Università
degli Studi
di Padova

Dipartimento di Fisica
e Astronomia
Galileo Galilei

Transformer architecture

Exploring GPT-2 model

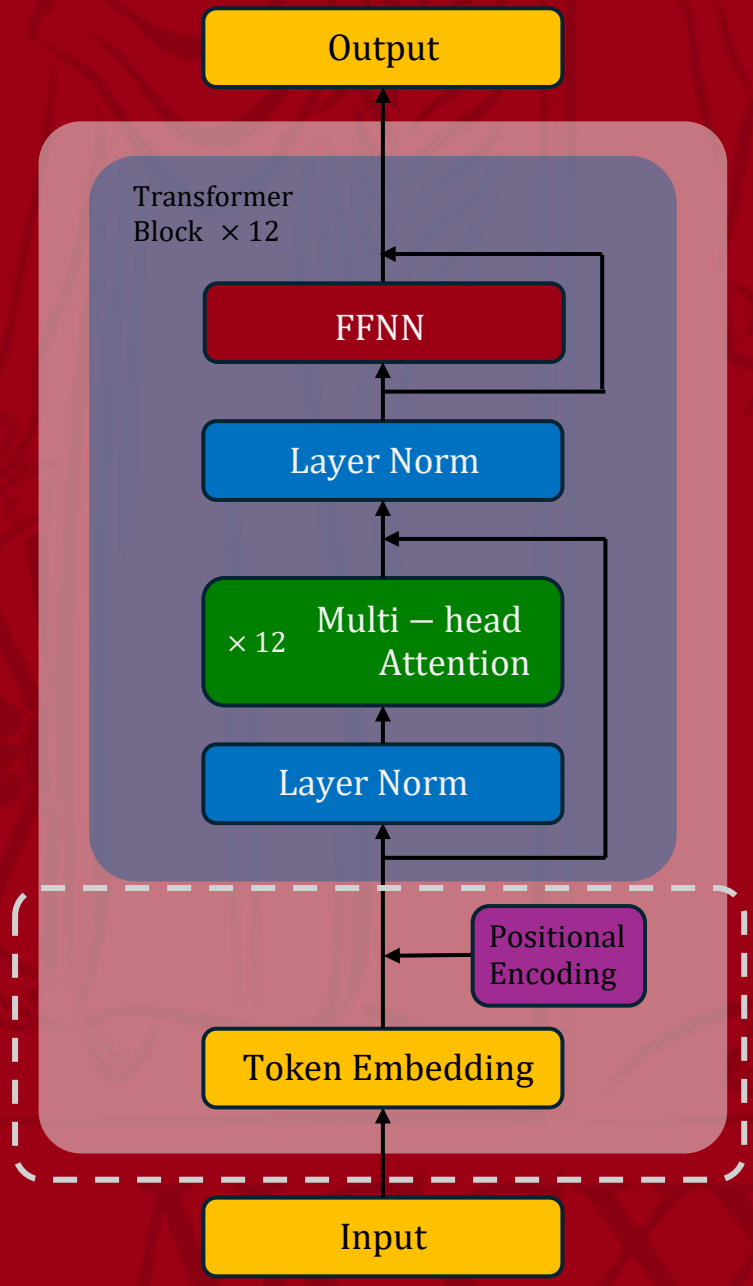
Laboratory of computational physics – MOD. B

Prof. Jeff Byers

Riccardo Corte
Alessandro Miotto
Lorenzo Rizzi

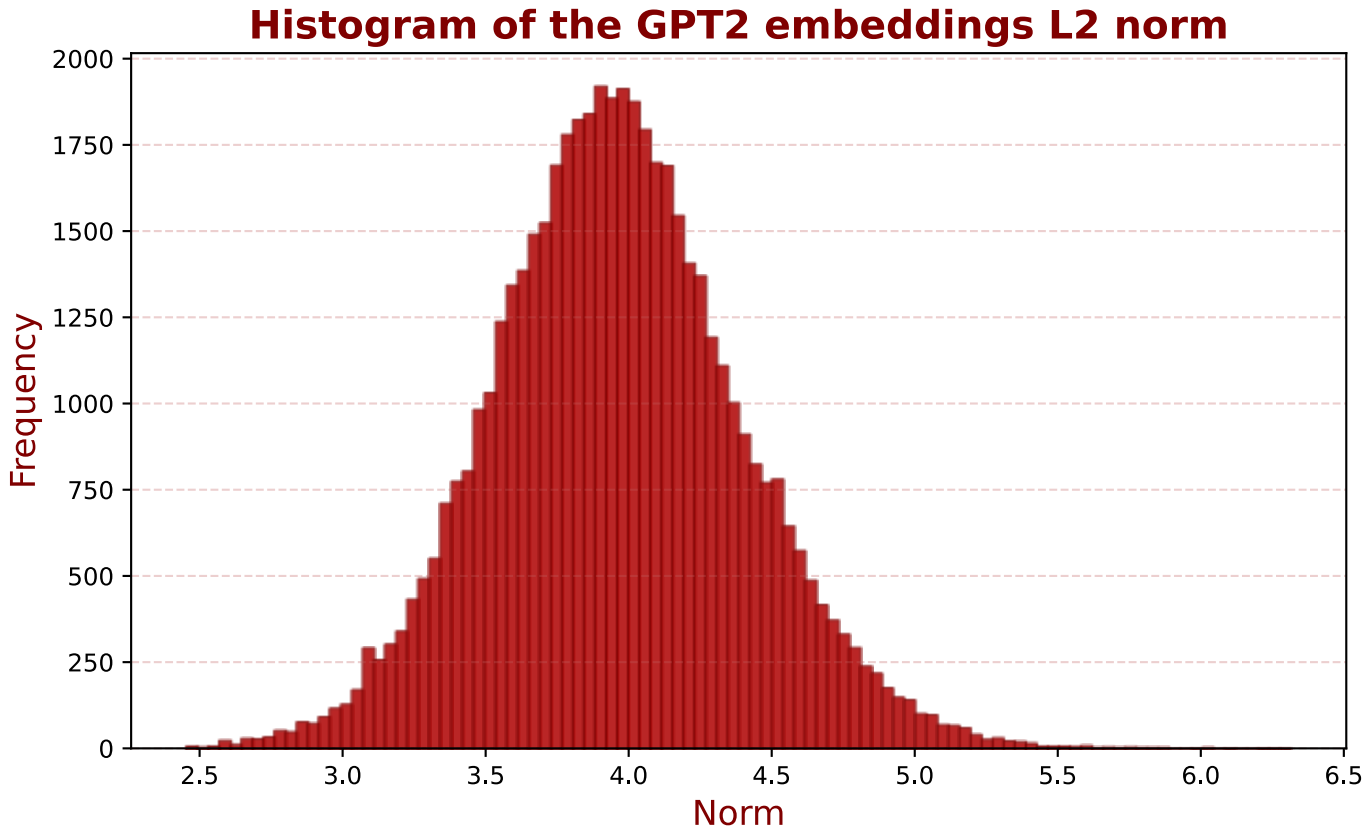
Part 1

EXPLORING GPT-2 EMBEDDING GEOMETRY



Part 1: Geometry of the embedding space

How are the embedding tokens distributed in the 784-dimensional space? Let's perform some analysis to gain some insights on the geometric property of the embedding space



Formally speaking, the embedding space is $\mathbb{R}^{784}$. Through the learning process, GPT-2 has decided how each token in the vocabulary should be represented, and thus we have a sequence of V points $\{x_1, x_2, \dots, x_V\} \subset \mathbb{R}^{784}$

We can study the norm (in the L2 sense) of those points. They are not normalized to a unique value and their average value is about 4.

Part 1: Distribution of cosine similarities

Let's compute the average **angular distance** between randomly chosen tokens x_i and x_j measured as their **cosine similarity**, i.e.:

$$\cos(x_i, x_j) = \frac{x_i \cdot x_j}{\|x_i\| \|x_j\|}$$

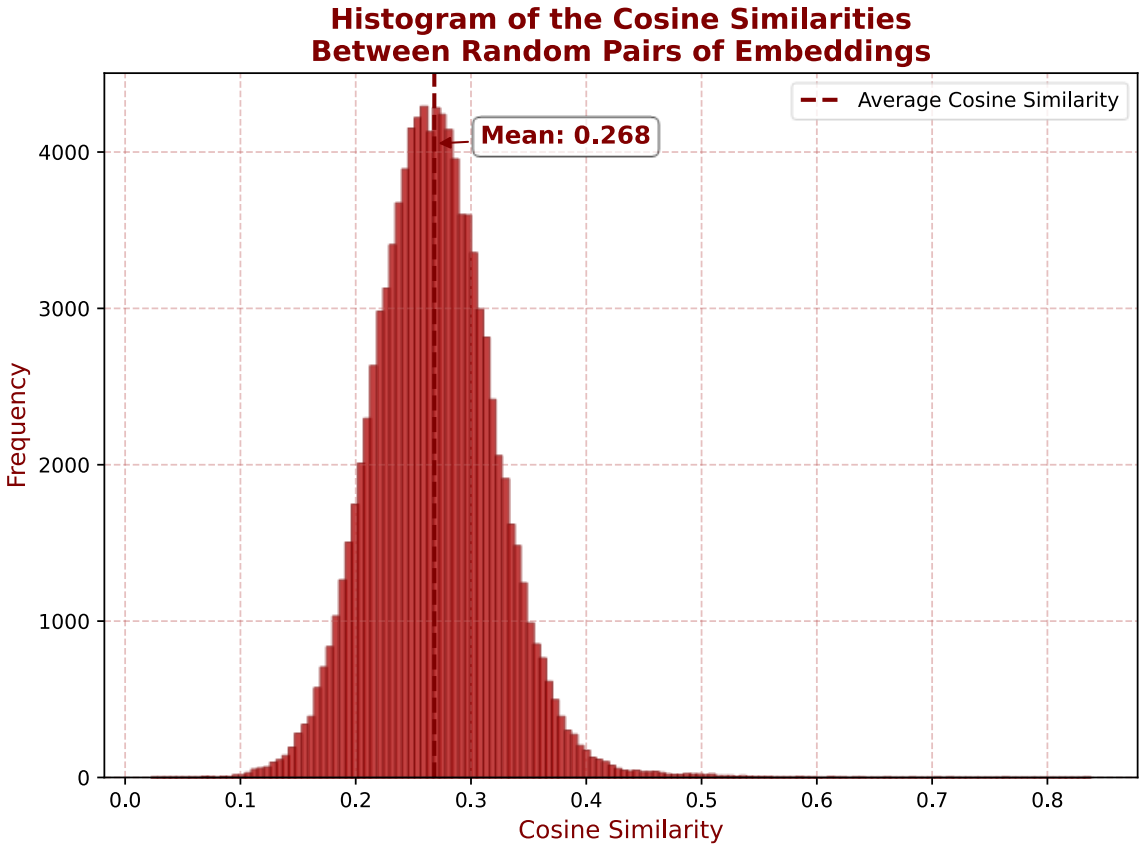
So, on average, randomly sampled tokens are **more probably acute** than obtuse. Let x_i and x_j be two vectors. Then, if we sample tokens independently, we expect that:

$$\langle \cos(x_i, x_j) \rangle = \langle x_i \cdot x_j \rangle \approx \langle x_i \rangle \cdot \langle x_j \rangle$$

If we assume that they are **homogenously distributed** around the origin, so that on average $\langle x_i \rangle = 0$, then we would have:

$$\langle \cos(x_i, x_j) \rangle = 0$$

Instead we get: $\langle \cos(x_i, x_j) \rangle \approx 74{,}45°$



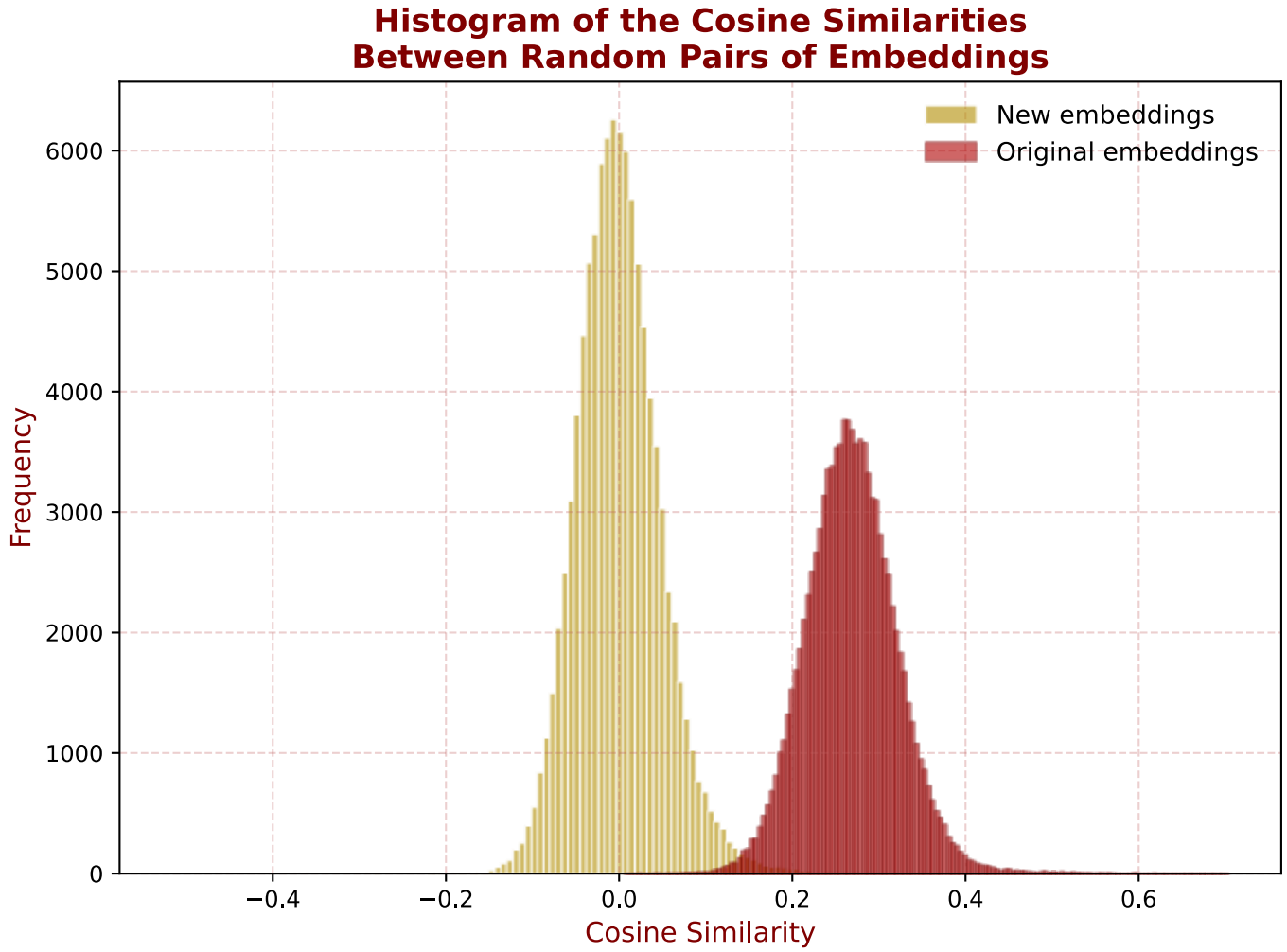
Part 1: Center of mass

This bias is due to a global center of mass (CoM) different than 0. So yes, the tokens have a specific average value different from zero.

Why is that? Why did GPT2 learn such a bias?

Let's get rid of this average value:

After removing the center of mass, the distribution of cosine similarities becomes much more symmetric around zero



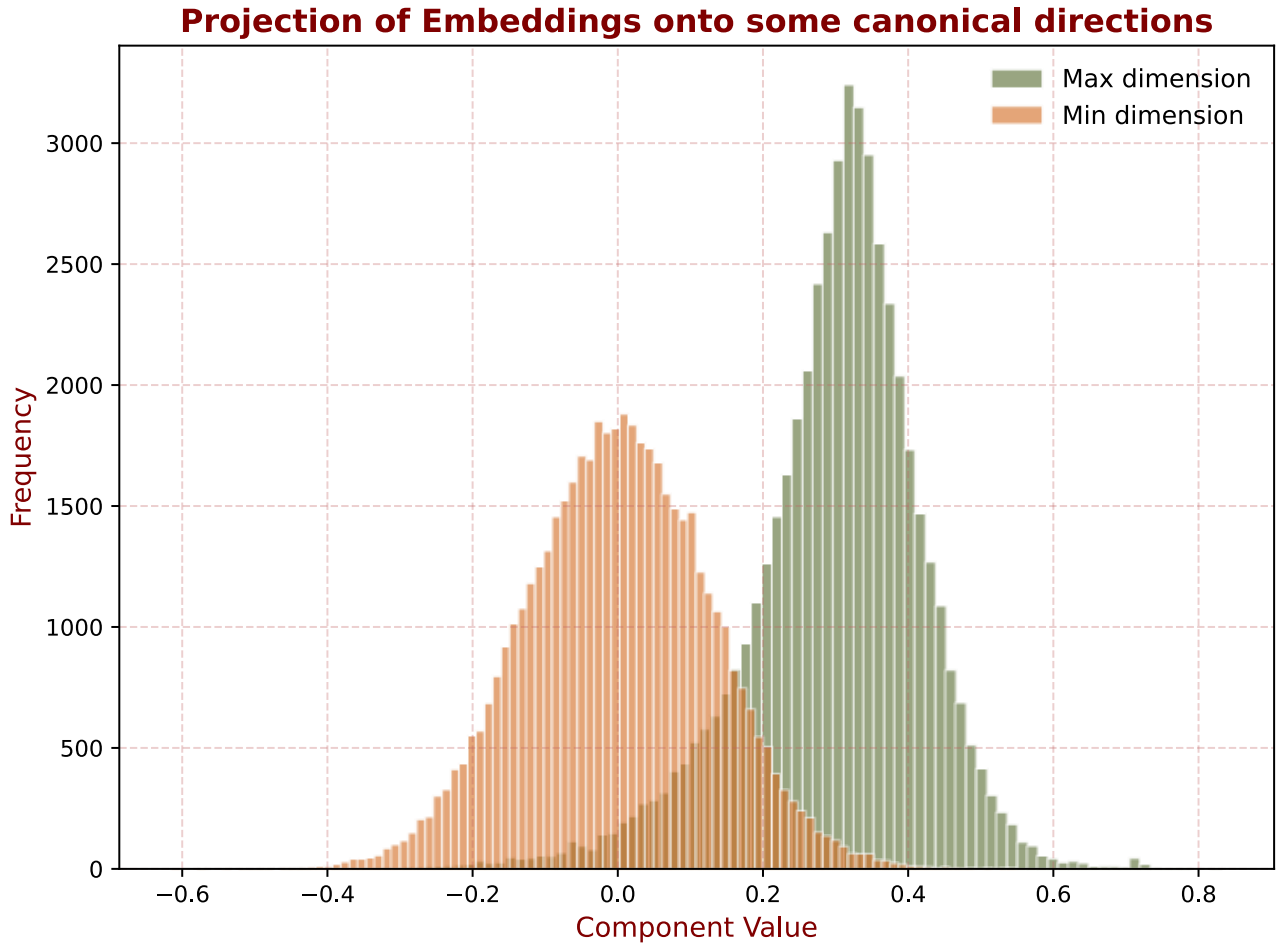
Part 1: Center of mass bias

To be completely sure, let's select one component for each token. We choose the dimension j such that the j-th component of the center of mass is the largest

$$j = \arg \max_{1 \le i \le 768} (\mathrm{CoM}_i)$$

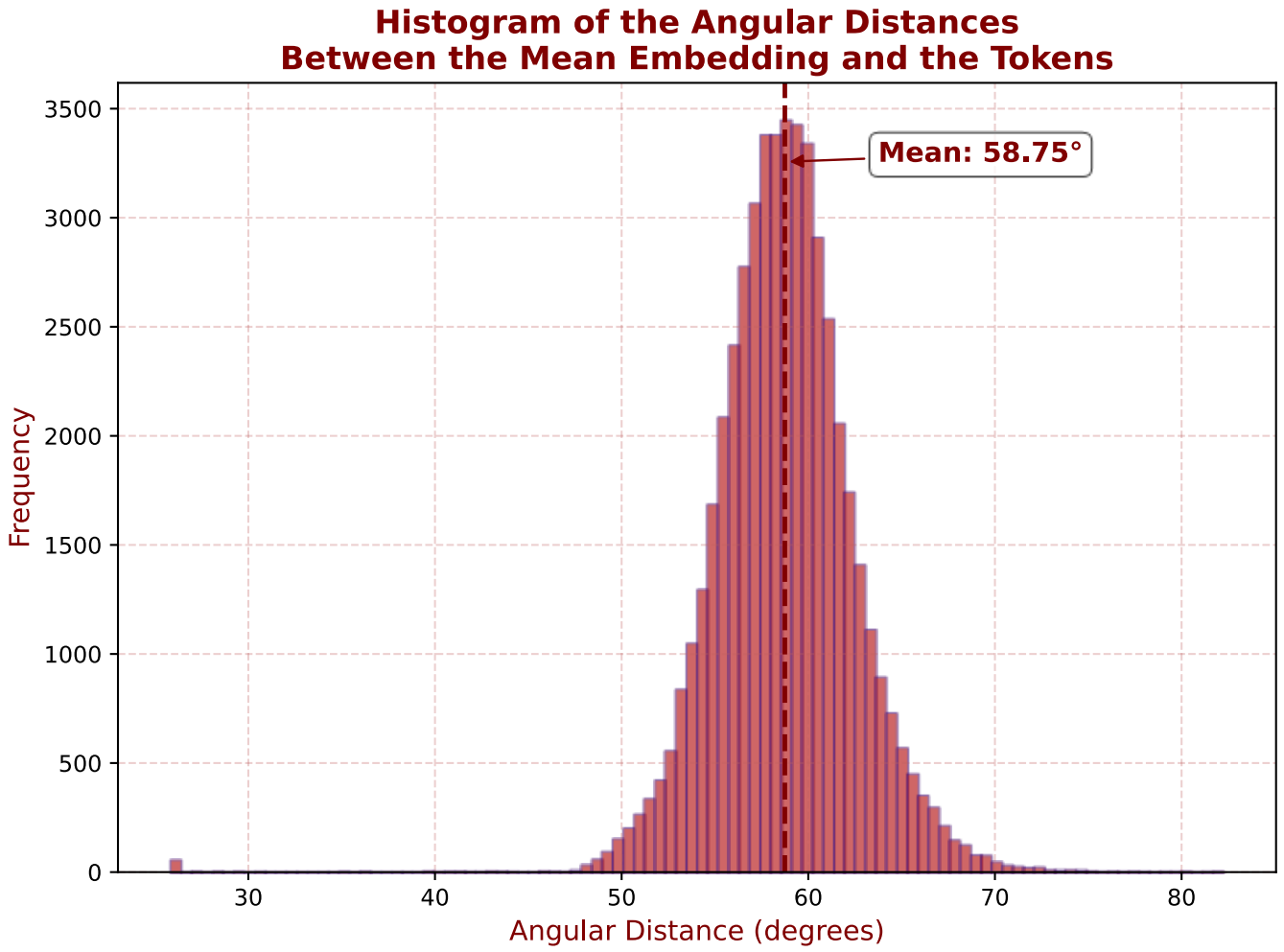
In other words, we are looking for the dimension along which the asymmetry is most pronounced.

