# Understanding Counting in Small Transformers: The Interplay between Attention and Feed-Forward Layers

**Freya Behrens** [1] **Luca Biggio** [1] **Lenka Zdeborová** [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 solution a model can implement depends non-trivially on the precise choice of the attention 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. From a broader perspective, our analysis offers a framework to understand how the interaction of different architectural components of transformer models shapes 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 in unleashing 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 token-mixing attention mechanism and a standard fully connected 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 transformers, there is no clear consensus on the distinct roles of their components, how they interact, or if they can be substituted with alternative modules (Tolstikhin et al., 2021; Bozic et al., 2023; Gu and 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; Ouellette 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; Ouellette et al., 2023) by considering the histogram task as a prototypical problem (Weiss et al., 2021). This task consists of 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 attention mechanism and a multi-layer perceptron.

In our analysis, we provide explicit constructions (parameter configurations) for a range of such architectures reaching perfect accuracy in a model-dependent hyperparameter regime. In a subsequent 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 strikingly different solutions for the histogram task, with a strong dependence on the scale of the model's hyperparameters and the type of token-mixing mechanism utilized. In particular, we identify two main algorithmic strategies a model can implement to succeed in the histogram task, which we term *relation-* and

[1]Statistical Physics of Computation Laboratory, École polytechnique fédérale de Lausanne (EPFL), CH-1015 Lausanne. Correspondence to: Freya Behrens <freya.behrens@epfl.ch>.

$2^{nd}$ *Workshop on Mechanistic Interpretability @ICML*, Vienna, Austria. 2024. Copyright 2024 by the author(s).

*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, the 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 done by the token-mixing attention module. In line with the solution identified by Weiss et al. (2021) for a one-head, one-layer architecture with a BOS token, we find that relation-based counting is more generally realized through the formation of an appropriate counter direction within the token embedding space.

- **Inventory-based counting**, instead, can be implemented based on an input-independent token-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 cost of increased memory and compute requirements.

- We study under which conditions both mechanisms emerge through training. In particular, we verify the feasibility regimes, identified for each mechanism from the explicit constructions, within the parameter space determined by each architecture's two primary hyperparameters: the embedding dimension and the size of the hidden layer in the feed-forward block.

- When the embedding dimension is comparatively smaller than the alphabet, we show that non-orthogonal embeddings can still result in models attaining perfect accuracy. Remarkably, we show that the softmax operator enables to effectively disentangle distinct tokens, hence allowing models with small hidden dimension to perform optimally.

In Section 2 we discuss the related literature and in Section 3 we provide the necessary background and introduce our notation. In Section 4 we describe our experimental setup and in Section 5 we present our theoretical and experimental results[1]. Sections 6 and 7 present the limitations, conclusion and open questions of our work.

## 2. Related Work

**Mechanistic Interpretability and Counting.** The emergence of algorithmic capabilities in transformers (Olsson et al., 2022; Power et al., 2022) has led to numerous investigations aimed at reverse-engineering trained models into human-understandable mechanisms (Zhong et al., 2023; Nanda et al., 2023; Quirke and Barez, 2024). Previous work has investigated a variety of counting tasks and mechanisms (Gould et al., 2023; Chollet et al., 2020; Ouellette et al., 2023; Cui et al., 2024). We consider the histogram task which was introduced in the context of the RASP(-L)-language (Weiss et al., 2021; Abbe et al., 2023). Weiss et al. (2021) predict that single layer transformers with one head require an additional BOS token as a scratchpad (Nye et al., 2021) to be able to solve the task. However, we find that the task does not necessarily require the BOS token and we give explicit constructions for several of such one-layer architectures. Our main focus is the interpretation of the hyperparameter scaling of several distinct models in relation to their performance and explicit constructions of different algorithms, similar to the studies in (Zhong et al., 2023; Quirke and Barez, 2024). We give precise theoretical conditions on the model and task that lead to perfect explicit constructions. While many works in this area focus on causal interventions (Vig et al., 2020; Meng et al., 2023) to gain insights into the model's computational mechanisms or attributes relevance scores to model's components (nostalgebraist, 2020; Elhage et al., 2021), we mainly gain intuition through the direct introspection of the models components.

**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 transitions from storing data points to discovering generalizing 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 single direction in embedding space, the counting subspace.

**Algorithmic Alignment.** While theoretical work has outlined the computational capacity 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

[1]All results and code to reproduce them is available at https://github.com/SPOC-group/counting-attention.

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 discovered evidence for the superposition of different computational graphs in a single model (Elhage et al., 2022) – we complement this analysis with a toy model that is able to disentangle non-orthogonal, hence superimposed, embedding directions in some parameter regimes.

## 3. 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

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

with the token mixing matrix $\mathbf{A} : \mathbb{R}^{L\times d} \to \mathbb{R}^{L\times L}$ and the token-wise feature transformation $f : \mathbb{R}^d \to \mathbb{R}^C$. The embedding $\bar{\mathbf{x}} \in \mathbb{R}^{L\times d}$, where $\bar{x}_\ell$ denotes its $\ell$-th row, is obtained by passing the input sequence $\mathbf{x}$ into a standard embedding layer (learnable lookup-table) 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 $x_\ell$. We do not include positional embeddings due to the due to the inherent permutation equivariance of the counting task. The input to the feature transformation, $\bar{x}'_\ell$, is obtained by first combining the tokens via the mixing matrix $\mathbf{A}$ and then adding back the original tokens, $\bar{x}_\ell$, through a residual connection. We will refer to the vector $\bar{x}'_\ell$, for each position $\ell = 1, \cdots, L$, as the mixed token. The final prediction of the model for each position 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* (`lin`), 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* (`dot`), 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. Note that, for the sake of simplicity, we assume the value matrix to be the identity. The number of parameters for dot-product mixing is $2d^2$.

In line with previous work (Weiss et al., 2021), for architectures employing the dot-product mixing, we also analyze models utilizing the so-called beginning-of-sequence (BOS) token. This special token, indicated with the symbol \$, is appended to the original input $\mathbf{x}$ resulting in a new sequence $\tilde{\mathbf{x}} = (\$, x_1, x_2, \cdots, x_L)$ of length $L + 1$. We will refer to the architecture that includes the BOS token as `bos`.

**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 identically to every mixed token $\bar{x}'_\ell$ for $\ell = 1, \cdots, L$, as:

$$f(\bar{x}'_\ell) = \text{ReLU}(\bar{x}'_\ell W_1 + b_1)W_2 + b_2 \quad (6)$$

where $f(\bar{x}'_\ell) : \mathbb{R}^d \to \mathbb{R}^C$ 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 $b_1 \in \mathbb{R}^C$.

## 4. 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 a token $t$ in the sequence $\mathbf{x}$ at position $\ell$ as $hist_{\mathbf{x}}(\ell)$. 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. C 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 data samples 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. 3, i.e. `lin` and `dot`, with or without the softmax (`+sftm`), where the token embeddings are jointly learned with the model parameters. Their relevant hyperparameters 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 the BOS token prior to entering a dot-product mixing layer (both with or without softmax). Previous studies (Weiss et al., 2021; Kazemnejad et al., 2023) 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.

## 5. Learning Regimes in Counting

We analyze the performance of the above-stated 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 accuracy attained by different learned models for $L = 10$ and $T = 32$.

We observe that the models exhibit both high and low accuracy across various parameter regimes. In App. D, we provide additional experiments for $T = 64, L = 15$, confirming the same phenomenology. Our objective in the following is to investigate the underlying reasons for these variations and how they are influenced by the model architecture.

We first study the regime where the model dimension $d$ is at least as big as the number of tokens $T$. In this case, we demonstrate how explicit constructions can be theoretically designed for a variety of models to achieve perfect accuracy. We begin our analysis by demonstrating that the BOS token can represent a counter subspace, in line with prior research (Kazemnejad et al., 2023; Weiss et al., 2021). We then demonstrate that the BOS token is not always necessary: other 1-layer architectural choices not employing this special token can indeed solve the histogram task. We classify them into two sub-classes: relation-based and inventory-based counting.

In *relation-based counting*, the model uses a dot-product mixing mechanism and a counting subspace, requiring only a low-capacity feed-forward hidden layer to project the mixed tokens onto this subspace – akin to the model with the BOS token.

Conversely, in *inventory-based counting*, the model memorizes token embeddings in the first layer parameters of the multi-layer perceptron, necessitating more hidden neurons and hence greater capacity.

We demonstrate that linear mixing adopts the latter strategy due to the absence of a token-dependent mixing mechanism for comparing token embeddings. Interestingly, in `dot+sftm`, the use of the softmax operator hinders the formation of a privileged counting direction (necessary for relation-based counting), forcing the model to rely on a greater number of hidden neurons in the feed-forward block and resorting to inventory-based counting. We show that these explicit constructions correlate with the accuracies achieved through supervised training in Fig. 1. In addition, Fig. 2 illustrates the performance of the considered models as a function of their parameter count, showcasing the superiority of relation-based counting in terms of efficiency. In particular, 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.

In the second learning regime, when $d < T$, the assumption that different tokens can be represented by orthogonal vectors is no longer valid. However, we find that the considered models are able to tolerate the noise stemming from non-orthogonality and still manage to solve the task perfectly. The extent to which this is possible depends on the specific model architecture and the size of the hidden layer of the feed-forward module. The characterization of how each model effectively process non-orthogonal embedding enables us to further extend the feasibility regimes for the histogram task.

