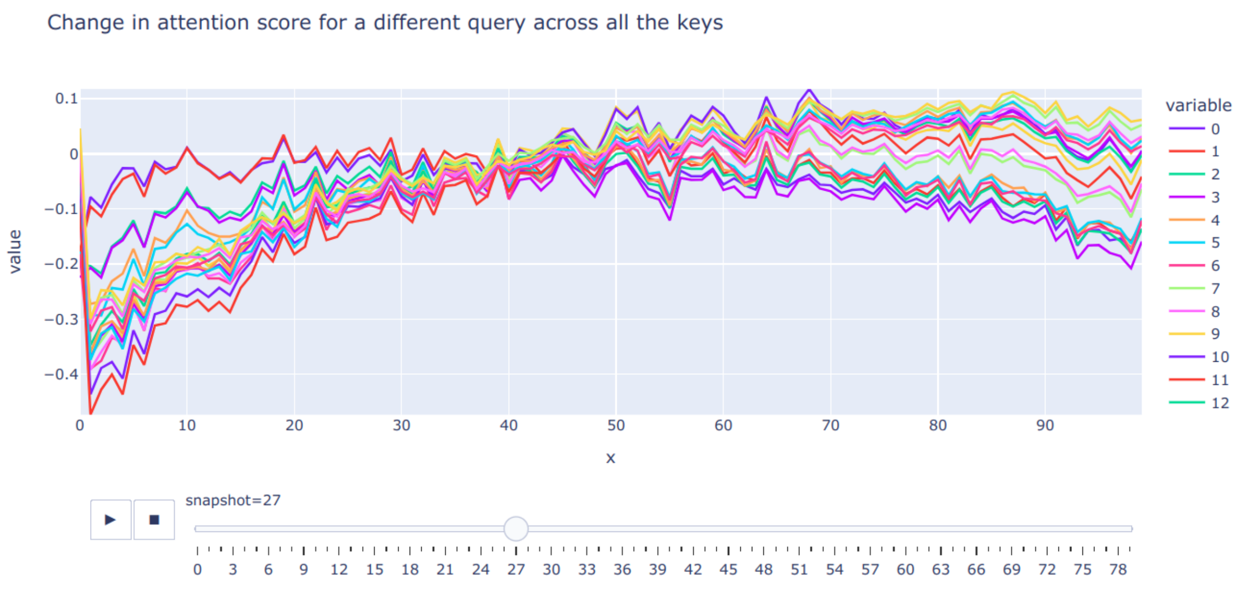
- We plotted the attention value which the 2nd last (9th) token pays while predicting the last number in sorting, we find that the heads pay most attention to the position containing the maximum element consistently. This pattern was also observed when we checked different positions, i.e. attention value of the 8th token consistently paid most attention to the position containing the second largest number.

What is the effect of current number on the prediction?

From the attention scores we see that the current CLS token (last predicted token) only pays attention to the inputs and 0 (depending on the head) but never on the previously predicted tokens. This would mean that the prediction is a function of the input sequence and the current value. To achieve sorting with this information, the model would need to learn to find the smallest number that is larger than the current number. We hypothesise that this is how it would work, but upon testing we see that it mostly predicts the number that belongs in that position rather than the number larger than the current number, indicating the significance of positional embedding of the predicted tokens.

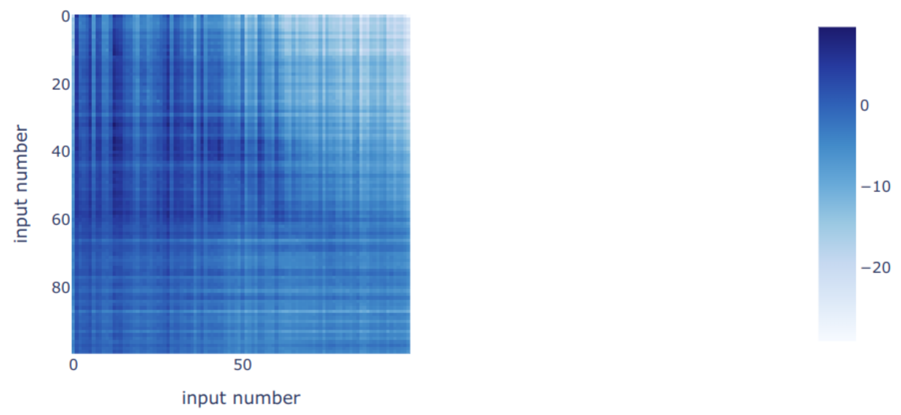
Multiple heads are complicated so we also tried one head (still complicated)