And indeed, when selecting the maximum dimension j, the j-th components of all the embedded tokens are distributed in a highly asymmetric manner, predominantly positive.



Part 1: A special cone

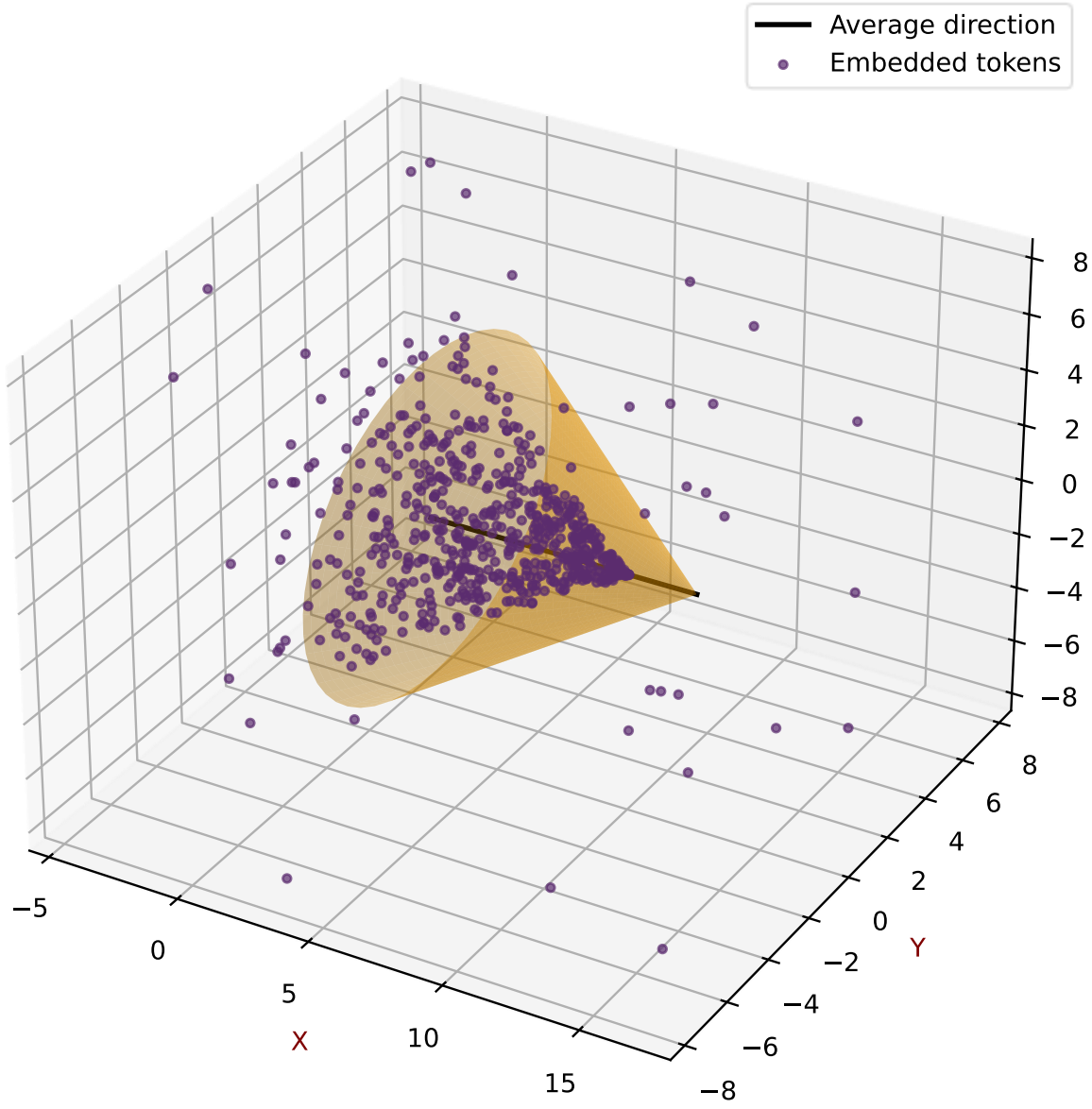
Given the average direction (i.e., the center of mass normalized), let's compute the angle between each token and the average direction.



This result is fascinating.
On average, each token forms an angle of about $60°$ with the mean direction. In other words:

All the tokens lie within a hyper-cone
with an average opening angle of approximately $60°$

Part 1: A special cone



Part 1: CoM bias

Why do we have this average value? Is there a small bias? Does it represent something? And does it disappear when `LayerNorm` is applied?

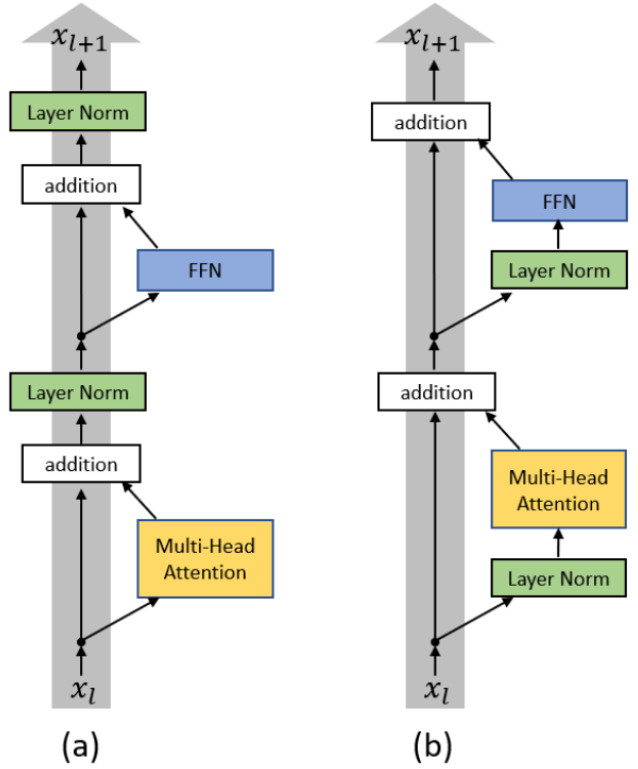


Figure 1. (a) Post-LN Transformer layer; (b) Pre-LN Transformer layer.

The effect of the `LayerNorm` is, given x a row in the prompt matrix,

$$\mathrm{LN}(\boldsymbol{x}) = \frac{\boldsymbol{x} - \boldsymbol{\mu}}{\sigma} \cdot \boldsymbol{\gamma} + \boldsymbol{\beta}$$

where μ, σ are the average and standard deviation along the row, so for each token.

So, we may ask ourselves why the algorithm has learnt this specific bias and whether it is meaningful. We can explore this issue my tweaking the embedding matrix E that GPT2 uses to perform its computations and removing this bias

Part 1: Modifying GPT inner working

Given prompt = "*The year is 2194, and humanity no longer lives on Earth. After the Collapse, we fled to floating arcologies orbiting the gas giants. Every child knows the stories of our homeworld, but none of us have seen a tree, felt rain, or walked on ...*"

With the initial bias:

(normal GPT2)

$\log p$	Token	Idx
−1.04	‘ water’	1660
−1.58	‘ a’	257
−2.16	‘ the’	262
−3.64	‘ ice’	4771
−3.72	‘ fire’	2046

With the modified version of E:

(removal of CoM from `model.wte.weight`)

$\log p$	Token	Idx
−1.14	‘ water’	1660
−1.26	‘ a’	257
−1.91	‘ the’	262
−3.76	‘ fire’	2046
−3.79	‘ air’	1633

Part 1: Modifying GPT inner working

Given prompt = "*Astronomy is the study of celestial bodies and phenomena beyond Earth's atmosphere. Using telescopes and spacecraft, scientists have discovered evidence for galaxies, black holes, and ...*"

With the initial bias:

(normal GPT2)

$\log p$	Token	Idx
−1.52	' other'	584
−3.13	' even'	772
−3.16	' dark'	3223
−3.20	' planets'	14705
−3.37	' super'	2208

With the modified version of E:

(removal of CoM from `model.wte.weight`)

$\log p$	Token	Idx
−1.61	' other'	584
−2.10	' planets'	14705
−2.33	' even'	772
−3.04	' stars'	5788
−3.18	' dark'	3223

Part 1: Rotating the tokens

Removing the center of mass (CoM) does not seem to significantly affect the probability density function over the vocabulary space.

To further validate our approach, we explore an alternative modification of the embedding space: a 784-dimensional rotation of the embedding space!

To construct a rotation matrix in $\mathbb{R}^{768}$:

- **Begin with the identity matrix $\mathbb{I}_d$ and choose a dimension j. Replace the j-th column with a normalized vector v of your choice (which defines the new direction for that basis vector)**
- **Move this modified column to the first position (swap it with the first column)**
- **Apply Gram-Schmidt orthogonalization starting from the second column onward**
- **Finally, swap the first and the j-th columns back to restore the original order**

The resulting matrix A is a proper rotation matrix: an orthogonal transformation that preserves vector norms and dot products (i.e., $A^T A = \mathbb{I}$, and $\det(A) = 1$).

(ref. n Dimensional Rotation Matrix)

Part 1: Rotating the tokens

$$E \xrightarrow{\text{remove CoM}} E - \text{mean}(E, \text{dim} = 1) \xrightarrow{\text{apply rotation}} A \cdot E$$

Given prompt = "*The year is 2194, and humanity no longer lives on Earth. After the Collapse, we fled to floating arcologies orbiting the gas giants. Every child knows the stories of our homeworld, but none of us have seen a tree, felt rain, or walked on ...***"**

With the initial bias:

(normal GPT2)

$\log p$	Token	Idx
−1.04	' water'	1660
−1.58	' a'	257
−2.16	' the'	262
−3.64	' ice'	4771
−3.72	' fire'	2046

With the modified version of E:

(removal of CoM and applying rotation)

$\log p$	Token	Idx
−1.14	'theless'	9603
−1.26	'soDeliveryDate'	39811
−1.91	'SPONSORED'	37190
−3.76	'interstitial'	294
−3.79	''	35266

Rotating by even a tiny bit the embedding matrix, the result is catastrophic…

The embedding is translationally invariant, but not isotropic

Part 1: Estimating differences in PDFs

To get a sense of how different the modified GPT is from the original, we compare the resulting probability distributions over the vocabulary space.

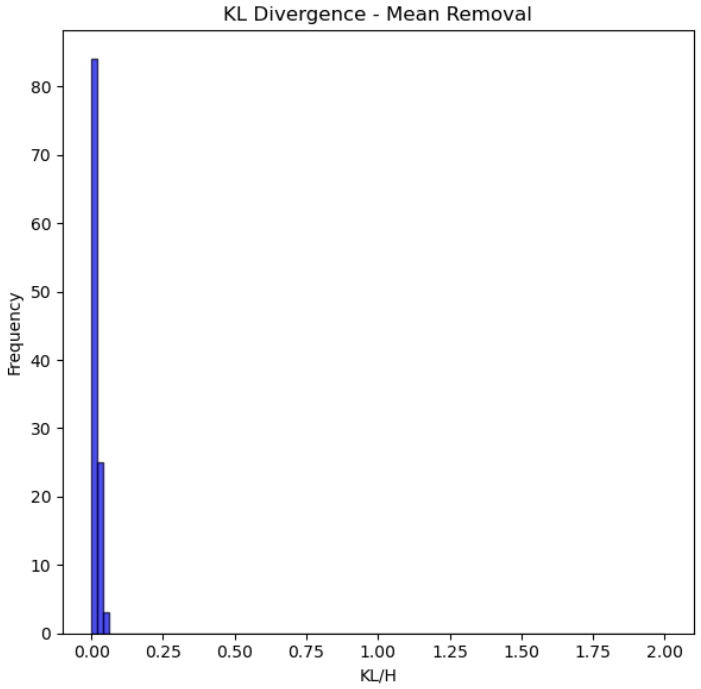
We use the Kullback-Leibler (KL) divergence:

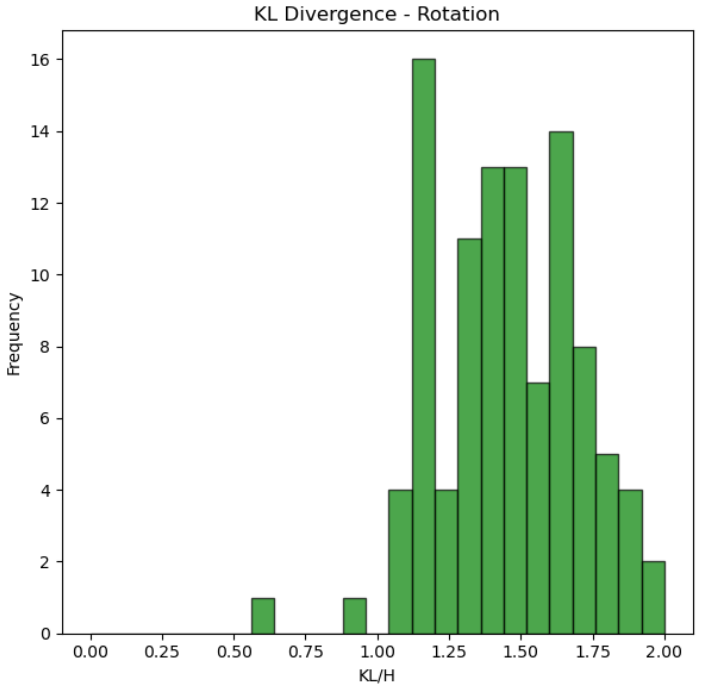
$$\mathrm{D}(p||q) = \sum_{x \in V} p(\boldsymbol{x}) \ln\left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\right)$$

normalized by the entropy of the first distribution:

$$\mathrm{H}(p) = -\sum_{x \in V} p(\boldsymbol{x}) \ln\big(p(\boldsymbol{x})\big)$$

If the ratio $\boldsymbol{D(p||q)/H(p) < 1}$, then the distribution q does not deviate significantly from q.





Part 1: Results

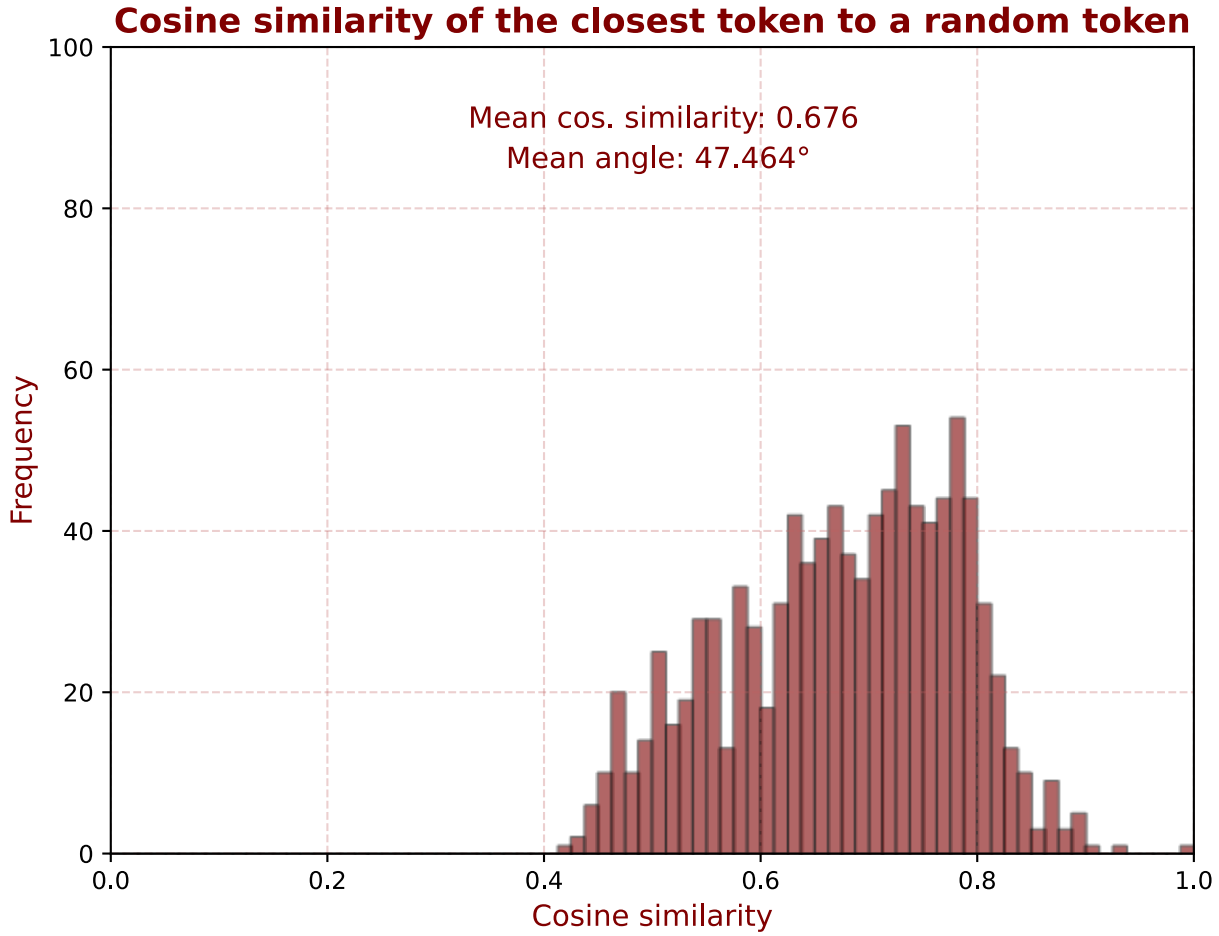
In conclusion, the center of mass appears to have little to no effect on GPT-2's behavior. Although some differences were observed (as detailed in the accompanying Jupyter notebook), they are insufficient to draw definitive conclusions.

- **Mu, J., & Viswanath, P. (2018). *All-but-the-Top: Simple and Effective Postprocessing for Word Representations*. https://arxiv.org/abs/1702.01417**

  *" Observation: Every representation we tested, in many languages, has the following properties: [...] the word representations have non-zero mean – indeed, word vectors **share a large common vector** (with norm up to a half of the average norm of word vector) [...] In this paper, we demonstrate a very simple postprocessing technique – **eliminate the common mean vector** and a few top dominating directions from the word vectors – that renders off-the-shelf representations **even stronger** "*

Part 1: Token Separation in Embedding Space

We randomly choose r points from the embedding space. To evaluate how well **separated** they are (i.e., how well resolved the semantic meanings are), we compute the cosine similarity between each chosen point and its closest neighbor.



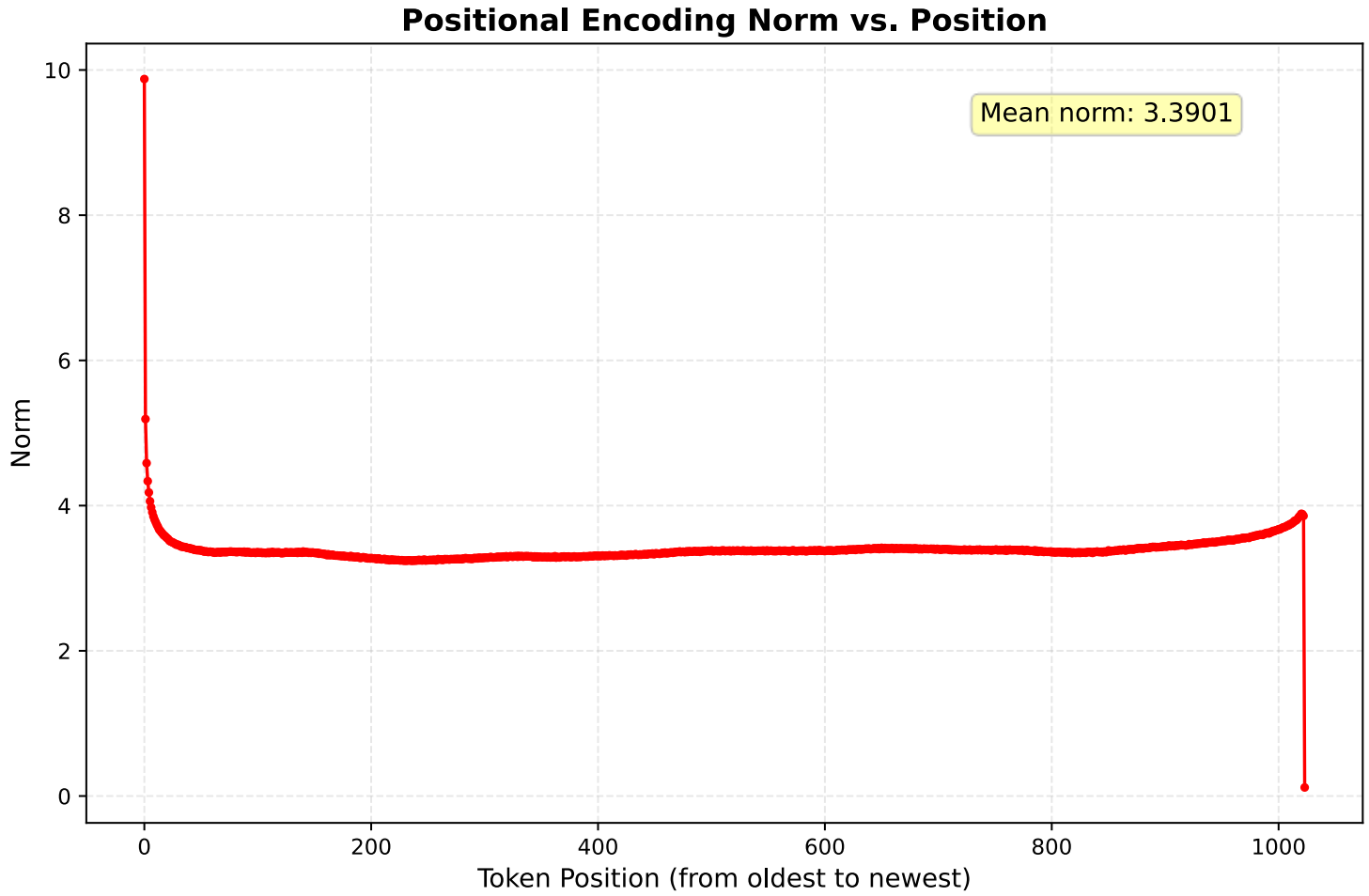
This provides an intuition on how well tokens are separated. The average angle between closest tokens is approximately $48°$ (in any direction). This insight will be useful later.

