Positional Information Can Emerge Through Causal Attention Making Nearby Token Embeddings Similar Even Without Positional Encodings

Chunsheng Zuo
Department of Computer Science
Johns Hopkins University
czuo3@jh.edu

Pavel Guerzhoy
Department of Mathematics
University of Hawai'i at Mānoa
pavel@math.hawaii.edu

Michael Guerzhoy
Division of Engineering Science
University of Toronto
guerzhoy@cs.toronto.edu

Abstract

Transformers with causal attention can solve tasks that require positional information without using positional encodings. In this work, we propose and investigate a new hypothesis about how positional information can be stored without using explicit positional encoding. We observe that nearby embeddings are more similar to each other than faraway embeddings, allowing the transformer to potentially reconstruct the positions of tokens. We show that this pattern can occur in both the trained and the randomly initialized Transformer models with causal attention and no positional encodings over a common range of hyperparameters.

1 Introduction

Recent results by Haviv et al. [2022], Kazemnejad et al. [2023], and Chi et al. [2023] suggest that positional encodings are not necessary when training decoder-only Transformer language models. These results motivate our investigation of how Transformers might represent positional information without positional encodings.

As shown in [Tsai et al., 2019] [Zuo and Guerzhoy, 2024], the non-causal attention mechanism is equivariant to the permutation of the input tokens —- the prediction for input token $n + 1$ is invariant to permutations of tokens $1, 2, ..., n - 1$. Therefore, without positional encodings, the causal attention mechanism is required for the Transformer to consider the order of the input tokens. Chi et al. [2023] hypothesize that causal attention allows positional information to be stored using the variance (taken across the indices of the embedding vector – essentially the norm) of the embeddings, which generally decreases for tokens at larger positions. They argue that the variance will tend to decrease because, when using causal attention, embeddings n is computed using embeddings $1, 2, ..., n - 1$ in the previous layer, whereas embeddings $n + k$ will be computed using k more input embeddings, leading to the variance shrinkage for embeddings $n + k$.

We identify a different possible way of representing positional information that also arises from the fact that embeddings at smaller positions is computed using fewer embeddings from the previous layer compared to those at larger positions. Specifically, we observe that embeddings at nearby indices will tend to have cosine similarity. This property could, in principle, enable the reconstruction of a token's position.

The rest of the paper is organized as follows. We first briefly review causal attention and establish the notions of the self-cosine-similarity matrix and the adjacency pattern, which quantify the similarity between token embeddings (Section 2.1). We then describe the metric and the tasks (Addition, Reversal, Ordering, and Indexing) that we use in our experiments (Section 3.2). We use them to examine the existence of the adjacency pattern in a variety of configurations. We report on experimental results in Section 4.

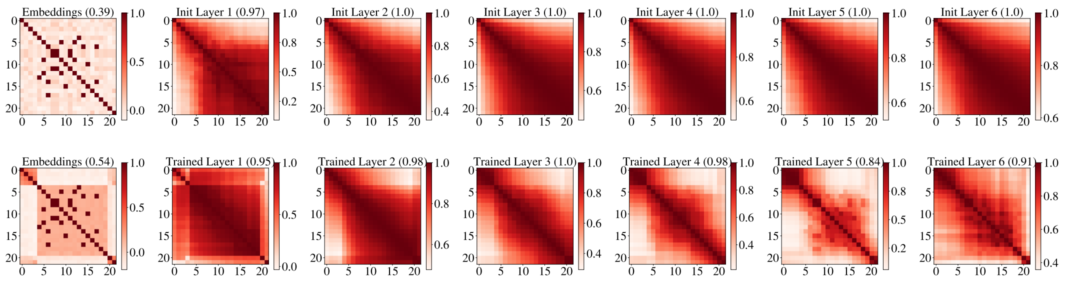

Figure 1: Self-cosine-similarity matrices of randomly initialized (first row) and trained (second row) 6-layer Transformers with causal attention and no positional encodings on the task of Reversal (22). The matrices are produced using a testing sample of 22 tokens, "rev(8502251258017069)=", as input, showing results from the embeddings to the output of layer 6 left to right for the initialized and trained models. The number in the bracket represents the Adjacency Probability Score.

2 Background

2.1 Transformers with causal attention store positional information without positional encodings

In a Transformer with non-causal attention, an output at the k-th position is agnostic to the change in positions of the inputs from other positions, a property known as permutation equivariance. Without positional encodings, permutation equivariance prevents the output of each layer from understanding the relative positions between any pairs of tokens. Consequently, the model will fail to distinguish between different ordering of the same input tokens. In contrast, as shown in [Tsai et al., 2019], Transformers with causal attention are not permutation-equivariant to the input sequence. This implies the possibility of the success of Haviv et al. [2022] in training causal Transformers without positional encodings —- non-causal attention could not accomplish that. [Zuo and Guerzhoy, 2024] demonstrate some of the experimental settings we use in this paper.