Attention score for every pair of inputs (Q-K analysis)

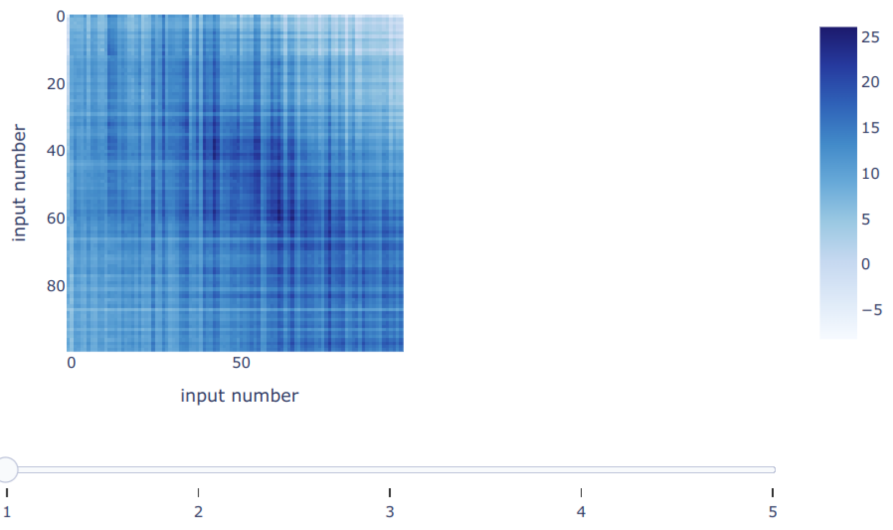


- We also plotted the Q-K attention scores for different input values and found that the attention scores variation are not as simple as the straight line curves observed with the max operation as it looks closer to a polynomial than a straight. We leave a detailed analysis for these attention scores as future work.

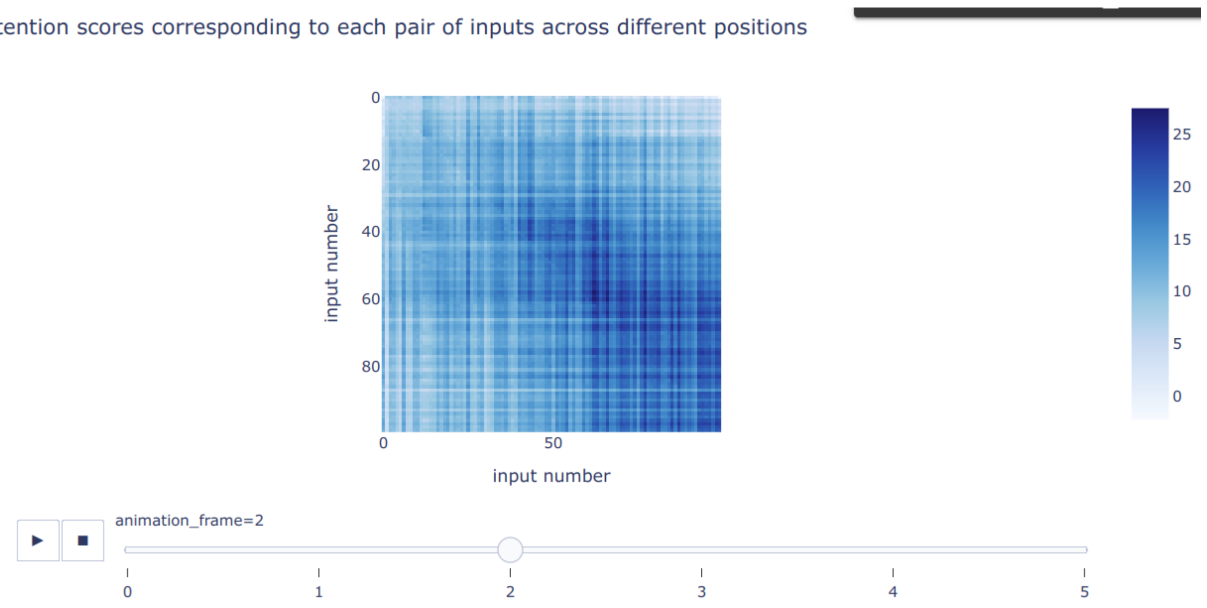
Attention scores corresponding to each pair of inputs across different positions



