The Interplay between Memory Capacity and Algorithmic Alignment: Understanding Counting in Transformers

Anonymous Authors[1]

Abstract

We provide a comprehensive analysis of simple transformer models trained on the histogram task, where the goal is to count the occurrences of each item in the input sequence from a fixed alphabet. Despite its apparent simplicity, this task exhibits a rich phenomenology that allows us to characterize how different architectural components contribute towards the emergence of distinct algorithmic solutions. In particular, we showcase the existence of two qualitatively different mechanisms that *implement* a solution, relation- and inventory-based counting. Which mechanism a model can implement depends non-trivially on the precise choice of the mixing mechanism, activation function, memorization capacity and the presence of a beginning-of-sequence token. By introspecting *learned* models on the counting task, we find evidence for the formation of both mechanisms: in the regime where the models *cannot implement* either mechanism precisely, we provide an intuition that, under some circumstances, they converge to approximate solutions resembling the aforementioned mechanisms. From a broader perspective, our analysis offers a framework to understand how the interaction of different architectural components of transformer models shape diverse algorithmic solutions and approximations.

1. Introduction

Transformers are the key neural network behind many recent deep learning advances, most notably Large Language Models (LLMs). Their success is partly due to their versatility in processing diverse data types, including text, images, and video, represented as sequences of tokens (Liu et al., 2021; Girdhar et al., 2019; Brown et al., 2020). While scale has been a key factor to unleash the potential of these models, it is remarkable that their architecture still largely follows the same simple template of the original transformer model proposed by Vaswani et al. (2017). At its core, a single transformer block primarily alternates two basic components: the attention mechanism and a standard multi-layer perceptron. At a high level, the attention mechanism mixes the tokens, while the multi-layer perceptron applies a nonlinear feature transformation identically to each token. Despite the widespread use of these models, there is no clear consensus on the distinct roles of these components, how they interact, or if they can be substituted with alternative elements (Tolstikhin et al., 2021; Bozic et al., 2023; Gu & Dao, 2023). In particular, the specific contribution of each architectural element to the model's hypothesis space –the range of algorithms it can learn and implement– remains opaque (Weiss et al., 2021; Delétang et al., 2023; Abbe et al., 2023).

In this work, we investigate this question from a mechanistic interpretability perspective (Cammarata et al., 2020; Olah et al., 2020; Elhage et al., 2021; Michaud et al., 2024) by considering the histogram task as a prototypical problem (Weiss et al., 2021). This task consists in predicting the number of appearances of each token in the input sequences processed by the model – counting. Despite its apparent simplicity, this task exhibits a rich phenomenology, allowing us to study the relative role of different architectural components and their impact on the final solutions implemented by the model in a controlled setting. To this end, we focus on models following the architectural template of primitive transformer blocks, i.e. alternating a token-mixing mechanism and a multi-layer perceptron.

In our analysis, we provide manual implementations (parameter configurations) for a range of such architectures that reach perfect accuracy in a model-dependent hyperparameter regime. In a second step, we compare these algorithms with the performance and mechanistic behavior of models trained from data. Our findings reveal that this class of models is capable of implementing drastically different solutions to the histogram task, with a strong dependence on the scale of the model's hyperparameters and the type of mixing mechanism used. In particular, we identify two main algorithmic strategies a model can implement to suc-

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

ceed in the histogram task, which we term *relation-* and *inventory-based counting*. Relation-based counting uses local pair-wise comparisons between tokens in a given sequence to obtain the number of occurrences conditioned on a given position. Inventory-based counting relies on the knowledge of the complete alphabet and counts the occurrences of all possible tokens to then extract the correct count for a given position.

The emergence of either mechanism during learning depends on the specifics of the architecture and the inductive bias it possesses in correspondence to the task. The main contributions of this work are as follows:

- We show that for some architectures, an implementation of relation-based counting is very memory and compute-efficient as it can leverage an attention-like dot-product mixing mechanism for comparison operations. It requires only a low-capacity feed-forward module, as most of the computation is effectively spent in the mixing module utilizing a token-independent counter subspace (Weiss et al., 2021).
- Inventory-based counting, instead, can be implemented based on an input-independent mixing mechanism. This weak inductive bias can be compensated via a feed-forward module with a large enough hidden layer that can memorize a lookup-table to implement a comparison operation (inventory): the model-task misalignment can be closed at the price of increased memory and compute requirements.
- We study under which conditions both mechanisms emerge through training. In particular, we verify the feasibility regime defined for each mechanism from the manual implementations within the phase space determined by each architecture's two primary hyperparameters: the embedding dimension and the size of the hidden layer in the feed-forward block.
- For the hyperparameter regimes where we do not know of perfect implementations, especially when the model dimension is comparatively smaller than the alphabet, we show that non-orthogonal embeddings can be disentangled more favorably by dot-product mixing and the softmax activation, whereas linear mixing fails to give high-quality approximations. This gives insights into the interpretable insights into the models failiure modes.

In Section 2 we provide the necessary background and introduce our notation; in Section 3 we describe our experimental setup and in Section 4 we present our theoretical and experimental results. Section 5 discusses the related literature and Sections 6 and 7 present the limitations, conclusion and open questions of our work.

2. Background and Notation

Architecture. As inputs, we consider sequences of tokens $\mathbf{x} = (x_1, x_2, \cdots, x_L) \in \mathcal{T}^L$. Each token stems from the set $\mathcal{T} = \{1, \cdots, T\}$ of size T. The corresponding sequence of outputs $\mathbf{y} = (y_1, \cdots, y_L)$ has the same length as the input sequence, where each output token belongs to the output vocabulary $\mathcal{C}$ of size C, i.e. $y_\ell \in \{1, ..., C\}$, with $C \leq L$. In this work, we analyze several 1-layer model architectures where a token-mixing mechanism is followed by a per-token feature transformation. This setup includes the case of a single transformer block where the dot-product attention mechanism is followed by a point-wise feed forward network. Formally, we consider a model $F : \mathcal{T}^L \to \mathcal{C}^L$ defined for the positions $\ell = 1, \cdots, L$ as

$$F(\mathbf{x})_\ell = \underset{c \in \{1, \cdots, C\}}{\arg\max}\ f(\bar{x}'_\ell)_c\,;\ \bar{x}'_\ell = \bar{x}_\ell + [\mathbf{A}(\bar{\mathbf{x}})\bar{\mathbf{x}}]_\ell \quad (1)$$

with the token mixing matrix $\mathbf{A} : \mathbb{R}^{L\times d} \to \mathbb{R}^{L\times L}$ and the feature transformation $f : \mathbb{R}^d \to \mathbb{R}^C$. The embedding $\bar{\mathbf{x}} \in \mathbb{R}^{L\times d}$ is obtained by passing the input sequence $\mathbf{x}$ into an embedding layer of dimension d. We refer to the embedding associated with token $t \in \mathcal{T}$ as $e_t \in \mathbb{R}^d$ or $e_{x_\ell} \in \mathbb{R}^d$ for the embedding of the token x_ℓ, and to the vector $\bar{x}'$ as the mixed token. The final prediction of the model is obtained from the maximum value of the vector of size c output by the feature transformation. In the following, we consider several implementations of $\mathbf{A}$ and f.

Token Mixing. We consider two types of mixing mechanisms with different activation functions. We refer to the case where the function $\mathbf{A}$ is constant in $\bar{\mathbf{x}}$ as linear mixing, e.g.

$$\mathbf{A}_{\text{lin}}(\bar{\mathbf{x}}) = A \quad (2)$$

$$\mathbf{A}_{\text{lin+sftm}}(\bar{\mathbf{x}}) = \text{softmax}(A) \quad (3)$$

where $A \in \mathbb{R}^{L\times L}$ is a learnable matrix and the softmax operator is applied row-wise. The number of learnable parameters is therefore L^2.

As an alternative mixing structure, which we refer to as dot-product mixing, we consider the popular attention mechanism which constructs the matrix $\mathbf{A}$ to be explicitly dependent on the inputs, i.e.

$$\mathbf{A}_{\text{dot}}(\bar{\mathbf{x}}) = \frac{1}{\sqrt{d}}\bar{\mathbf{x}}W_Q W_K^T \bar{\mathbf{x}}^T \quad (4)$$

$$\mathbf{A}_{\text{dot+sftm}}(\bar{\mathbf{x}}) = \text{softmax}\left(\frac{1}{\sqrt{d}}\bar{\mathbf{x}}W_Q W_K^T \bar{\mathbf{x}}^T\right) \quad (5)$$

where W_Q and W_K are learnable $d \times d$ matrices, and the softmax function is applied row-wise. The number of parameters for dot-product mixing is $2d^2$.