### 5.1. $d \geq T$: 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 $e_t \in \mathbb{R}^d$

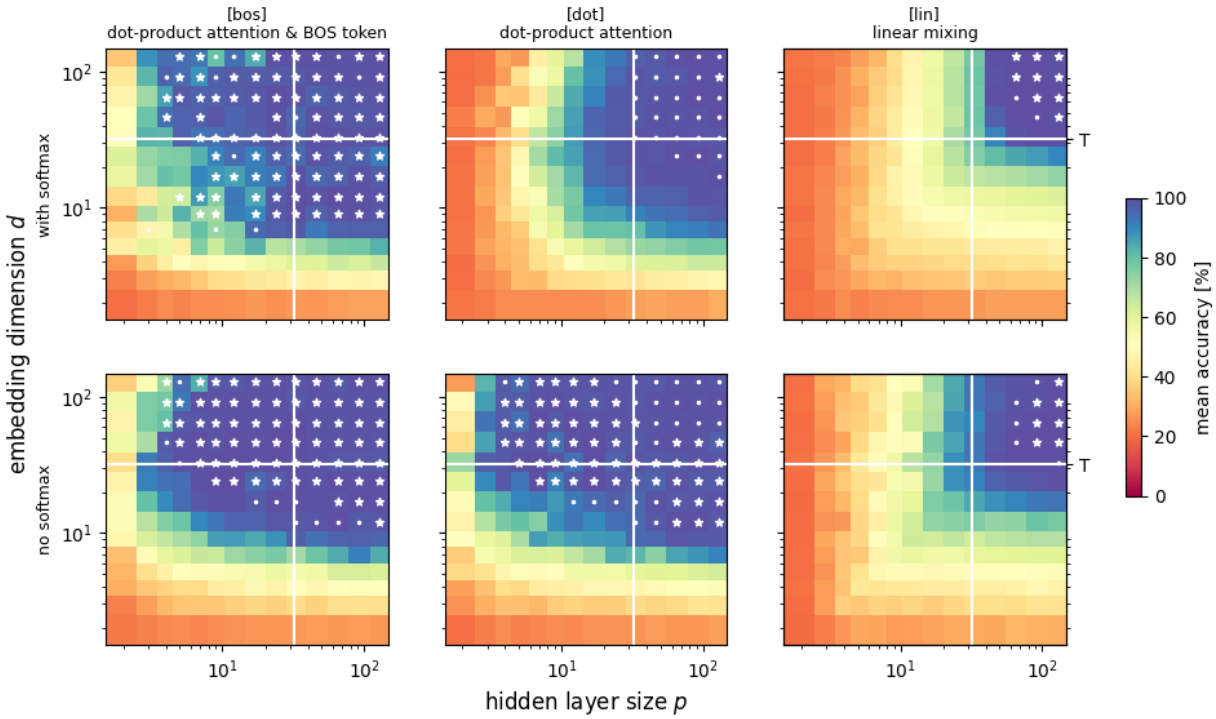


*Figure 1. Performance on the histogram task for different 1-layer transformer architectures.* Mean accuracy for varying embedding dimension $d$, hidden layer dimension $p$, for fixed $T = 32$ and $L = 10$ for the different token mixing mechanisms `dot`, `bos` and `lin`. *(Top)* Models with softmax; *(Bottom)* Models without softmax. Average over 5 runs for every $d, p \in \{1, 2, 3, 4, 6, 8, 12, 16, 23, 32, 45, 64, 91, 128\}$. 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.

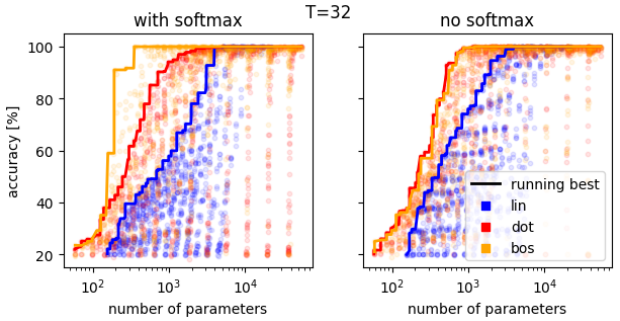


*Figure 2. Accuracy vs. Parameter count.* The data is the same as generated for Fig. 1, and every data point is a single experiment.

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 *theoretically* construct the weights of a model for each mixing mechanism to solve the histogram task for all possible sequences when $d \geq T$ and that this is reflected in the models learned from data. Remarkably, the different mixing mechanisms require a different number of hidden neurons $p$. 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 (requiring $p \geq T$), and for others, relation-based counting (where $p \geq 1$ is sufficient).

**Relation-based counting: The BOS direction acts as a counter in a low-dimensional subspace.** When an extra beginning-of-sequence token $t_{\text{BOS}}$ is available in `bos`, it can be used as a to extract information about a token's count $hist_{\mathbf{x}}(\ell)$ in the attention layer of the network through its attention score. In the literature, the 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 count-

ing algorithm can be theoretically implemented in these two architectures by construction.

When we set the $t_{\text{BOS}}$ 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 $t_{\text{BOS}}$ is at the first position of the sequence of now length $L+1$, a clean dot-product operation in the attention mechanism (with $Q, K = d^{\frac{1}{4}} \mathbb{I}_d$) 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 $t_{\text{BOS}}$, means that $\langle \bar{x}'_\ell, e_{\text{BOS}} \rangle = T + hist_{\mathbf{x}}(\ell) + 1$, i.e. the embedding $e_{\text{BOS}}$ can be taken as the single relevant direction that reveals the count of $x_\ell$. 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. A.2).

Investigating learned models we find instances achieving 100% accuracy that, although not exactly corresponding to the algorithm we prescribed theoretically, exhibit a close correspondence up to symmetries. In Fig. 3, we show for an example of `bos+sftm`, that $t_{\text{BOS}}$ 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 $x_\ell$, as it has different values for tokens that occur a different amount of times. We emphasize here that 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. We also show in Fig. 3 how the presence of the $t_{\text{BOS}}$ 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` in the 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. A, one can add a single common direction $e_{\text{cnt}}$ to all previuosly orthogonal token embeddings. Using a clean dot-product mixing, this leads to $a_{\ell m} = a_{\neq}$ when $x_\ell$ 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. $\langle e_{\text{cnt}}, \bar{x}'_\ell \rangle \propto 1 + hist_{\mathbf{x}}(\ell) a_{=} + (L - hist_{\mathbf{x}}(\ell)) a_{\neq}$. We can, therefore, obtain a perfect accuracy implementation in the regime where $d \geq T$ with only a 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. 4. 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 the 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 by adding the extra token with a special functionality that 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 an 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. A.3.

**Inventory-based counting: Memorization in the feature transform.** For `lin` we can prescribe weights theoretically that achieve 100% accuracy when $d \geq T$ and $p \geq T$, 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. A.3.

Again, for the given model parameterizations, several solutions exist due to symmetries, and in the following we describe one of them for clarity. In the linear mixing layer,

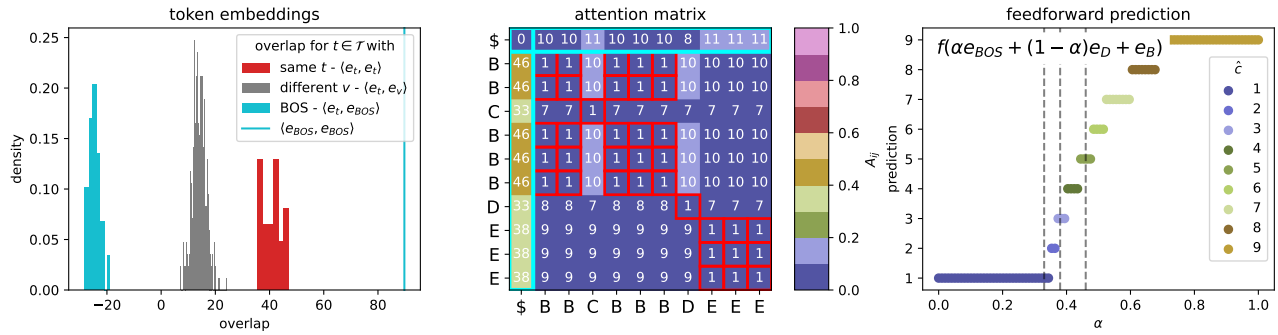


*Figure 3. Relation-based counting with* `bos+sftm` *($T = 32, L = 10, p = 2, d = 45$).* This model achieves 99.9% accuracy. It was selected as the best model from all our experiments with $p = 2$. *(Left)* The tokens overlap (cosine similarity) with the same tokens (red), different tokens (grey) and the BOS (light blue) all concentrate on different values. *(Middle)* This is reflected in the attention matrix after the application of the row-wise softmax. The $t_{\text{BOS}}$ ('\$') in the first column $a_{\ell,0}$ becomes a proxy for the count of $x_\ell$. *(Right)* To demonstrate that the feedforward network is only sensitive to this direction, we show its count predictions for a mix of tokens $\alpha e_{\text{BOS}} + (1-\alpha)e_D + e_B$, where the contribution $\alpha$ of the BOS token is varied and $D, B$ are two specific elements of the alphabet $\mathcal{T}$. The same experiment is repeated for different elements of the alphabet in App. D.6. We mark the $a_{\ell,0}$ obtained from the left as vertical lines, the prediction is correct for all counts independent of the precise token.

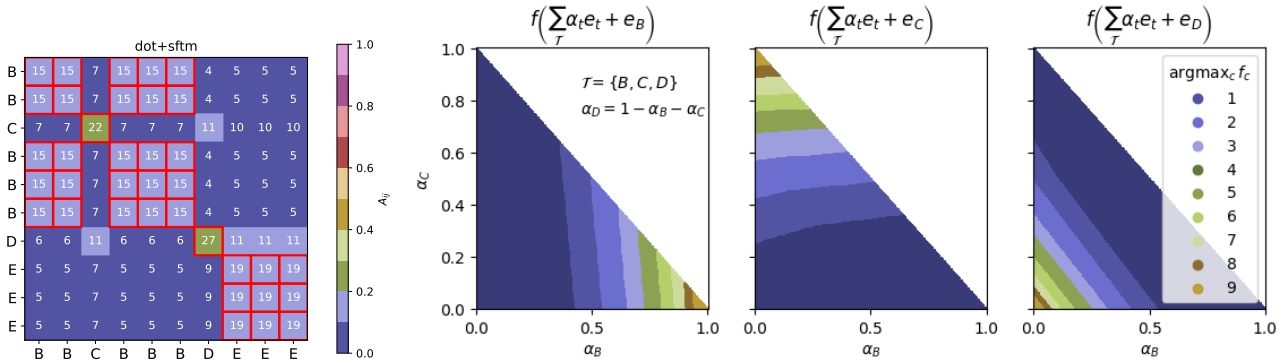


*Figure 4. Inventory-based counting with* `dot+sftm` *($T = 32, L = 10, p = 32, d = 32$).* This model achieves 99.47% accuracy. *(Left)* The attention matrix for a given sequence differentiates between similar and different tokens. However in this case, any counting direction that could emerge in token space is evidently not usable, as $p \geq T$ is required (see Fig. 1). *(Right)* This is reflected in the output from the feature transformation $f$, shown here for a linear combination of three different tokens from the alphabet, $B, C, D$. The prediction strongly depends on the coefficient $\alpha_t$ associated with the token $t$ present in the residual connection and only weakly on the others. The non-linear scaling of the decision boundaries is due to the softmax activation function.

we set the mixing weights for different positions $\ell, m$ 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 $x_\ell$ can therefore be extracted, via:

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

By design, only the hidden neuron activated according to the overlap with token $e_t = x_\ell$ – the token associated with the residual connection – has a non-zero activation. This equation can exactly be implemented by the first layer of the feature transformation in equation 6 with $p \geq T$ hidden neurons. The output layer can then be designed to activate the correct output vector corresponding to the count $hist_{\mathbf{x}}(\ell)$ (refer to the visualization in App. A.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. 5, 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 $\sim 0.11$ and a diagonal of $\sim 0.08$. 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