2.2 The self-cosine-similarity matrix and the adjacency pattern

The *self-cosine-similarity matrix* is a method to visualize the similarity between all pairs of vectors within a sequence of embeddings. To create the self-cosine-similarity matrix $\boldsymbol{C}$ for a sequence of n token embeddings $\boldsymbol{X} \in \mathbf{R^{n\times d}}$ of dimension d, we define each entry $\boldsymbol{D}_{ij}$ as the cosine similarity between the i^{th} and j^{th} token embeddings, namely, $\boldsymbol{D}_{ij} = similarity(\boldsymbol{X}_i, \boldsymbol{X}_j)$. Since the cosine similarity operation is commutative, $\boldsymbol{D}_{ij} = \boldsymbol{D}_{ji}$, resulting in the self-cosine-similarity matrix being diagonally symmetrical.

The *adjacency pattern* is a pattern we observe in self-cosine-similarity matrices, an indication of token embeddings at nearby positions having higher similarity than further ones. An example of this pattern can be found in Figure 1, where the matrix is darker (higher values) closer to the diagonal and brighter (lower values) further way, indicating that each embedding vector is more similar to closer ones and less to further ones. We posit that embeddings that exhibit the adjacency pattern contain positional information, since the position of a token can be approximated by observing which embeddings are close to which.

The self-cosine-similarity matrix is used in Wang and Chen [2020] to visualize various positional encodings, some of which, such as the sinusoidal embeddings, demonstrate an adjacency pattern. In our work, the self-cosine-similarity matrix is applied to the causal attention's output embeddings directly to examine their adjacency pattern.

3 The adjacency pattern appears in both non-trained and trained architectures in a variety of configurations

In this section, we explore the settings in which the adjacency pattern in causal Transformers with no positional encodings ("Causal-NoPE") appears. We define the way we measure the adjacency pattern, describe the tasks we are using, and provide the experimental details.

3.1 The adjacency probability score

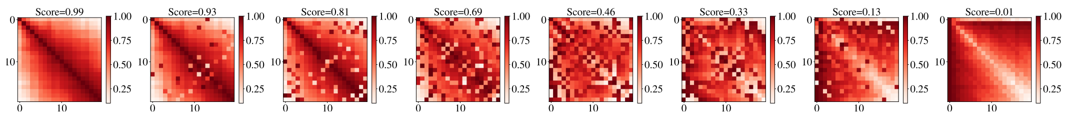

Figure 2: Synthetic matrices with different adjacency probability score values.

We propose the *adjacency probability score* as a metric to quantify the intensity of the adjacent patterns. The score is constructed to correlate with the amount of positional information that can be inferred from the self-similarity matrix.

We compute the proportion of time that the embeddings of tokens with closer positions have higher cosine similarity than those farther away, which can be derived directly from the self-cosine-similarity matrix. Consider the entries at the k^{th} row of a self-cosine-similarity matrix from the 0^{th} column up to k^{th} column (the column of the diagonal entry), denoted by $\boldsymbol{C}_{k1}, \boldsymbol{C}_{k2}, \ldots, \boldsymbol{C}_{kk}$. The row-wise adjacency probability score for this row is defined as:

$$P_{\text{Adj},k} = \mathbb{P}\left[\boldsymbol{C}_{ki} < \boldsymbol{C}_{kj} \text{ if } i < j\right] = \frac{1}{\binom{k}{2}} \sum_{i=0}^{k} \sum_{j=0}^{i} \mathbb{I}_{[\boldsymbol{C}_{ki} < \boldsymbol{C}_{kj}]} \tag{1}$$

where $\mathbb{I}_{[\boldsymbol{C}_{ki} < \boldsymbol{C}_{kj}]}$ is 1 when $\boldsymbol{C}_{ki} < \boldsymbol{C}_{kj}$ and 0 otherwise. The adjacency probability score for the entire self-cosine-similarity matrix is calculated as the average row-wise adjacency probability score over all rows. Notice that only the lower triangular portion of the matrix is involved in the calculation (See Appendix A.1).

3.2 Tasks

We trained Causal-NoPE Transformers for a variety of tasks that require positional information. The tasks were selected for being trainable from scratch and always requiring positional information.

Addition: The Addition task involves generating the completion of strings like `"123+456="`. Following Lee et al. [2024], whose code base we also use, we train NanoGPT to generate the answer in reverse order. The input length is typically 9 for a sample of 3-digit Addition equation.

Reversal: The Reversal task requires the model to generate the reversed sequence. As an example, for the prompt "rev(1234)=", in an auto-regressive manner, the model is supposed to output "4“ as the next token and continue to generate "3", "2", and "1", which ends up completing the prompt as "rev(1234)=4321". The input length is typically 22 for reversing 16 digits.

Indexing: The Indexing task requires the model to locate the position of the first occurrence of a number in the sequence. As an example, for the prompt "wherex(134504392,4)=", the model is supposed to output "2", which is the index for the first occurrence of "4". The input length is typically 19 for reversing 16 digits.