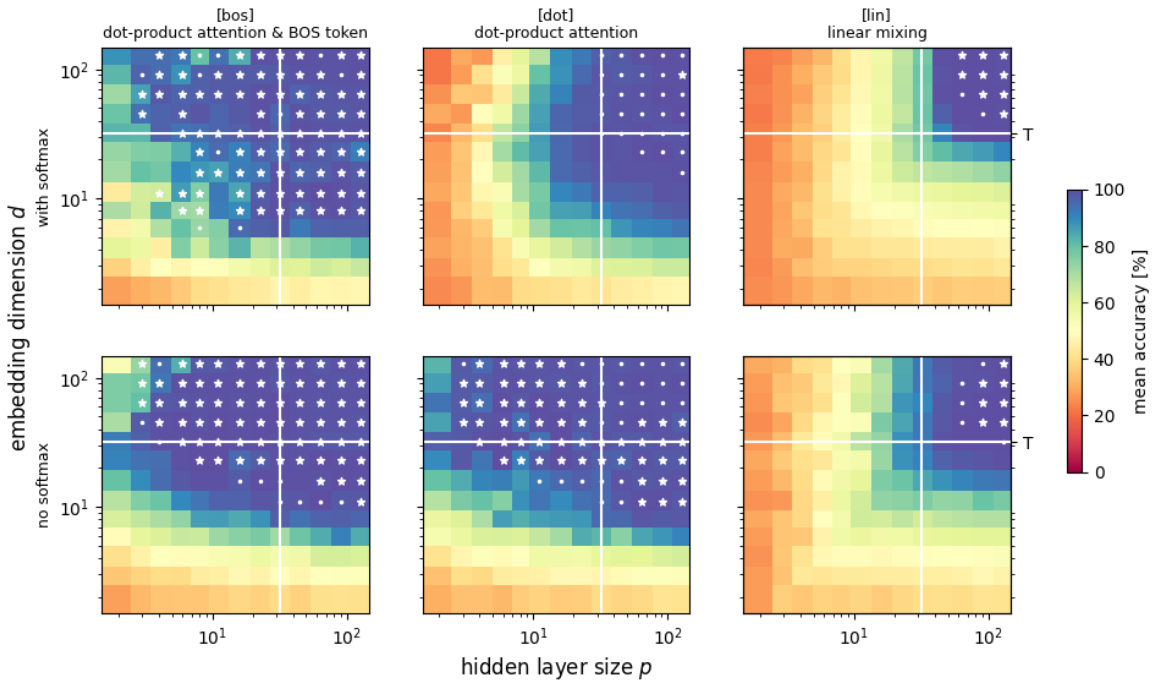

Figure 1. Mean accuracy for varying embedding dimension d, hidden layer dimension p, for fixed $T = 32$ and $L = 10$ for the different token mixing mechanisms `dot`, `bos` and `lin`. *(Top)* Models without softmax; *(Bottom)* Models with softmax. Average over 5 runs for every $d, p \in \{2, 3, 4, 6, 8, 12, 16, 23, 32, 45, 64, 91, 128\}$.Vertical and horizontal lines indicate $p = T$ and $d = T$ respectively. White stars mark the parameter configurations, where a 100% accuracy configuration was found during training in at least one of the five runs. White dots mark the same for $\geq 99\%$ accuracy configurations.

Feature Transformation. The feature transformation is a single hidden layer perceptron with ReLU activations. The hidden layer is of dimension p. The function f is applied to the mixed vector with the residual addition $\bar{x}'$ as

$$f(\bar{x}') = \mathrm{ReLU}(\bar{x}' W_1 + b_1) W_2 + b_2 \tag{6}$$

where $f(\bar{x}') : \mathbb{R}^d \to \mathbb{R}^{L \times C}$ and where the weights have the appropriate dimensions to accommodate a hidden layer of size p, i.e. $W_1 \in \mathbb{R}^{d \times p}, b_1 \in \mathbb{R}^p, W_2 \in \mathbb{R}^{p \times C}$ and $b_1 \in \mathbb{R}^C$.

3. Experimental Setup

Task and Dataset. We consider a simple algorithmic task that is referred to as *histogram*: given a sequence of tokens, the goal is to return a sequence of the same length where each entry represents the number of times the corresponding input token appears in the entire sequence. For example, given $\mathbf{x} = [A, B, D, D, B, B]$, the output will be $\mathbf{y} = [1, 3, 2, 2, 3, 3]$. We define the count of token t in the sequence $\mathbf{x}$ as $hist_{\mathbf{x}}(t)$. In our experiments, we consider i.i.d. distributions of sequences of length L from an input vocabulary of size T, where $L \leq T$. Our sampling strategy relies on first sampling a set of partitions, and then assigning a token to each partition (see App. A for details). Note that this strategy is different from the one used by Weiss et al. (2021) as it avoids skewing the training distributions to targets which largely exhibit the same counts.

Models and Training. We investigate the performance on the histogram task of the four different variants of the token mixing models described in Sec. 2, i.e. `lin` and `dot`, with or without the softmax (`+sftm`), where the token embeddings are jointly learned with the model parameters. Their relevant parameters, in addition to those resulting from the mixing type, are the dimension of the embedded tokens d, and the hidden layer size p of the feature transformation. Additionally, we consider the model `bos` where every input sequence is prefixed with a special token – the so-called beginning-of-sequence (BOS) token – prior to entering a dot-product mixing layer (both with or without softmax). Previous studies have demonstrated that transformer networks consistently attend to BOS tokens, despite their lack of semantic content, and we explore this point in our experiments.

All models are trained with Adam with a learning rate of $\nu = 10^{-3}$ on the cross-entropy loss for 500 epochs with a batch size of 32. We consider the online learning setting where for each new epoch we generate a dataset of $10,000$ data samples. We compute the accuracy attained by each model based on a set of $3,000$ independent data samples, which covers a large range of all possible input sequences.

4. Learning Regimes in Counting

We analyze the performance of these models with varying mixing mechanisms in different learning regimes characterized by different values of the embedding dimension d and number of hidden neurons p of the feed-forward module. Fig. 1 shows the the accuracy attained by different learned models for $L = 10$ and $T = 32$. Our goal is to understand why some regimes for different models enable a perfect fit of the data (feasible regimes), while others can only approximate the task solution (approximation regimes).

In App. B, we provide additional experiments for $T = 64, L = 10$, as well as a comparison between the number of parameters of each model and the corresponding accuracy levels. In the following sections, we separately examine the different regimes for $d \geq T$ and $d < T$, investigating the inner mechanisms of the models in each sub-case and providing manual implementations of perfect accuracy model configurations for some regimes.

4.1. $d \geq T$: Orthogonal token embeddings are separable

Comparisons in orthogonal embedding space. When the model dimension d is at least as big as the number of tokens T, tokens can be represented by embeddings that are mutually orthogonal to one another. Assuming all tokens $t \in \mathcal{T}$ have such mutually orthogonal embeddings $e_t \in \mathbb{R}^d$ with a norm of 1, the overlap is $\langle e_s, e_t \rangle = 0$ for distinct tokens $t \neq s$ and it is 1 when $t = s$.

In such a scenario, a linear combination of token embeddings preserves the information about single tokens. By leveraging knowledge about the embeddings of the alphabet, a weighted sum of tokens, denoted as $e' = \sum_{t \in \mathcal{T}} \alpha_t e_t$, can be broken down into the original tokens using projections on the original token embeddings, where $\alpha_t = \langle e_t, e' \rangle / \|e_t\|_2^2$. In the following, we show that this property can be used to *manually* configure the weights of a model for each mixing mechanism to solve the histogram task for all possible sequences when $T = d$ and that this is reflected in the models learned from data. Remarkably, the different mixing mechanisms require a different number of hidden neurons p: either $p = T$ or $p = 1$. We show that this depends on the interplay of the mixing layer and the feature transform: for some mixing mechanisms the latter needs to implement inventory-based counting ($p = T$) and for others relation-based counting ($p = 1$).

Relation-based counting: The BOS direction acts as a counter in a low-dimensional subspace. When an extra beginning of sequence token t_{BOS} is available in `bos`, it can be used as a reference counter in the attention part of the network for $hist_{\mathbf{x}}(x_\ell)$. In the literature, beginning (or end) of sequence tokens have been linked to model-internal computations such as counting. In (Weiss et al., 2021), it is shown that the RASP language can solve the histogram task with one layer and one attention head. We confirm empirically that `bos` and `bos+sftm` reach (close to) 100% accuracy whenever $d > T$ and we verify that a relation-based counting algorithm can be manually implemented in these two architectures.

When we set the t_{BOS} embedding to $e_{\text{BOS}} = \sum_{t \in \mathcal{T}} e_t$, this embedding has an overlap of 1 with every other token embedding, assuming that all other token embeddings e_t are normalized to have norm 1. Assuming that t_{BOS} is at the first position of the sequence of now length $L+1$, a clean dot-product operation in the attention mechanism (with $Q, K = \sqrt{\sqrt{d}}\mathbb{I}_d$) will lead to an attention matrix with entries:

$$a_{\ell m} = \begin{cases} T & \text{if } \ell = m = 1 \\ 1 & \text{if } \ell > 1, m = 1 \\ 1 & \text{if } \ell, m > 1, \text{and } x_\ell = x_m \\ 0 & \text{if } \ell, m > 1, \text{and } x_\ell \neq x_m \end{cases}.$$

Without softmax, projecting the mixed token onto the t_{BOS}, means that $\langle \bar{x}'_\ell, e_{\text{BOS}} \rangle = T + hist_{\mathbf{x}}(x_\ell) + 1$, i.e. the embedding e_{BOS} can be taken as the single relevant direction that reveals the count of x_ℓ. Therefore, a *single* hidden neuron $p = 1$ suffices and the output layer can transfer the count into a categorical representation. The same principle applies to `bos+sftm`, although $\langle \bar{x}'_\ell, e_{\text{BOS}} \rangle$ scales non-linearly in the token count due to the softmax normalization (App. D.2).

Investigating learned models we find models achieving 100% accuracy that do not exactly correspond to the algorithm we prescribed manually, but we find a close correspondence up to symmetries. In Fig. 2, we show for an example of `bos+sftm` that t_{BOS} indeed plays a special role in the *learned* model: in the attention matrix it can already be observed that its activation could be interpreted as a proxy for the number of occurrences of x_ℓ, as it has different values for tokens that occur a different amount of times. We emphasize that here the comparison operation naturally provided by the dot-product allows the model to extract the count of the same tokens, for each token in the sequence. For all other elements of the input sequence, the activations of the attention matrix are comparatively low when the compared tokens are the same and high when they are different. In the embedding space, its special role stands out when measuring the two first principal components. We also show how the presence of the t_{BOS} determines the final prediction of the feature transformation f.