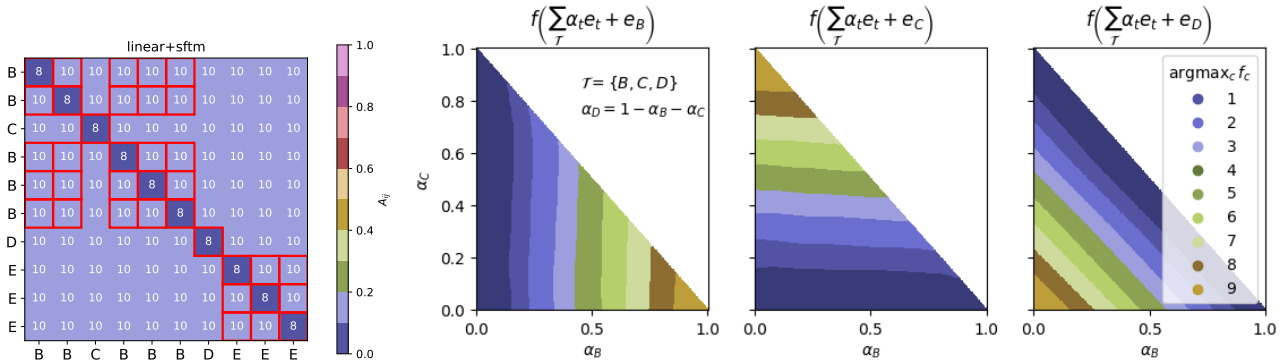


*Figure 5. Inventory-based counting with* `lin+sftm` *($T = 64, L = 10, p = 128, d = 128$).* The model achieves 99.97% accuracy. *(Left)* Attention matrix learned by Adam, which is constant in the input sequence $\mathbf{x}$. The different score on the diagonal assigns a different weight to the token at the current position $\ell$ than to all other tokens. *(Right)* Predictions on an 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.

$\alpha_t$ corresponding to the token embedding fed through the residual connection. Notably this behavior is close to Fig. 4 (Right) and 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 in principle be implemented.

To investigate this point, in Fig 6 we show the singular value decomposition of $W_1$, for models where both $p \geq T$ and $d \geq T$, for $T = 32$ with at least 99% accuracy. For models that can *only* implement inventory-based counting, the ordered singular values visibly dip after the 32nd largest singular value. As inventory-based counting requires the feed-forward hidden layer to have at least $T = 32$ neurons to detect the complete alphabet, this observation supports that the learned model is aligned with our explicit construction. Conversely, for models capable of implementing both inventory-based and relation-based counting, this decline is less marked. However, some model instances, such as in `bos+sftm`, exhibit a singular value distribution that deviates from the typical pattern for this model and closely resembles the distribution seen in inventory-based-only models. This suggests that these exceptional models may have adopted a different mechanism, potentially integrating elements of both algorithmic strategies. Nevertheless, this does not give a clear understanding of whether the memory-intensive solution is preferred when the memory is available. 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.

### 5.2. $d < T$: Entangled Embeddings

**Non-orthogonality and the discrete nature of counting.** The scenario where $d < T$ fundamentally differs from the one explored in Section 5.1 because the embeddings for different tokens can no longer be orthogonal. This results in some token pairs having a non-zero overlap due to their linear dependence, causing the mixing of tokens to entangle count information across different directions in the embedding space. This phenomenon is illustrated in Fig. 15 (Appendix, bottom row) where it is shown empirically that the overlaps between different tokens indeed become continuously more spread out in `dot` (hence resulting in more noise) as we decrease the dimension $d$, which is accompanied by more errors in the prediction in Fig. 15 (Appendix, top row).

This key difference prevents us to apply the same rationale used to derive the explicit constructions described in the previous section.
Nevertheless, the discrete nature of the histogram task, i.e. the fact that every token can only be mapped to $L$ different counts, makes the prediction inherently more robust to the effect of noise stemming from entanglement. Consider for example the token $x_l$ appearing $k_{x_l}$ times in the input sequence. Assuming $p = 1$ without loss of generality, for a model capable of solving the histogram task for $d = T$, we would like the scalar associated with the hidden node of the feed-forward module, to be easily transformed into a $L$ dimensional vector of logits whose maximum entry

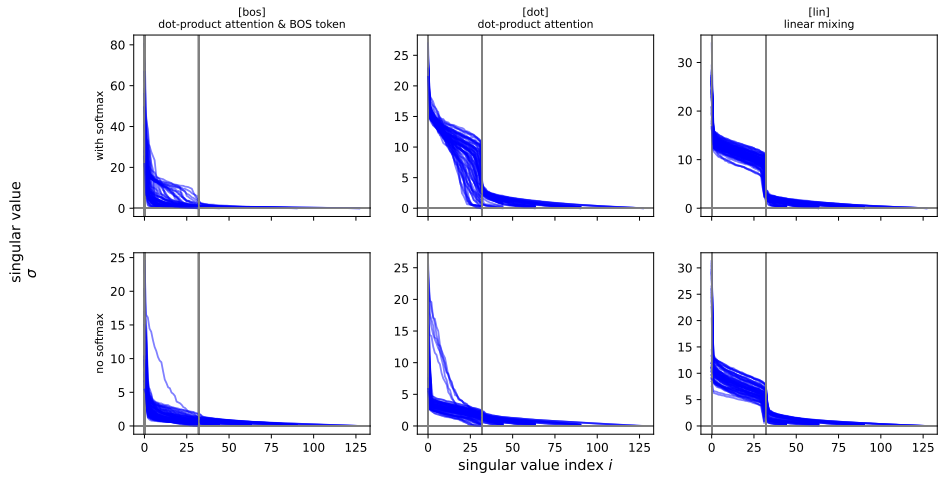


*Figure 6. Singular values of $W_1$.* We show the results for all models from Fig. 1 with $T = 32$, where $p, d \geq T$ and the accuracy is at least 99%. Some qualitative differences are visible for `bos` and `dot`.

corresponds to the true count. The latter transformation, implemented by the last layer of the feed-forward module and the argmax operator, will then act as a mapping outputting the final counts as a function of the value of the hidden node. It turns out that, in light of the discrete nature of the task, such a function will be associated with a decision boundary characterized by some margins separating the different classes (counts). This concept is illustrated in Fig. 7 (Appendix) for the `dot+sftm` model. This means that, as long as the value of the hidden node falls within the margin determined by the decision boundaries and these boundaries do not overlap, the model can still solve the task with perfect accuracy. The variability of the hidden node's value within the margin is a result of the perturbation effect caused by the non-orthogonality of the embeddings.

Building on this intuition, in Appendix B, we derive explicit constraints on the embedding dimension $d < T$, such that the histogram task can still be solved with 100% accuracy. Our analysis is based on two key points: first, the mutual coherence of a set of $T$ unit-norm $d$ dimensional vectors has a lower bound known as the Welch bound, which depends on $d$ and $T$; second, the softmax operator, when rescaled by an inverse temperature parameter, naturally concentrates around the largest values in its argument. Detailed derivations are provided in Appendix B. Below, we summarize the main concepts and results of our analysis.

**Mutual Coherence and Welch bounds.** The *mutual coherence* $\mathcal{M}$ of a set of $T$ unit norm vectors $\{v_1, \ldots, v_T\} \subset \mathbb{R}^d$ is defined as

$$\mathcal{M} = \max_{i \neq j} |\langle v_i, v_j \rangle| \,. \tag{7}$$

In words, this quantity measures the maximal similarity between pairs of vectors within a specified subset of $\mathbb{R}^d$. This value is lower bounded for a given matrix by the Welch bound (Welch, 1974)

$$\mathcal{M} \geq \sqrt{\frac{T-d}{d(T-1)}} = \mathcal{W}(T,d) \,, \tag{8}$$

and equality can only be attained if $T < d^2$ (Strohmer and Heath, 2003). By using this bound and the above-stated observation that decision boundaries associated with individual counts should not overlap to attain perfect accuracy, we are able identify less stringent feasibility regimes for some of the considered models. Notably, such regimes vary depending on the type of mixing mechanism and are summarized below:

$$(\texttt{lin}, \texttt{lin+sftm}; p = T)\text{:} \ \left\lceil \frac{T(2L-3)^2}{T-1+(2L-3)^2} \right\rceil \leq d,$$

$$(\texttt{dot}, \texttt{bos}; p = 1)\text{:} \ \left\lceil \frac{T(2L-3)^2}{T-1+(2L-3)^2} \right\rceil + 1 \leq d,$$

$$(\texttt{dot}, \texttt{bos}; p = T)\text{:} \ \left\lceil \frac{T(2L-3)}{T-1+(2L-3)} \right\rceil \leq d.$$

By evaluating the obtained bounds for the values of $T$ and $L$ in Fig. 1, we observe that the feasibility regimes exhibited by learned models are very close to our theoretical predictions.

**The softmax operator increases the separation between distinct tokens.** Recall the softmax function is defined as

$$\text{sftm}(\kappa \mathbf{z})_i = \frac{e^{\kappa z_i}}{\sum_{j=1}^{n} e^{\kappa z_j}} \quad \text{for} \quad i = 1, 2, \ldots, n \,, \tag{9}$$

where $\kappa > 0$ is the so-called inverse temperature parameter (whose default value is 1). When $\mathbf{z}$ of length $L$ contains only different two values, one with $k$ and the other with

$L-k$ occurrences, then as $\kappa \to \infty$ the mass concentrates only on the larger value of the two, and sets the other to zero. We use this intuition to create token embeddings that fulfill for all $t, s = 1, \ldots, T$ and $s \neq t$

$$\langle e_t, e_t \rangle = 1\,, \tag{10}$$

$$\langle e_t, e_s \rangle < 1 + \epsilon\,, \tag{11}$$

where $\epsilon > 0$. The idea is that the softmax with a high enough inverse temperature sets the term for distinct tokens, $\langle e_t, e_s \rangle$, *close enough* to zero, essentially eliminating the noise. Note that equation 10 is a weaker condition on the set of token embeddings than for example the bound of the mutual coherence in terms of the sequence length $L$ `bos` with $p = 1$ in Section B.2.2. It allows us to obtain perfect accuracy with smaller $d$. In particular:

(`bos+sftm`; $p = 1$): $d \geq \lceil \log_2(T+1) \rceil + 2$.

(`dot+sftm`; $p = T$): $d \geq \lceil \log_2(T+1) \rceil + 2$.

Again the newly identified feasibility regimes are in good agreement with our empirical results in Fig. 1. In Appendix B.3.1 we show that this bound can be even further improved, but at the cost of increasingly precise computations as $T$ and $L$ grow, and seems therefore unrealistic.

## 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 or positional encodings. 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; Engels et al., 2024). Given its specificity, it is unclear if and how similar memory-architecture phenomena would emerge for different simple tasks (e.g. sorting or lookup).