Ordering: Given a sequence of numbers and its reordered version, the Ordering task requires the model to output the new order of the original indices based on the reordered sequence. As an example, for the prompt "order(67812,28716)=", the model is supposed to generate the answer "42130" auto-regressively. The input length is typically 17.

3.3 Experimental setup

We first want to examine whether the adjacency pattern persists for models trained for different tasks that require positional information. We train the baseline 6-layer NanoGPT with 10.6 million parameters on each of the tasks. By default, all models are initialized by the normal distribution $\mathcal{N}(0, 0.02)$. The training for each configuration is repeated for 5 different random seeds. Each task has 20000 training and 20000 testing samples. The rest of the training configurations follow the work by Lee et al. [2024] that trained the NanoGPT model to converge on the 3-digit Addition task. All experiments are conducted using an NVIDIA RTX4090 graphics card, with each trial being approximately 15 minutes.

Additionally, we want to compare different configurations of the model, particularly the number of layers and hidden dimensions. We choose the task of reversal and train models with 6, 12, and 24 layers and 192, 384, and 768 hidden dimensions, respectively, with the same train-test split. Unless further specified, the most trained models have reached above 90% accuracy in the testing set.

4 Results

Tasks	Embeddings	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
Addition (9) Init	0.48 ± 0.09	1.00 ± 0.01	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
Addition (9) Trained	0.49 ± 0.06	0.96 ± 0.04	0.98 ± 0.01	1.00 ± 0.01	0.99 ± 0.01	0.91 ± 0.03	0.81 ± 0.04
Ordering (18) Init	0.52 ± 0.05	0.98 ± 0.05	0.99 ± 0.05	0.99 ± 0.06	0.99 ± 0.05	0.99 ± 0.05	1.00 ± 0.04
Ordering (18) Trained	0.55 ± 0.04	0.85 ± 0.07	0.99 ± 0.05	0.94 ± 0.05	0.73 ± 0.04	0.84 ± 0.05	0.77 ± 0.07
Reversal (22) Init	0.54 ± 0.07	0.97 ± 0.11	0.98 ± 0.09	0.98 ± 0.09	0.99 ± 0.08	0.98 ± 0.08	0.99 ± 0.07
Reversal (22) Trained	0.58 ± 0.07	0.90 ± 0.10	0.96 ± 0.06	0.99 ± 0.07	0.90 ± 0.07	0.80 ± 0.06	0.79 ± 0.09
Indexing (20) Init	0.51 ± 0.05	0.99 ± 0.05	1.00 ± 0.04	1.00 ± 0.04	1.00 ± 0.04	1.00 ± 0.04	1.00 ± 0.04
Indexing (20) Trained	0.57 ± 0.04	0.82 ± 0.06	0.97 ± 0.04	0.95 ± 0.04	0.87 ± 0.05	0.82 ± 0.05	0.87 ± 0.05

Table 1: Layer-wise adjacency probability score (mean $\pm$ standard deviation) for the 4 tasks, with initialization and trained results, each averaged over 256 samples. The number in the bracket beside each task indicates the length (maximum and most frequent) of the equations in the task.

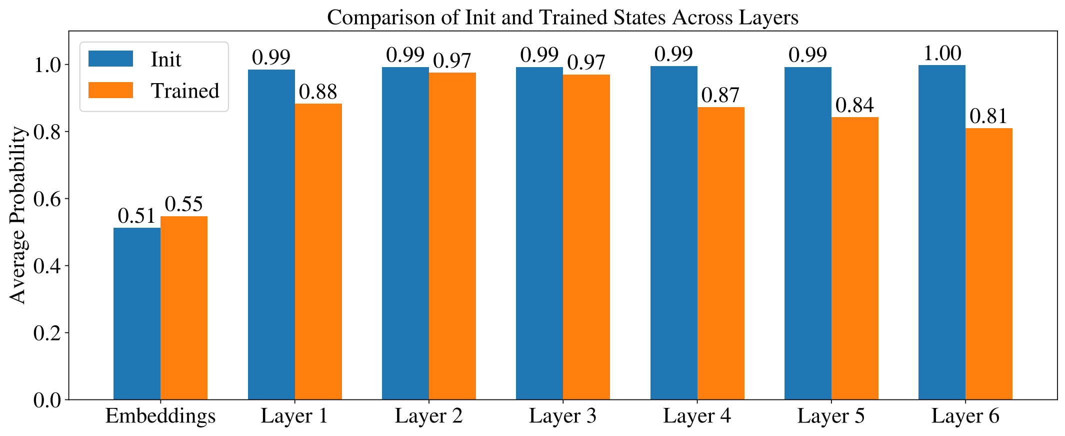

Figure 3: The layer-wise adjacency probability score for randomly initialized and trained models averaged over the 4 tasks, correspond to the values presented in Table 1

4.1 Random initialization produces the adjacency pattern

We computed the self-cosine-similarity matrix and the adjacency score across settings. We use the sample in Figure 1 to represent the general samples that we see. In Figure 1, while there is no adjacency pattern in the matrices of the zeroth layer (i.e., the token embeddings), the adjacency pattern starts to appear in the output of the first attention layer and continues in the rest of the layers. The adjacency probability scores in the zeroth layer (i.e., the token embeddings) —- 0.39 and 0.54

for the randomly initialized and trained models respectively —– are much lower than in the other layers (where the minimum is 0.84). In those upper layers, the embeddings have been through at least 1 layer of causal attention. Hence, one layer of causal attention could be sufficient to generate the adjacency pattern.