Relation-based counting can be implemented without a BOS token. Surprisingly, the `dot` model (without the softmax) reaches a comparatively good empirical performance as `bos` regime $d \geq T$, even though it does not have a dedicated counter token available. It turns out that this model can implement a similar mechanism using a counting subspace but without explicitly resorting to any extra token. As we show in App. D, one can add a single

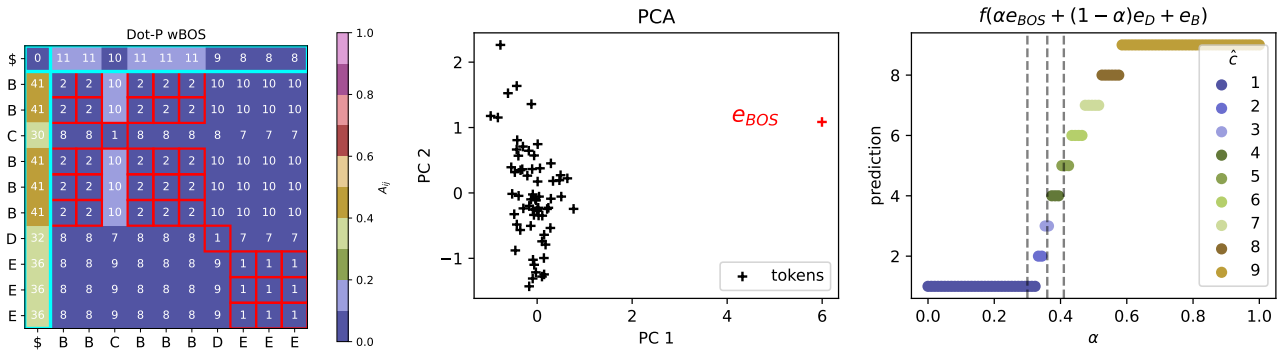

Figure 2. **bos+sftm** ($T = 64, L = 10, p = 10, d = 128$) – This model achieves 98.19% accuracy. *(Left)* The attention values on the t_{BOS} ('\$') in the first column $a_{\ell,0}$ is a proxy for the count of x_ℓ. *(Middle)* The e_{BOS} varies strongly from the embedded tokens in the first two principal components. *(Right)* The feature transformation maps a mixed input $\alpha e_{\text{BOS}} + (1-\alpha)e_D + e_B$ to the count prediction. This is an artifical mixing, that serves to demonstrate the predictions independence of the precise tokens that are mixedwith the weight $(1-\alpha)$ (we show more examples in App. E). Indeed, marking the $a_{\ell,0}$ obtained from the right as vertical lines, the prediction is correct for all counts independent of the precise token. This indicates that after the mixing step, only the direction of e_{BOS} is used for computing the count.

common direction e_{cnt} to all –up to that point orthogonal– token embeddings. Using a clean dot-product mixing, this leads to $a_{\ell m} = a_{\neq}$ when x_ℓ is different from x_m, and $a_{\ell m} = a_{=}$ when tokens are the same. Then, the number of counts can be easily extracted from the dot-product $\langle e_{\text{cnt}}, \bar{x}'_\ell \rangle$ of the counting token with the mixed token $\bar{x}'_\ell$, i.e. $\langle e_{\text{cnt}}, \bar{x}'_\ell \rangle \propto 1 + hist_{\mathbf{x}}(x_\ell)a_{=} + (L - hist_{\mathbf{x}}(x_\ell))a_{\neq}$. We can therefore obtain a perfect accuracy implementation in the regime where $d \geq T$ with a only single hidden neuron, explaining the observed empirical performance without a BOS token.

Dot-product attention with softmax cannot implement relation-based counting. Since the dot-product mechanism is well-suited for relation-based counting, one might expect that the `dot+sftm` model also implements this mechanism. However, and maybe surprisingly so, we empirically observe a marked difference between `dot` and `dot+sftm` in Fig. 1. `dot` only starts performing close to 100% accuracy when both the model dimension d and the number of hidden neurons p are larger than the number of tokens T. To understand why this is the case, we show the attention matrix of `dot+sftm` in Fig. 3. Notably, it is based on the semantics, as $(\mathbf{A}_{\text{dot+sftm}})_{\ell m}$ is higher when $x_\ell = x_m$ than otherwise. However, the normalization effect of the softmax activation prevents the development of a meaningful counter subspace that is needed in the relation-based algorithm. As a result of normalization, the attention scores are $\sum_m a_{\ell m} = 1$, so any direction present in all tokens (and by symmetry of the task, it would need to be present in all tokens) would be uninformative after the token mixing – its weight would be 1 regardless of the input sequence and would therefore not carry information about the count. Before, the model `bos+sftm` circumvented this problem via adding the extra token with a special functionality, as itself does not need to be counted. Because this is not possible for `dot+sftm`, the architecture fails to perform well for $p = 1$ – it now needs to measure more than one direction in the feed-forward module. In the following, we show that good performance can still be achieved as this is an instance of inventory-based counting algorithm, where the complete alphabet is memorized in the feed-forward layer. We detail this in the following paragraphs, for the example of `lin`, but show an implementation for `dot+sftm` in App. D.3.

Inventory-based counting: Memorization in the feature transform. For `lin` we can prescribe weights manually that achieve 100% accuracy when $T = d = p$, which roughly corresponds to the regime where all *trained* models perform almost perfectly (Fig. 1). For `lin+sftm` and `dot+sftm` the same ideas explain their good performance and are elaborated in App. D.3. Again, for the given parameterizations of the models, several solutions exist due to symmetries, and in the following we describe one of them for clarity. In the linear mixing layer, we set the mixing weights for different positions ℓ, m in the sequence as $a_{\ell m} = a$ and for the same position $\alpha_{\ell\ell} = 2a$, with $a = 1/(L+1)$. Hence, the mixed token $\bar{x}'_\ell$ (after the residual connection) can be written as

$$\bar{x}'_\ell = \sum_{m=1}^{L} a_{\ell m} e_{x_m} + e_{x_\ell} \,.$$

The correct count of x_ℓ can therefore be extracted, via:

$$hist_{\mathbf{x}}(x_\ell) = \frac{1}{a} \sum_{t \in \mathcal{T}} \text{ReLU}\left(\langle \bar{x}'_\ell, e_t \rangle - (1+a)\right)$$

By design, only the hidden neuron activated according to the overlap with token $e_t = x_\ell$ has a non-zero activation. This equation can exactly be implemented by the first layer of the feature transformation in equation 6 with $p = T$

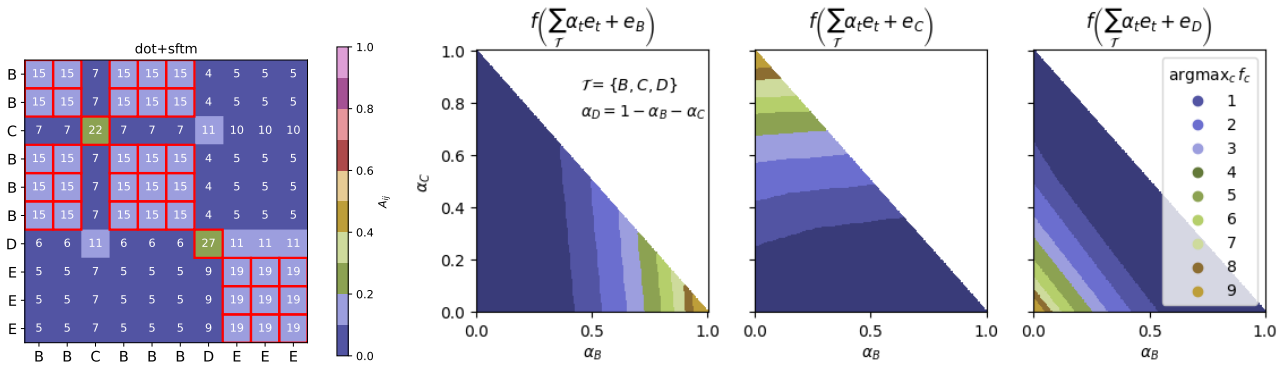

Figure 3. **dot+sftm** ($T = 64, L = 10, p = 64, d = 64$) learned with 98.8% accuracy. *(Left)* Attention matrix for a given sequence. *(Right)* Prediction of the feature transformation f for a linear combination of three tokens. The prediction directly depends on the coefficient associated with the residual connection. The non-linear scaling of the decision boundaries is due to the softmax.

hidden neurons. The output layer can then be designed to activate the correct output vector corresponding to the count $hist_{\mathbf{x}}(x_\ell)$ (refer to the visualization in App. D.4). Since we selected $a \in [0, 1]$ and $\sum_{m=1}^{L} a_{\ell m} = 1$ the same procedure can be implemented by a matrix which is passed through the softmax operator for `lin+sftm`. This means that the feed-forward module is correlated with the complete alphabet, acting as a look-up table, or inventory. In Fig. 4, we inspect the attention matrix $\mathbf{A}_{\text{lin}}$ and the feature transformation f which is *learned* by Adam for `lin+sftm` for parameters in the regime where $p \sim T \sim d$. The mixing has an off-diagonal of ~ 0.11 and a diagonal of ~ 0.08. Feeding the feature transformation f with a weighted combination of 3 tokens, B, C, D, we observe that the final prediction of the network depends mainly on the coefficient α_t corresponding to the token embedding fed through the residual connection. Notably, this suggests that the feature transformation must have encoded the information of the token embedding in its weights, hence requiring at least $p = T$ hidden neurons.

Superposed or selective implementations. Some of the models capabilities overlap, for example the models that can implement relation-based counting for $p = 1$ can also implement inventory-based counting for $p \geq T$. In Fig. 1 we observe that the model `dot` (which is able to implement relation-based counting) witnesses a very slight decrease in maximal learned performance from 100% accuracy to 99% when the number of hidden neurons is *increased* to $p = T$ and inventory-based counting can be implemented in principle. We do not have a clear understanding of whether the memory-intensive solution is preferred. It is possible that, from a random initialization, inventory-based basins of attraction are suddenly favored, which in turn appear to be harder to optimize to 100% accuracy (as visible empirically for `dot+sftm`). We intend to explore this issue further in the future, considering approaches like those in (Henighan et al., 2023), which examine the phenomenon of feature versus data learning from a double descent perspective.

4.2. $d < T$: Entangled Embeddings

Linear mixing fails due to entanglement. The case $d < T$ differs from the scenario investigated in Section 4.1 for a fundamental reason: the embedding associated with different tokens cannot be orthogonal anymore. This implies that every pair of different tokens will have a non-zero overlap due to linear dependence, and therefore mixing tokens entangles count information over the different directions in embedding space.

An immediate consequence of this aspect is that for `lin`, $p = T$ is no longer sufficient, as it is not anymore possible to obtain precise counts by projecting directly onto the T different directions from the alphabet. Following our manual implementations, we would obtain an overlap between non-identical embeddings of tokens s and t as $\langle e_s, e_t \rangle = \epsilon_{s,t} > 0$ due to linear dependence. Projecting the mixed token $\bar{x}'_\ell$ (for `lin`: a weighted sum of all input tokens in the sequence with a residual connection) onto an embedding e_t would not only give the weight α_t associated with the count of t, but also additional terms $\epsilon_{s,t} > 0$ when other tokens appear in the sequence. This effect may result in α_t being associated with a count that does not correspond to the true value. This effect is reflected in our empirical analysis in Fig. 1, where we see that `lin`'s performance degrades as d is decreased even at $p > T$ (same for `lin+sftm` and `dot+sftm`). However, this decrease is gradual rather than discrete and suggest that approximations are feasible.