## 7. Conclusion

We study how different components of simple transformer models contribute to the emergence of different solutions to the histogram task. Our analysis shows that the hyperparameter space with perfect accuracy for these models is influenced by the choice of the mixing mechanism and its inter-dependency with the feed-forward transformation. We identify two distinct algorithmic approaches that 1-layer transformers can utilize to solve the histogram task: relation-based counting and inventory-based counting. The relation-based method employs a dot product mixing mechanism combined with a low-capacity feed-forward transformation and relies on the presence of an appropriate counter direction within the token embedding space. In contrast, the inventory-based method involves memorizing the token embeddings within the feed-forward module's weights, thus requiring more parameters. By characterizing the feasible regimes of these mechanisms in the phase space defined by the embedding dimension $d$ and the hidden dimension $p$ of the feed-forward module, we confirm that learned models converge to solutions resembling these mechanisms. In certain regimes, both strategies can potentially be implemented, and our experiments indicate that some learned models exhibit features of superimposed algorithmic mechanisms. In the regime where the embedding dimension $d$ is smaller than the vocabulary size $T$, tokens cannot form an orthogonal basis and solve the task directly via a linear projection. Despite this, we find that the considered models exhibit different levels of robustness to the noise stemming from non-orthogonality. Our analysis precisely characterizes how different models cope with this aspect and identifies less stringent feasibility regimes in terms of the embedding dimension. Furthermore, we find that the softmax activation can be particularly effective in maximizing the distance between distinct tokens, hence reducing the impact of non-orthogonality. This is particularly relevant to real world models, where the alphabet size is usually much larger than the model dimension.

**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 superposed) 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 through the analysis of the feasible regime, we find mechanistic investigations at or close to the "failing" regime a relevant and challenging research direction.

*Generalizing to arbitrary functions.* It is unclear how to solve the histogram task for arbitrary objects, i.e. $T = \infty$, as done in-context learning in (Reddy, 2024). Preliminary explorations indicate that none of the models considered here learn to reach perfect accuracy, and that at least two layers are required. 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. (2023). Generalization on the unseen, logic reasoning and degree curriculum. In *ICML*.

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

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. (2020). Language models are few-shot learners. *CoRR*, abs/2005.14165.

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

Chollet, F., Tong, K., Reade, W., and Elliott, J. (2020). Abstraction and reasoning challenge.

Chughtai, B., Cooney, A., and Nanda, N. (2024). Summing up the facts: Additive mechanisms behind factual recall in LLMs.

Cui, H., Behrens, F., Krzakala, F., and Zdeborová, L. (2024). A phase transition between positional and semantic learning in a solvable model of dot-product attention.

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. (2023). Neural networks and the chomsky hierarchy. In *11th International Conference on Learning Representations*.

Donoho, D. L. and Elad, M. (2003). Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ¡sup¿1¡/sup¿ minimization. *Proceedings of the National Academy of Sciences*, 100(5):2197–2202.

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. (2023). Faith and fate: Limits of transformers on compositionality. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in Neural Information Processing Systems*, volume 36, pages 70293–70332. Curran Associates, Inc.

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. (2022). Toy models of superposition.

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. (2021). A mathematical framework for transformer circuits. *Transformer Circuits Thread*. https://transformer-circuits.pub/2021/framework/index.html.

Engels, J., Liao, I., Michaud, E. J., Gurnee, W., and Tegmark, M. (2024). Not all language model features are linear.

Fickus, M. and Mixon, D. G. (2016). Tables of the existence of equiangular tight frames.

Geva, M., Schuster, R., Berant, J., and Levy, O. (2021). Transformer feed-forward layers are key-value memories. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t., editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

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

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

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

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

Jiang, Q., Li, S., Bai, H., de Lamare, R. C., and He, X. (2017). Gradient-based algorithm for designing sensing matrix considering real mutual coherence for compressed sensing systems. *IET Signal Processing*, 11(4):356–363.

Jyothi, R. and Babu, P. (2022). Telet: A monotonic algorithm to design large dimensional equiangular tight frames for applications in compressed sensing. *Signal Processing*, 195:108503.

Kazemnejad, A., Padhi, I., Ramamurthy, K. N., Das, P., and Reddy, S. (2023). The impact of positional encoding on length generalization in transformers.

Liu, B., Ash, J. T., Goel, S., Krishnamurthy, A., and Zhang, C. (2023). Transformers learn shortcuts to automata. In *The Eleventh International Conference on Learning Representations*.

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

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

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

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

nostalgebraist (2020). interpreting GPT: the logit lens — LessWrong.

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. (2021). Show your work: Scratchpads for intermediate computation with language models. *CoRR*, abs/2112.00114.

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., and Carter, S. (2020). Zoom in: An introduction to circuits. *Distill*. 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. (2022). In-context learning and induction heads.

Ouellette, S., Pfister, R., and Jud, H. (2023). Counting and algorithmic generalization with transformers. *arXiv preprint arXiv:2310.08661*.

Petzka, H., Trimmel, M., and Sminchisescu, C. (2020). Notes on the symmetries of 2-layer relu-networks. In *Proceedings of the northern lights deep learning workshop*, volume 1, pages 6–6.

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

Quirke, P. and Barez, F. (2024). Understanding addition in transformers. In *The Twelfth International Conference on Learning Representations*.

Raventos, A., Paul, M., Chen, F., and Ganguli, S. (2023). Pretraining task diversity and the emergence of non-bayesian in-context learning for regression. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Reddy, G. (2024). The mechanistic basis of data dependence and abrupt learning in an in-context classification task. In *The Twelfth International Conference on Learning Representations*.

Strohmer, T. and Heath, R. W. (2003). Grassmannian frames with applications to coding and communication. *Applied and Computational Harmonic Analysis*, 14(3):257–275.

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. (2021). Mlp-mixer: An all-mlp architecture for vision.

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

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

Weiss, G., Goldberg, Y., and Yahav, E. (2021). Thinking like transformers. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11080–11090. PMLR.

Welch, L. R. (1974). Lower bounds on the maximum cross correlation of signals (corresp.). *IEEE Trans. Inf. Theory*, 20:397–399.

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

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

# Appendices

## A. Explicit Constructions for Orthogonal Embeddings $d = T$

### A.1. Overview

In the supplementary code[2], we provide the *explicit construction* for 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 explicit constructions give the correct prediction for all input token sequences. For all the models we describe below, we define the sum of the hidden layer neurons as:

$$y_\ell = \sum_{i=1}^{p} \mathrm{ReLU}(z_{\ell,i}) = \sum_{i=1}^{p} \mathrm{ReLU}(W_1 \bar{x}'_\ell + b_1)_i \tag{12}$$

In many cases, a simple linear regression can map the scalar $y_\ell$ to the correct count of tokens $x_\ell$, and we describe how to achieve this mapping to the classification problem in Section A.4.

In the following, we characterize which parameters $W_1, b_1$ in equation 12 allow for a correct mapping in each mechanism. Importantly, the architecture exhibits numerous symmetries due to the feed-forward ReLU network (Petzka et al., 2020). To demonstrate feasibility, we select one specific implementation. In the main text we observe that there is no one-to-one correspondence between our explicit constructions and the learned weights, even though both functions achieve the same perfect accuracy. Throughout, unless otherwise specified, we assume that $E \in \mathbb{R}^{d\times T}$ is an orthonormal basis of $\mathbb{R}^d$, which we will use to create different forms of token embeddings.

### A.2. Relation-based counting

A.2.1. (DOT; $p = 1$)

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. We choose the embeddings of the tokens as

$$e_t = \tilde{e}_t + \tilde{e}_{\text{cnt}} \quad \forall t = 1, \dots T \tag{13}$$

where the set $E = \{\tilde{e}_t\}_{t=1}^T$ 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 key and query matrix to the scaled identity $W_K = W_Q = d^{1/4} I_d$. The first layer weights $W_1, b_1 \in \mathbb{R}^d$ can be fixed as

$$W_1 = \tilde{e}_{\text{cnt}}/(T+1)\,; \;\; b_1 = -(1 + L(T+2))\,, \tag{14}$$

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

$$(W_2)_1 = -1 + \frac{1}{L+1}\,, \qquad (b_2)_1 = 0\,; \tag{15}$$

$$(W_2)_\ell = -1 + \frac{\ell}{L+1}\,, \qquad (b_2)_\ell = ((W_2)_{\ell-1} - (W_2)_\ell)\,(\ell - 0.5) + b_{\ell-1}\,, \qquad \forall \ell = 2, \dots, L. \tag{16}$$

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

$$\langle e_t, e_s \rangle = \begin{cases} 2 + T & \text{if } t \neq s\,, \\ 3 + T & \text{if } t = s\,. \end{cases} \tag{17}$$

Because of our choice of the query and key matrices, it directly follows that for tokens $x_\ell, x_m$ from a given sequence $\mathbf{x}$, their attention score is

$$a_{\ell m} = \begin{cases} 2 + T & \text{if } x_\ell \neq x_m\,, \\ 3 + T & \text{if } x_\ell = x_m\,. \end{cases} \tag{18}$$

[2] https://github.com/SPOC-group/counting-attention

Hence, the mixed token after applying the residual connection is

$$\bar{x}'_\ell = \bar{x}_\ell + \sum_{m:x_\ell=x_m} (T+3)\bar{x}_m + \sum_{m:x_\ell\neq x_m} (T+2)\bar{x}_m \tag{19}$$

so that computing

$$\begin{aligned}
\bar{x}'_\ell W_1 &= \left\langle \bar{x}'_\ell, \frac{\tilde{e}_{\text{cnt}}}{1+T} \right\rangle && (20)\\
&= \left\langle \bar{x}_\ell, \frac{\tilde{e}_{\text{cnt}}}{1+T} \right\rangle + \sum_{m:x_\ell=x_m} (T+3) \left\langle \bar{x}_m, \frac{\tilde{e}_{\text{cnt}}}{1+T} \right\rangle + \sum_{m:x_\ell\neq x_m} (T+2) \left\langle \bar{x}_m, \frac{\tilde{e}_{\text{cnt}}}{1+T} \right\rangle && (21)\\
&= 1 + hist_{\mathbf{x}}(\ell)(T+3) + (L - hist_{\mathbf{x}}(\ell))(T+2) && (22)\\
&= hist_{\mathbf{x}}(\ell) + 1 + L(T+2). && (23)
\end{aligned}$$

Then the single hidden unit has the value $y_\ell = \text{ReLU}(\bar{x}'_\ell W_1 + b_1) = hist_{\mathbf{x}}(\ell)$. It is easy to show (analogous to Fig. 7) 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

$$\underset{i=1\ldots L}{\arg\max}\, c_i(y) = \begin{cases} 1 & y = 1 \\ 2 & y = 2 \\ \ldots & \\ L & y = L\,, \end{cases} \tag{24}$$

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.