Layers	Embeddings	Layer 1	Layer 2	Layer 3	Layer n-2	Layer n-1	Layer n
6	0.38 ± 0.10	0.90 ± 0.17	0.95 ± 0.15	0.96 ± 0.13	0.84 ± 0.11	0.76 ± 0.11	0.93 ± 0.16
12	0.49 ± 0.09	0.90 ± 0.17	0.93 ± 0.08	0.96 ± 0.11	0.86 ± 0.11	0.81 ± 0.08	0.84 ± 0.09
24	0.51 ± 0.10	0.84 ± 0.15	0.94 ± 0.09	0.84 ± 0.10	0.90 ± 0.10	0.78 ± 0.09	0.75 ± 0.09

Table 2: Layer-wise adjacency probability score (mean $\pm$ standard deviation) for models with different numbers of layers trained on the Reversal (22) task, each averaged over 256 samples.

Dimensions	Embeddings	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
192	0.49 ± 0.11	0.93 ± 0.17	0.96 ± 0.14	0.96 ± 0.14	0.92 ± 0.14	0.81 ± 0.12	0.73 ± 0.11
384	0.38 ± 0.10	0.90 ± 0.17	0.95 ± 0.15	0.96 ± 0.13	0.84 ± 0.11	0.76 ± 0.11	0.93 ± 0.16
768	0.50 ± 0.11	0.96 ± 0.15	0.96 ± 0.15	0.95 ± 0.15	0.94 ± 0.15	0.90 ± 0.12	0.93 ± 0.15

Table 3: Layer-wise adjacency probability score for models with different numbers of hidden dimensions trained on the Reversal (22) task, averaged over 256 samples. The only configuration that did not achieve over 90% test accuracy is the model with 192 dimensions, which has an accuracy of 56%. Yet, we observed that in most cases where the model makes an error, the majority of digits are correct, with only a few being incorrect.

4.2 Adjacency pattern across different models and datasets

The adjacency probability scores of models trained on various tasks and with different numbers of hidden dimensions and the number of layers are listed in Tables 1, 2, and 3. Each column of the table indicates the location where the embeddings are taken to produce the self-cosine-similarity matrices. Figure 3 presents the adjacency probability scores for the embeddings at each layer, averaged across different tasks. For Table 2, and 3, without loss of generality, we present the ablation study results for the Reversal (22) task because we observed that the effect of having different numbers of layers and embedding is the same across different tasks.

For most configurations, the adjacency probability scores spike up from around 50% in the token embeddings to more than 80% at the first layer and maintain in the rest. This is consistent regardless of the task type, the training state (initialized/trained), the number of layers, and the dimensions. As a general trend, the adjacency score is the highest for output embeddings in the second layer, while declines gradually from there to the end.

4.3 The adjacency pattern across different initializaitons

We further test different initialization schemes, showing that the adjacency pattern is robust for the commonly used initialization schemes. Table 4 and Table 5 show the results for the adjacency probability scores obtained in models initialized by Normal distribution with different means ($\mu_{init} \in \{0, 4, 8\}$) and different standard deviations ($\mu_{init} \in \{0.002, 0.02, 0.2\}$). The highlighted adjacency probability scores indicate a lack of discernible adjacency patterns qualitatively. The adjacency pattern is missing when the mean and the standard deviation are large enough ($\mu_{init} = 4$ and $\sigma_{init} = 0.2$), which are not typical values for initialization. It can be inferred that the mean has a smaller influence than the variance, since the first layer for the model with $\mu_{init} = 4$ can still produce the adjacency pattern. Yet, it is likely that the large μ_{init} only causes the variance after the first layer to be large, which is why the adjacency pattern for the rest of the layers is removed.

μ_{init}	Embeddings	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
0	0.39 ± 0.10	0.92 ± 0.17	0.94 ± 0.18	0.95 ± 0.17	0.95 ± 0.15	0.95 ± 0.17	0.95 ± 0.17
4	0.46 ± 0.15	0.91 ± 0.18	0.95 ± 0.10	0.93 ± 0.06	0.95 ± 0.07	0.95 ± 0.08	0.96 ± 0.06
8	0.36 ± 0.11	0.91 ± 0.18	$\mathbf{0.54} \pm 0.09$	$\mathbf{0.41} \pm 0.04$	$\mathbf{0.46} \pm 0.06$	$\mathbf{0.51} \pm 0.08$	$\mathbf{0.60} \pm 0.11$

Table 4: Layer-wise adjacency probability score for models initialized by Gaussian distribution with different means μ_{init}, averaged over 256 samples from the Reversal (22) tasks.