Almost-orthogonality aids approximations. A first intuition is that the overlap between different tokens can be minimized globally, allowing the model to operate in a regime of *quasi-orthogonality* and thereby minimizing the values $\epsilon_{s,t}$. Since the absolute value of ϵ is lower bounded by the mutual coherence (the minimal overlap for T vectors in d dimensions) all models deteriorate for very small d. Fig. 5 shows that the overlaps between tokens indeed become continuously more spread out (hence resulting in more noise) as we decrease d, which is accompanied by more errors in the

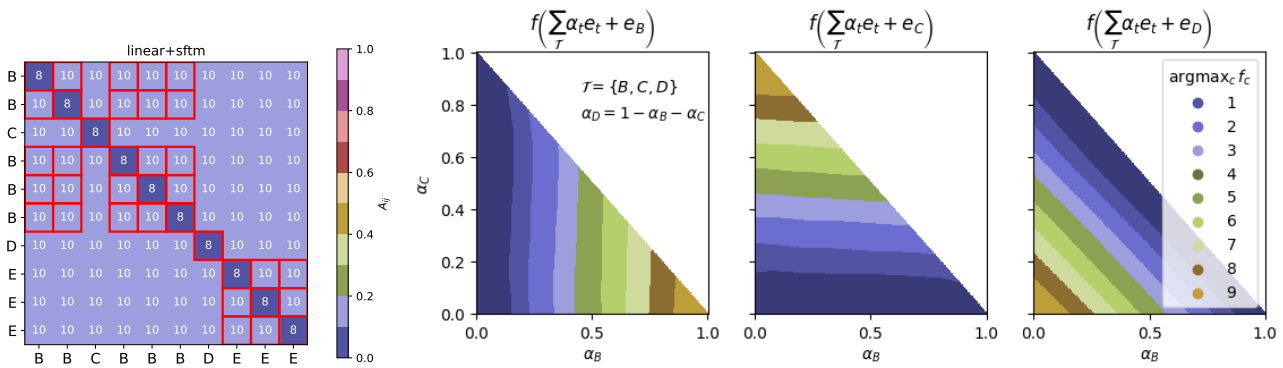

Figure 4. **`lin+sftm`** ($T = 64, L = 10, p = 128, d = 128$) – This model achieves 99.97% accuracy. *(Left)* Attention matrix learned by Adam, which is constant in the input sequence $\mathbf{x}$. Even though it does not correspond exactly to our implementation, it assigns a constant score to all tokens except on the diagonal. The different score on the diagonal in addition to the residual connection help to identify which token of the inventory should be counted. *(Right)* Predictions on artificial mix of learned embeddings for the three tokens B, C and D. The prediction depends on the token in the residual connection, but is largely independent of the presence of other tokens in the mixing. This indicates that f projects the mixed token onto the alphabet $\mathcal{T}$ and is able to extract tokens due to orthogonality.

prediction. However, when $p > T$ this intuition holds for inventory-based counting with both `lin` and with `dot`, but the approximations from `dot` is still much more favorable.

Dot-product mixing allows 100% accuracy solutions despite entanglement. In fact, we observe that all learned models *with a dot-product mixing* outperform the linear models, and relation-based models even achieve perfect performance in regimes where we do know of manual implementations (Fig. 1). A striking example is the existence of a 100% accuracy solution for `bos+sftm` at $d = 6, p = 8, T = 32$. A possible reason for this success stems from the dependence of the dot-product mechanisms on the semantics of the tokens. A token t can amplify (or diminish) relevant (or irrelevant) directions as a function of its similarity (dot-product) with another token – which is not possible with a constant linear mixing. This aspect allows for an "error-correction" by reducing the magnitude of the weights α_s for tokens s different from e_t, and therefore the corruption $\epsilon_{s,t}$ introduced onto the relevant direction the mixed token both for the inventory- (`dot+sftm`) and relation-based counting (`dot`, `bos`, `bos+sftm`) can be minimized. This hypothesis implies that the attention scores a_{ij} are such that (up to ± 1 symmetries) $a_{ij} < a_{ii}$ for all $x_i \neq x_j$. The difference between the largest $\max_j a_{ii} - a_{ij}$ determines the maximal possible corruption on the mixed token. A construction for a set of such embeddings with a maximal distance of $1 - 1/\sqrt{2}$ is possible for $d = \log(T)$ dimensions with vectors representing the binary encoding of each token, normalized to one. We show that learning the embedding is a substantial advantage over *random but fixed* embeddings when $d < T$ (App. B.2).

The softmax non-linearity improves disentanglement. The softmax activation allows a model to non-linearly reduce the noise ϵ stemming from not-equal comparisons

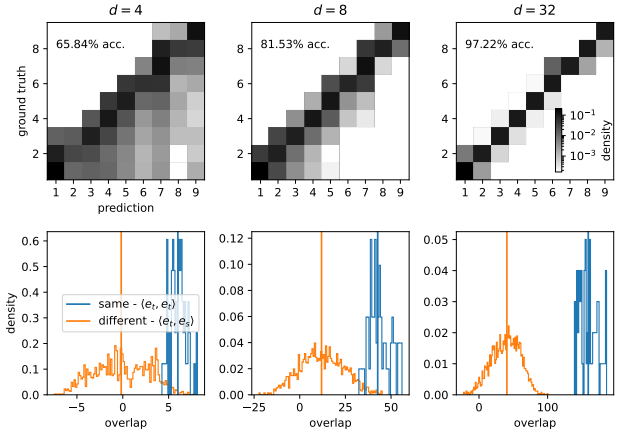

Figure 5. **Approximations.** We show examples of `dot` for $T = 64, L = 10, p = 128$ for varying the model dimension d. The top row shows the confusion matrix of ground truth and predicted counts. Below is the overlap distribution between same and different token embeddings.

even further. This provides a possible explanation as to why `bos+sftm` generally reaches better performance in the small d and small p regime than `bos` and `dot`.

Increased sensitivity to initialization. When $d < T$, the interplay between token mixing mechanism and feed-forward module becomes very intertwined in the optimization process: when the first fails to find an optimal embedding for the given dimension, the second inevitably requires larger values of p to disentangle the counts. We hypothesize that this aspect makes the loss landscape more challenging to optimize, resulting in a larger variability in accuracy across different runs in these more critical regimes (see App. B.1). Here, the overparameterized regime seems preferable from an optimization perspective.

5. Related Work

Mechanistic Interpretability. The emergence of algorithmic capabilities in transformers (Olsson et al., 2022; Power et al., 2022) has led to the numerous investigations aimed at reverse-engineering the trained models into human-understandable mechanisms (Zhong et al., 2023; Nanda et al., 2023; Quirke & Barez, 2024). In this work, we examine the histogram task, previously studied in the context of the RASP(-L)-language (Weiss et al., 2021; Abbe et al., 2023), but, to our knowledge, not from an interpretability perspective (except for counting as a sequence of associations (Gould et al., 2023), rather than summarizing). We confirm that an extra BOS token in the input can be used as a scratchpad (Nye et al., 2021) to extend the capabilities of small models. Additionally, we emphasize that a counting subspace does not necessarily require extra tokens; with a suitable model architecture, it can be an intrinsic property of the token embeddings themselves. While many works in this area focus on causal interventions (Vig et al., 2020; Meng et al., 2023) to gain insights, our approach aligns more with those attributing relevance scores to model components (nostalgebraist, 2020; Elhage et al., 2021). Our main focus is the manual interpretation of the hyperparameter scaling of several distinct models in relation to their performance and known implementations of different algorithms, similar to the studies in (Zhong et al., 2023; Quirke & Barez, 2024).

Memorization. The role of feed-forward layers as memorization modules has been investigated in the context of factual recall for language models (Geva et al., 2021; Meng et al., 2023; Chughtai et al., 2024). Henighan et al. (2023) study a double decent phenomenon where the purpose of the feed-forward layer moves from memorization of datapoints to discovering data features as a function of increasing training data diversity (Raventos et al., 2023). In the histogram task, we observe a similar phenomenon as a function of the architecture: the feed-forward layer acts either as a look-up table or a feature detector for a counting subspace.

Algorithmic Alignment. While theoretical work has outlined the computational capacities of a range of (autoregressive) neural networks (Weiss et al., 2021; Yun et al., 2019; Delétang et al., 2023; Liu et al., 2023), hallucinations and failure modes on seemingly trivial tasks in real world transformers are the rule rather than an exception. Dziri et al. (2023) postulate that this may be due to a misalignment between the computational graph of a model and the task itself. In this work, we show that subtle differences in components such as the mixing type and layer width play a crucial role in terms of algorithmic alignment. Previous work also discovered evidence for the superposition of different computational graphs in a single model (Elhage et al., 2022) – we complement this analysis an example model that disentangles superimposed embedding directions.

6. Limitations

Similar to other works in mechanistic interpretability (Zhong et al., 2023), we focus on 1-layer transformers as a simplified model for modern transformers. Our models are not autoregressive and do not account for the impact of causal masks. While more complex models could lead to more intricate interdependencies between the components, potentially limiting the applicability of our findings to such architectures, it seems plausible that similar vector arithmetic could emerge in subspaces of large transformers (Gould et al., 2023). Given its specificity, it is unclear how similar memory-architecture phenomena would emerge for different simple tasks (e.g., sorting or lookup). Even in our simple example, we have not fully understood the $d < T$ regime to a satisfactory degree, yet.

7. Conclusion

We investigate the role of feed-forward layers and their relation with token-mixing components in transformer-like architectures for counting algorithms. Our analysis shows that their optimal design is highly influenced by the model dimension and the type of token-mixing mechanism. In particular, we find that the memorization requirements of the feed-forward block increase when the mixing mechanism is poorly aligned with the task under consideration and verify that this is the case in learned models. Our introspection of learned models reveals meaningful "counter" subspaces, as predicted by the theory. Additionally, we gain intuition, supported by empirical evidence, that explains the model's good or bad performance when token embeddings are entangled. In this context, the dot-product and softmax activation functions prove to be more advantageous than linear mixing mechanisms.