#### A.2.2. (BOS+SFTM; $p = 1$)

For the model with the beginning-of-sequence token, every sequence $\mathbf{x}$ if prefixed with $t_{\text{BOS}}$ 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 $E$. Analogous to the background token, $p = 1$ is enough and we set

$$W_1 = e_{\text{BOS}};\;\; b_1 = -1. \tag{25}$$

For a given token $x_\ell$ 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 $a = e/((k_{x_\ell} + 1)e + (L - k_{x_\ell}))$ (where $e$ is Euler's number) for comparing $x_\ell$ to $t_{\text{BOS}}$ and to all the tokens where $x_\ell = x_m$, and $b = 1/((k_{x_\ell} + 1)e + (L - k_{x_\ell}))$ otherwise, where, $k_{x_\ell} = hist_{\mathbf{x}}(\ell)$. Hence, the mixed token is:

$$\bar{x}'_\ell = a e_{\text{BOS}} + a k_{x_\ell} \bar{x}_\ell + \sum_{x_m \neq x_\ell} b \bar{x}_m + \bar{x}_\ell. \tag{26}$$

Applying $W_1$ and $b_1$, we obtain:

$$\begin{aligned}
y_\ell &= aT + a k_{x_\ell} + b(L - k_{x_\ell}) \\
&= aT + a k_{x_\ell} + 1 - a(k_{x_\ell} + 1) \\
&= a(T-1) + 1
\end{aligned} \tag{27}$$

since $(k_{x_\ell} + 1)a + (L - k_{x_\ell})b = 1$ by normalization via the softmax function. The value of $y_\ell$ can be readout into the correct classification as shown Fig. 7.

#### A.2.3. (BOS; $p = 1$)

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

### A.3. Inventory-based counting

A.3.1. (LIN; $p = T$).

As embeddings we directly use $E$, s.t. $e_t = E_t$ where the latter is a column of $E$. We set

$$\mathbf{A}_{\text{lin}} = \begin{bmatrix} a & a & \cdots & a \\ a & a & & \\ \vdots & & \ddots & \\ a & & & a \end{bmatrix}; \quad W_1 = E; \quad b_1 = -1, \tag{28}$$

where $a = 1/L$. We start by writing $z_{\ell,t}$ for $t \in \{1, ..., p = T\}$, the t-th activation of the first hidden layer of the feed-forward module

$$z_{\ell,t} = \sum_{m=1}^{L} a_{\ell m} \langle e_{x_m}, e_t \rangle + \langle e_{x_\ell}, e_t \rangle - 1 \,. \tag{29}$$

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

$$z_{\ell,t} = k_{x_\ell} a + 1 - 1 = a k_{x_\ell} \,, \tag{30}$$

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

$$z_{\ell,t} = a k_{e_t} + 0 - 1 = a k_{e_t} - 1 \leq 0 \,. \tag{31}$$

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. Hence the count can be read off from $y_\ell$. Since only one neuron is activated at a time, the readout from the same procedure as in `bos+sftm` can be applied to all hidden neurons $z_{\ell,t}$ simultaneously, instead of only one.

A.3.2. (LIN+SFTM: $p = T$)

Same as `lin`, except

$$\mathbf{A}_{\text{lin+sftm}} = \begin{bmatrix} a & a & \cdots & a \\ a & a & & \\ \vdots & & \ddots & \\ a & & & a \end{bmatrix} = \text{softmax}\left( \begin{bmatrix} \alpha & \alpha & \cdots & \alpha \\ \alpha & \alpha & & \\ \vdots & & \ddots & \\ \alpha & & & \alpha \end{bmatrix} \right), \tag{32}$$

where $a = 1/L$ which implicitly defines a choice of $\alpha$.

A.3.3. (DOT+SFTM: $p = d = T$)

As before, we set the key and query matrix to the scaled identity $W_K = W_Q = d^{1/4} I_d$. 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. For $k_{x_\ell} = hist_{\mathbf{x}}(\ell)$, then, after applying the softmax, we have $a_{lm} = \frac{e}{(L - k_{x_\ell}) + e k_{x_\ell}}$ if $x_m = x_\ell$ and $a_{lm} = \frac{1}{(L - k_{x_\ell}) + e k_{x_\ell}}$ otherwise. Hence, for $e_t \neq e_{x_\ell}$

$$\langle \bar{x}'_\ell, e_t \rangle = \frac{k_{e_t}}{(L - k_{x_\ell}) + e k_{x_\ell}} < 1, \tag{33}$$

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

$$\langle \bar{x}'_\ell, e_{x_\ell} \rangle = \frac{k_{x_\ell} e}{(k_{x_\ell} e + (L - k_{x_\ell}))} + 1, \tag{34}$$

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

$$W_1 = E; \quad b_1 = -1, \tag{35}$$

and applying the ReLU activation, equation 33 will be 0, while equation 34 will implicitly give us the counts as:

$$k_{x_\ell} = (L y_\ell)/(-e y_\ell + y_\ell + e) \tag{36}$$

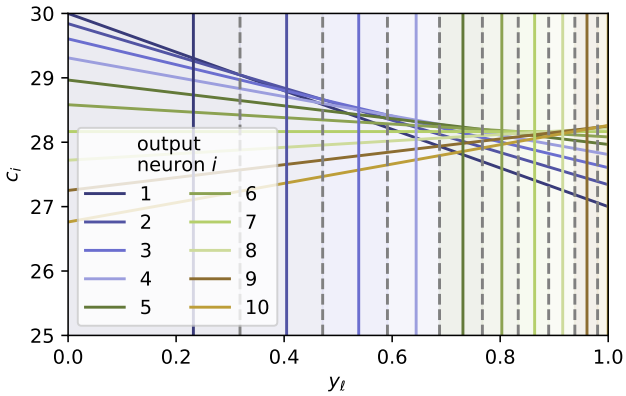


*Figure 7.* Demonstration of a hidden neuron output $y_\ell$ which is mapped to $L = 10$ different neurons $c_i$ using a single output layer, according to the decision boundaries shown by the dotted lines. After applying the argmax function to the 10 output neurons, the highest value gives the discrete output. The solid lines mark the values a single hidden neuron would achieve for different counts in the explicit construction of the `dot+sftm` model.

While the final layer cannot immediately implement non-linear functions in $y_\ell$, it can take advantage of the fact that $y_\ell$ can take only $L$ different values. Since eventually we need to map the $L$ values of $y_\ell$ to the counts $[1, \cdots, L]$ the linear output layer is sufficient to implement this non-linear discrete map (see Fig. 7 which shows exactly this example where solid lines are $y_\ell$ lying in intervals bounded by dashed lines).

### A.4. Mapping a scalar to a categorical one-hot encoding

It is straightforward to map a single scalar $y_\ell$ to a series of neurons which activate one after another. This is needed as the second part of the feed-forward parameters to transform the count measured by the sum of the hidden neurons $y_\ell$ 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 uninterrupted 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. 7 for `dot+sftm`. In Fig. 8 we show the outputs for the `lin+sftm` model with the best accuracy for $T = 32$ for every $p, d$ ran in Fig. 1. While it is possible to learn the count from one hidden neuron only using inventory-based counting for each neuron, for some examples the count information seems to be spread out over several hidden neurons: The output logits are non-linear in the count and can hence not rely on a single hidden neuron only.

## B. Explicit Constructions for Linearly Dependent Embeddings $d < T$

### B.1. Overview

In this section, we discuss the scenario when $d < T$, i.e. when the embeddings are necessarily linearly dependent. In that case, we can no longer assume that there exist embeddings with $\langle e_t, e_s \rangle = 0$ for all $t \neq s$. Nonetheless, also in this regime for some models it is possible to provide explicit constructions of the weights that have 100% accuracy. This relies on the fact that the prediction problem is inherently discrete, i.e. it chooses exactly one among $L$ classes. When we examine $y_\ell \in \mathbb{R}$ from equation 12 which is mapped to the discrete class through the readout layer (see for example Fig. 7), we notice that the class boundary (the gray dashed class borders) can be placed variably in the margin between the values that $y_\ell$ assumes for different counts $k_{x_\ell}$ (solid lines). In the following explicit constructions, our goal is to design embeddings with $d < T$ in such a way that we maximize the aforementioned margin: there will be pairs of token embeddings in the alphabet that have non-zero similarity $\langle e_t, e_s \rangle$, and in equation 12 this will create non-zero terms that will alter the value of $y_\ell$. This means that for every possible sequence with $k$ occurrences of token $x_\ell$, the hidden activation $y_\ell$ will assume values in a certain range $[y_\ell^{\text{lower}}(k), y_\ell^{\text{upper}}(k)]$. If these ranges overlap for different $k$, the count cannot be identified. However, we construct embeddings such that every for every $k = 1, \ldots, L-1$ it holds that

$$y_\ell^{\text{upper}}(k) < y_\ell^{\text{lower}}(k+1)\,, \tag{37}$$

so we can still use a construction as in Fig. 7 to correctly compute the final count. In the remainder of this section, we introduce explicit constructions with $d < T$ for a given $L$, both for the cases where we have relation-based counting and inventory-based counting (the same argument as above transfers to $z_{\ell,t}$ from equation 29). Notably, for the explicit