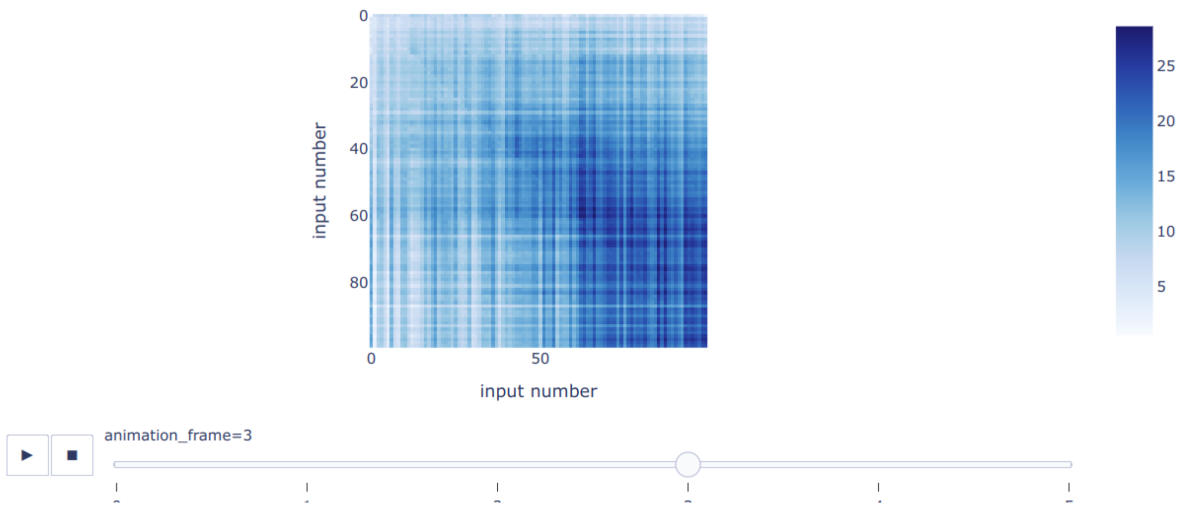
Attention scores corresponding to each pair of inputs across different positions