σ_{init}	Embeddings	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
0.002	0.36 ± 0.11	0.93 ± 0.17	0.93 ± 0.19	0.95 ± 0.14	0.94 ± 0.16	0.95 ± 0.16	0.94 ± 0.17
0.02	0.39 ± 0.10	0.92 ± 0.17	0.94 ± 0.18	0.95 ± 0.17	0.95 ± 0.15	0.95 ± 0.17	0.95 ± 0.17
0.2	0.35 ± 0.10	**0.43** ± 0.09	**0.49** ± 0.08	**0.54** ± 0.09	**0.59** ± 0.12	**0.57** ± 0.12	**0.65** ± 0.13

Table 5: Layer-wise adjacency probability score for models initialized by Gaussian distribution with different standard deviation σ_{init}, averaged over 256 samples from the Reversal (22) tasks.

5 Discussion

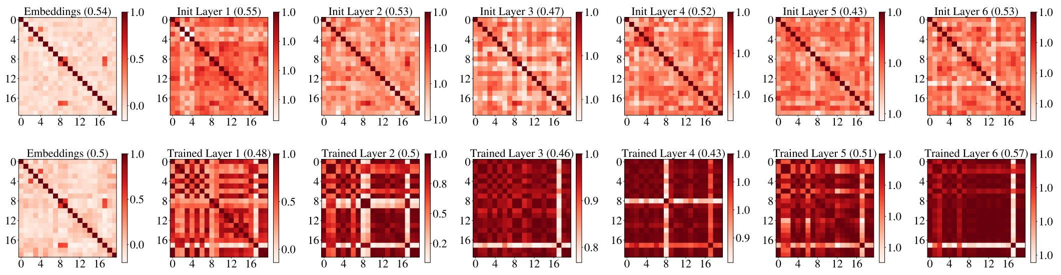

Figure 4: Self-cosine-similarity matrices of randomly initialized (first row) and trained (second row) 6-layer Transformers with normal attention and learned absolute positional encodings on the task of Indexing (20). The matrices are produced using a testing sample of 20 tokens, "wherex(299517340,9)=", as input.

5.1 Is the adjacency pattern unique to causal attention

Yes. We also applied a self-cosine-similarity matrix to Transformers with normal attention and confirmed that there is no adjacency pattern. An example is shown in Fig 4, where the self-cosine-similarity matrices look random and the adjacency scores are low. There is a learned absolute positional embedding added to the token embeddings of this model only to let the model converge.

5.2 The origin of the adjacency pattern

Though we haven't had a formal proof to determine the sufficient and necessary condition of the adjacency pattern, our analysis roughly points out two conditions. The first is to have a small variance in the attention weight, creating something we call the "averaging effect". The second is to have distinctive value vectors. Further details are in Appendix A.2.

6 Conclusions and future work

In Transformers with causal attention and no positional encodings, the adjacency pattern can occur for models with a wide range of hyperparameters, including the number of layers, hidden dimensions, and initialization schemes. It exists in the output embeddings of the Transformer's first causal attention layer and persists throughout the rest of the layers. For randomly initialized weights, the adjacency pattern can be observed for various initializations, especially for the ones commonly occurring in practice. For trained models, it is typical that the adjacency pattern in the first few layers is more prominent than in later ones, which we consider reasonable because knowing enough positional information in the earlier layers may allow the models to focus on other more contextual information required by the tasks in later layers.

References

T.-C. Chi, T.-H. Fan, L.-W. Chen, A. I. Rudnicky, and P. J. Ramadge. Latent positional information is in the self-attention variance of transformer language models without positional embeddings. *arXiv preprint arXiv:2305.13571*, 2023.

A. Haviv, O. Ram, O. Press, P. Izsak, and O. Levy. Transformer language models without positional encodings still learn positional information. *arXiv preprint arXiv:2203.16634*, 2022.

A. Kazemnejad, I. Padhi, K. N. Ramamurthy, P. Das, and S. Reddy. The impact of positional encoding on length generalization in transformers. *arXiv preprint arXiv:2305.19466*, 2023.

N. Lee, K. Sreenivasan, J. D. Lee, K. Lee, and D. Papailiopoulos. Teaching arithmetic to small transformers. *International Conference on Learning Representations*, 2024.

Y.-H. H. Tsai, S. Bai, M. Yamada, L.-P. Morency, and R. Salakhutdinov. Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. In K. Inui, J. Jiang, V. Ng, and X. Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4344–4353, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1443. URL `https://aclanthology.org/D19-1443`.

Y.-A. Wang and Y.-N. Chen. What do position embeddings learn? an empirical study of pre-trained language model positional encoding. *arXiv preprint arXiv:2010.04903*, 2020.