Part 1: Positional encoding

In GPT2, the PE matrix is learnt, not deterministically set. Let's analyze a bit its structure

The first positional token has a surprisingly high norm with respect to the others. The trend is bizarre: it starts very high and then stabilize. The first tokens in the prompt are thus significantly perturbed

How much are tokens from the embedding space perturbed when adding the PE?

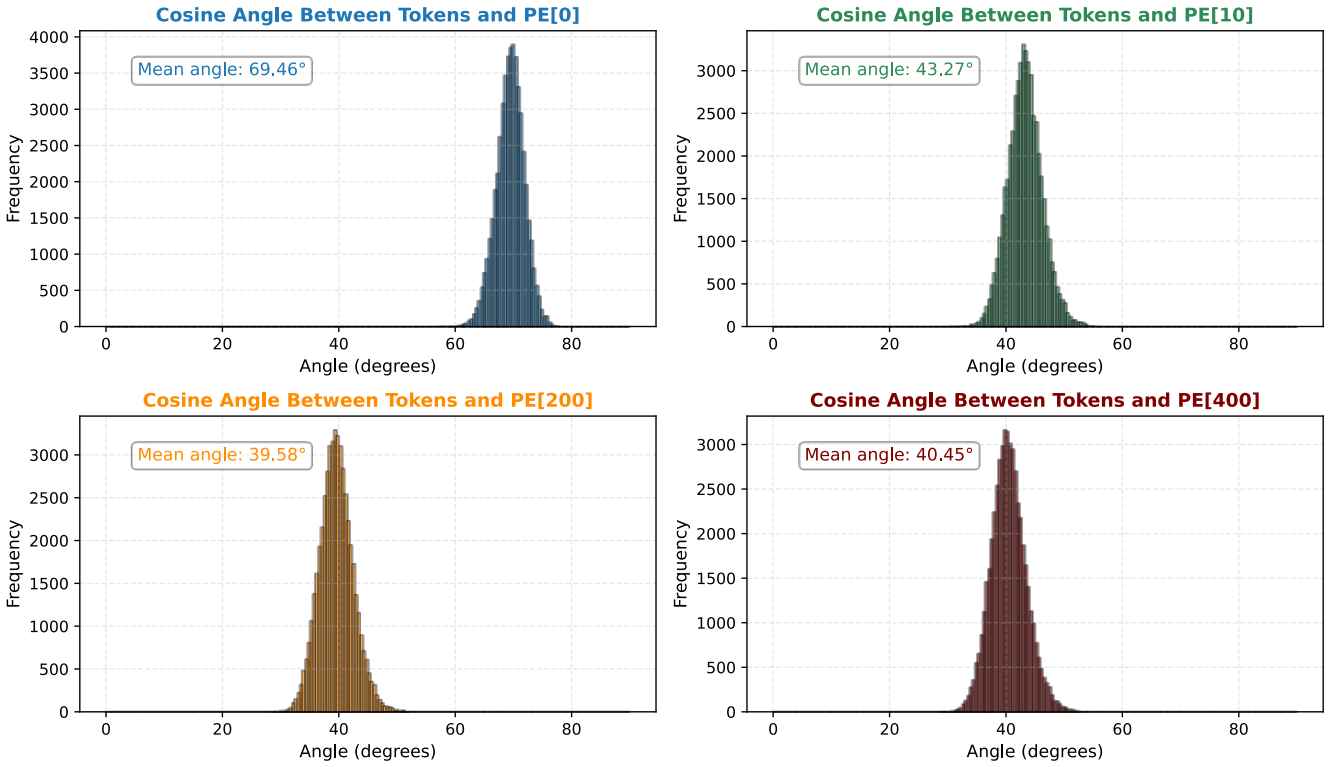


Part 1: Perturbation Induced by Positional Encoding

To quantify the semantic distortion induced by positional encoding, choose a vector x:

$$\boldsymbol{x} \rightarrow \boldsymbol{x} + \mathrm{PE}[i] \rightarrow \mathrm{angle}(\boldsymbol{x}, \boldsymbol{x} + \mathrm{PE}[i])$$

This angle reflects how much the direction of x is perturbed by the addition of the positional encoding at position $i \in \{0, 10, 200, 400\}$. A large angle implies that the semantic direction of the token is significantly altered, while a small angle means that the core meaning is mostly preserved.



Early positions (e.g., position 0) cause a much stronger distortion compared to later ones. For positions beyond ≈ 10, the shift is smaller than the average angular resolution between tokens, suggesting that the semantic geometry is largely preserved for further away tokens.

Part 1: Positional perturbation

Why this strange behavior for the first tokens? Two explanations:

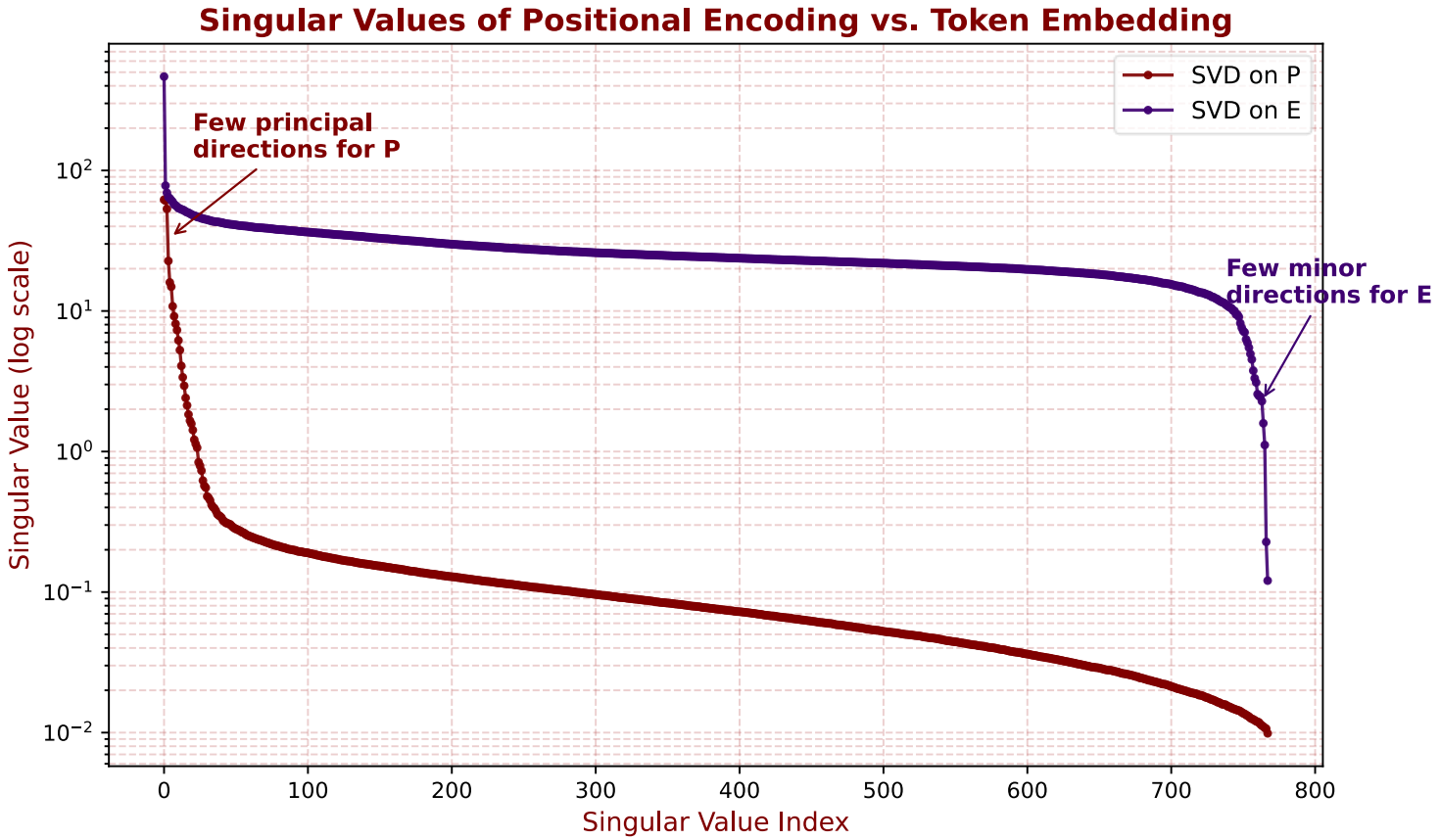
- **Low semantic load:** **Initial tokens are frequently articles, punctuation, or other high-frequency terms with low semantic content. The model can afford to displace them more effectively pu*shing them away* without harming the overall meaning, freeing up directional space for more informative tokens. Basically, the model really seem to care a lot about the first few tokens and *differentiate* those tokens more by spacing them out**
- **Training dynamics:** **The first positions are always populated (never masked or padded) in the training phase. As a result, their positional encodings receive disproportionately more gradient updates during training, which may naturally lead to larger norms.**

N.B. Some sources suggest that the GPT-2 tokenizer internally prepends a `<|endoftext|>` *token to improve inference robustness. Under this assumption, the first positional embedding would correspond to this special token. However, I did not observe such behavior, and the literature appears to be inconsistent on this point.*

Part 1: Embedding space ⊥ Positional space

Let E be the embedding matrix and P the positional matrix:

$$E, P \xrightarrow{\text{Apply SVD}} V_E, V_P$$



- The positional encoding matrix P is effectively low-rank.
- Its structure is largely captured by just the top ~ 30 singular directions.
- By contrast, the token embedding matrix E has much higher rank: its spectral mass is distributed across many singular directions.

The following procedure is inspired by
https://james-simon.github.io/blog/gpt2-positional-encs/

Part 1: Embedding space ⊥ Positional space

We demonstrate that the top principal directions of P align with the bottom principal directions of E, indicating that these two subspaces are largely disjoint. To quantify this relationship, we proceed as follows:

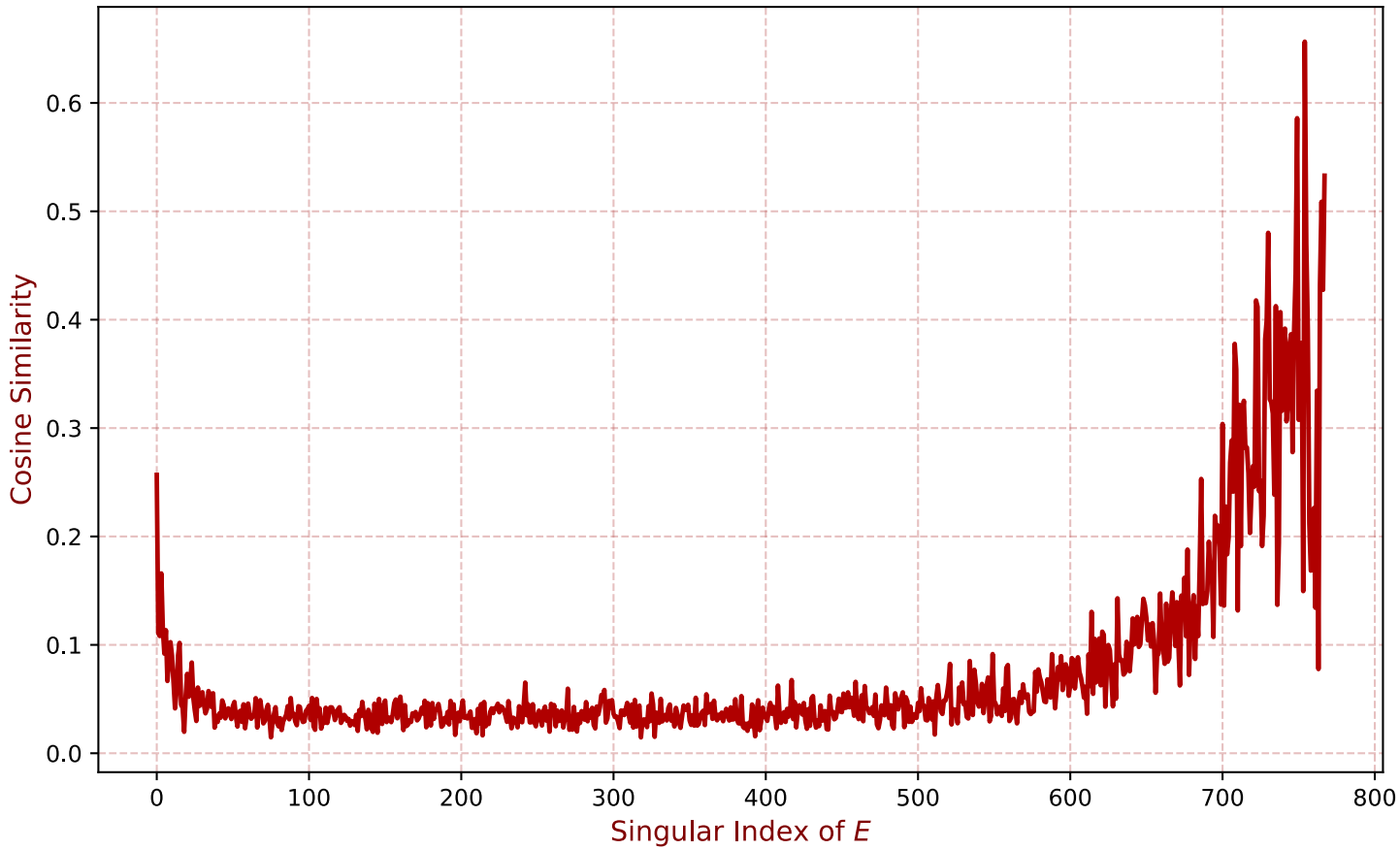
- **Select the top 10 principal directions $\tilde{V}_P$ of the positional matrix P, i.e., the first 10 rows of V_P, and construct the projector onto this subspace:**

$$\Pi_{10} = \tilde{V}_P^T \tilde{V}_P$$

- **For each singular direction $v_j \in V_E$ of the embedding matrix E, compute its projection onto this subspace using the cosine similarity:**

$$p_j = \langle \boldsymbol{v_j} | \Pi_{10} | \boldsymbol{v_j} \rangle$$

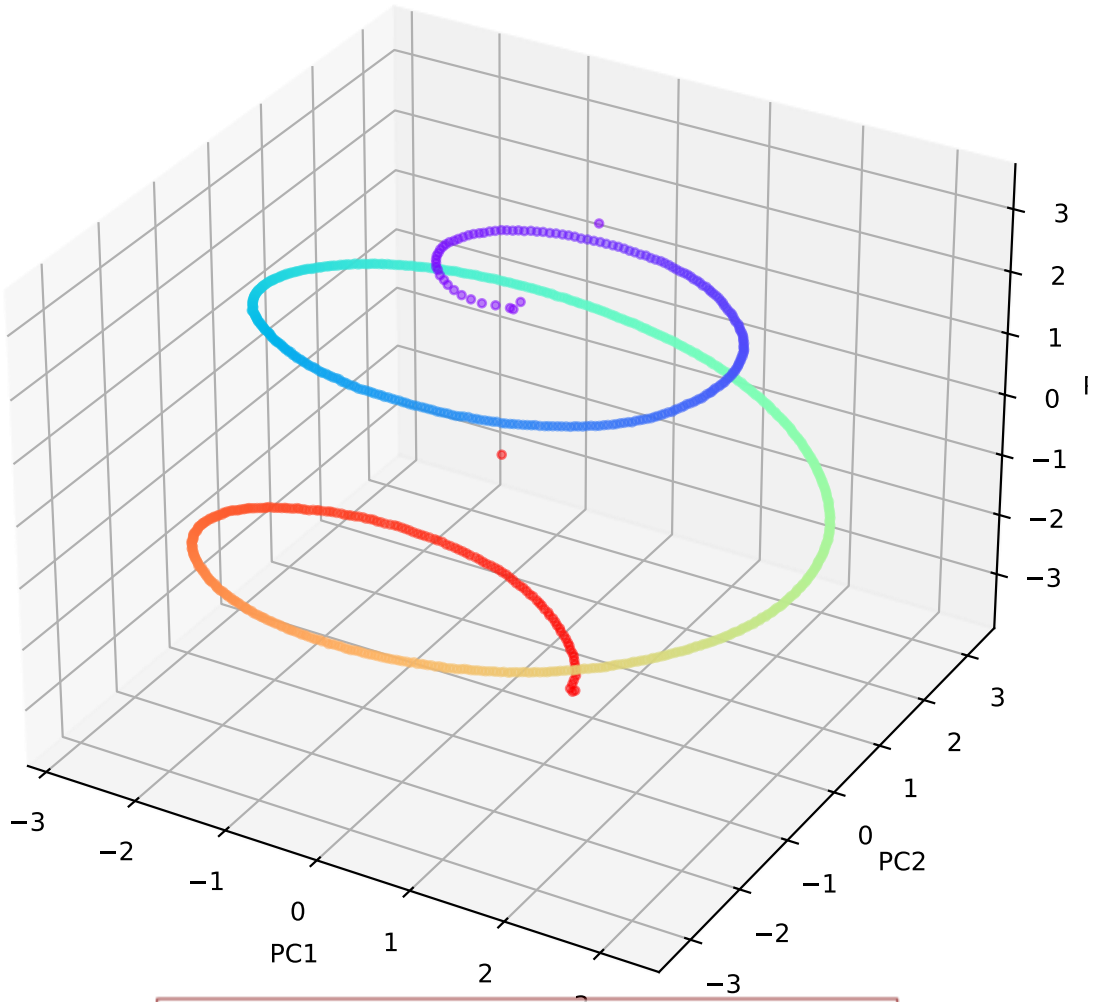
Part 1: Embedding space ⊥ Positional space



The concentration of values on the right side of the plot indicates that the top subspace of the positional encodings closely aligns with the bottom subspace of the token embeddings.

In other words, the two matrices span nearly orthogonal subspaces, suggesting that positional encodings and token embeddings operate independently, without interfering with each other.

Part 1: Positional encoding is a helix



The positional encoder P is an extremely-low rank matrix. Let us build Π_3 and project P into the subspace of its three most prominent directions

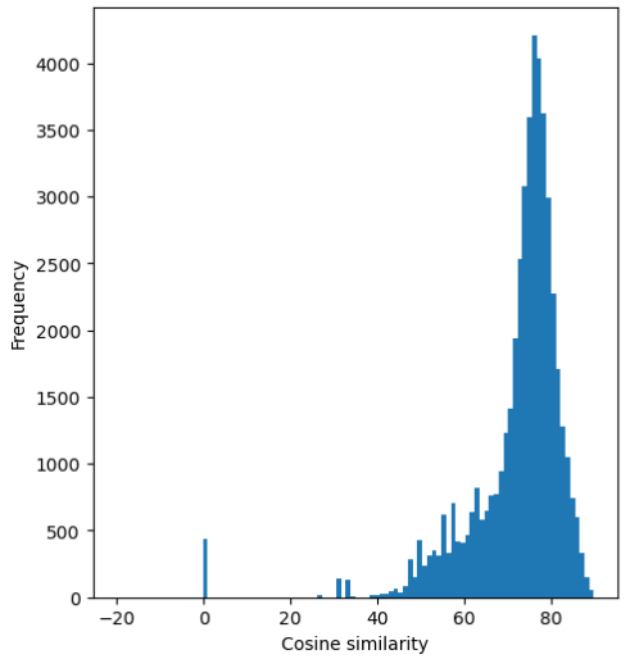
The positional matrix in GPT-2 is learned, not fixed during pre-training. In the famous paper "*Attention Is All You Need*", authors proposed a sinusoidal structure for positional embeddings. However, even when GPT-2 learns the positional structure on its own, it still appears to develop a representation with **strong periodicity**.

Part 1: Positional Encoding + LayerNorm

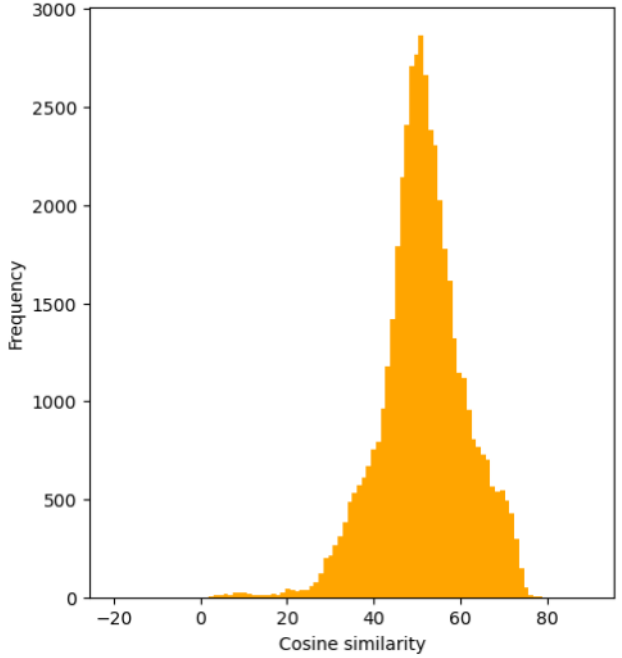
After adding the positional encoding (PE), GPT-2 applies a `LayerNorm` operation. In previous slides, we analyzed the statistical properties across the entire vocabulary. Here, we focus specifically on the tokens within selected prompts.