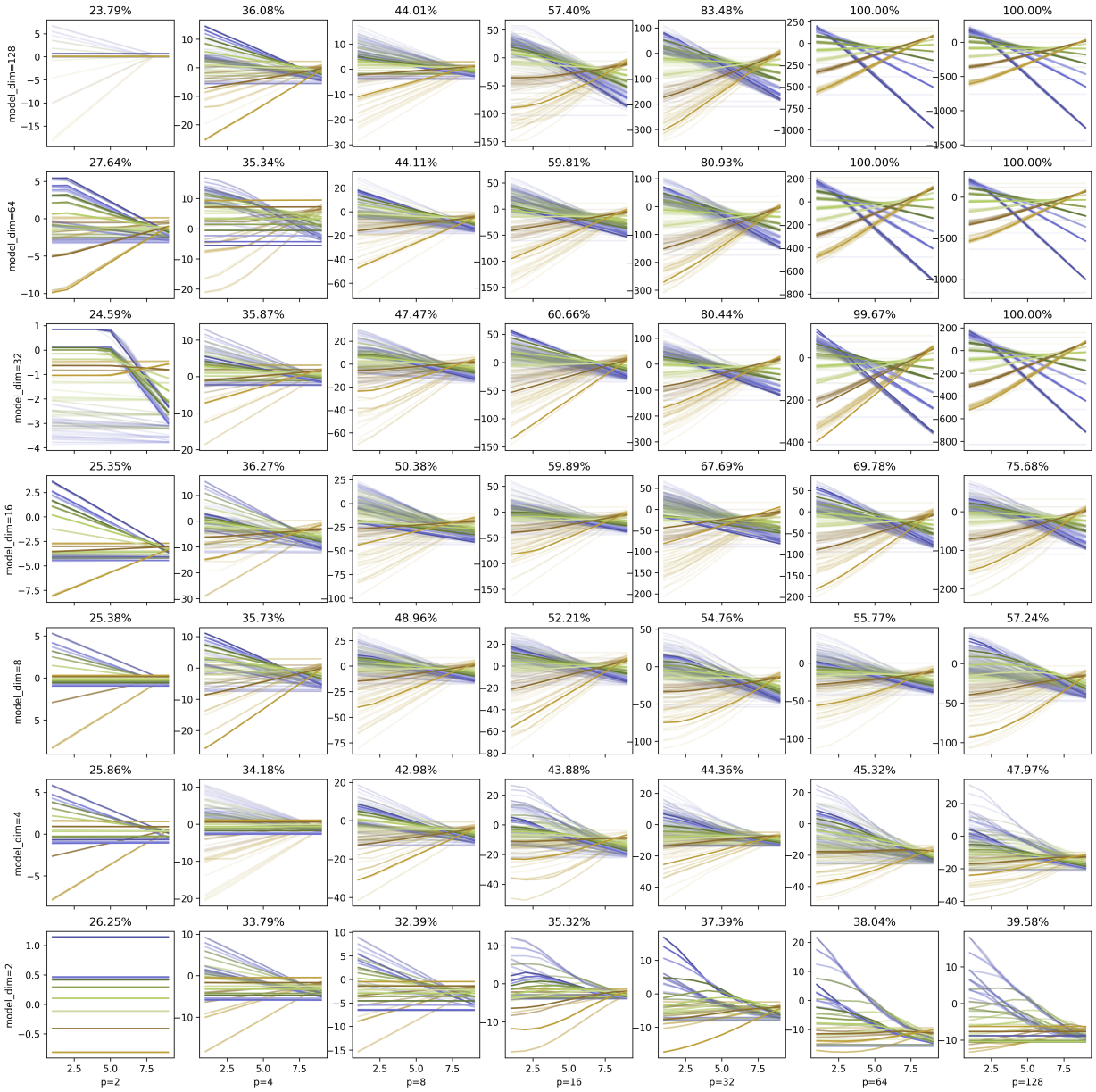


*Figure 8.* The output neurons $c_i(\mathbf{x}_\ell)$ visualized for examples of a learned version of `lin+sftm` for several model dimensions $d$ and hidden layer sizes $p$. Note that differently from Fig. 7, in this case the x-axis shows the number of occurrences $hist_{\mathbf{x}}(\ell = 0) = 1, \ldots, L-1$ of the token $t$ in an input sequence $\mathbf{x} = [t, \cdots, t, v, \cdots, v]$ that contains otherwise only a token $v \neq t$ (and *not* the activation of a hidden neuron). We show the activations $c_i$ of the final layer output neurons activations (logits) in terms of the number of occurrences of a given token in the input. The colors represent the different output predictions and are as in the explicit construction from Fig. 7. We show several activations for different tokens $t \in \mathcal{T}$, where $T = 32$, and we highlight one of the example tokens $t$ with a wider line. While similar to the explicit construction from Fig. 7, the models with 100% accuracy are not necessarily linear in the count.

constructions we propose, the function of the lowest achievable $d(p, T, L)$ differs across different mixing types. To summarize:

- For models with $\mathbf{A}$ constant in the inputs or models *without* softmax activation, our explicit construction relies on an embedding matrix with a small mutual coherence. The mutual coherence is a concept from compressed sensing and coding theory that ensures that the maximal similarity between pairs of vectors is small (Donoho and Elad, 2003). We can upper bound the mutual coherence that the margins of the construction can tolerate to still achieve perfect accuracy

in terms of a given $L$. At the same time, the mutual coherence of a set of vectors is naturally lower bounded in terms of the number of vectors $T$ and their respective dimension $d$, known as the Welch bound (Welch, 1974). When this bound can be attained and $T, L$ are given, this leads to the following bounds on $d$ for the different models

(lin, lin+sftm; $p = T$): $\left\lceil \frac{T(2L-3)^2}{T-1+(2L-3)^2} \right\rceil \leq d$,

(dot, bos; $p = 1$): $\left\lceil \frac{T(2L-3)^2}{T-1+(2L-3)^2} \right\rceil + 1 \leq d$,

(dot, bos; $p = T$): $\left\lceil \frac{T(2L-3)}{T-1+(2L-3)} \right\rceil \leq d$.

- For bos+sftm we rely on the fact that the softmax function accentuates the largest value and thereby can drive attention scores for equal tokens $a_{ii}$ higher relative to attention scores of non-equal tokens $a_{ij}$. This distinguishes it from the previous case, and allows us to describe an explicit construction with perfect accuracy for

  (bos+sftm; $p = 1$): $d \geq \lceil \log_2(T+1) \rceil + 2$.
  (dot+sftm; $p = T$): $d \geq \lceil \log_2(T+1) \rceil + 2$.

  Notably there is no explicit dependence on $L$. However, the smaller the dimension $d$ the more accurate computations and softmax numerical stability are required. With infinitely precise computations we show it is even possible to achieve perfect accuracy with $d = 2$.

### B.2. Explicit construction for bounded mutual coherence

We define the *mutual coherence* $\mathcal{M}$ of a set of $T$ unit norm vectors $\{v_1, \ldots, v_T\} \subset \mathbb{R}^d$ as

$$\mathcal{M} = \max_{i \neq j} |\langle v_i, v_j \rangle| \,. \tag{38}$$

This value is lower bounded for a given matrix by the Welch bound (Welch, 1974)

$$\mathcal{M} \geq \sqrt{\frac{T-d}{d(T-1)}} = \mathcal{W}(T, d) \,, \tag{39}$$

and equality can only be attained if $T < d^2$ (Strohmer and Heath, 2003). There is a large body of work in coding theory and compressed sensing concerning the existence and construction of a set of vectors that attains $\mathcal{M}$ at or close to $\mathcal{W}(T, d)$. Explicit constructions exist but are not known for every combination of $T$ and $d$. A list with existing constructions for the real space for small $T, d$ can be found in (Fickus and Mixon, 2016), but otherwise gradient-based optimization has been used to find good candidate matrices (Jiang et al., 2017; Jyothi and Babu, 2022).
For a given $T$, $L$ and $p$, we can derive an upper bound on the mutual information of the embeddings in terms of $L$, which is required to obtain perfect accuracy. The form of this upper bound depends on the precise mixing strategy and the choice of $p$. Through the Welch lower bound on $\mathcal{M}$ we can in turn obtain a lower bound on $d$ in terms of $L$ and $T$. Note that the Welch bound cannot be attained for $T < d^2$ and in this case the bound on $d$ is strict.

#### B.2.1. (LIN, LIN+SFTM; $p = T$)

We analyze the inventory-based construction for lin in equation 28. Given that $p = T$, and $L$ is given, let us assume that there exists set of $T$ unit norm vectors $\{e_1, \ldots, e_T\} \subset \mathbb{R}^d$ with mutual coherence $\mathcal{M}$. We use these vectors as our embeddings. With the same parameters as in equation 28, this model still attains perfect accuracy. The value $z_{\ell,t}$ for $t = x_\ell$, with $W_1 = [e_1, \ldots, e_T]$ and $b_1 = -1$ is

$$z_{\ell,t} = a k_{x_\ell} + a \sum_{m: x_m \neq t} \langle e_{x_m}, e_t \rangle \,, \tag{40}$$

and using that fact that the mutual coherence bounds the absolute value of the inner product

$$a k_{x_\ell} - a\mathcal{M}(L - k_{x_\ell}) \leq z_{\ell,t} \leq a k_{x_\ell} + a\mathcal{M}(L - k_{x_\ell}) \,. \tag{41}$$

Similarly, for $t \neq x_\ell$ and $a = 1/L$ it still holds that

$$z_{\ell,t} = a k_t + a(L - k_t)\mathcal{M} - 1 \leq aL - 1 \leq 0 \,, \tag{42}$$

so the ReLU sets all hidden neurons $z_{\ell,t}$ to zero when $t \neq x_\ell$. Then, defining

$$y_\ell^{\text{lower}}(k) = ak - a\mathcal{M}(L-k)\,, \tag{43}$$
$$y_\ell^{\text{upper}}(k) = ak + a\mathcal{M}(L-k)\,, \tag{44}$$

we have that indeed for a sequence where $x_\ell$ occurs $k = 1, \ldots, L$ times it holds that

$$y_\ell^{\text{lower}}(k) \leq y_\ell(k) \leq y_\ell^{\text{upper}}(k). \tag{45}$$

From equation 37 we have the condition that for all $k = 1, \ldots, L$ it holds that

$$y_\ell^{\text{upper}}(k) < y_\ell^{\text{lower}}(k+1)\,, \tag{46}$$
$$k + (L-k)\mathcal{M} < (k+1) + (L-k-1)(-\mathcal{M})\,, \tag{47}$$
$$\mathcal{M} < \frac{1}{2(L-k)-1}\,, \tag{48}$$

and since we assume that there exist at least two different tokens in the sequence, minimizing the bound over $k$ leaves for $k = 1$

$$\mathcal{M} < \frac{1}{2L-3}\,. \tag{49}$$

which is valid provided that $L \geq 2$. Combining this with the Welch bound equation 39 yields the condition

$$d \geq \left\lceil \frac{T(2L-3)^2}{T-1+(2L-3)^2} \right\rceil \tag{50}$$

for the given explicit construction to obtain a perfect accuracy. For `lin` and `lin+sftm` the construction and condition is the same.