Future Work. *Implications for complex architectures and broader tasks.* To understand the capabilities of transformers it would be interesting to see efforts to characterize them grounded in vector arithmetic to a similar degree of generality as RASP (Weiss et al., 2021) to allow for entangled (or superpositioned) algorithms and representations (Elhage et al., 2022; Gurnee et al., 2023).
Understanding the link between failure modes, architecture and tasks. At this moment examples for hallucinations and failures of LLM's are as numerous as their success stories. While we provide only a limited example here, we find mechanistic investigations in the "failing" regime a relevant and challenging research direction.
Generalizing to arbitrary comparison functions. It is unclear how to solve the histogram task for $T = \infty$, as for in-context learning in (Reddy, 2024). Preliminary explorations indicate that none of the models considered here can reach perfect accuracy. It is interesting to understand under which conditions more general functions can be implemented or learned.

References

Abbe, E., Bengio, S., Lotfi, A., and Rizk, K. Generalization on the unseen, logic reasoning and degree curriculum. In *ICML*, 2023. URL https://arxiv.org/abs/2301.13105.

Bozic, V., Dordevic, D., Coppola, D., Thommes, J., and Singh, S. P. Rethinking attention: Exploring shallow feed-forward neural networks as an alternative to attention layers in transformers. *arXiv preprint arXiv:2311.10642*, 2023.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020. URL https://arxiv.org/abs/2005.14165.

Cammarata, N., Carter, S., Goh, G., Olah, C., Petrov, M., Schubert, L., Voss, C., Egan, B., and Lim, S. K. Thread: circuits. *Distill*, 5(3):e24, 2020.

Chughtai, B., Cooney, A., and Nanda, N. Summing up the facts: Additive mechanisms behind factual recall in LLMs, 2024. URL https://openreview.net/forum?id=P2gnDEHGu3.

Delétang, G., Ruoss, A., Grau-Moya, J., Genewein, T., Wenliang, L. K., Catt, E., Cundy, C., Hutter, M., Legg, S., Veness, J., and Ortega, P. A. Neural networks and the chomsky hierarchy. In *11th International Conference on Learning Representations*, 2023.

Dziri, N., Lu, X., Sclar, M., Li, X. L., Jiang, L., Lin, B. Y., Welleck, S., West, P., Bhagavatula, C., Le Bras, R., Hwang, J., Sanyal, S., Ren, X., Ettinger, A., Harchaoui, Z., and Choi, Y. Faith and fate: Limits of transformers on compositionality. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 70293–70332. Curran Associates, Inc., 2023.

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., DasSarma, N., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., Grosse, R., McCandlish, S., Kaplan, J., Amodei, D., Wattenberg, M., and Olah, C. Toy models of superposition, 2022.

Geva, M., Schuster, R., Berant, J., and Levy, O. Transformer feed-forward layers are key-value memories. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t. (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.446. URL https://aclanthology.org/2021.emnlp-main.446.

Girdhar, R., Carreira, J., Doersch, C., and Zisserman, A. Video action transformer network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 244–253, 2019.

Gould, R., Ong, E., Ogden, G., and Conmy, A. Successor heads: Recurring, interpretable attention heads in the wild, 2023.

Gu, A. and Dao, T. Mamba: Linear-time sequence modeling with selective state spaces, 2023.

Gurnee, W., Nanda, N., Pauly, M., Harvey, K., Troitskii, D., and Bertsimas, D. Finding neurons in a haystack: Case studies with sparse probing, 2023.

Henighan, T., Carter, S., Hume, T., Elhage, N., Lasenby, R., Fort, S., Schiefer, N., and Olah, C. Superposition, memorization, and double descent. *Transformer Circuits Thread*, 2023.

Liu, B., Ash, J. T., Goel, S., Krishnamurthy, A., and Zhang, C. Transformers learn shortcuts to automata. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=De4FYqjFueZ.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. *CoRR*, abs/2103.14030, 2021. URL https://arxiv.org/abs/2103.14030.

Meng, K., Bau, D., Andonian, A., and Belinkov, Y. Locating and editing factual associations in gpt, 2023.

Michaud, E. J., Liu, Z., Girit, U., and Tegmark, M. The quantization model of neural scaling, 2024.

Nanda, N., Chan, L., Lieberum, T., Smith, J., and Steinhardt, J. Progress measures for grokking via mechanistic interpretability, 2023.

nostalgebraist. interpreting GPT: the logit lens — LessWrong, January 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

Nye, M. I., Andreassen, A. J., Gur-Ari, G., Michalewski, H., Austin, J., Bieber, D., Dohan, D., Lewkowycz, A., Bosma, M., Luan, D., Sutton, C., and Odena, A. Show your work: Scratchpads for intermediate computation with language models. *CoRR*, abs/2112.00114, 2021. URL https://arxiv.org/abs/2112.00114.

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., and Carter, S. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001. https://distill.pub/2020/circuits/zoom-in.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Johnston, S., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. In-context learning and induction heads, 2022.

Power, A., Burda, Y., Edwards, H., Babuschkin, I., and Misra, V. Grokking: Generalization beyond overfitting on small algorithmic datasets, 2022.

Quirke, P. and Barez, F. Understanding addition in transformers. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=rIx1YXVWZb.

Raventos, A., Paul, M., Chen, F., and Ganguli, S. Pretraining task diversity and the emergence of non-bayesian in-context learning for regression. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=BtAz4a5xDg.

Reddy, G. The mechanistic basis of data dependence and abrupt learning in an in-context classification task. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=aN4Jf6Cx69.

Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., Lucic, M., and Dosovitskiy, A. Mlp-mixer: An all-mlp architecture for vision, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need, 2017.

Vig, J., Gehrmann, S., Belinkov, Y., Qian, S., Nevo, D., Singer, Y., and Shieber, S. M. Causal mediation analysis for interpreting neural NLP: the case of gender bias. *CoRR*, abs/2004.12265, 2020. URL https://arxiv.org/abs/2004.12265.

Weiss, G., Goldberg, Y., and Yahav, E. Thinking like transformers. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11080–11090. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/weiss21a.html.

Yun, C., Bhojanapalli, S., Rawat, A. S., Reddi, S. J., and Kumar, S. Are transformers universal approximators of sequence-to-sequence functions? *CoRR*, abs/1912.10077, 2019. URL http://arxiv.org/abs/1912.10077.

Zhong, Z., Liu, Z., Tegmark, M., and Andreas, J. The clock and the pizza: Two stories in mechanistic explanation of neural networks, 2023.

A. Data Generation

Every sample $\mathbf{x} = (x_1, \cdots, x_L)$ is generated recursively as follows, starting from size $K = L$ and alphabet $\mathcal{T}' = \mathcal{T}$:

1. Sample an integer k uniformly from $[1, \cdots, K]$.
2. Sample a token t uniformly from $\mathcal{T}'$.
3. Set $x_i = t$ for all $i = k, \cdots, K$.
4. Set $\mathcal{T}' = \mathcal{T}' \setminus \{t\}$ and $K = k$.
5. If $K \neq 0$, repeat from 1.
6. Set $\mathbf{x} = \texttt{shuffle}(\mathbf{x})$.

In contrast to sampling the elements of each sequence uniformly at random from the alphabet, this simple strategy enables us to better control the distribution of counts in the training dataset.

B. Supplementary Material for Experiments

B.1. Additional Metrics

In this section, we give a more in depth analysis of the experiments performed in Fig. 1.
In Fig. 6, we show the best reached accuracy during training over the five sample runs. This gives insights into the feasibility of implementing a counting solution for a given combination of parameters T, d, p of a model. Interestingly, even for dimensions $d < T$ some models reach an accuracy very close to 100%.
In Fig. 7 we explore the influence of initialization on the performance via the variability of the final results. Especially in the $p, d < T$ regime where `bos+sftm` is able to reach an accuracy relatively close to 100%, the variability of the accuracies resulting from different initializations is quite large.

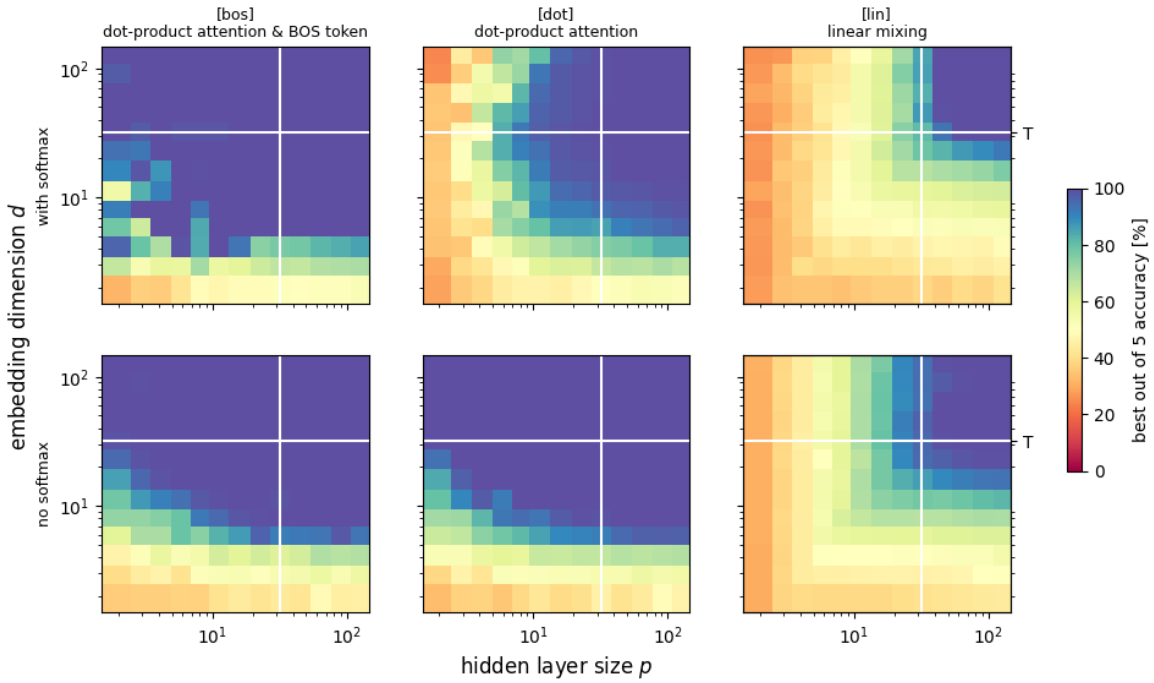

Figure 6. Experiments from Fig. 1 with $T - 32$, best accuracy during training reached from the 5 randomly initialized runs per model/hyperparameter configuration.