We consider the same 112 prompts and, for each, compute the distribution of cosine similarities at three stages:

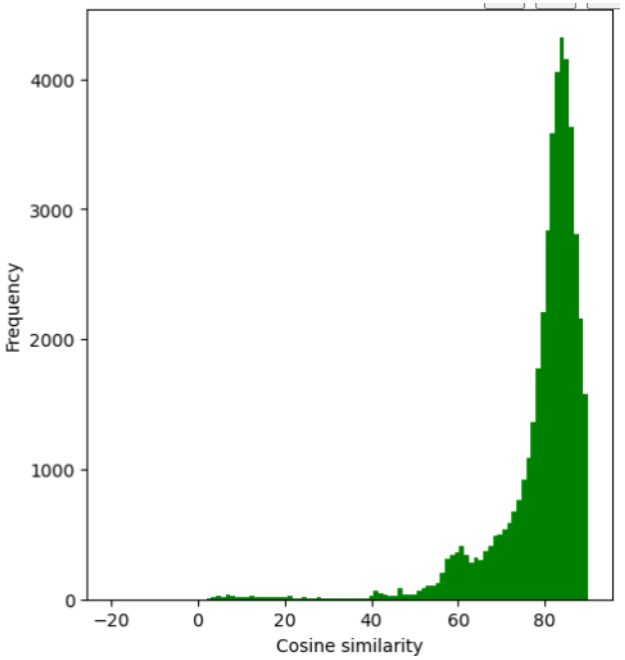
Immediately after the initial embedding



After adding positional encoding



After applying `LayerNorm`



Part 1: Norm of the Mean vs. mean of the norms

A useful metric to characterize the distribution of points is the ratio between the norm of the mean and the mean of the norms, often called the coherence metric

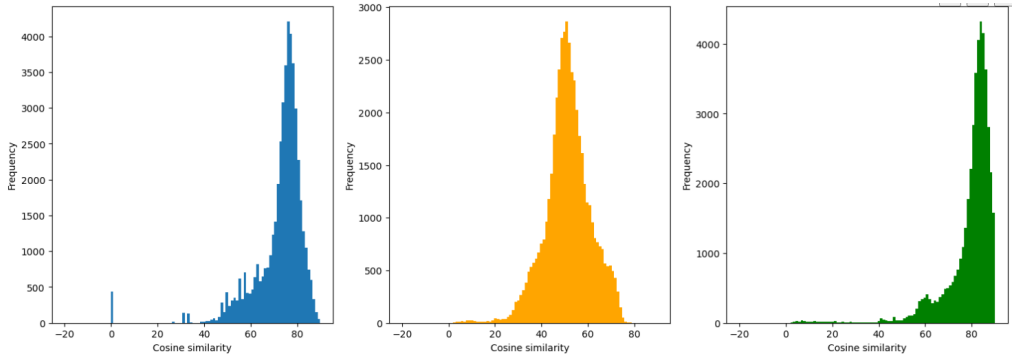
$$R = \frac{\|\langle \boldsymbol{x} \rangle\|_2}{\langle \|\boldsymbol{x}\|_2 \rangle}$$

The advantage of using R is that it is computed over a specific prompt and not on the entire ensemble, i.e., the whole vocabulary, as we did above. It's less statistically rigorous, but more empirical

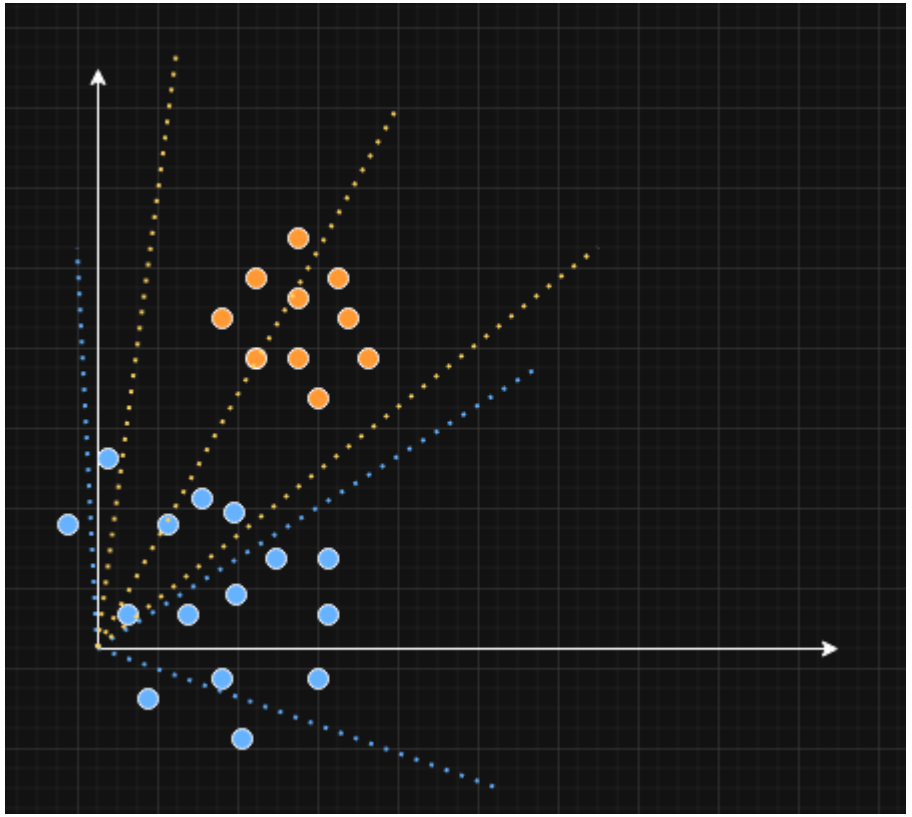
If the points are uniformly distributed on a hypersphere centered at the origin, we expect $R \approx 0$. It can be shown that the ratio satisfies $0 \leq R \leq 1$.

- **When $R = 1$, all points are identical or perfectly aligned, meaning the norm of the mean equals the mean of the norms.**
- **Values of R between 0 and 1 indicate varying degrees of directional coherence among the points.**

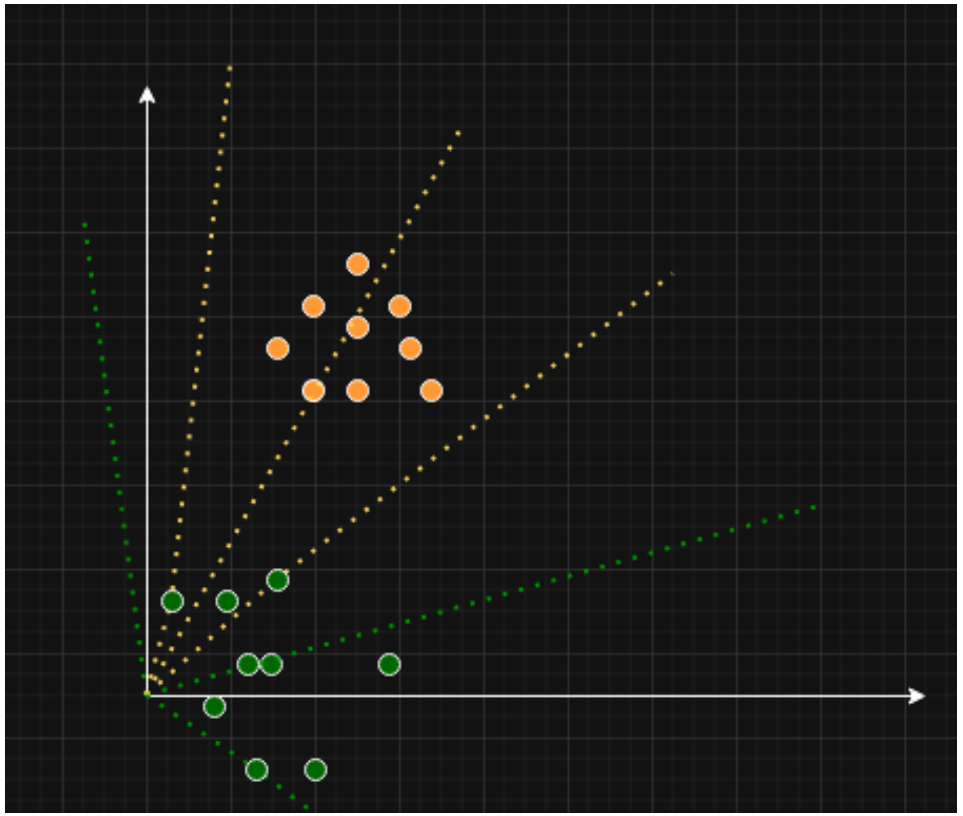
	R
Before PE	0.553
After PE	0.784
After LN	0.435



Part 1: A picture of the embedding space



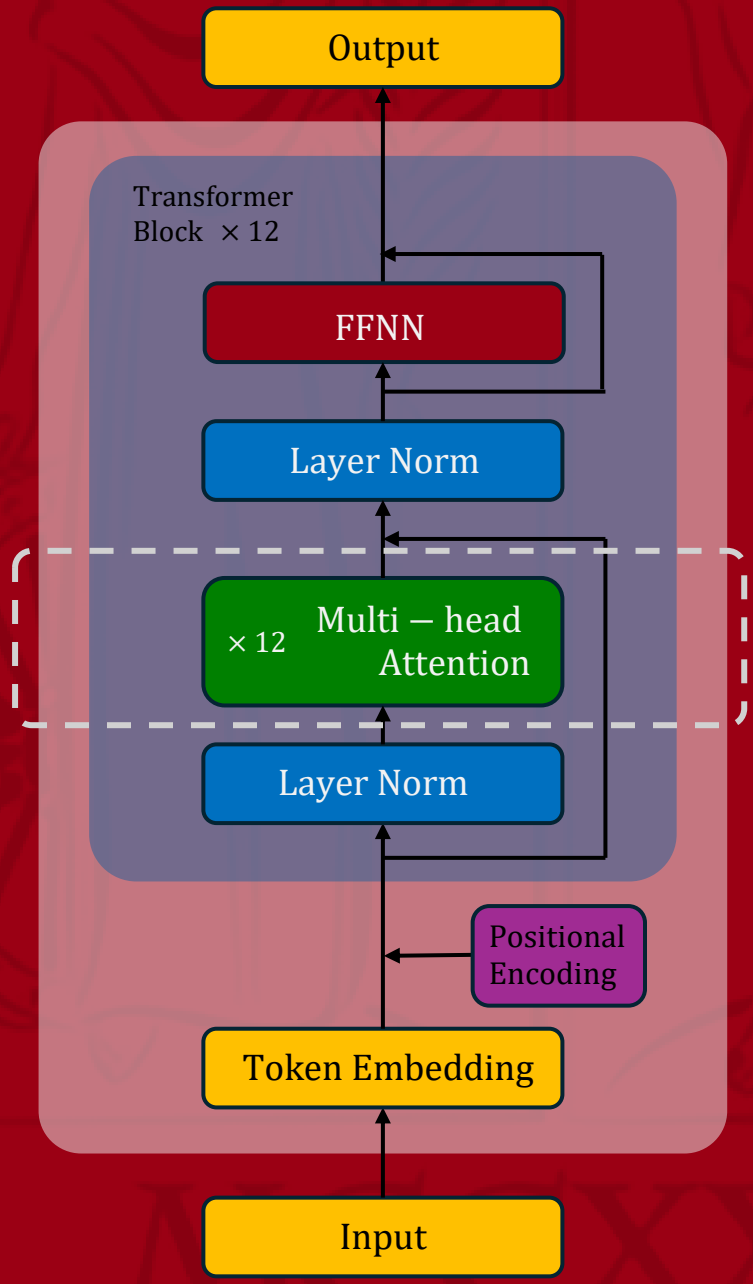
Before PE — After PE



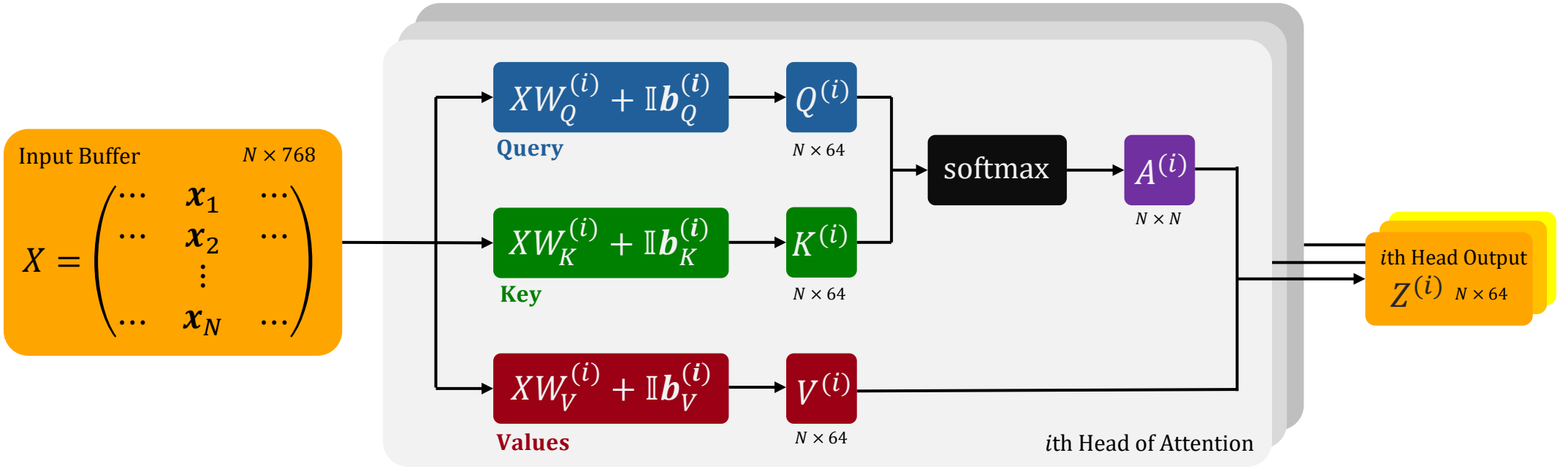
After PE — After LN

Part 2

MULTI-HEAD ATTENTION GEOMETRY



Part 2: GPT-2 Multi head attention



Input Buffer: X $(N \times 768)$

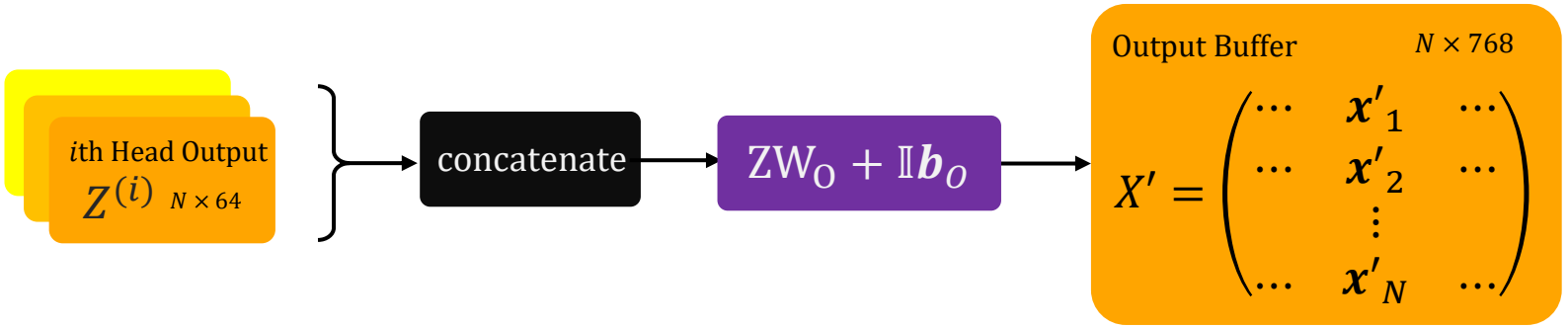
Weight matrix: W (768×64)

Bias vector: $\boldsymbol{b}$ (1×64)

Attention score matrix: A $(N \times N)$

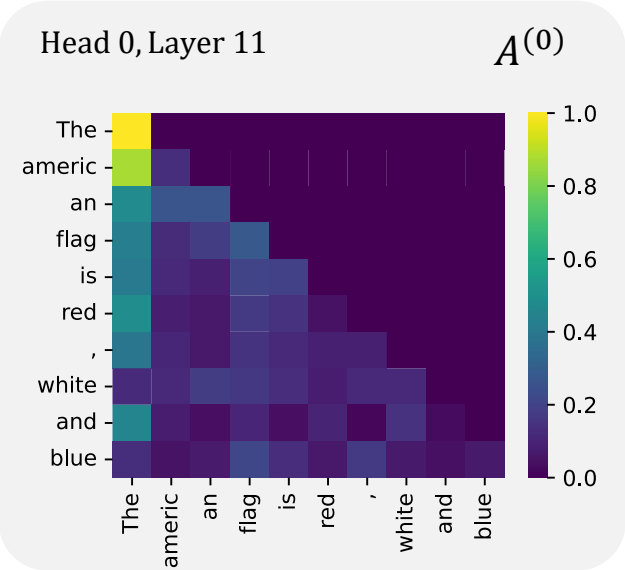
Output matrix: W_O (768×768)

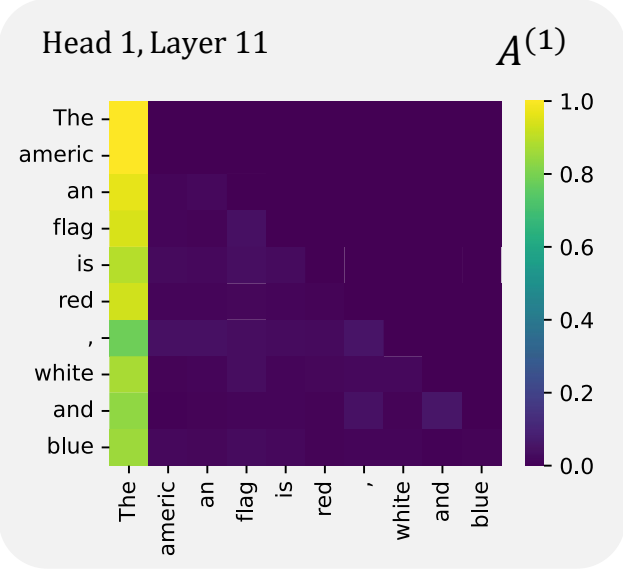
Output bias: $\boldsymbol{b}_O$ (1×768)



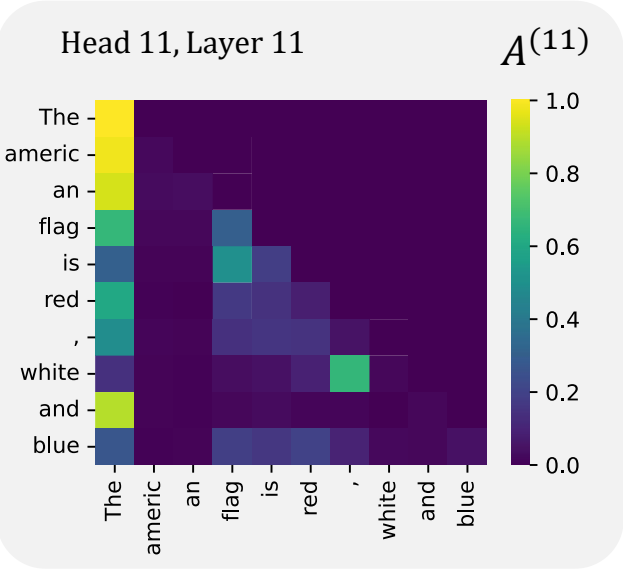
Part 2: Attention score matrix

A common-sense assumption is that, during training, each attention head becomes specialized in capturing certain aspects or features of the input tokens. These features could relate to syntactic structure, semantic roles, positional patterns, or other latent properties of language



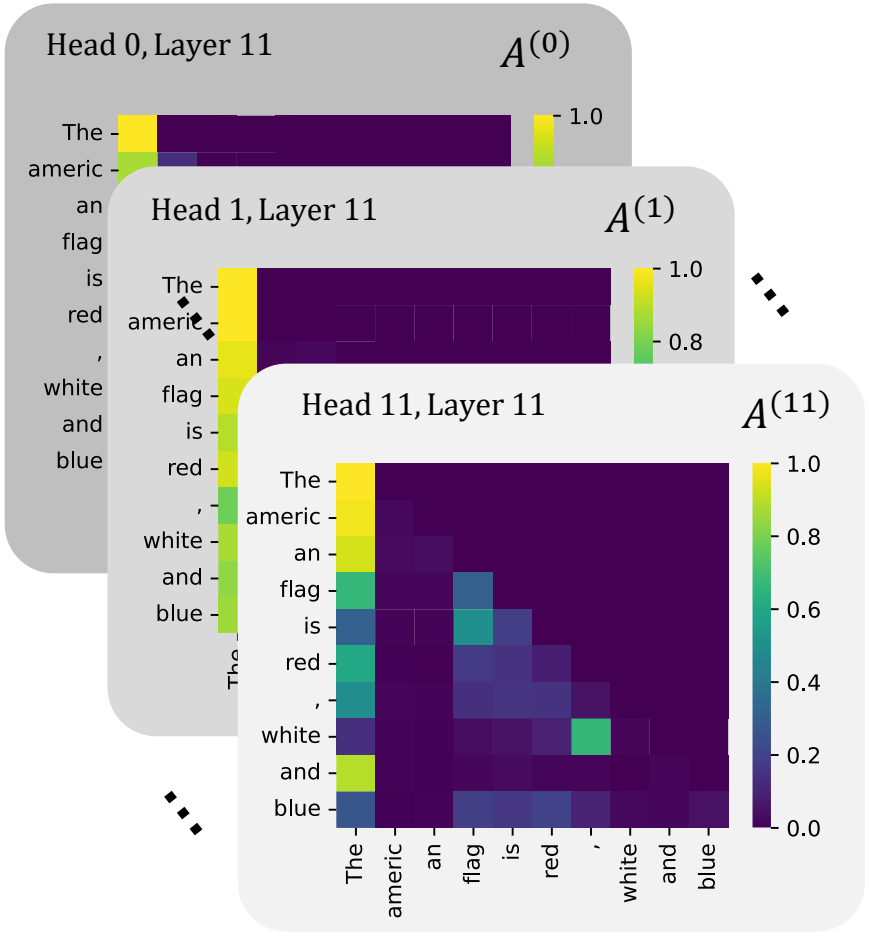


...