B.2.2. (DOT, BOS; $p = 1$)

For the relation-based explicit construction for `dot` (which holds equivalently for `bos` when setting the BOS token to zero) with $p = 1$, given $T$ and $L$ we assume that we have a set of $T$ unit norm vectors with $\{v_1, \ldots, v_T\} \subset \mathbb{R}^{d-1}$ with mutual coherence $\mathcal{M}$. We set the entries of the $T$ embedding vectors $e_t \in \mathbb{R}^d$ to be

$$e_t = \begin{bmatrix} v_t \\ \alpha \end{bmatrix}. \tag{51}$$

The shared counting subspace is defined on the last coordinate of the vectors via $e_{cnt} = [0, 0, \ldots, 1/\alpha]$. Then

$$\langle e_t, e_t \rangle = 1 + \alpha^2 \tag{52}$$
$$|\langle e_t, e_s \rangle| \leq \mathcal{M} + \alpha^2 \tag{53}$$

The mixed token with the residual connection at position $\ell$ for a given input sequence is

$$\bar{x}'_\ell = \sum_{m=1}^{L} \langle e_{x_m}, e_{x_\ell}, \rangle e_{x_m} + e_{x_\ell} \tag{54}$$

and the single hidden neuron $y_\ell$ for a bias term $b_1 = 0$ and $W_1 = e_{cnt}$

$$y_\ell = \langle e_{cnt}, \bar{x}'_\ell \rangle = \sum_{m=1}^{L} \langle e_{x_m}, e_{x_\ell}, \rangle \langle e_{x_m}, e_{cnt} \rangle + \langle e_{x_\ell}, e_{cnt} \rangle \tag{55}$$
$$= k_{x_\ell}(1+\alpha^2) + \sum_{m: x_m \neq x_\ell} \langle e_{x_m}, e_{x_\ell}, \rangle + 1 \tag{56}$$

So that we can achieve for a given count $k$ the $y_\ell^{\text{lower}}(k) \leq y_\ell(k) \leq y_\ell^{\text{upper}}(k)$ with

$$y_\ell^{\text{lower}}(k) = k(1+\alpha^2) - (L-k)(\mathcal{M}+\alpha^2) + 1\,, \tag{57}$$

$$y_\ell^{\text{upper}}(k) = k(1+\alpha^2) + (L-k)(\mathcal{M}+\alpha^2) + 1\,. \tag{58}$$

Finally, the condition from equation 37 yields

$$\mathcal{M} < \frac{1}{2(L-k)-1} - \frac{2(L-k)-2}{2(L-k)-1}\alpha^2 \tag{59}$$

under the condition that $0 < \alpha < \sqrt{\frac{1}{2(L-k)-2}}$. Again, assuming there exist at least two different tokens in the sequence, the r.h.s. of the above expression is minimized for $k = 1$ as

$$\mathcal{M} < \frac{1}{2L-3} - \frac{2L-4}{2L-3}\alpha^2 \tag{60}$$

which is always positive assuming $L \geq 2$. Again, combining this with the Welch bound equation 39 lead to

$$d - 1 \geq \left\lceil \frac{T(\frac{2L-3}{1-(2L-2)\alpha^2})^2}{T-1+(\frac{2L-3}{1-(2L-2)\alpha^2})^2} \right\rceil\,, \tag{61}$$

and when we choose $\alpha > 0$ close to zero, as for `lin` before

$$d \geq \left\lceil \frac{T(2L-3)^2}{T-1+(2L-3)^2} \right\rceil + 1. \tag{62}$$

B.2.3. (DOT, BOS; $p = T$)

We can decrease the required dimension $d < T$ even further than previously, when we have $p = T$ and implement inventory-based counting in the `dot` model (and equivalently in the `bos` model). In that case, the lower bound on $d$ becomes more loose, because we combine the ideas we saw in `lin` and $p = T$ for inventory-based counting and the effects on the margin in `dot` and $p = 1$.

In our construction, for a given $T$ and $L$, we assume that there is a set of $T$ unit norm vectors $\{e_1, \ldots, e_T\} \subset \mathbb{R}^d$ with mutual coherence $\mathcal{M}$ upon which we build our embeddings. Note that the only difference to the previous relation-based case $p = 1$ is that this time there is no extra counting direction. Importantly, we set $K = d^{1/4} I_d$ as before, but $Q = \frac{1}{L} d^{1/4} I_d$. This gives an extra factor in the attention scores. Further, we set $b_1 = -1$ and the columns of $W_1 \in \mathbb{R}^{d\times T}$ to the embeddings $e_t$, as we did for `lin`. This results in a mixed token $\bar{x}'_\ell$ according to equation 54. The hidden neuron is

$$z_{\ell,t} = \frac{1}{L}\sum_{m=1}^{L} \langle e_{x_m}, e_{x_\ell}\rangle\langle e_t, e_{x_m}\rangle + \langle e_t, e_{x_\ell}\rangle - 1 \tag{63}$$

then with $t = x_\ell$ we have

$$z_{\ell,t} = \frac{1}{L}\left(k_{x_\ell} + \sum_{m:x_m\neq t} \langle e_{x_m}, e_t\rangle^2\right)\,. \tag{64}$$

Note that the square in equation 63 is what differs from the $z_{\ell,t}$ in equation 40. This is because the term $\langle e_{x_m}, e_t\rangle$ is once introduced through the dot-product attention and once through the dot-product via $W_1$. Conversely, with $t \neq x_\ell$ it becomes

$$z_{\ell,t} = \frac{1}{L}\left(k_t\langle e_t, e_{x_\ell}\rangle + \sum_{m:x_m\neq t} \langle e_{x_m}, e_{x_\ell}\rangle\langle e_{x_m}, e_t\rangle\right) + \langle e_t, e_{x_\ell}\rangle - 1 \tag{65}$$

$$\leq \frac{1}{L}\left(k_t\mathcal{M} + (L-k_t)\mathcal{M}\right) + \langle e_t, e_{x_\ell}\rangle - 1 \tag{66}$$

$$\leq 2\mathcal{M} - 1 \tag{67}$$

if we set $\mathcal{M} < 0.5$, which we need anyways for $L \geq 2$ by the stronger upper bound on $\mathcal{M}$ that we develop in the following we finally have for $t \neq x_\ell$

$$z_{\ell,t} < 0\,. \tag{68}$$

Again, negative $z_{\ell,t}$ are set to zero via the ReLU, and the final outcome $y_\ell = z_{\ell,t=x_\ell}$ depends only on a single hidden neuron equation 63. This eventually leads to

$$y_\ell^{\text{lower}}(k) = \frac{1}{L}(k - \mathcal{M}^2(L-k))\,, \tag{69}$$

$$y_\ell^{\text{upper}}(k) = \frac{1}{L}(k + \mathcal{M}^2(L-k))\,, \tag{70}$$

and using the same concept as before, while minimizing over $k$ and applying the Welch bound, to the upper bound

$$\mathcal{M} < \sqrt{\frac{1}{2L-3}}\,. \tag{71}$$

The final bound is more loose as we only require

$$d \geq \left\lceil \frac{T(2L-3)}{T-1+(2L-3)} \right\rceil\,. \tag{72}$$

### B.3. Explicit Construction with binary representations and softmax

In our final construction we examine the key difference between the models `bos+sftm` and `bos` – the softmax activation. Before, we needed to construct embeddings with a low mutual coherence, because the term $\langle e_t, e_s\rangle$ introduced an error on the mixed token, when $t$ and $s$ were not equal. Now, with the softmax activation applied to the mixing coefficients, the model can use the non-linearity of this transform to its advantage. Recall the softmax function is

$$\text{sftm}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^n e^{z_j}} \quad \text{for} \quad i = 1, 2, \ldots, n\,, \tag{73}$$

and when we compute $\text{sftm}(\kappa\mathbf{z})_i$ we say it is a softmax with a inverse temperature $\kappa > 0$. When $\mathbf{z}$ of length $L$ contains only two different values, one with $k$ and the other with $L-k$ occurrences, then as $\kappa \to \infty$ the mass concentrates only on the larger value of the two, and sets the other to zero. We use this intuition to create token embeddings that fulfill for all $t, s = 1, \ldots, T$ and $s \neq t$

$$\langle e_t, e_t\rangle = 1\,, \tag{74}$$

$$\langle e_t, e_s\rangle < 1 + \epsilon\,, \tag{75}$$

where $\epsilon > 0$. The idea is that the softmax with a high enough inverse temperature sets the term for different tokens, $\langle e_t, e_s\rangle$, *close enough* to zero, essentially eliminating the noise. Note that equation 74 is a weaker condition on the set of token embeddings than for example the bound of the mutual coherence in terms of the sequence length $L$ `bos` with $p = 1$ in Section B.2.2. It allows us to obtain perfect accuracy with smaller $d$. In the following we describe the construction of the matrix explicitly.

#### B.3.1. (BOS+SFTM, $p = 1$)

In our explicit construction we introduce the inverse temperature by scaling the query matrix. We set $K = d^{1/4} I_d$ as before, but $Q = \kappa d^{1/4} I_d$. The embeddings vectors correspond to the binary representation of the token index $t = 1, \ldots, T$ in $d' = \lceil \log_2(T+1) \rceil$ dimensions

$$e_t = \begin{bmatrix} \text{bin}(t)\langle \text{bin}(t), \text{bin}(t)\rangle^{-1} \\ \\ \alpha \\ 0 \end{bmatrix}\;;\quad e_{BOS} = \begin{bmatrix} \text{bin}(0)\langle \text{bin}(0), \text{bin}(0)\rangle^{-1} \\ \\ 1/\alpha \\ 1 \end{bmatrix}\,. \tag{76}$$

where $\mathrm{bin}(t) = [v_1, \ldots, v_{d'}] \in \{0,1\}^{d'}$ with $t = \sum_{i=1}^{d'} v_i 2^{i-1}$. We select $\alpha > 0$. Then we have that

$$\langle e_t, e_t \rangle = 1 + \alpha^2 \,, \tag{77}$$

$$\alpha^2 \leq \langle e_t, e_s \rangle \leq \sqrt{1 - \frac{1}{d'}} + \alpha^2 \leq 1 + \alpha^2 - \epsilon \,, \tag{78}$$

$$\langle e_t, e_{BOS} \rangle = 1 \,, \tag{79}$$

where $\sqrt{\frac{d'-1}{d'}} = \langle e_{2^{d'}-1}, e_{2^{d'}-2} \rangle$, which has the largest overlap among all possible non-equal pairs of tokens, and the lower bound comes from all coordinates being positive. Using a readout on the direction only present in the $e_{BOS}$ token, namely, $W_1 = [e_{cnt}] = [0, \ldots, 0, 1] \in \mathbb{R}^d$ and $b_1 = 0$, we construct

$$y_\ell = \langle e_{cnt}, \bar{x}'_\ell \rangle = \mathrm{sftm}(EE_\ell^T)_0 \langle e_{BOS}, e_{cnt} \rangle + \sum_{m=1}^{L} \mathrm{sftm}(EE_\ell^T)_{m+1} \langle e_{x_m}, e_{cnt} \rangle + \langle e_{x_\ell}, e_{cnt} \rangle \tag{80}$$

$$= \mathrm{sftm}(EE_\ell^T)_0 \tag{81}$$

$$= \mathrm{sftm}([\langle e_\ell, e_{BOS} \rangle, \langle e_\ell, e_1 \rangle, \ldots, \langle e_\ell, e_L \rangle])_0 \tag{82}$$

The goal of applying the softmax function is to diminish the contributions of error equation 78, while having the final dimension of the $e_{BOS}$ token be representative of the count of $x_\ell$. The maximum error is induced when the upper bound equation 78 is attained for all tokens in the sequence $\mathbf{x}$ that are not equal to $x_\ell$. The minimum error is obtained when these different tokens attain the lower bound. Without loss of generality on the ordering, this implies that for a given length $L$ and a softmax activation function with an inverse temperature $\kappa$, we have that

$$y_\ell^{\mathrm{lower}}(k) = \frac{e^{\kappa 1}}{e^{\kappa 1} + k e^{\kappa(1+\alpha^2)} + (L-k) e^{\kappa(1+\alpha^2-\epsilon)}} \,, \tag{83}$$

$$y_\ell^{\mathrm{upper}}(k) = \frac{e^{\kappa 1}}{e^{\kappa 1} + k e^{\kappa(1+\alpha^2)} + (L-k) e^{\kappa \alpha^2}} \,. \tag{84}$$

We explicitly need $\epsilon$ strictly greater than zero, since otherwise there is no information about the count in $y_\ell$ since it is independent of the count $k$. Notice, that this time it holds that $y_\ell$ that correspond to higher values correspond to smaller counts, since a larger count corresponds to a larger denominator, i.e. a smaller $y_\ell$. Due to this inverse relationship, for this model, we want that for all counts $k = 1, \ldots, L-1$ that it holds that

$$y_\ell^{\mathrm{upper}}(k+1) < y_\ell^{\mathrm{lower}}(k) \,. \tag{85}$$

This can be achieved by setting the inverse temperature $\kappa$ accordingly.