C. Zuo and M. Guerzhoy. Breaking symmetry when training transformers. *NAACL Student Research Workshop*, 2024.

A Appendix / supplemental material

A.1 More about the adjacency probability score

We only consider each row up to the diagonal because, for causal attention, each self-cosine-similarity matrix $S \in \mathbb{R}^{n \times n}$ of size n contains n sub-matrices, from $S_1 \in \mathbb{R}^{1 \times 1}$ to $S_n \in \mathbb{R}^{n \times n}$. For a sub-matrix of length $k \in [1, .., n]$, it is formed by embeddings resulting exactly from the first k out of n tokens of the original sequence. Therefore, each row-wise adjacency probability score at row k measures the last row of sub-matrix S_k. Another way to think of this is that causal attention at the current token only considers anything before it. Hence, we measure just the adjacency probability score for anything up to the current token, which is up to the diagonal of each row.

Figure 2 demonstrates different adjacency probability scores with their respective sample matrix. A higher adjacency probability score can be interpreted as the model being more likely to know the exact ordering of other tokens before a certain token. Meanwhile, although a zero adjacency probability score will also allow the model to know the token order oppositely, it is unachievable in a self-cosine-similarity matrix unless all embeddings are the same. For random matrices, the adjacency probability score is about 0.5.

A.2 How the adjacency pattern arises

In this section of the appendix, we want to provide some evidence to show that small variances inside the model weights facilitate high adjacency scores inside the causal attention's output embeddings. The Lemma 3 in Chi et al. [2023] also indicates that the initialized Transformers with causal attention typically have small variances in each row of the attention matrix. However, while Chi et al. [2023] are using Lemma 3 for their prove of the decreasing variance in the output embeddings of the causal attention, we would like to use Lemma 3 differently to show that the adjacency pattern in the self-cosine-similarity in the Transformer also relies on this.

A.2.1 Empirical evidence

The following is a possible origin of the adjacency pattern. It is the way that the vectors at positions $k-1$, k, and $k+1$ are computed–the linear combinations of the value vector sets $\{e_1, e_2, \ldots, e_{k-1}\}$, $\{e_1, e_2, \ldots, e_{k-1}, e_k\}$, and $\{e_1, e_2, \ldots, e_{k-1}, e_k, e_{k+1}\}$, respectively–that gives rise to this property. This claim originates from our empirical simulation of the attention output embeddings' computation at initialization.

We first simulate the value vectors used in attention by a set of random normal 128-dimensional vectors $\{v_1, ..., v_k\}$ and the causal attention weights at the 4th, 5th, and 6th row by the following i.i.d. random coefficient sets $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$, $\{\beta_1, \beta_2, \beta_3, \beta_4\}$, $\{\gamma_1, \gamma_2, \gamma_3, \gamma_4\, \gamma_5\, \gamma_6\}$. We then mimic the attention output embeddings at token positions 4, 5, and 6 by the following linear combination of vectors: $a = \left(\sum_{i=1}^{4} \alpha_i v_j, b = \sum_{i=1}^{5} \beta_i v_j, c = \sum_{i=1}^{6} \gamma_i v_j\right)$. Denote the consine similarity as "sim". We want to determine the condition for $sim(a, b)$ to be consistently higher than $sim(a, c)$, as well as for $sim(c, b)$ to be higher than $sim(c, a)$. We simulate with a range of standard deviations σ_{init} from the set $\{0.001, 0.01, 0.11, 10, 100\}$, and for each we repeat for 10000 trials and record $sim(a, b) - sim(a, c)$ and $sim(c, b) - sim(c, a)$ for each trial. The resulting histogram is plotted in Fig 5, where the first and second rows are for $sim(a, b) - sim(a, c)$ and $sim(c, b) - sim(c, a)$, respectively. The distribution is narrow and above zero for only small values of σ_{init}, corresponding to the condition that allows $sim(a, b)$ to be consistently higher than $sim(a, c)$ (same for $sim(c, b)$ and $sim(c, a)$. This also corroborates with the experimental results in Table 5.

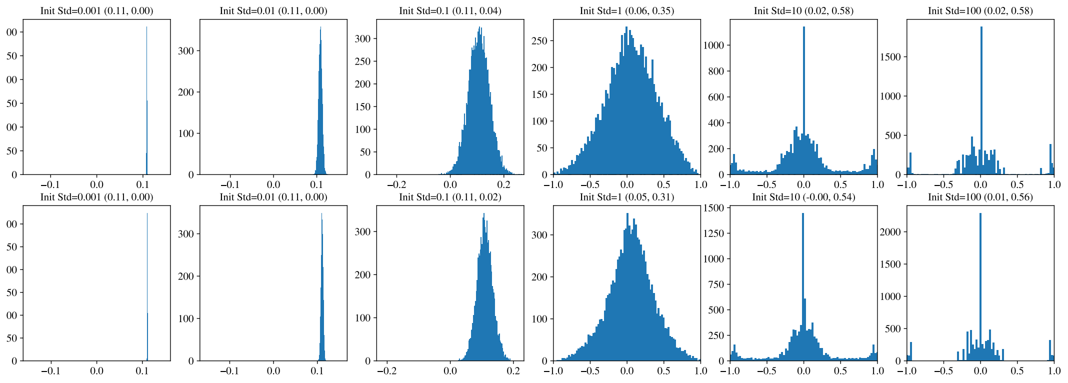

Figure 5: Histograms on the differences between the cosine similarity of nearby tokens and further ones.