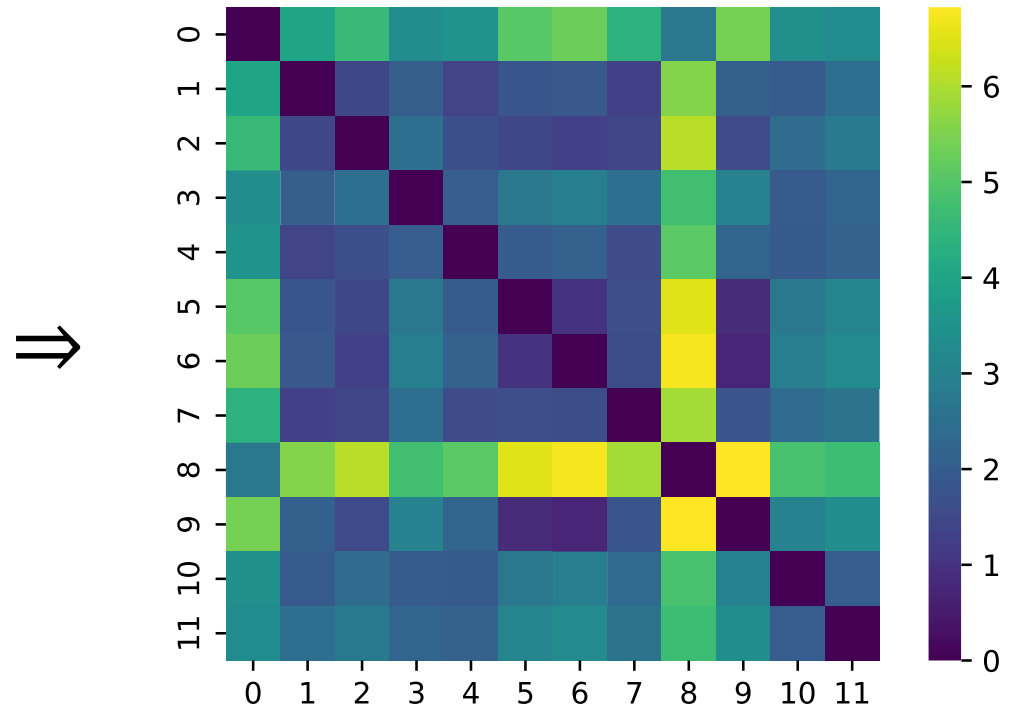
Some heads might learn to focus on similar patterns, resulting in overlapping attention behaviors, while others may diverge and capture entirely different features

Part 2: Attention score matrix



For each layer l and for each pair of heads i and j, we can empirically compute the difference between heads using the Frobenius norm:

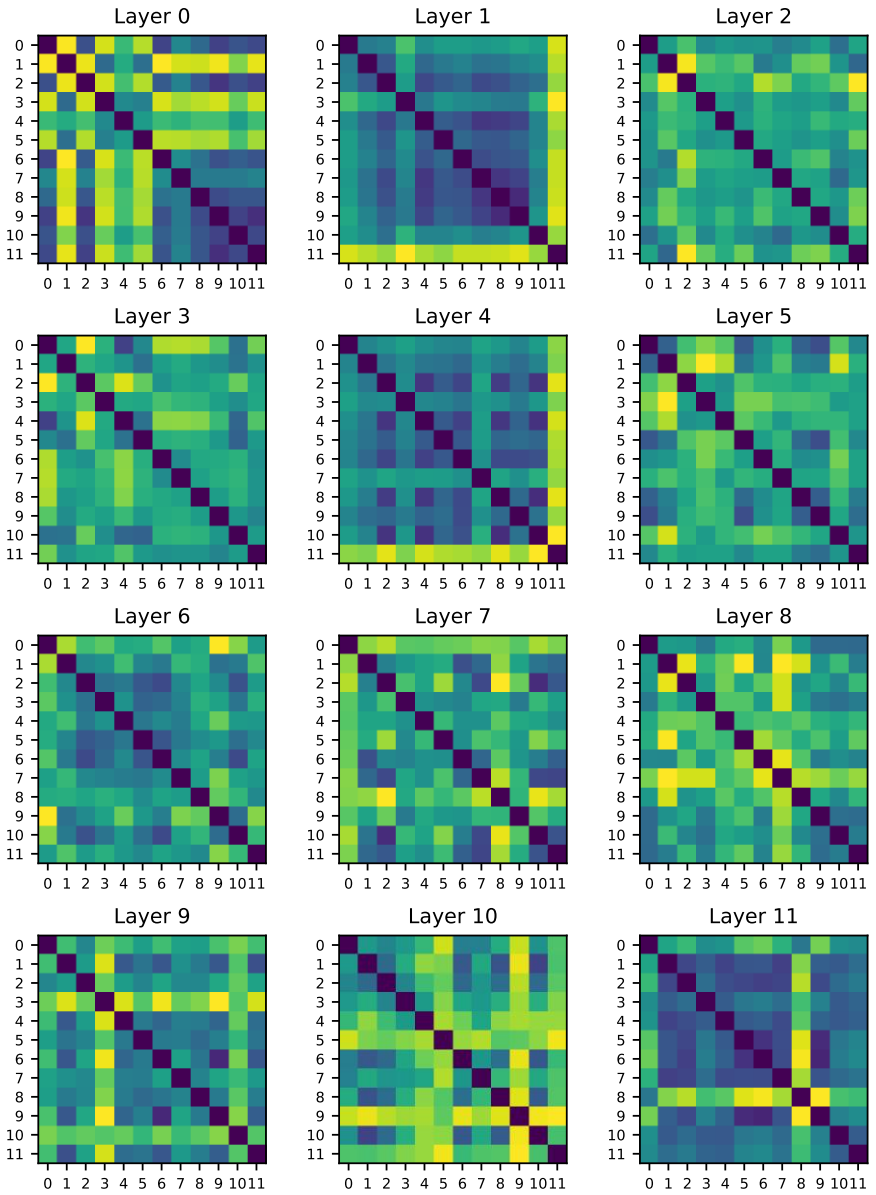
$$\Delta_l^{ij} = \left\| A_l^{(i)} - A_l^{(j)} \right\|$$



Layer 11 – average on 1278 tokens

*Heads **5**, **6**, and **9** attends **similar feature**. In contrast, head **8** shows a distinctly **different pattern**, suggesting it is specialized in capturing features that are independent from those used by the other heads.*

Part 2: Attention score matrix



These are empirical results for each layer of GPT-2.

Can we describe the attention mechanism using purely geometric intuition? Can we reproduce the observed similarities between heads without relying on any token input, using only the learned weights?

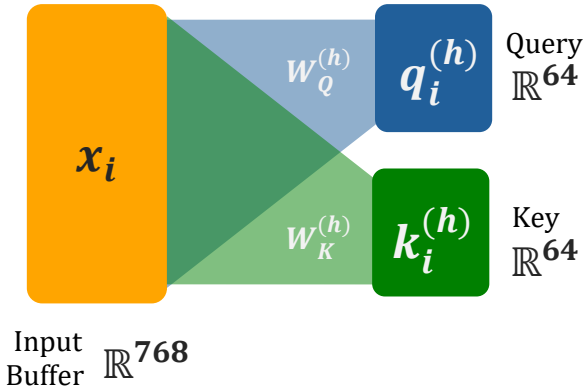
GOAL: understand how attention heads relate to one another in a token-agnostic way, by analyzing the geometric structure of their learned subspaces, without requiring actual input data.

Part 2: Bilinear form formulation

Tokens are embedded in $x_i \in \mathbb{R}^{768}$ and then they are projected using W_Q, W_K in $q, k \in \mathbb{R}^{64}$

The element ij of the attention score matrix can be written as:

$$A_{ij}^{(h)}(x_i, x_j) \propto q_i^{(h)} \cdot k_j^{(h)}$$



$$q_i^{(h)} = x_i W_Q^{(h)} + b_Q^{(h)}$$

$$k_i^{(h)} = x_i W_K^{(h)} + b_K^{(h)}$$

For a specific head h we aim to express the attention matrix A as a bilinea form:

$$A_{ij} \propto \tilde{x}^T J \tilde{y} = B(\tilde{x}, \tilde{y}) \qquad \text{where } J = \begin{pmatrix} W_Q W_K^T & W_Q b_K \\ b_Q W_K^T & b_Q^T b_K \end{pmatrix}, \tilde{x} = \begin{pmatrix} x_i \\ 1 \end{pmatrix}, \tilde{y} = \begin{pmatrix} x_j \\ 1 \end{pmatrix}$$

This bilinear form, however does not introduce an inner product, as the matrix J is generally neither symmetric nor positive definite, furthermore is a low rank matrix with rank r

Part 2: Bilinear form formulation

To interpret the bilinear form in a way that resembles an inner product, we can perform the singular value decomposition

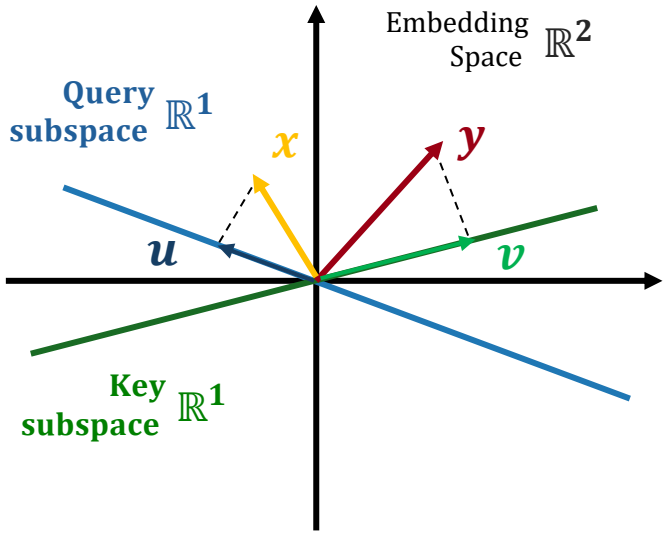
$$B(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}) = \tilde{\boldsymbol{x}}^T U \Sigma V^T \tilde{\boldsymbol{y}} = \langle U^T \tilde{\boldsymbol{x}}, V^T \tilde{\boldsymbol{y}} \rangle_\Sigma = \langle \boldsymbol{u}, \boldsymbol{v} \rangle_\Sigma$$

This expression can be interpreted as a generalized inner product between two transformed vectors u and v, each living in different coordinate systems defined by the linear transformation:

$$\pi_U : U^T \boldsymbol{x} \mapsto \boldsymbol{u}$$

$$\pi_V : V^T \boldsymbol{y} \mapsto \boldsymbol{v}$$

Each attention head maps token pairs **from the original embedding** space into distinct **query and key subspaces**, and then performs a dot product weighted by* Σ



**note that this transformation is not a projection in the strict sense, but rather a submersion from $\mathbb{R}^n$ to $\mathbb{R}^r$*

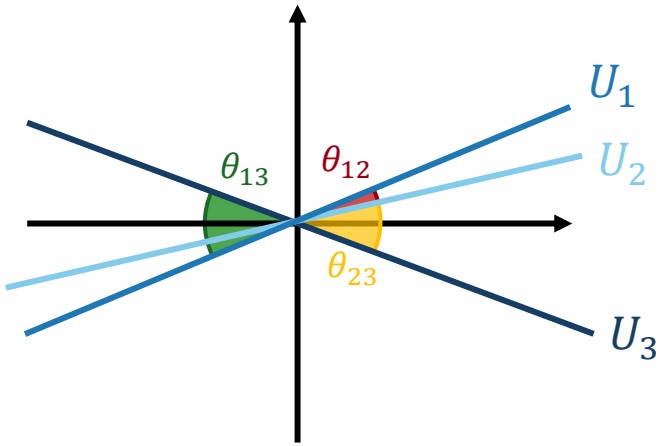
U and V are $n \times r$ matrices
Σ is a $r \times r$ diagonal matrix
$(\mathrm{n} = 768, \mathrm{r} = 64)$

Part 2: Distance between heads

We want to characterize the bilinear form associated with each attention head using a unified mathematical framework

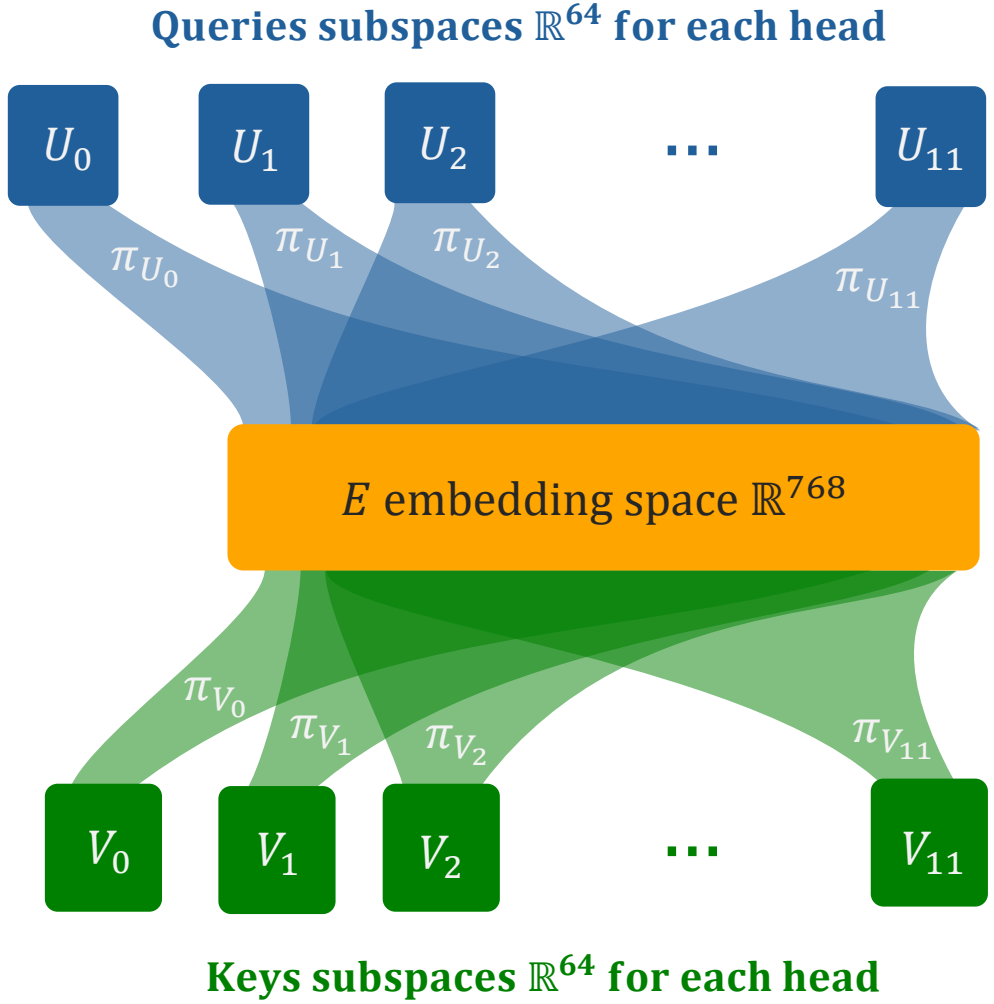
U and V consists of r linearly independent vectors in $\mathbb{R}^n$, and thus defines a Grassmann manifold $\mathrm{Gr}_r(\mathbb{R}^n)$. We can compare the bilinear forms B_i, B_j across different heads using the Grassmann distance.

$$\boxed{d(B_i, B_j)^2 = d(U_i, U_j)^2 + d(V_i, V_j)^2} \qquad \text{where } d(U_i, U_j) = \left(\sum_{l=1}^{r} \theta_l^2\right)^{1/2}$$



The Grassmann distance is the geodesic between the subspaces spanned by different basis and is based on the principal angles $\{\theta_1, \theta_2, \ldots, \theta_r\}$. The distance is zero if the subsapces U_i and U_j are identical, and is $\sqrt{\pi} r/2$ if the subspaces are fully orthogonal

Part 2: Distance between heads

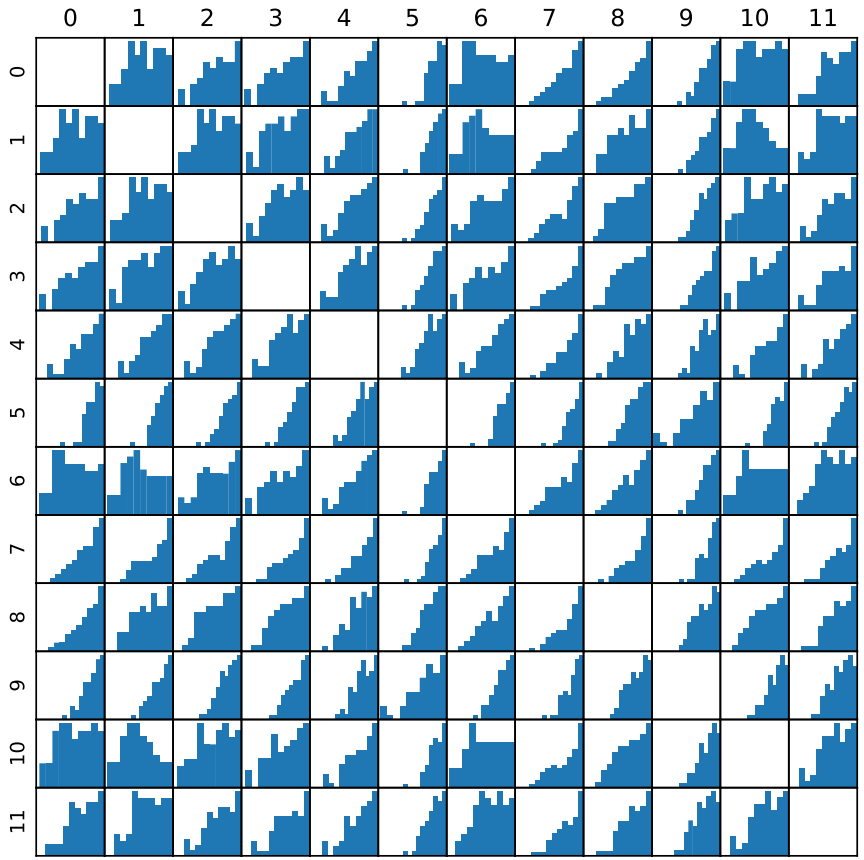


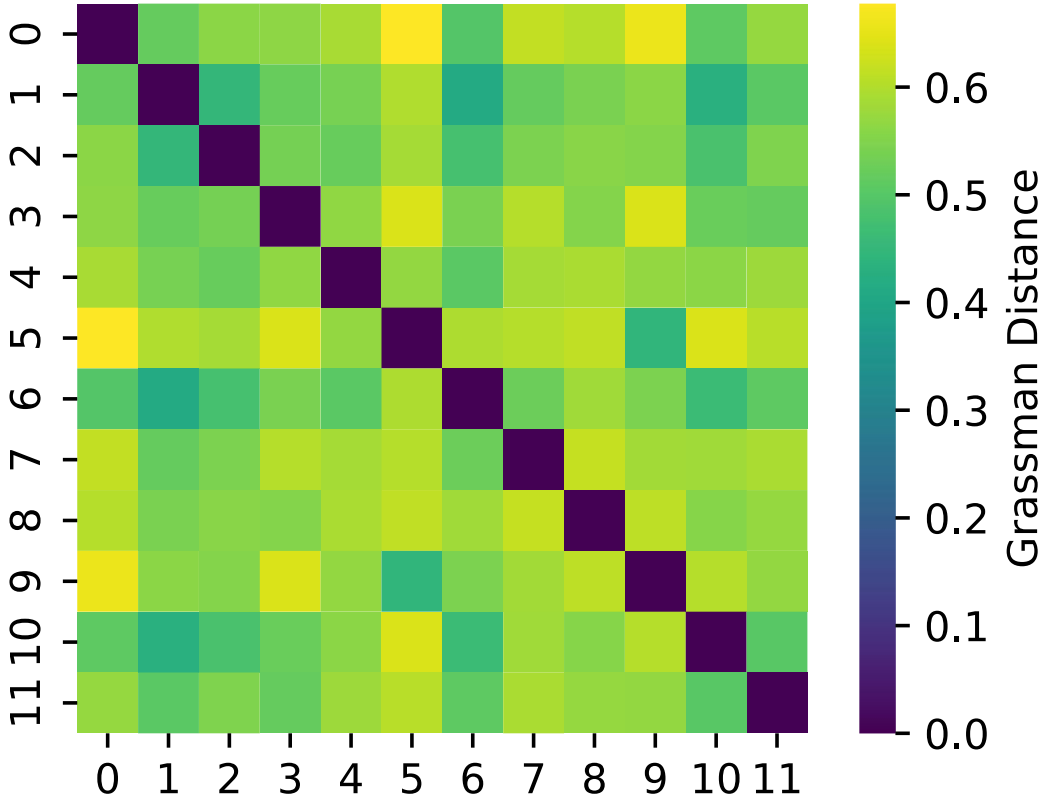
We can interpret each layer as decomposing the full embedding space into a set of lower-dimensional subspaces, defined by linear maps π_{U_i} and π_{V_i}

Since each attention head projects into a subspace of rank $r = 64$, and there are 12 heads per layer, we expect the 768-dimensional embedding space to be partitioned into twelve distinct 64-dimensional subspaces.

We expect minimal overlap (i.e., large distances) between these subspaces, which would suggest that training has been effective: the attention heads are not redundant, but rather specialized, with each focusing on distinct aspects of the input.

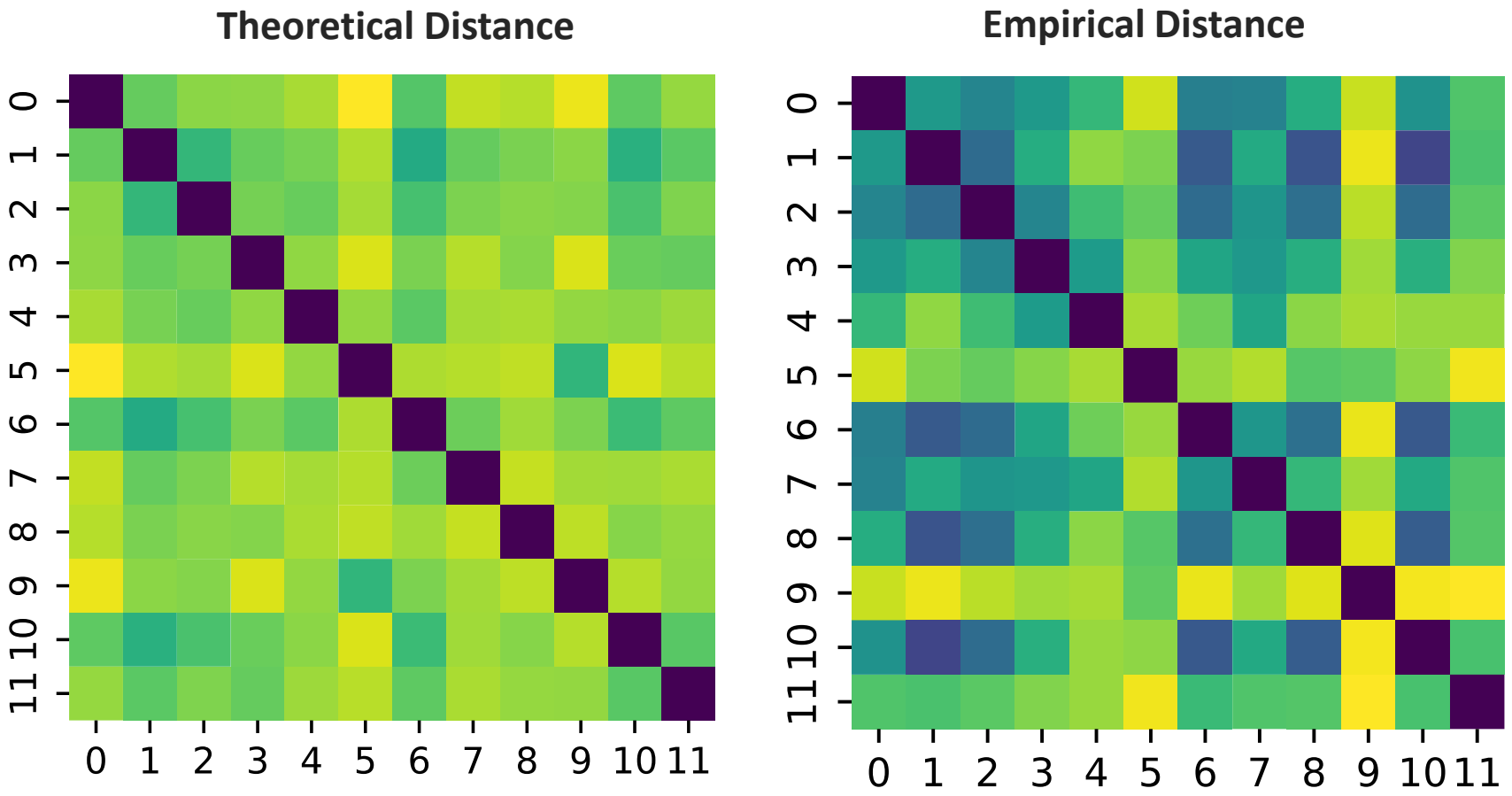
Part 2: Results





Distribution of the principal angles and total Grassmann distances between heads in layer 10. Larger distances correspond to pairs of heads whose subspaces are mostly orthogonal, whereas heads that frequently 'fire together' (i.e., exhibit similar attention behavior) tend to have more aligned subspaces, reflected by smaller distances.