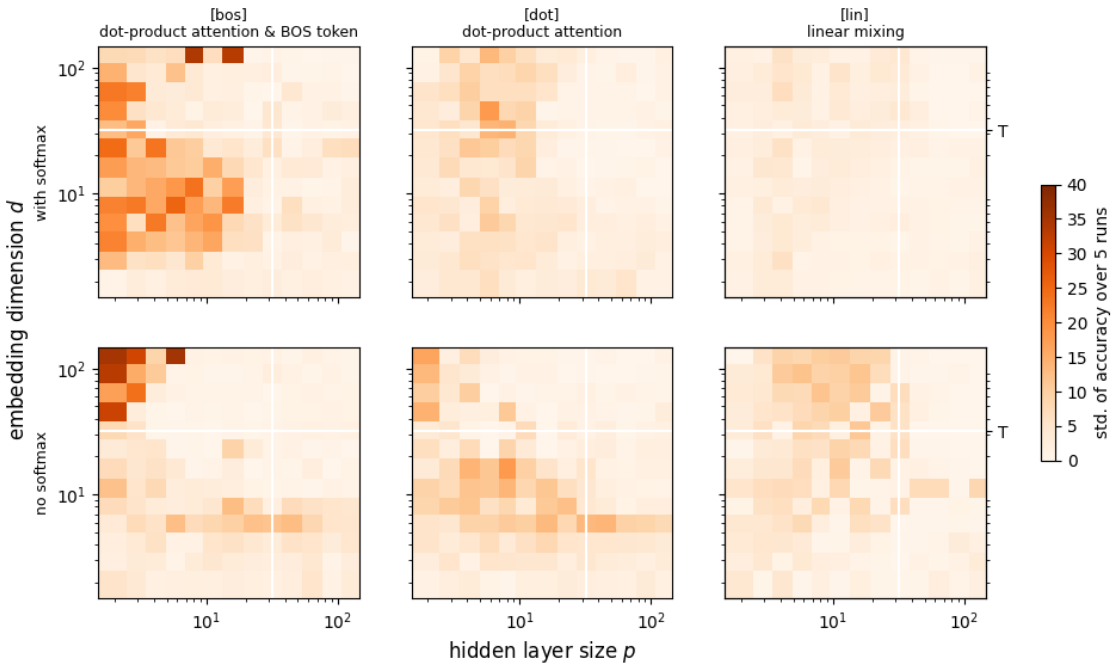

Figure 7. Experiments from Fig. 1 with $T - 32$, standard deviation of the accuracy reached after training from the 5 randomly initialized runs per model/hyperparameter configuration.

B.2. Random but fixed embeddings

In Fig. 8 we repeat the experiments of Fig. 1, but for embeddings that are frozen throughout training (also 5 runs). In the regime $d < T$ where there is no mutual orthogonality possible, the random embeddings result in worse performance. Especially for `bos+sftm`, learning the embeddings increases the performance strongly in some regimes.

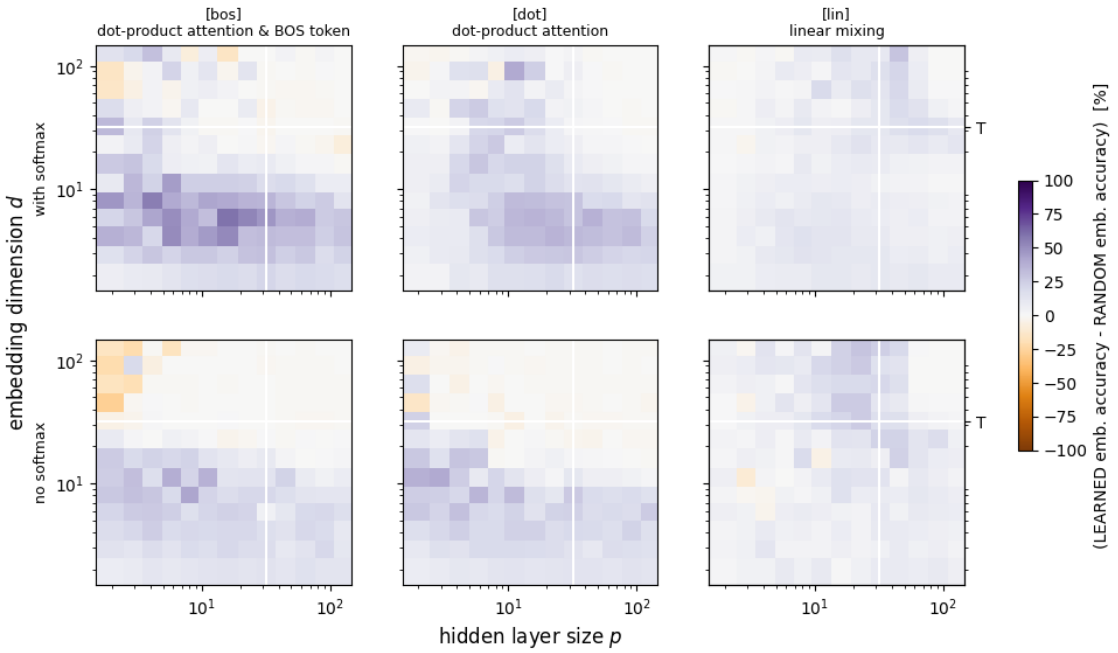

Figure 8. The difference between learned and random embeddings for $T = 32$. Orange indicates that the random embeddings perform better on average. Purple indicates that the learned embeddings perform better on average. Experimental settings as in Fig. 1.

B.3. Alternative T

We repeat the experiments presented in Fig. 1 with for $T = 64$ in Fig. 9, leading to the same phenomenology, in line with our hypothesis that indeed the number of tokens T determines the relevant transition point.

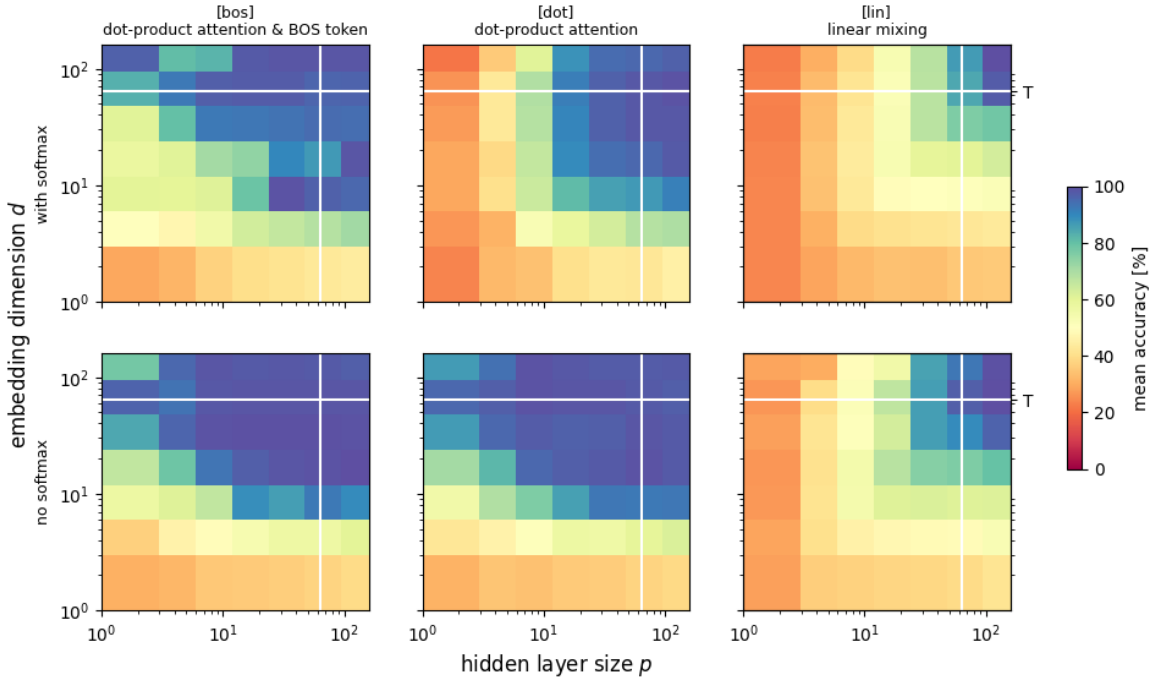

Figure 9. Phase diagrams for varying p and d for fixed $T = 64$ and $L = 10$ for `lin`, `dot` and `bos`. *(Top)* Models without softmax; *(Bottom)* Models with softmax. With five runs for every $d, p \in \{2, 4, 8, 16, 32, 64, 128\}$.

C. Parameter counts

In Fig. 10 we show how the accuracy attained by different models changes as a function of the corresponding number of parameters. The maximum achieved accuracy for a given parameter count is marked, showing that models implementing relation-based counting (`dot`, `bos`, `bos+sftm`) require fewer parameters to achieve high/perfect accuracy. In contrast, models implementing inventory-based counting (`dot+sftm`, `lin`, `lin+sftm`) need to increase their parameter count, and specifically the number of hidden neurons in the feed-forward module, to attain perfect accuracy.