A.2.2 The averaging effect provably arises in the first layer

Here, we show that we can expect that, in the second layer (i.e., the first layer after the embeddings), the angle between embedding $k + t$ and embedding $k + t + 1$ is smaller than the angle between embedding $k + t$ and embedding $k + t + 2$, implying an adjacency pattern.

Assume that the embeddings $\{e_1, e_2, ..., e_k, ..., e_n\}$ are high-dimensional and normalized, and therefore approximately orthogonal. We are computing the next layer, with coefficients α, α', β, and β'.

We would like to show that the angle between $\sum_{i=1}^{k+t} \alpha_i e_i$ and $\sum_{i=1}^{k+t+1} \beta_i e_i$ would tend to be smaller than the angle between $\sum_{i=1}^{k+t} \alpha_i e_i$ and $\sum_{i=1}^{k+t+2} \beta'_i e_i$. The weights α, β, and β', which correspond to the attention weight in a causal architecture, would all sum to 1: $\sum_{i=1}^{k} \alpha_i = \sum_{i=1}^{k+t} \beta_i = \sum_{i=1}^{k+t+1} \beta'_i$ = 1.

Instead of the angles, we compute the dot products, and show that we can expect the difference between the dot products to be positive, namely

$$\left(\sum_{i=1}^{k+1} \alpha_i v_i \cdot \sum_{i=1}^{k+t} \beta_i v_i\right) - \left(\sum_{i=1}^{k+1} \alpha_i v_i \cdot \sum_{i=1}^{k+t+1} \beta'_i v_i\right) > 0.$$

Indeed, the difference between the left and right sides is

$$\sum_{i=1}^{k+1} \alpha_i v_i \cdot \sum_{j=1}^{k+t+1} (\beta_j - \beta'_j) v_j$$
$$\approx \sum_{i=1}^{k+1} \alpha_i (\beta_i - \beta'_i) v_i \cdot v_i$$
$$\approx ||v|| \sum_{i=1}^{k+1} \alpha_i (\beta_i - \beta'_i) > 0,$$

where the approximate equalities follows from the approximate orthogonality of large n-dimensional vectors normalized to norm 1.

A.2.3 The norm alone may not be sufficient for Accurate Positional Information

Tasks	Embeddings	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
Addition (9) Init	0.47 ± 0.05	0.92 ± 0.04	0.96 ± 0.03	0.96 ± 0.04	0.98 ± 0.02	0.96 ± 0.03	0.93 ± 0.06
Addition (9) Trained	0.46 ± 0.04	0.91 ± 0.01	0.91 ± 0.03	0.63 ± 0.06	0.82 ± 0.04	0.79 ± 0.02	0.54 ± 0.04
Ordering (18) Init	0.47 ± 0.05	0.91 ± 0.03	0.96 ± 0.03	0.99 ± 0.01	0.98 ± 0.02	0.98 ± 0.02	0.94 ± 0.05
Ordering (18) Trained	0.41 ± 0.02	0.87 ± 0.02	0.84 ± 0.13	0.86 ± 0.04	0.67 ± 0.05	0.58 ± 0.05	0.49 ± 0.04
Reversal (22) Init	0.49 ± 0.02	0.78 ± 0.06	0.88 ± 0.08	0.84 ± 0.10	0.83 ± 0.12	0.95 ± 0.05	0.93 ± 0.07
Reversal (22) Trained	0.51 ± 0.02	0.90 ± 0.02	0.84 ± 0.06	0.78 ± 0.04	0.75 ± 0.05	0.70 ± 0.05	0.53 ± 0.05
Indexing (20) Init	0.51 ± 0.08	0.90 ± 0.05	0.94 ± 0.03	0.97 ± 0.03	0.96 ± 0.03	0.96 ± 0.02	0.96 ± 0.03
Indexing (20) Trained	0.52 ± 0.02	0.86 ± 0.03	0.96 ± 0.03	0.93 ± 0.03	0.82 ± 0.05	0.73 ± 0.06	0.67 ± 0.06

Table 6: Layer-wise adjacency probability score of the norm of embeddings (mean $\pm$ standard deviation) for the 4 tasks, with initialization and trained results. The number in the bracket beside each task indicates the input length involved in the task.