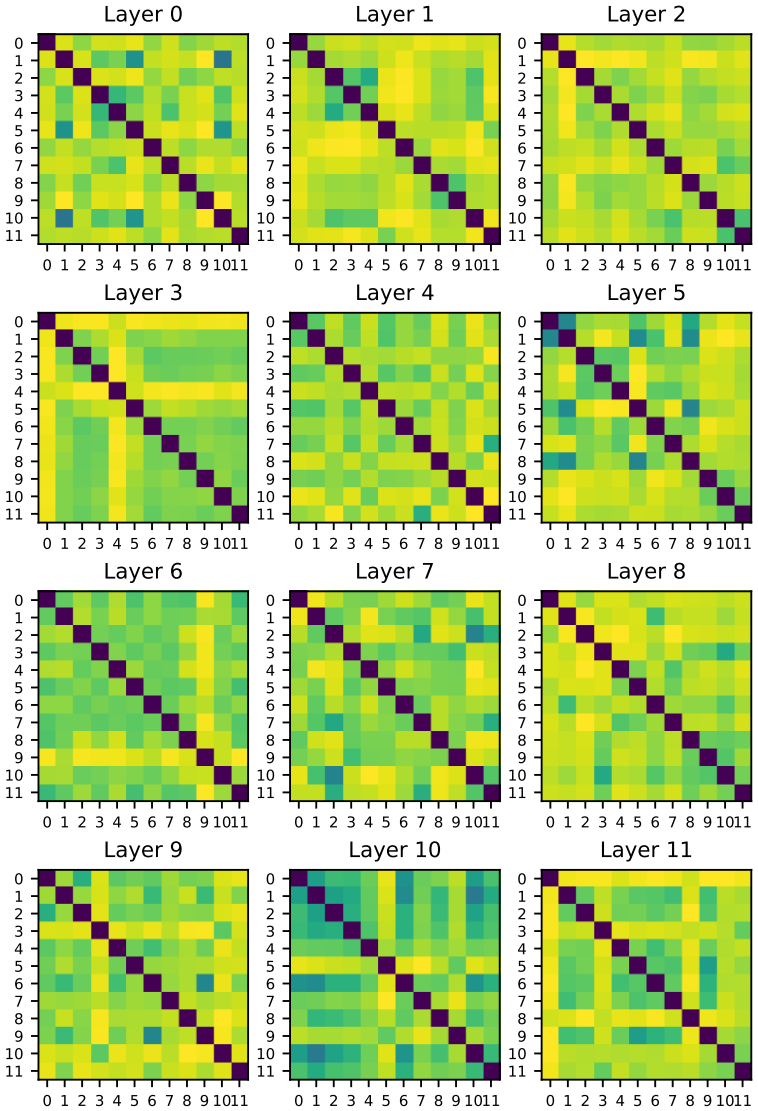
Part 2: Comparison with the empirical results



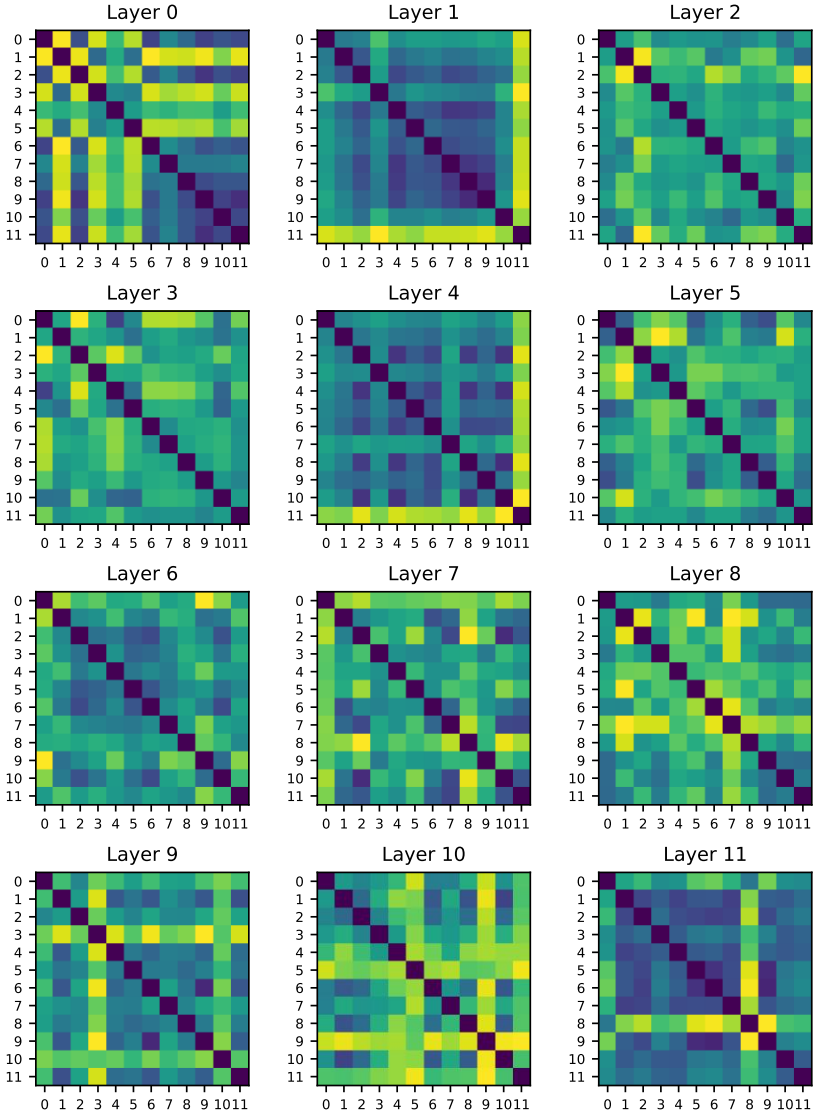
The left heatmap shows the theoretical distances, computed as the sum of Grassmann distances between the query and key subspaces for each pair of attention heads across all layers. The right plot displays empirical distances, obtained by averaging the differences between attention score matrices across 1278 tokens from a test set. The comparison highlights how geometric similarity between subspaces aligns with actual attention behavior.

Part 2: Comparison with the empirical results

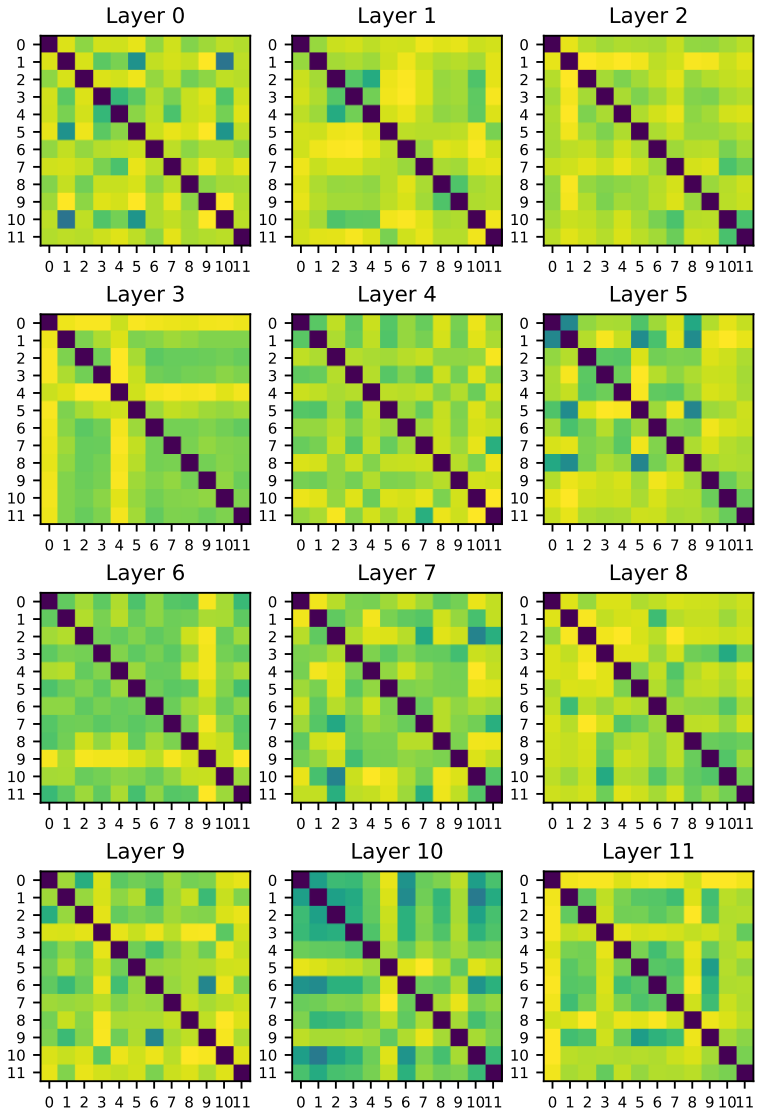
Theoretical Distance



Empirical Distance



Part 2: Conclusion



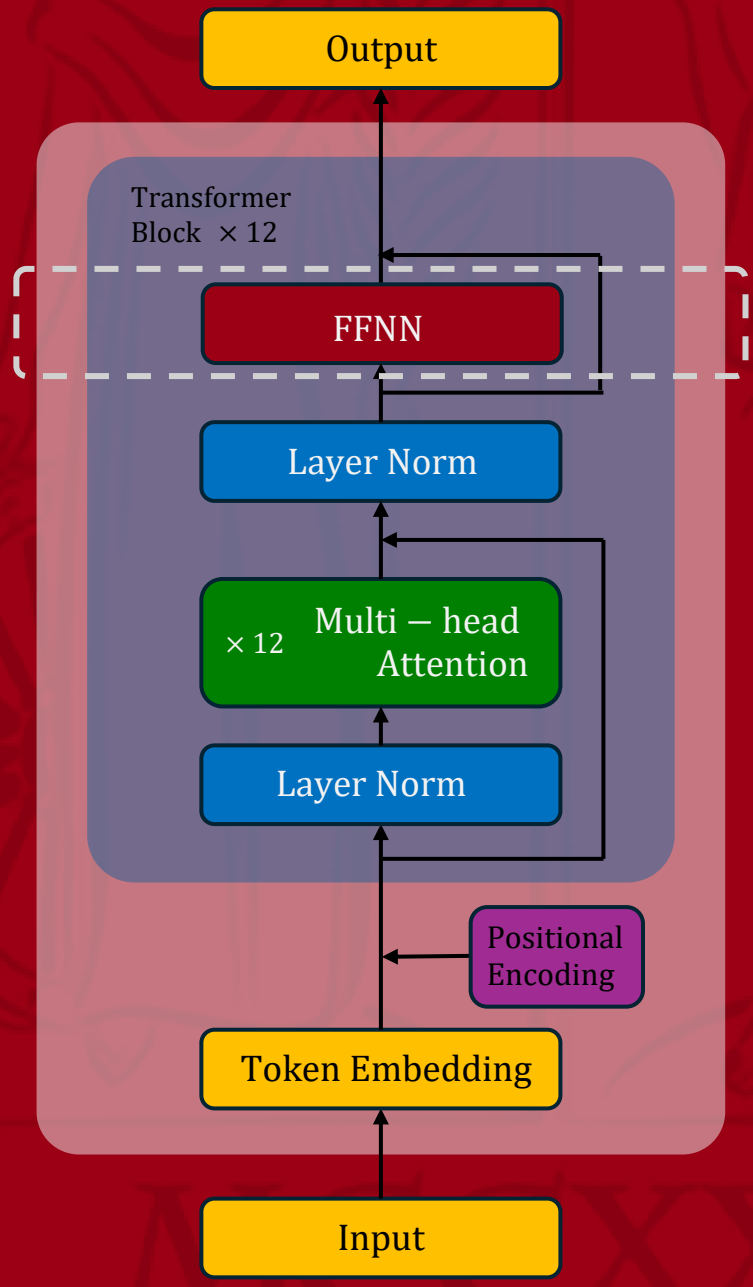
Despite the simplicity of our geometric model, the **alignment between the token-agnostic interpretation and the empirical results** from actual token inputs is quite remarkable, especially in the deeper layers of GPT-2. This suggests that the learned attention structure could be meaningfully understood through subspace geometry.

There is still room for deeper exploration. Our current approach simplifies several aspects, for instance we **excluded the role of singular values Σ** in the interaction and **combined the query and key subspaces U and V in a naïve way**.

An insightful direction for future work is to explore the **null spaces** of the attention heads. Since null space components are effectively ignored by the attention mechanism, injecting tokens with controlled **null-space perturbations** could reveal how different heads filter information

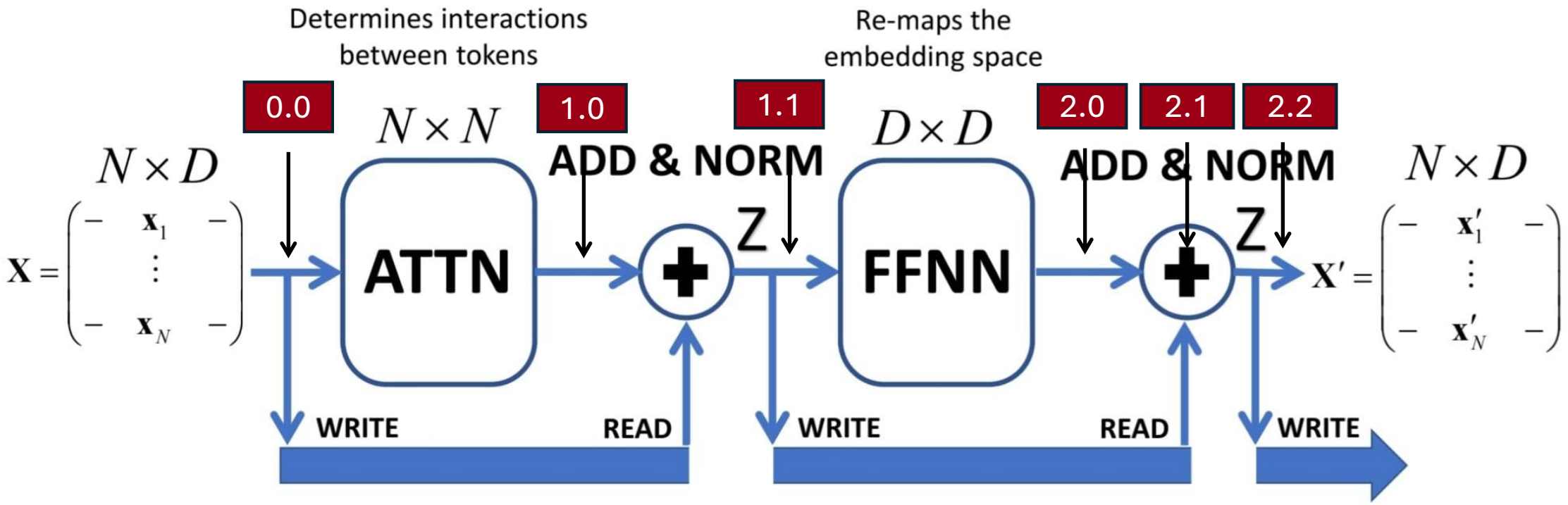
Part 3

FEED-FORWARD NEURAL NETWORK



Residual Connections and Layer Normalization

We tend to think of *residual connections* as "going around sub-modules" in the stack.
Instead, consider the sub-modules as performing "read/write" to a data stream.

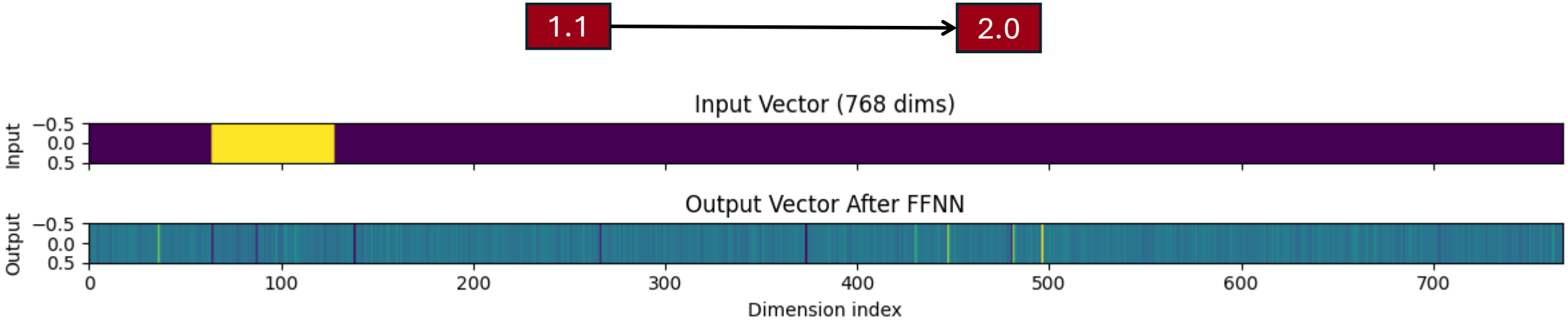


Residual Connections

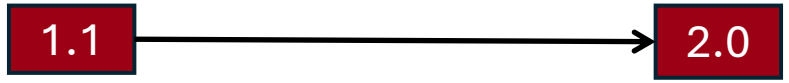
Part 3: FFNN Mapping

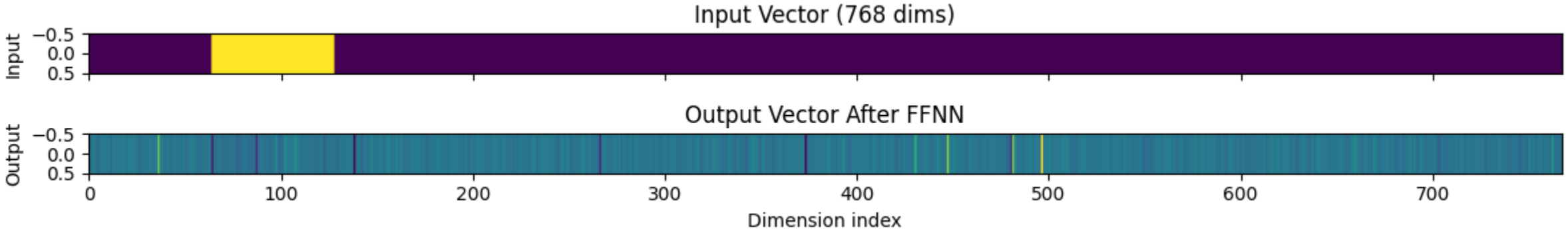
GOAL: We want to gather some insights over the transformation that the buffer undergoes while passing through this block of the transformer.
These phenomenological approach is a consequence of the non linear mapping happening inside the Neural Network.

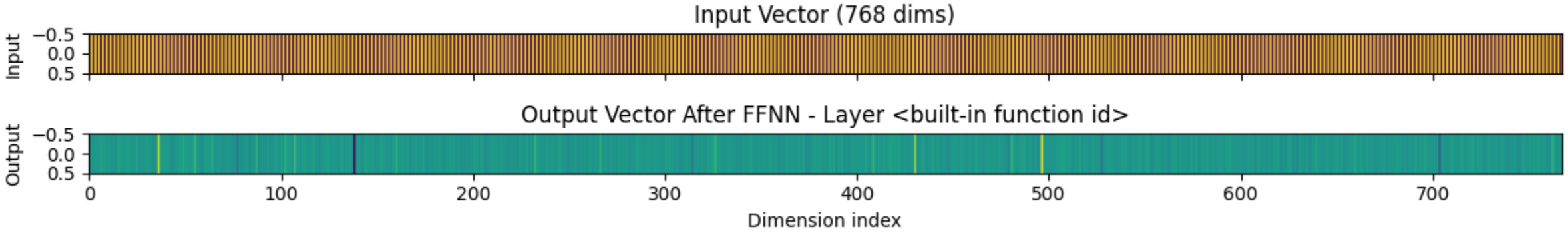
To do so we will study the transformation of artificial buffers through all the stages of the Neural Network.