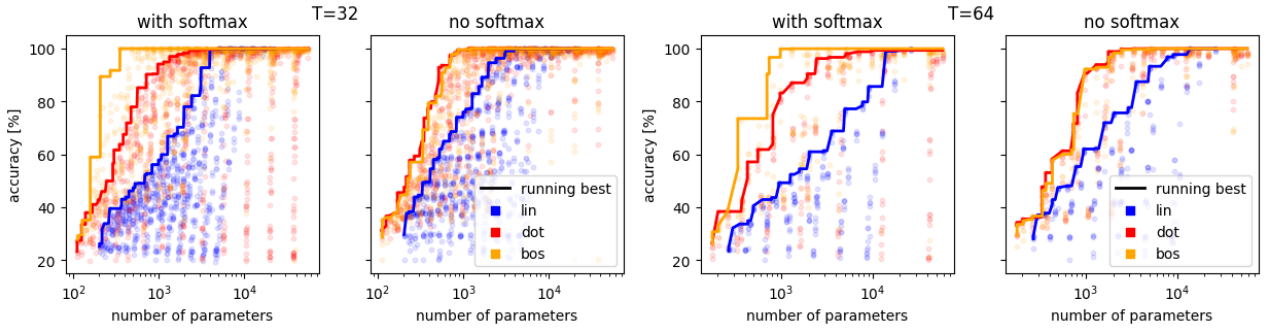

Figure 10. Comparison of the parameter counts for the model with $T = 32$ and $T = 64$ with $L = 10$ for different values of p and d taken. The data is the same as generated for Fig. 1 and Fig. 9, and every data point is a single experiment.

In Fig. 11 we compare the models with learned token embeddings with the same models but where the embeddings are

fixed from initialization. Note that these new models generally have dT parameters less than their learned counterparts. Surprisingly, some curves almost collapse onto one another, showing that the lack of learned embeddings is compensated elsewhere in the model using the same number of parameters. This is the case for all models when no softmax is used and always for linear mixing. However, `dot+sftm` and `dot` with learned embeddings outperform fixed random embeddings, indicating that learned embeddings can indeed lead to an advantage.

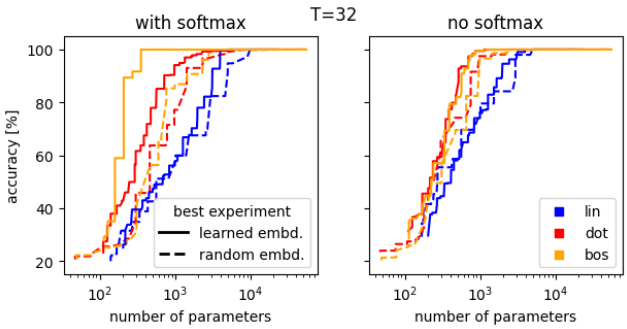

Figure 11. Comparison of the parameter counts for the model with $T = 32$ $L = 10$ for different values of p and d – for models with random and fixed embeddings or learned embeddings. The data is the same as generated for Fig. 1, and the data for fixed but random embeddings was generated in the same fashion.

D. Manual Implementations

D.1. Overview

In the supplementary code, we provide the *manual* implementation of different models on the histogram task in `pytorch`. In the parameter regime where there is always an orthonormal basis of size T in $\mathbb{R}^d$, these manual models give the correct prediction for all input token sequences. For all the models we describe below, we end up with the proxy count:

$$y_\ell = \sum_{i=1}^{p} \mathrm{ReLU}(z_{\ell,i}) = \sum_{i=1}^{p} \mathrm{ReLU}(W_1 \bar{x}'_\ell + b_1)_i \tag{7}$$

We first characterize the mapping in Eq. 7 for each mixing mechanism, and then follow with a description of the final readout map. Note that there are many symmetries due to the parameterization of the model, and we choose one for definiteness. We assume that $E \in \mathbb{R}^{d\times T}$ is an orthonormal basis of $\mathbb{R}^d$, which we will use for the token embeddings (i.e. vectors have norm 1).

D.2. Relation-based counting

`dot`: For $T \leq d; p = 1$, there are parameters for which the `dot` model solves the counting task for a sequence of length L at 100% accuracy. For example, if the embeddings of the tokens are chosen to be:

$$e_t = \tilde{e}_t + \tilde{e}_{\text{cnt}} \quad \forall t = 1, \dots T \tag{8}$$

where the set $\{\tilde{e}_t\}_{t=1}^T$ is an orthonormal basis of an arbitrary but fixed T-dimensional subspace of $\mathbb{R}^d$, and $\tilde{e}_{\text{cnt}} = \sum_{t=1}^T \tilde{e}_t$. We set the value and query matrix to the scaled identity $\sqrt{\sqrt{d}} I_d$. The first layer weights $W_1, b_1 \in \mathbb{R}^d$ can be fixed as

$$W_1 = \tilde{e}_{\text{cnt}}/(T+1)\,; \;\; b_1 = -(1 + L(T+2)) \tag{9}$$

and the second layer weights $W_2, b_2 \in \mathbb{R}^L$ follow the recursion

$$(W_2)_1 = -1 + \frac{1}{L+1}\,; \qquad (b_2)_1 = 0 \tag{10}$$

$$(W_2)_\ell = -1 + \frac{\ell}{L+1}\,; \qquad (b_2)_\ell = ((W_2)_{\ell-1} - (W_2)_\ell)\,(\ell - 0.5) + b_{\ell-1} \qquad \forall \ell = 2, \dots, L. \tag{11}$$

Given these parameters, it holds that for tokens $1 \leq t, s \leq T$ their dot-product is

$$\langle e_t, e_s \rangle = \begin{cases} 2+T & \text{if } t \neq s \\ 3+T & \text{if } t = s \end{cases}. \tag{12}$$

Because the key and query matrices are the identity, it directly follows that for tokens x_ℓ, x_m from a given sequence $\mathbf{x}$, their attention score is

$$a_{\ell m} = \begin{cases} 2+T & \text{if } x_\ell \neq x_m \\ 3+T & \text{if } x_\ell = x_m \end{cases}. \tag{13}$$

Hence, the mixed token after applying the residual connection is

$$\bar{x}'_\ell = \bar{x}_\ell + \sum_{m:x_\ell = x_m} (T+3)\bar{x}_m + \sum_{m:x_\ell \neq x_m} (T+2)\bar{x}_m \tag{14}$$

so that computing

$$\begin{aligned} \bar{x}'_\ell W_1 &= \left\langle \bar{x}'_\ell, \frac{\tilde{e}_{\text{cnt}}}{1+T} \right\rangle && (15) \\ &= \left\langle \bar{x}_\ell, \frac{\tilde{e}_{\text{cnt}}}{1+T} \right\rangle + \sum_{m:x_\ell = x_m} (T+3) \left\langle \bar{x}_m, \frac{\tilde{e}_{\text{cnt}}}{1+T} \right\rangle + \sum_{m:x_\ell \neq x_m} (T+2) \left\langle \bar{x}_m, \frac{\tilde{e}_{\text{cnt}}}{1+T} \right\rangle && (16) \\ &= 1 + hist_\ell(\mathbf{x})(T+3) + (L - hist_\ell(\mathbf{x}))(T+2) && (17) \\ &= hist_\ell(\mathbf{x}) + 1 + L(T+2). && (18) \end{aligned}$$

Then the single hidden unit has the value $y_\ell = \text{ReLU}(\bar{x}'_\ell W_1 + b_1) = hist_\ell(\mathbf{x})$. It is easy to show (analogous to Fig. 12) that the output logits $c = y_\ell W_2 + b_2$ with $c \in \mathbb{R}^L$, correctly identify the count for integer values $x \in [1, \ldots, L]$. This is because we constructed our recursion such that at a given input $x = \ell$ we have that $((W_2)_\ell(\ell - 0.5) + (b_2)_\ell = (W_2)_{\ell-1}(\ell - 0.5) + (b_2)_{\ell-1}$ and $(W_2)_\ell > (W_2)_{\ell-1}$, so it holds that

$$\underset{i=1\ldots L}{\arg\max}\, c_i(x) = \begin{cases} 1 & x = 1 \\ 2 & x = 2 \\ \ldots & \\ L & x = L\,, \end{cases} \tag{19}$$

which gives the correct classification output for all possible counts. Note however that this implementation is only one possible implementation, and many symmetries can lead to different but also 100% correct algorithms.

bos+sftm: $T = d; p = 1$**.** For the model with the beginning of sequence token, every sequence $\mathbf{x}$ if prefixed with t_{BOS} before it is fed into the embedding and then the mixing layer. We directly use the matrix E for embeddings, and set $e_{\text{BOS}} = \sum_{t=1}^{T} E_t$, where $e_t = E_t$ and the latter is a column of E. Like with the background token, we only use $p = 1$ and we set

$$W_1 = e_{\text{BOS}}; \;\; b_1 = -1. \tag{20}$$

For a given token x_ℓ we have that in the dot-product mechanism $\langle e_{\text{BOS}}, e_{x_\ell} \rangle = 1$, $\langle e_{x_\ell}, e_{x_m} \rangle = 1$ if $x_m = x_\ell$ and 0 otherwise. Due to the softmax, the mixing coefficient is $a = e/((k_{x_\ell} + 1)e + (L - k_{x_\ell}))$ for comparing x_ℓ to t_{BOS} and to all the tokens where $x_\ell = x_m$, and $b = 1/((k_{x_\ell} + 1)e + (L - k_{x_\ell}))$ otherwise. Hence, the mixed token is :

$$\bar{x}'_\ell = a e_{\text{BOS}} + a k_{x_\ell} \bar{x}_\ell + \sum_{x_m \neq x_\ell} b \bar{x}_m + \bar{x}_\ell \tag{21}$$

Applying W_1 and b_1, we obtain:

$$\begin{aligned} y_\ell &= aT + a k_{x_\ell} + b(L - k_{x_\ell}) = \\ &= aT + a k_{x_\ell} + 1 - a(k_{x_\ell} + 1) = \\ &= a(T-1) + 1 \end{aligned} \tag{22}$$

since $(k+1)a + (L-k)b = 1$ by normalization. Reordering the terms gives the count, which can be implemented in a discrete fashion as shown for the `dot+sftm` later.

`bos`: $T = d; p = 1$. By equivalence of the models, this model can either ignore t_{BOS} by setting it to zero, and implement the same mechanism as `dot` slightly modifying the final mapping to $hist_{\mathbf{x}}(x_\ell)$ *or* it can implement the mechanism of `bos`, but adapt the mapping so that it works without the softmax.