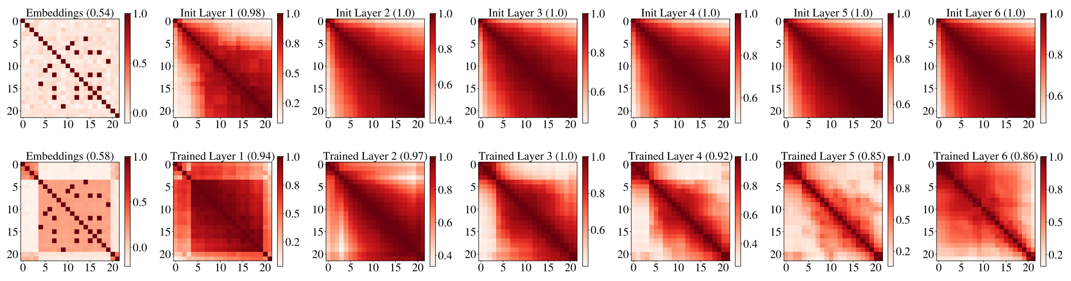

Figure 6: Layer-wise self-cosine-similarity matrices of randomly initialized (first row) and trained (second row) Causal-NoPE Transformers on the task of ordering, with "rev(1849364897192906)=" as the input.

A.3 More visualizations of experimental results

What mean and standard deviation do not show directly is the true distribution of the adjacency pattern in the sample. To determine if there are clusters of samples that exhibit extremely low to extremely high values, we check the distributions of the adjacency scores for all configurations. Typically, we observe distributions like the ones in Figure 9. In this example, while the distributions of adjacency scores concentrate around 1 for the untrained model, after training, only the adjacency scores for layer 2 and layer 3 distribute densely and closely to 1. In particular, the adjacency scores are the highest and most concentrated in layer 3 of the trained model, to an extent that matches the ones in the untrained model. We interpret these observations as an indication that the model learns to keep the adjacency pattern in earlier layers and gradually discard it in later ones.

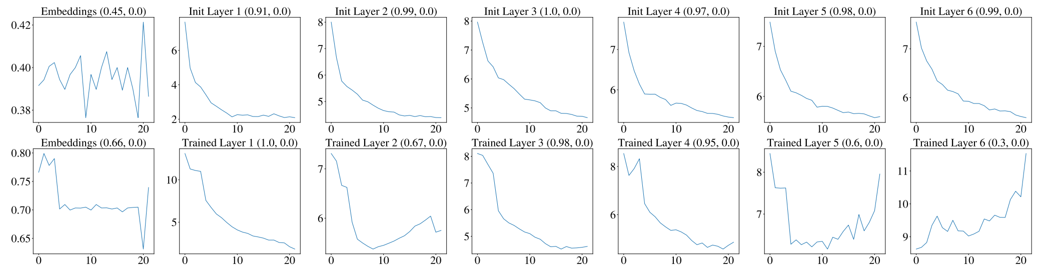

Figure 7: Layer-wise embedding norms for randomly initialized (first row) and trained (second row) Causal-NoPE Transformers on the task of Reversal (22), with "rev(1849364897192906)=" as the input.

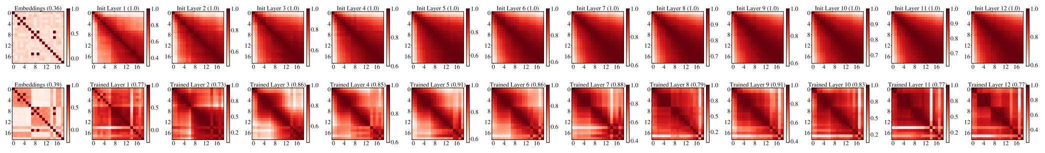

Figure 8: Self-cosine-similarity matrices of randomly initialized (first row) and trained (second row) 12-layer Transformers with causal attention and no positional encodings on the task of Indexing. The matrices are produced using a testing sample of 22 tokens, "wherex(8483561,8)=0", as input, showing results from the embeddings to the output of layer 12 left to right for the initialized and trained models. The number in the bracket represents the adjacency probability score.

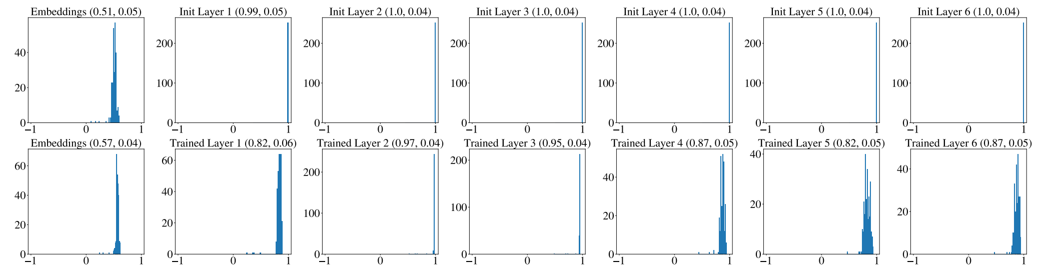

Figure 9: Distribution of Adjacency Probability Score for a model before and after training ("Init"/"Trained") on the indexing task. The sample size of the histograms is 256. The two numbers inside the brackets of the subplot titles are the distribution's mean and standard deviation. Notice that the 7 pairs of means and standard deviations for the trained model (the second row) correspond to the values presented in Table 1 for the indexing task.