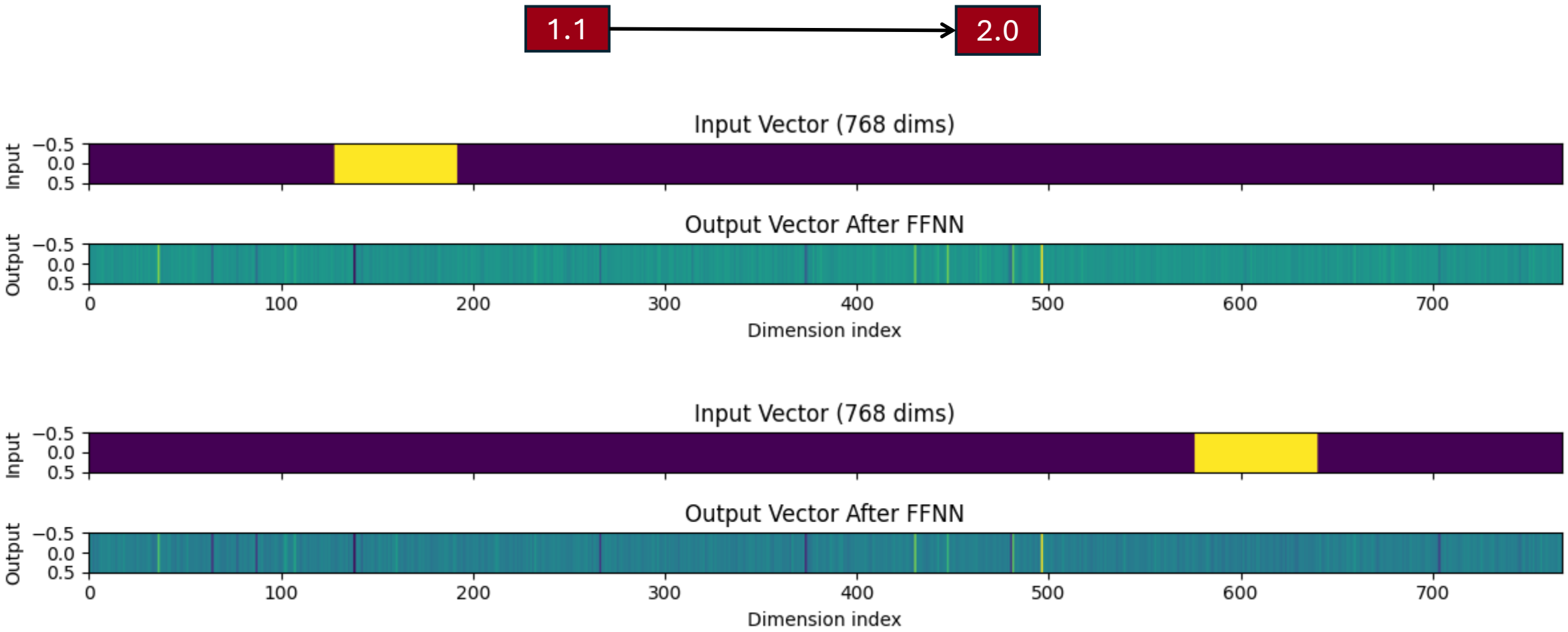
Part 3: Buffer transformation



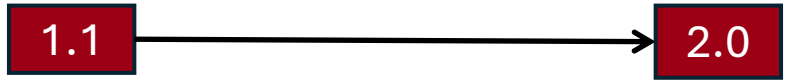


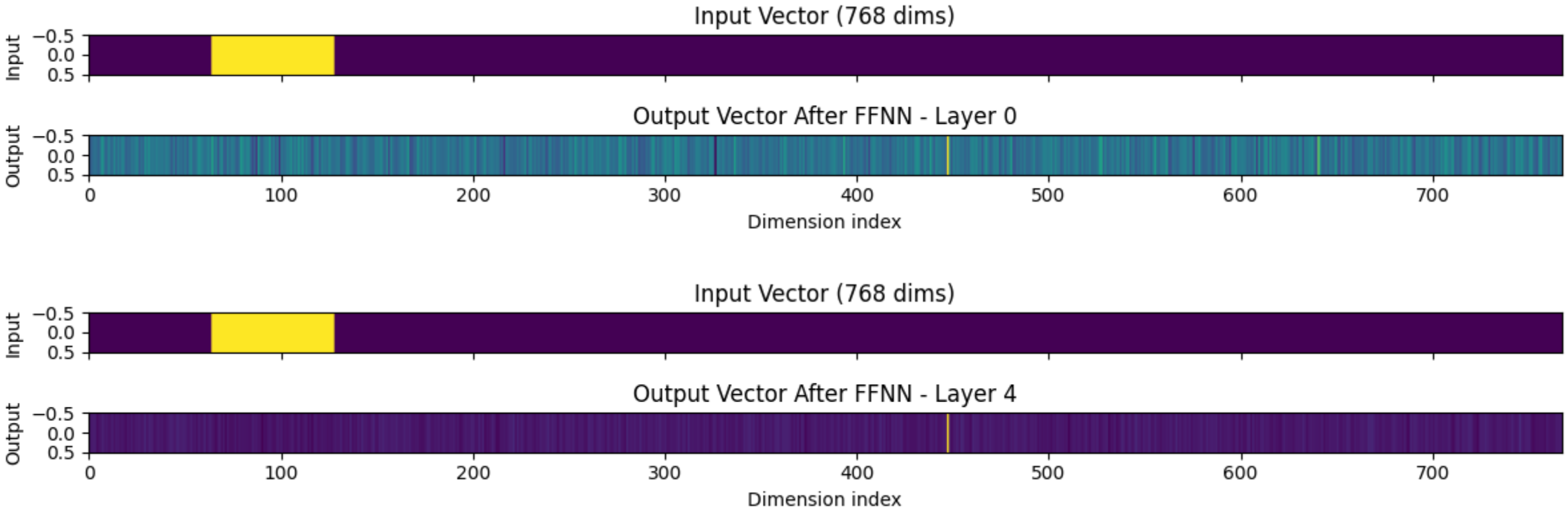


Part 3: Transformations across attention heads

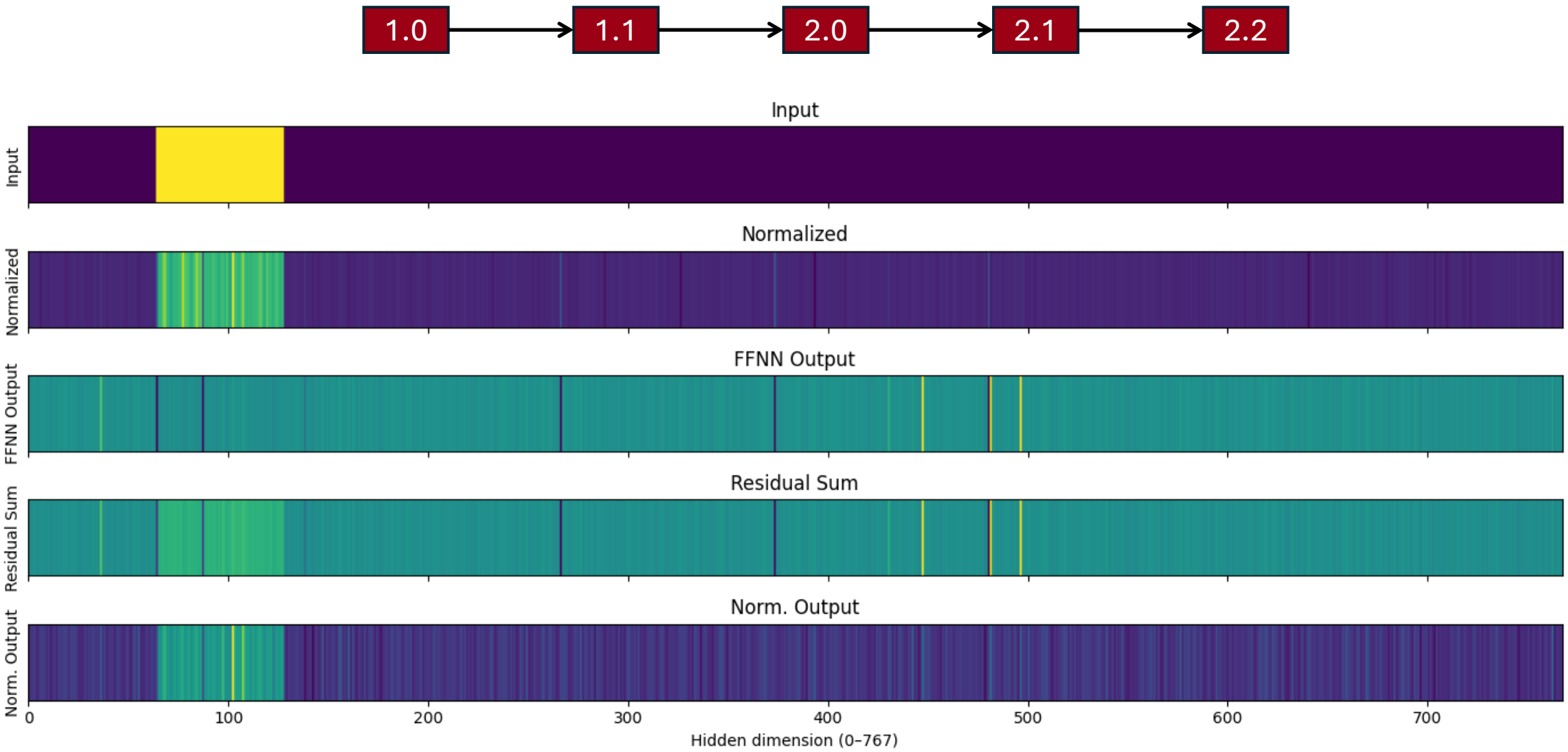


Part 3: Transformation across different layers



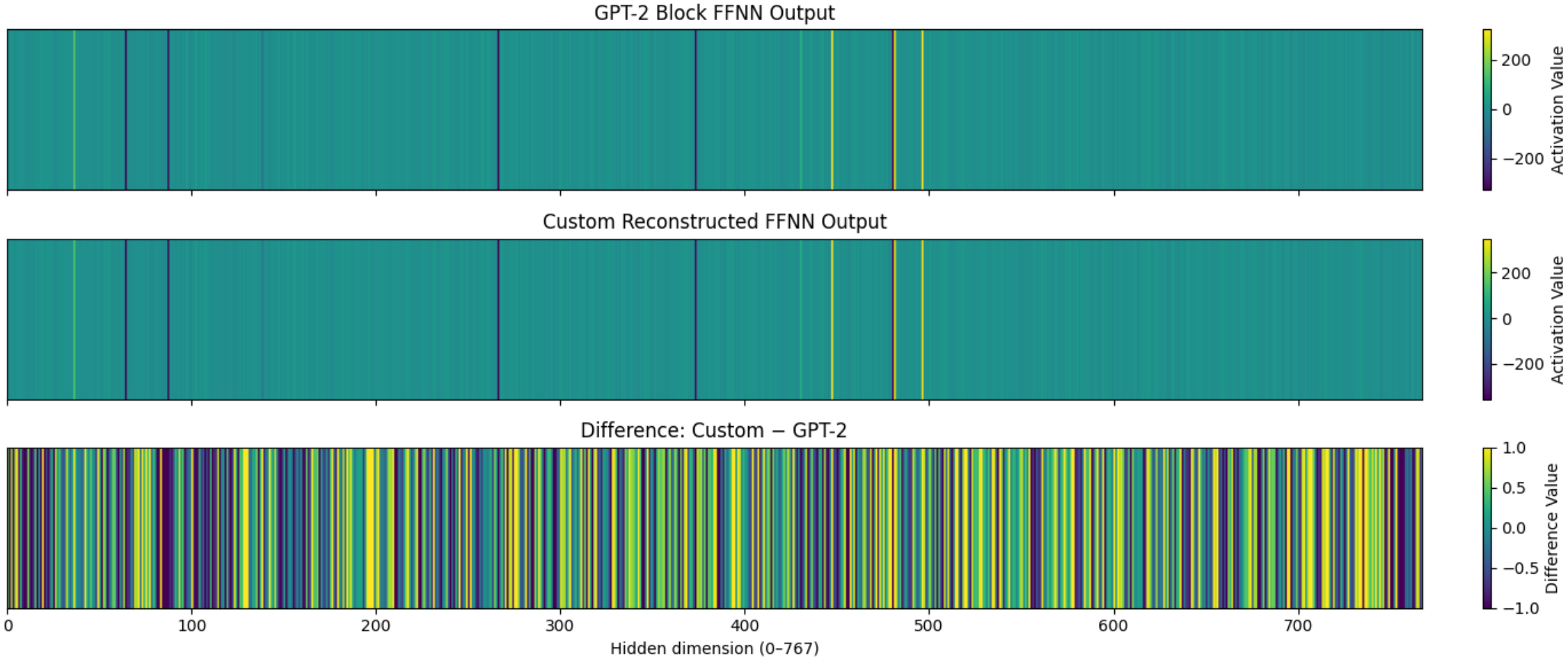


Part 3: Transformation through all steps

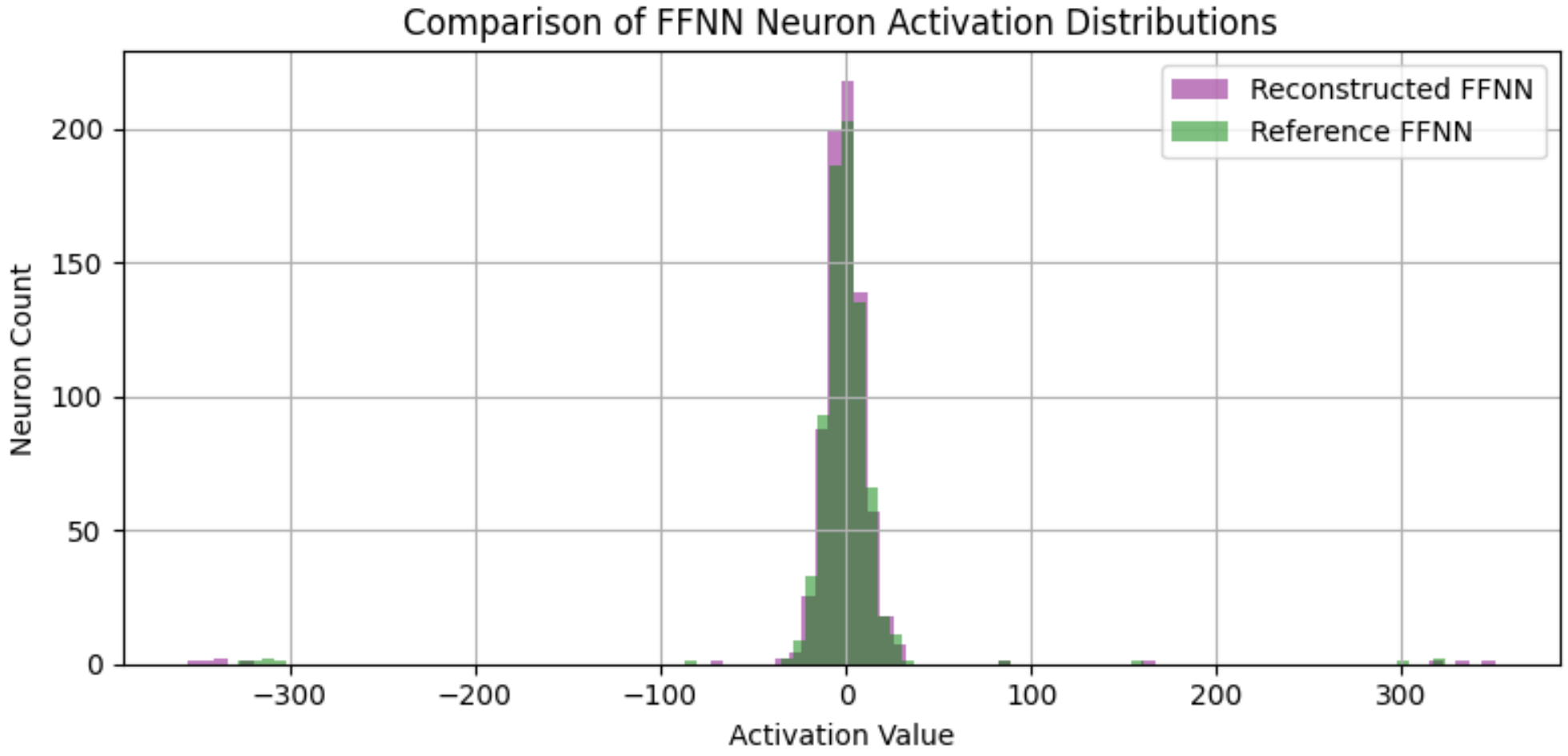


Part 3: FFNN Mapping

The goal is now to effectively visualize the effect of bias. To do so we reconstruct manually, using the matrix of weights, and removing the biases.
Afterwards we directly compare the result obtained with the custom FFNN and the original block.



Part 3: Neuron activation distribution



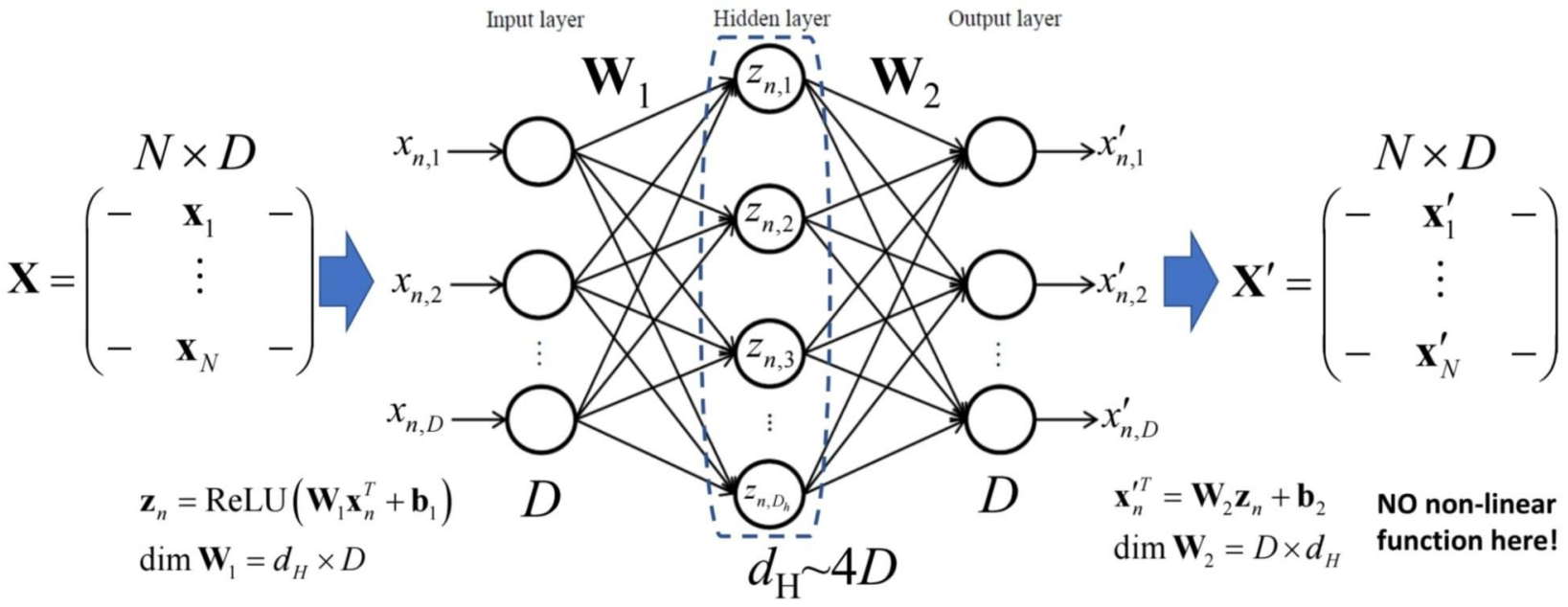
Part 3: Embedding Dimension Analysis

GOAL: We want to study how the feature space gets transformed in the FFNN, understanding how the dimensions are enhanced and compressed.

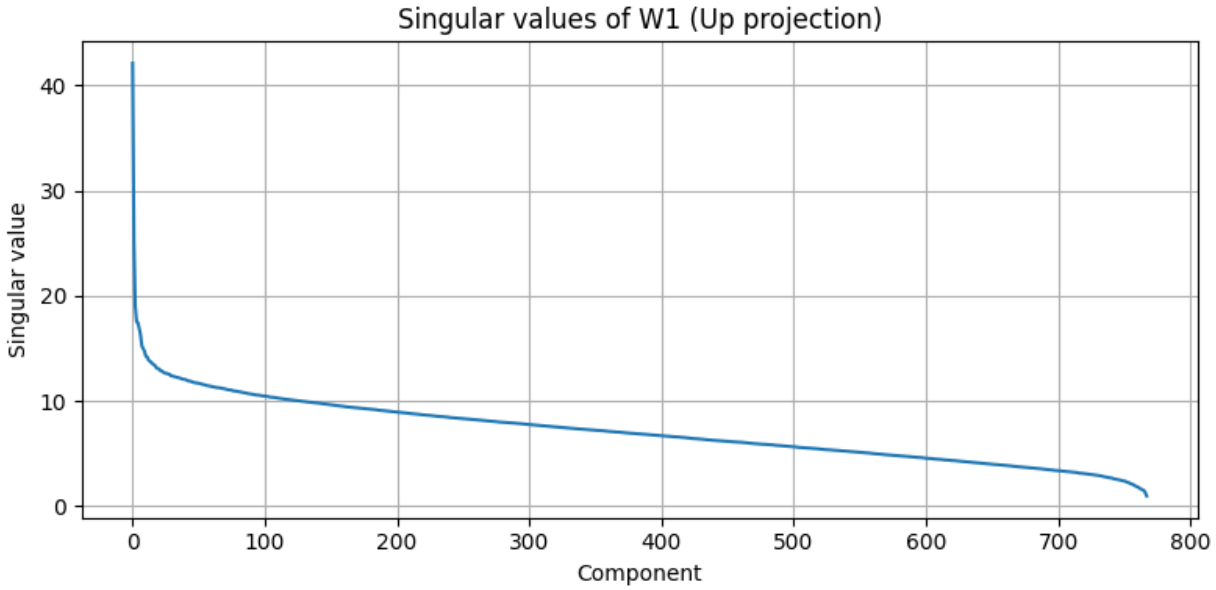
Two-Layer Fully Connected Feed Forward NN

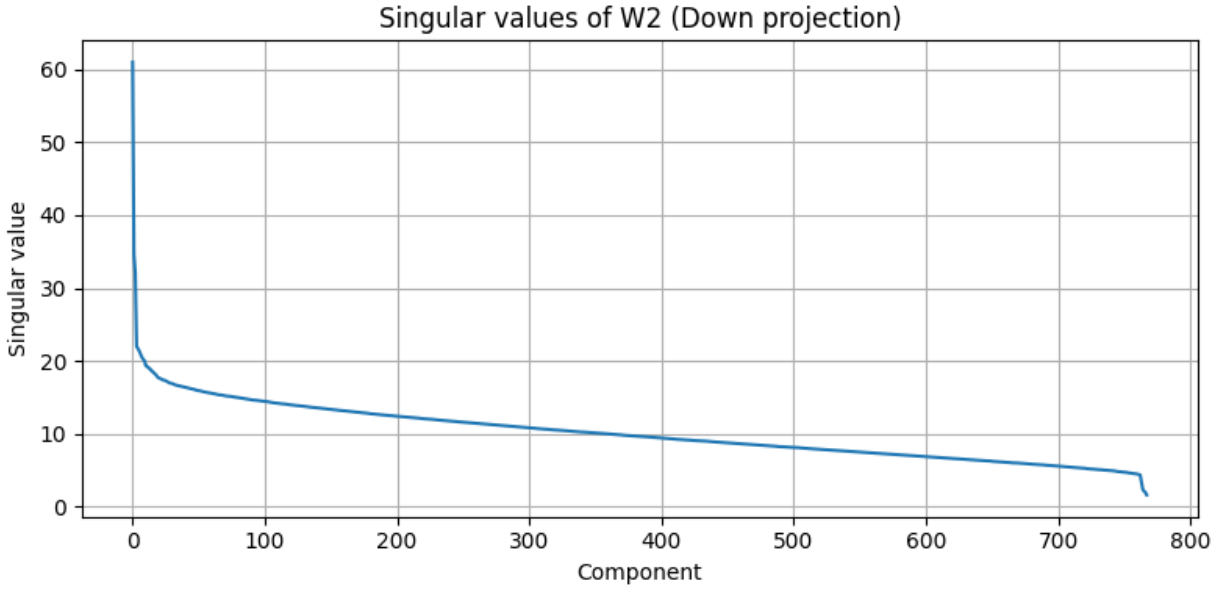
The *FFNN* acts independently on each vector, $\mathbf{x}_n$, of the buffer, $\mathbf{X}$, as if it's a batch of size N!