Attention scores corresponding to each pair of inputs across different positions

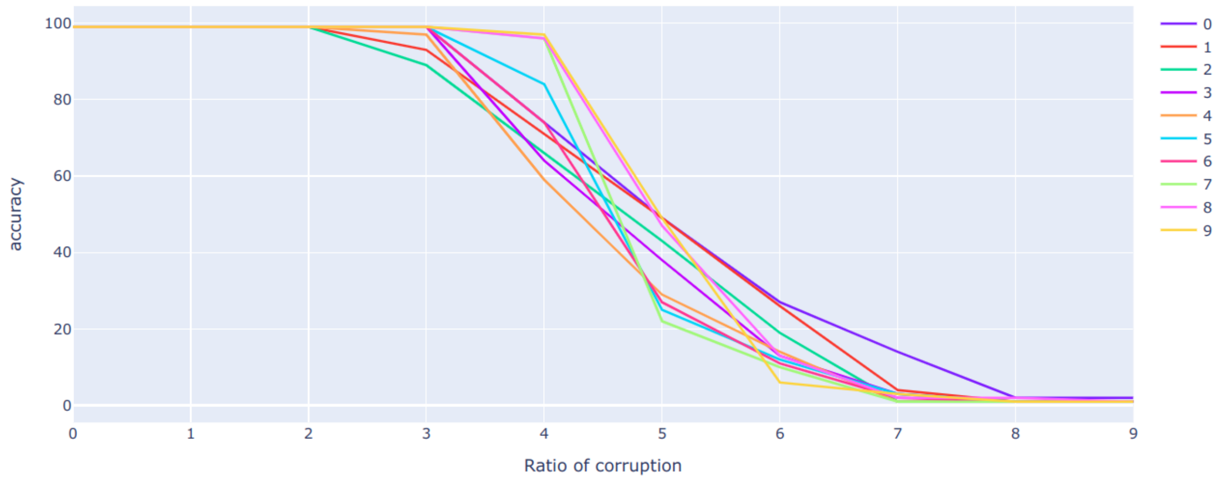


Attention scores corresponding to each pair of inputs across different positions



- The above graphs are similar to the graphs shown before but when input number 2 is kept at different positions while input number 1 is always kept at position 0. We didn't any patterns in the attention scores here so we leave interpreting the attention scores for future work.

- Robustness of the OV circuit: x-axis shows extent (0% to 100%) of adding noise (other competing v-value) and its effect on accuracy of copying in OV circuit. We see that we can add 30-40% percent of other v-value before accuracy drops from 100%. Robustness of copying shows redundancy in OV-circuit.



Conclusion

We trained and performed interpretation analysis on 3 tasks: max, median and sort.

References

[1] A. Power, Y. Burda, H. Edwards, I. Babuschkin, and V. Misra, “Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets,” Jan. 06, 2022, arXiv: arXiv:2201.02177. doi: 10.48a550/arXiv.2201.02177.

[2] N. Nanda, L. Chan, T. Lieberum, J. Smith, and J. Steinhardt, “Progress measures for grokking via mechanistic interpretability,” Oct. 19, 2023, arXiv:2301.05217. doi: 10.48550/arXiv.2301.05217.

[3] F. Charton, “Learning the greatest common divisor: explaining transformer predictions,” presented at the The Twelfth International Conference on Learning Representations, Oct. 2023. Accessed: Sep. 18, 2024. [Online]. Available: https://openreview.net/forum?id=cmcD05NPKa

[4] A. Gromov, “A simple and interpretable model of grokking modular arithmetic tasks,” Oct. 2023, Accessed: Sep. 18, 2024. [Online]. Available: https://openreview.net/forum?id=0ZUKLCxwBo

[5] N. Palumbo, R. Mangal, Z. Wang, S. Vijayakumar, C. S. Pasareanu, and S. Jha, “Mechanistically Interpreting a Transformer-based 2-SAT Solver: An Axiomatic Approach,” Jul. 18, 2024, arXiv: arXiv:2407.13594. doi: 10.48550/arXiv.2407.13594.

[6] Beren Millidge. Grokking ’grokking’, 2022. URL https://www.beren.io/ 2022-01-11-Grokking-Grokking/.

[7] F. Charton, “Linear algebra with transformers,” Nov. 08, 2022, arXiv: arXiv:2112.01898. doi: 10.48550/arXiv.2112.01898.

[8] B. Chughtai, L. Chan, and N. Nanda, “A Toy Model of Universality: Reverse Engineering How Networks Learn Group Operations,” May 24, 2023, arXiv: arXiv:2302.03025. doi: 10.48550/arXiv.2302.03025.

[9] N. Nanda, “200 COP in MI: Interpreting Algorithmic Problems,” Dec. 2022, Accessed: Sep. 18, 2024. [Online]. Available: https://www.alignmentforum.org/posts/ejtFsvyhRkMofKAFy/200-cop-in-miinterpreting-algorithmic-problems