D.3. Inventory-based counting

`lin`: $T = d = p$. As embeddings we directly use E, s.t. $e_t = E_t$ where the latter is a column of E. We set

$$\mathbf{A}_{\text{lin}} = \begin{bmatrix} 2a & a & \cdots & a \\ a & 2a & & \\ \vdots & & \ddots & \\ a & & & 2a \end{bmatrix}; \quad W_1 = E; \quad b_1 = -(1+a), \tag{23}$$

where $a = 1/(L+1)$. Let's start by writing $z_{\ell,t}$ for $t \in \{1, ..., p\}$, the t-th activation of the first hidden layer of the feed-forward module:

$$z_{\ell,i} = \sum_{m=1}^{L} a_{\ell m} \langle e_{x_m}, e_t \rangle + \langle e_{x_\ell}, e_t \rangle - (1+a) \tag{24}$$

If $e_t = e_{x_\ell}$, we have:

$$z_{\ell,i} = 1 + 2a + (k_{x_\ell} - 1)a - 1 - a = ak_{x_\ell} \tag{25}$$

where, $k_{x_\ell} = hist_{\mathbf{x}}(x_\ell)$, applying the ReLU to this scalar keeps its value unchanged. If $e_t \neq e_{x_\ell}$, we have:

$$z_{\ell,i} = ak_{e_t} - 1 - a = a(k_{e_t} - 1) - 1 \tag{26}$$

The right hand side of the above equation is negative given our choice of a, hence applying the ReLU returns 0. This means that, for each token in the input sequence, the contributions of orthogonal tokens cancel, leaving us with a single hidden hidden neuron activated. The counts can be easily read-off from this activation via a linear projection.

`lin+sftm`: $T = d = p$. Same as `lin`, except

$$\mathbf{A}_{\text{lin+sftm}} = \begin{bmatrix} 2a & a & \cdots & a \\ a & 2a & & \\ \vdots & & \ddots & \\ a & & & 2a \end{bmatrix} = \text{softmax}\left(\begin{bmatrix} \beta & \alpha & \cdots & \alpha \\ \alpha & \beta & & \\ \vdots & & \ddots & \\ \alpha & & & \beta \end{bmatrix} \right), \tag{27}$$

where $a = 1/(L+1)$ which implicitly defines a choice of β, α.

`dot+sftm`: $T = d = p$. We set $Q = \mathbf{I}_d$ and $K = \sqrt{d}\mathbf{I}_d$, where the factor $\sqrt{d}$ reverts the normalization given by convention in the $\mathbf{A}_{\text{dot}}$ and $\mathbf{A}_{\text{dot+sftm}}$ functions. In this model, we directly use the orthonormal basis E for the token embeddings. The pre-softmax mixing weights will be 1 for equal and 0 for different tokens due to the unit-norm token embeddings. If $k_{x_\ell} = hist_{\mathbf{x}}(x_\ell)$, then, after applying the softmax, we have $a_{lm} = \frac{e}{(L-k_{x_\ell})+ek_{x_\ell}}$ if $l = m$ and $a_{lm} = \frac{1}{(L-k_{x_\ell})+ek_{x_\ell}}$ otherwise. Hence, for $e_t \neq e_{x_\ell}$:

$$\langle \bar{x}'_\ell, e_t \rangle = \frac{k_t}{(L-k_\ell) + ek_\ell} < 1, \tag{28}$$

while for $e_t = e_{x_\ell}$:

$$\langle \bar{x}'_\ell, e_{x_\ell} \rangle = k_\ell e / (k_\ell e + (L - k_\ell)) + 1 \tag{29}$$

where the extra summand comes from the residual connection. Hence, by setting

$$W_1 = E; \quad b_1 = -1, \tag{30}$$

and applying the ReLU activation, Eq. 28 will be 0, while Eq. 29 will implicitly give us the counts as:

$$k_{x_\ell} = (Ly_\ell)/(-ey_\ell + y_\ell + e) \tag{31}$$

While the final layer cannot immediately implement non-linear functions in y_ℓ, it can take advantage of the fact that y_ℓ can take only L different values. Since eventually we need to map the L values of y_ℓ to the counts $[1, \cdots, L]$ the linear output layer is sufficient to implement this non-linear discrete map (see Fig. 12 which shows exactly this example, solid lines are y_ℓ, lying in intervals bounded by dashed lines).

D.4. Mapping a scalar to a categorical one-hot encoding

It is straightforward to map a single scalar α to a series of neurons which activate one after another. This is needed as the second part of the feed-forward configuration to transform the count measured by the sum of the hidden neurons y_ℓ to the discrete categorical representation of the output vector. Every output vector can be viewed as a linear function of the hidden neuron's value. Since we only map functions, where a given output neuron needs to be maximized in a single fixed and uninterupted interval $[a, b] \in R$, the mapping can work as a superposition of linear functions with increasing slope. A visual sample is given in Fig. 12.

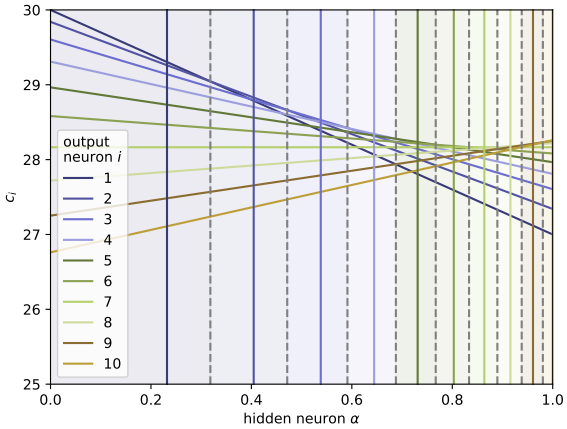

Figure 12. Demonstration of a hidden neuron output α which is mapped to 10 different neurons using a single output layer, according to the decision boundaries shown by the dotted lines. After applying the argmax function to the 10 output neurons, the highest value gives the discrete output. The solid lines mark the values a single hidden neuron would achieve for different counts in the `dot+sftm` model.

E. BOS mixing token

In Fig. 2 in the main, we describe how the t_{BOS} is the main predictor for the count. Here, we provide more evidence by showing how the count predictions for mixed tokens $\bar{x}'$ output by the feature transform f are invariant to the type of other token present in the mixed token. The results for four different tokens are shown in Fig. 13.

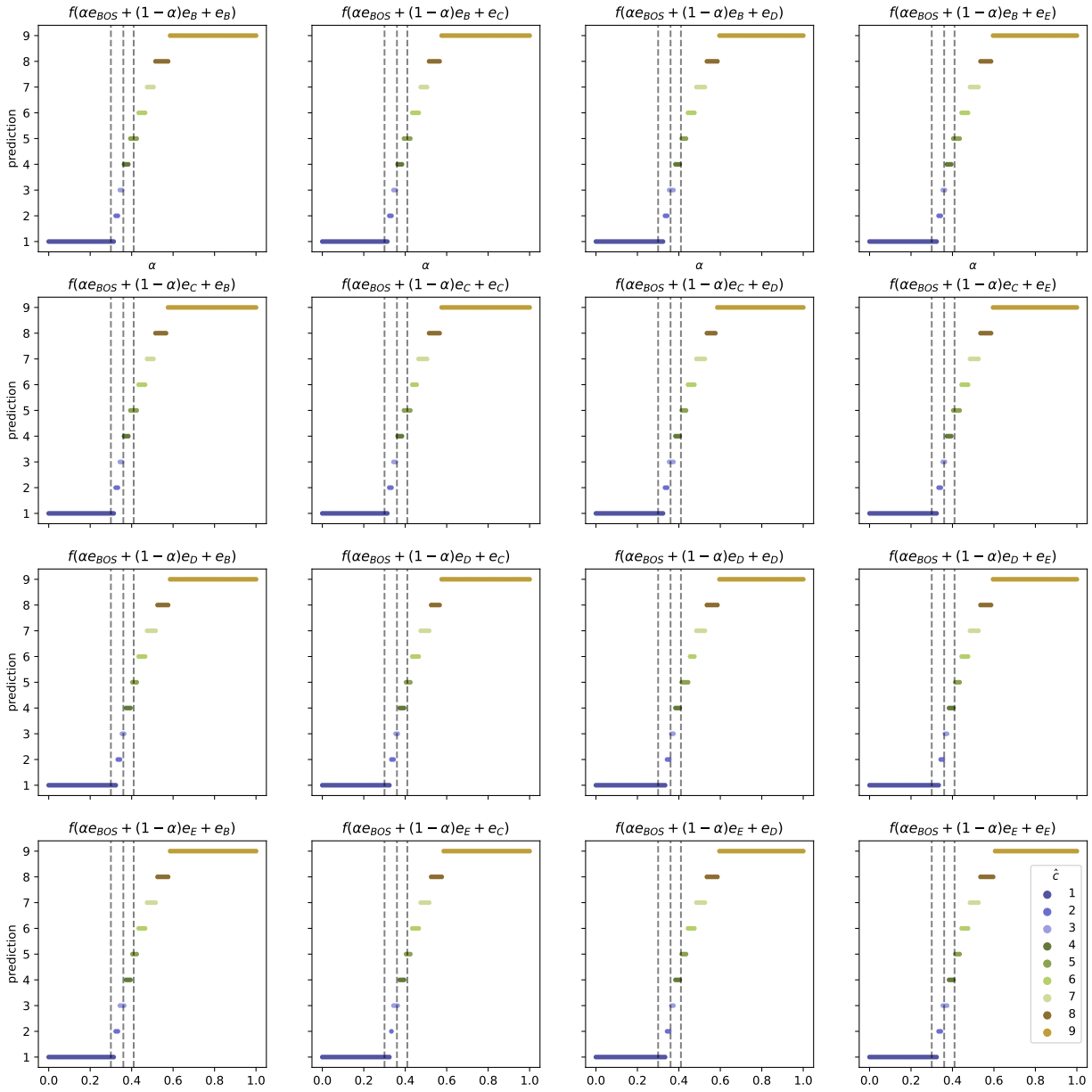

Figure 13. For the same model as in Fig. 2, we vary the inputs to the feature transformation f to show it is independent on the precise input sequence, but only depends on the prevalence of t_{BOS}. We vary the inputs between the learned tokens $[B, C, D, E]$.