Learnable parameters: $(D_h+1)\times D + D_h\times(D+1) \approx 8D^2$

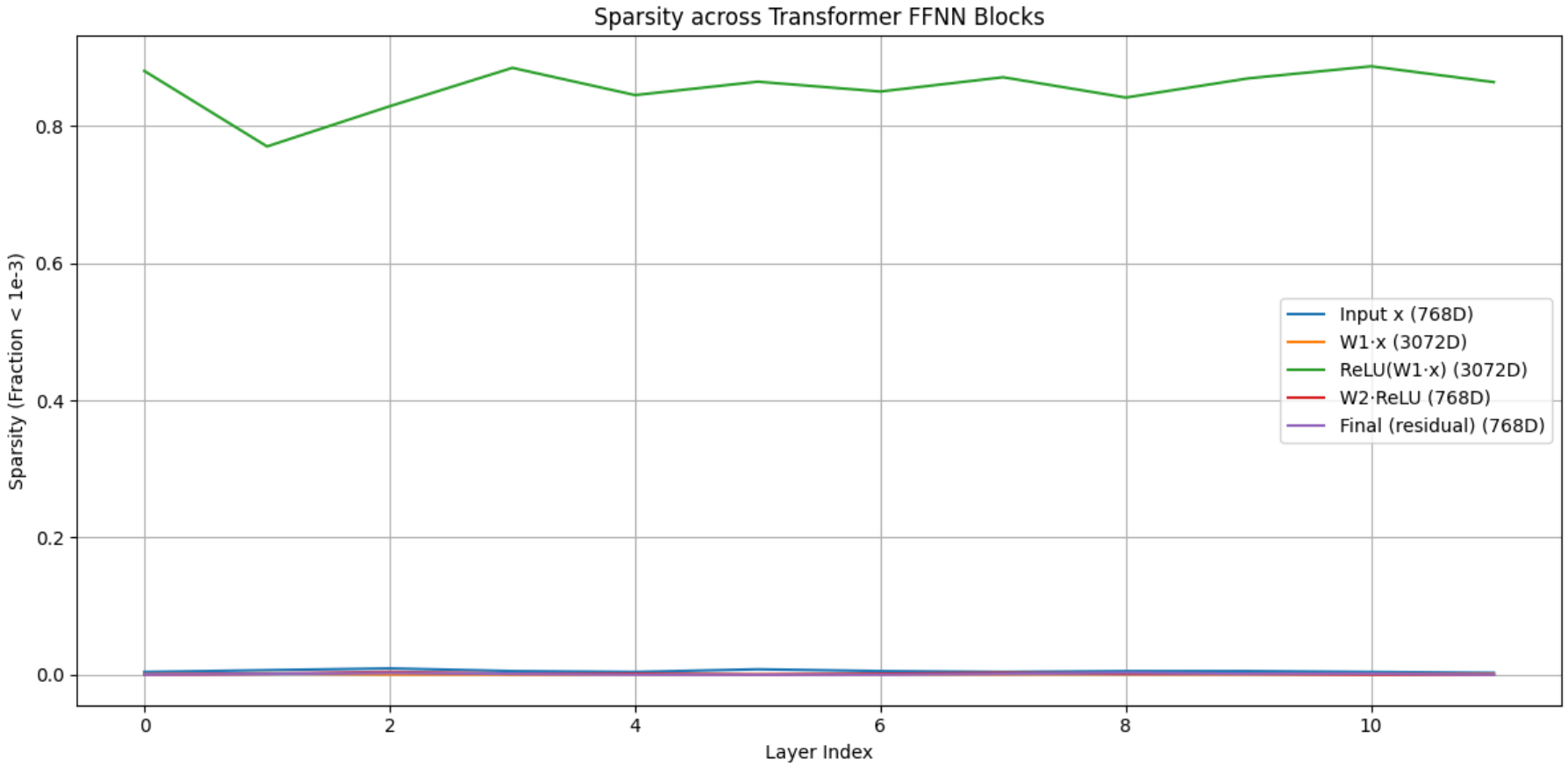


Part 3: SVD


Singular values of W1 (Up projection)
Singular value
Component
0
10
20
30
40
0
100
200
300
400
500
600
700
800


Singular values of W2 (Down projection)
Singular value
Component
0
10
20
30
40
50
60
0
100
200
300
400
500
600
700
800

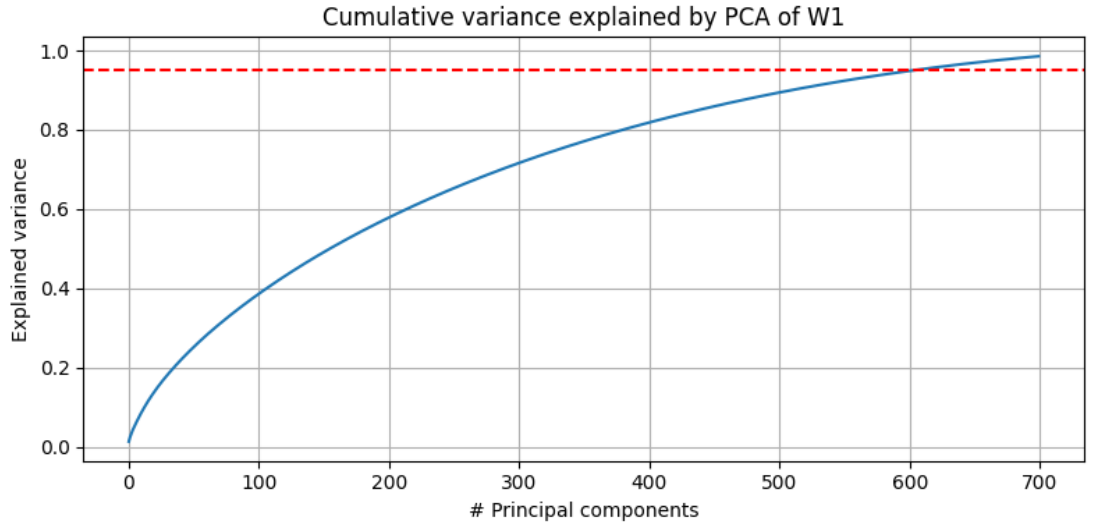
Part 3: Sparsity Analysis



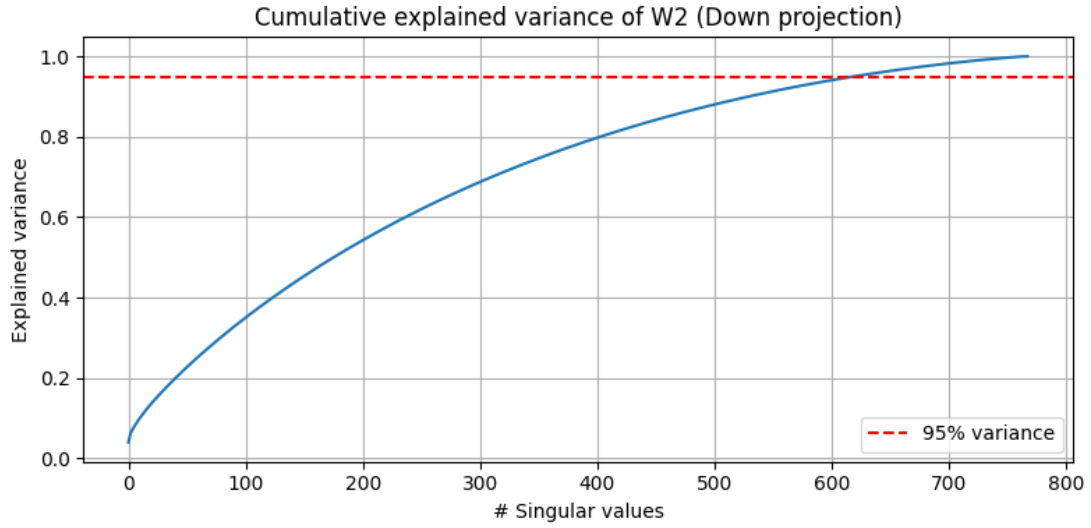
Part 3: Cumulative Variance

PCA, cumulative variance explained:

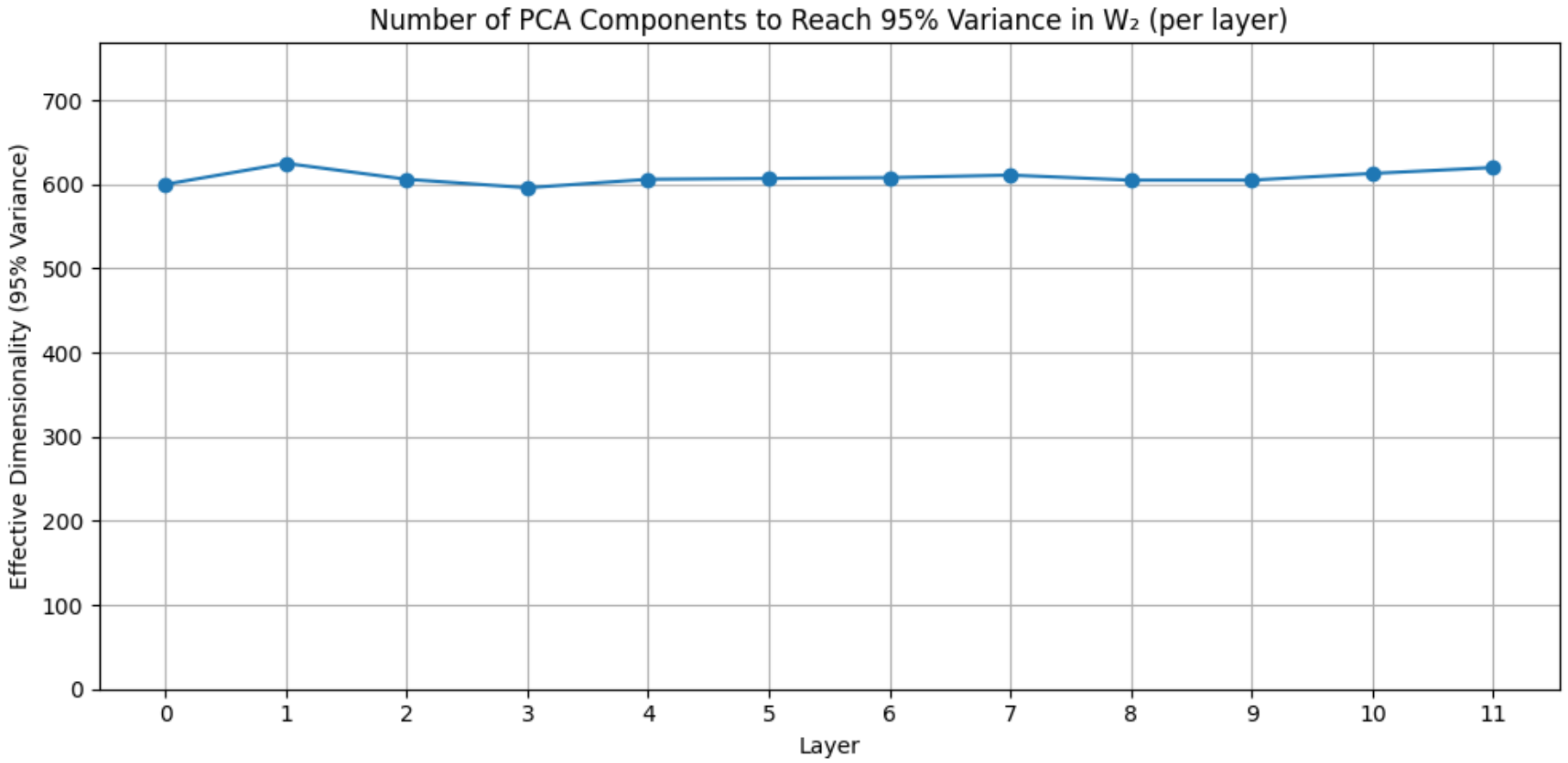
W1 projects into **~604** effective dimensions



W2 compresses to **~620** effective dimensions

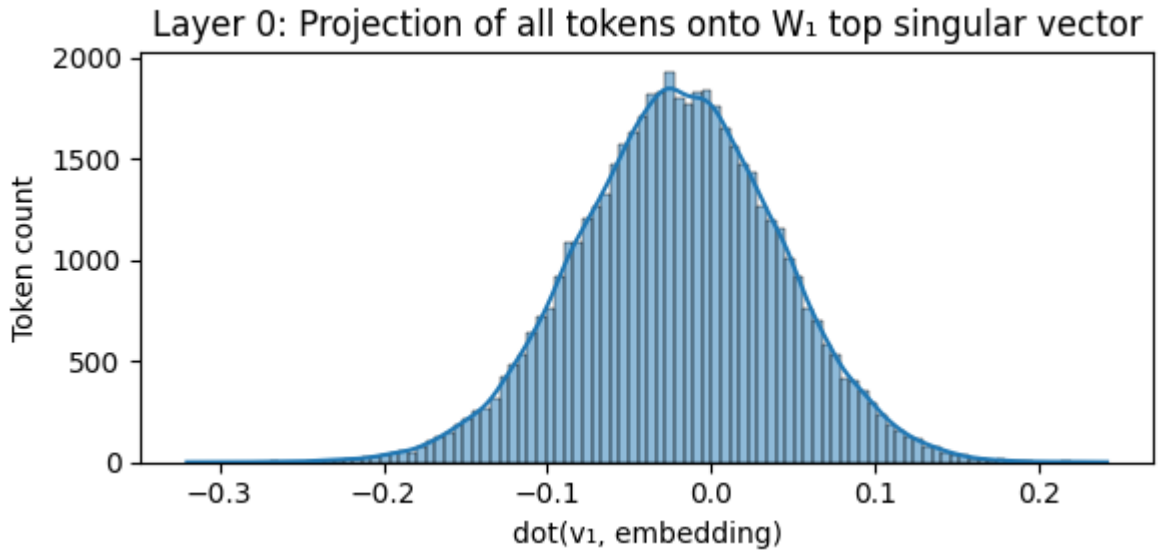


Part 3: Cumulative Variance per layer

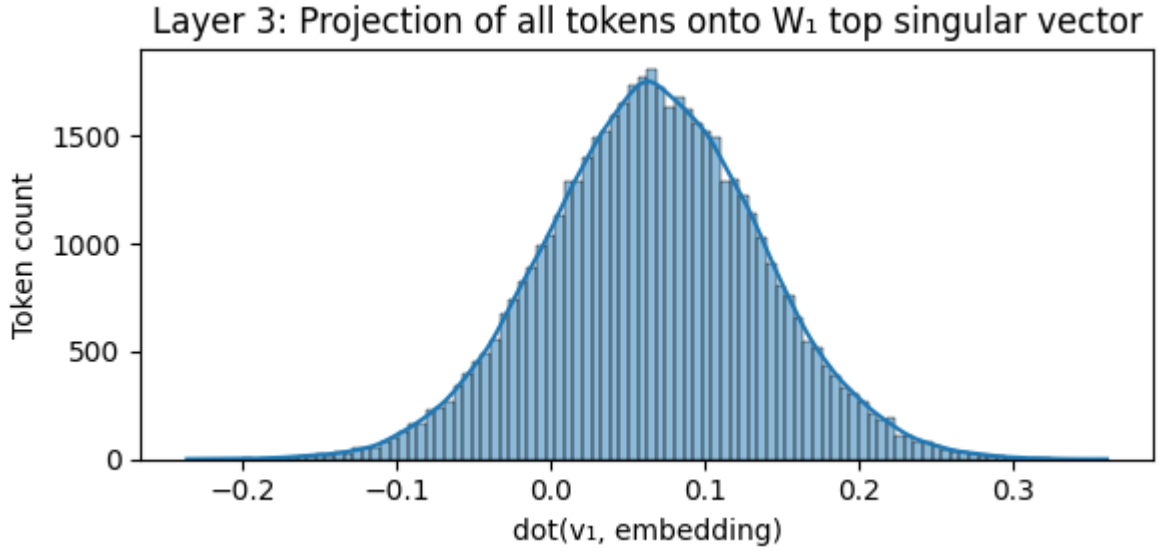


Part 3: Enhanced tokens

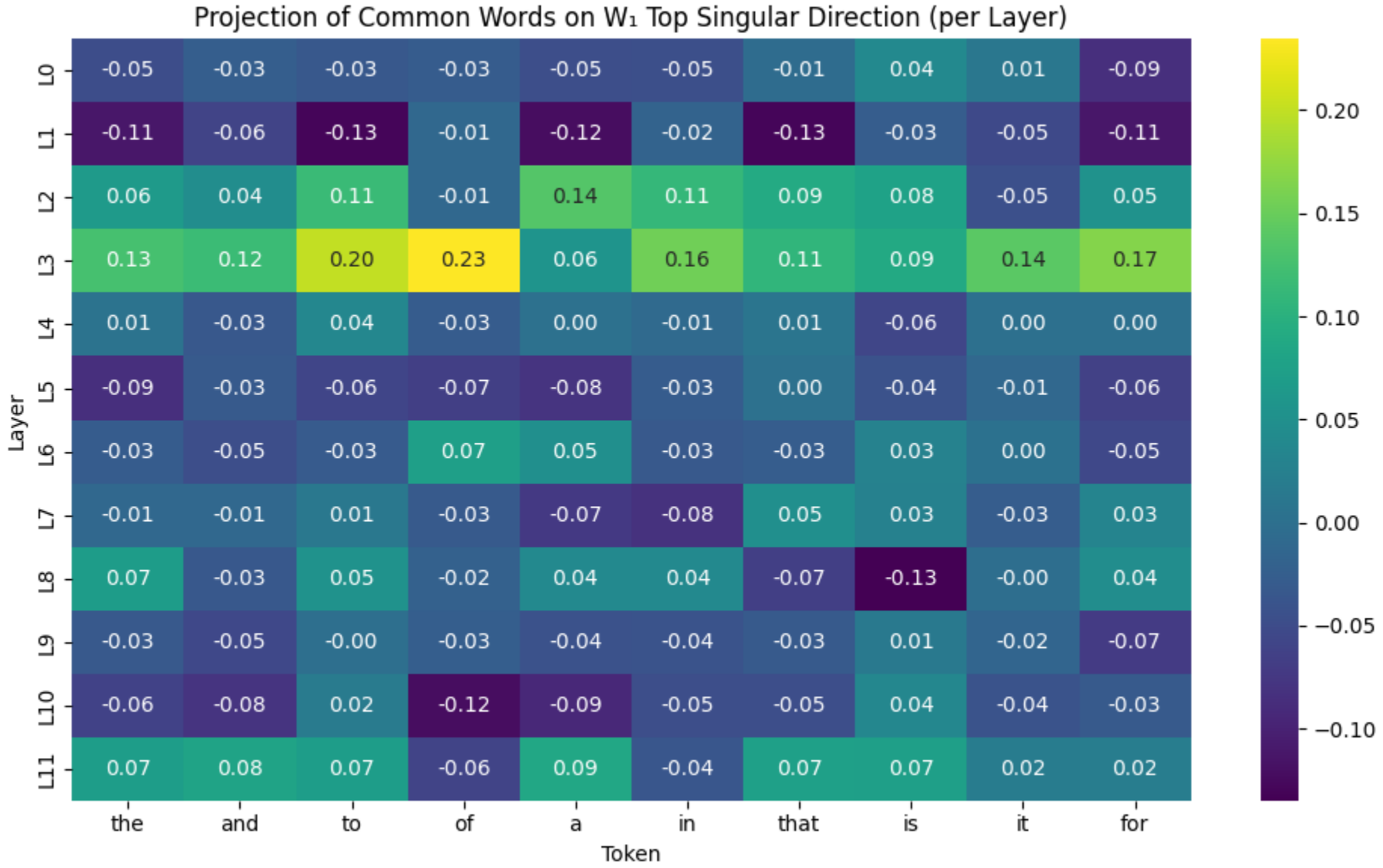
Layer 0 - Tokens most aligned with W1 top singular direction:
'due' (id=23301) → dot = 0.2414
'Untitled' (id=46332) → dot = 0.2279
'yss' (id=33968) → dot = 0.2228
'ousy' (id=41808) → dot = 0.2184
'izable' (id=13821) → dot = 0.2170
'ois' (id=10924) → dot = 0.2165
'rival' (id=43171) → dot = 0.2141
'emis' (id=30561) → dot = 0.2115
' Est' (id=10062) → dot = 0.2055
' semblance' (id=45960) → dot = 0.2028



Layer 3 - Tokens most aligned with W1 top singular direction
'Safe' (id=31511) → dot = 0.3610
' tidy' (id=43044) → dot = 0.3277
'Pg' (id=31743) → dot = 0.3253
'TPS' (id=28820) → dot = 0.3246
' captcha' (id=48972) → dot = 0.3231
'short' (id=19509) → dot = 0.3193
' vines' (id=44439) → dot = 0.3184
'eur' (id=23365) → dot = 0.3176
' anonym' (id=14571) → dot = 0.3172
'Yan' (id=49664) → dot = 0.3149



Part 3: Enhanced tokens




Thanks for The ATTENTION
TRANSFORMERS
RISE OF THE BEASTS