In the following we show that there exists a $\kappa$ which fulfills equation 85 for all $d' \geq 2$ and $L > 2$. Observe that $y_\ell^{\mathrm{upper}}(2) < y_\ell^{\mathrm{lower}}(1)$ implies the bounds for all other $k$. We define the distance or margin as

$$\mathrm{dist}(\kappa) = y_\ell^{\mathrm{lower}}(1) - y_\ell^{\mathrm{upper}}(2) \,. \tag{86}$$

Since at $\kappa = 0$ both $y_\ell^{\mathrm{upper}}(2) = y_\ell^{\mathrm{lower}}(1) = 1/(L+1)$, the distance is zero. However then it becomes impossible to distinguish $k = 1$ and $k = 2$, as they receive the same weight. We therefore need the additional condition that $y_\ell^{\mathrm{upper}}(2) \neq y_\ell^{\mathrm{lower}}(1)$. At $\kappa = 0$, we observe that this function has a negative derivative, as

$$\frac{\partial}{\partial \kappa} \mathrm{dist}(\kappa)|_{\kappa=0} = \mathrm{sftm}(\kappa z_{\mathrm{lower}})_0 \left( (z_{\mathrm{lower}})_0 - \sum_{i=0}^{L+1} (z_{\mathrm{lower}})_j \mathrm{sftm}(\kappa z_{\mathrm{lower}})_i \right) \tag{87}$$

$$- \mathrm{sftm}(\kappa z_{\mathrm{upper}})_0 \left( (z_{\mathrm{upper}})_0 - \sum_{i=0}^{L+1} (z_{\mathrm{upper}})_j \mathrm{sftm}(\kappa z_{\mathrm{upper}})_i \right) \tag{88}$$

$$= - \left( \frac{1}{L+1} \right)^2 \left( \left[ (1+\alpha^2) + (L-1)(1+\alpha^2-\epsilon) \right] - \left[ 2(1+\alpha^2) + (L-2)\alpha^2 \right] \right) \tag{89}$$

$$= - \left( \frac{1}{L+1} \right)^2 \left[ (L-2) - (L-1)\epsilon \right] \tag{90}$$

$$< 0 \tag{91}$$

where the last bound is met when $0 < \epsilon < 0.5$ which is fulfilled already for $d' = 2$ and when $L > 2$. As the distance function is continuous, there exists a $\kappa$ close to zero for which the $\mathrm{dist}(\kappa) < 0$. Simultaneously, as $\kappa \to \infty$, we have that due to the concentration of the softmax probabilities on the largest entry, which here is $1 + \alpha^2$, it holds that as $\kappa \to \infty$ we have $\mathrm{dist}(\kappa) \to 0$. At the same time, the function approaches infinity from the positive regime. For large enough $\kappa$ we have $y_\ell^{\mathrm{upper}}(2) < y_\ell^{\mathrm{upper}}(1)$.

When we select the smallest possible $\kappa > 0$, we avoid computing large exponentials. To find the non-trivial root of $\mathrm{dist}(\kappa)$ numerically, we consider a simplification of equation 86. We define $u = e^\kappa$. Then it holds that we can solve

$$\mathrm{dist}(\kappa) = 0 = (L-1)u^{(1-\epsilon)} - u - (L-2) \tag{92}$$

numerically for $\kappa > 0$. This shows that we can find an explicit construction with 100% accuracy with $p = 1$ and $d' > 2$ for the `bos+sftm` when we have

$$d = \lceil \log_2(T+1) \rceil + 2\,. \tag{93}$$

For example, for the case of $L = 10$ and $T = 32$ this allows for a dimension $d = 7$ with $\alpha = 0.01$ (and for $T = 31$ with the same settings $d = 6$ suffices).

**Remark ($d = 2$).** In principle, it is enough to have some $\epsilon > 0$ that ensures that overlaps between different token embeddings are strictly less than one. In principle, we can find an arbitrary number of tokens $T$ the satisfies this condition for just $d' = 2$. Take for example the following construction. For $t = 1, \ldots, T$ tokens with $T$ odd we can design the set of embeddings

$$v_t = \begin{bmatrix} \sqrt{\frac{t}{T}} \\ \sqrt{\frac{T-t}{T}} \end{bmatrix}\,. \tag{94}$$

Each $\langle e_t, e_t \rangle = 1$ and for $t \neq s$ the overlap $\langle e_t, e_s \rangle \leq \langle e_{(T+1)/2}, e_{(T-1)/2} \rangle = \sqrt{T^2-1}/T$.

This implies that $\epsilon \to 0$ as $T \to \infty$ at a rate $1/T$. Since smaller $\epsilon$ imply larger values of the temperature to solve equation 92, this might become problematic when this exceeds the accuracy of computations. Previously, for the binary representation construction from equation 76, we had that $\epsilon$ shrinks at a rate $\sim 1/\log_2(T)$. For the intermediate regime between $\log_T(T) + 1$ and $\log_2(T)$ dimensions, one can generalize this principle to arbitrary bases, e.g. $\log_3(T) > 2$, resulting in a smaller dimension but also less favorable (smaller) $\epsilon$ – this construction thus comes with a clear trade-off.

#### B.3.2. (DOT+SFTM; $p = T$)

For this model, the explicit construction is analogous to the previous one. Instead of using $p = 1$ we use $p = T$. The selection of the embeddings is analogous, but instead of a counting direction we read off all the weight directions. Not having a counting direction also saves the additional two dimensions over `bos+sftm` with $p = 1$. In the feedforward pass with $W_1$ the explicit construction considers again $z_{\ell,t}$ for every token $t \in \mathcal{T}$. The selection of the temperature is also analogous, with the exception that one has $L$ terms in the softmax instead of $L + 1$.

## C. 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 $\mathbf{x} = \mathtt{shuffle}(\mathbf{x})$.

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.

# D. Additional Experiments

## D.1. Best Accuracy

In Fig. 9, 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.

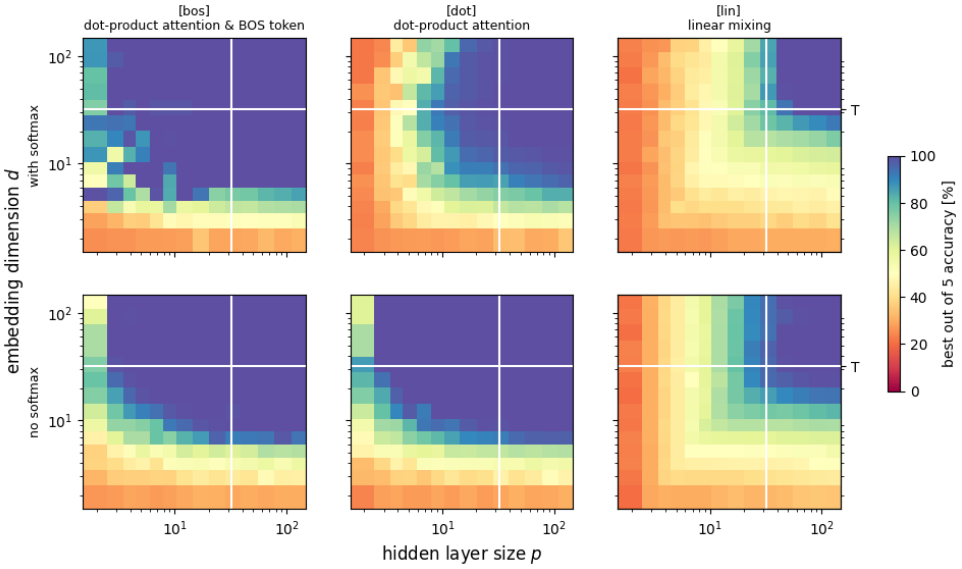


*Figure 9.* Experiments from Fig. 1 ($T = 32$), we show only the *best accuracy* during training reached from the 5 randomly initialized runs per model/hyperparameter configuration.

## D.2. Variability

In Fig. 10 we explore the influence of initialization on the performance via the variability of the final accuracy for several runs. 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.

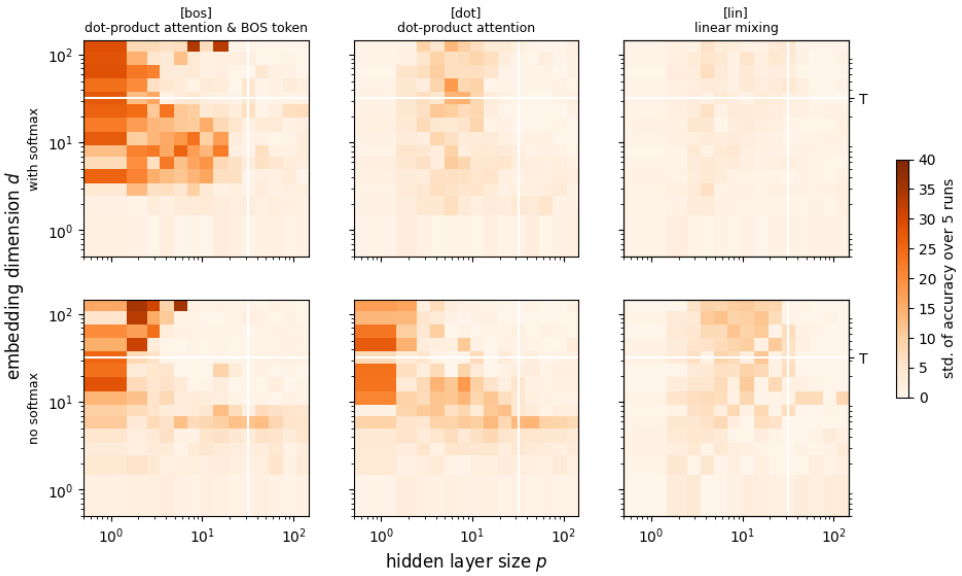


*Figure 10.* Experiments from Fig. 1 with $T = 32$, standard deviation of the accuracy reached after training from the 5 randomly initialized runs per model/hyperparameter configuration.

### D.3. Model with Random but Fixed Embeddings

In Fig. 11, 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 than the learned ones. Especially for `bos+sftm`, learning the embeddings increases the performance strongly in some regimes. This indicated that the models indeed learn adapted embeddings here.

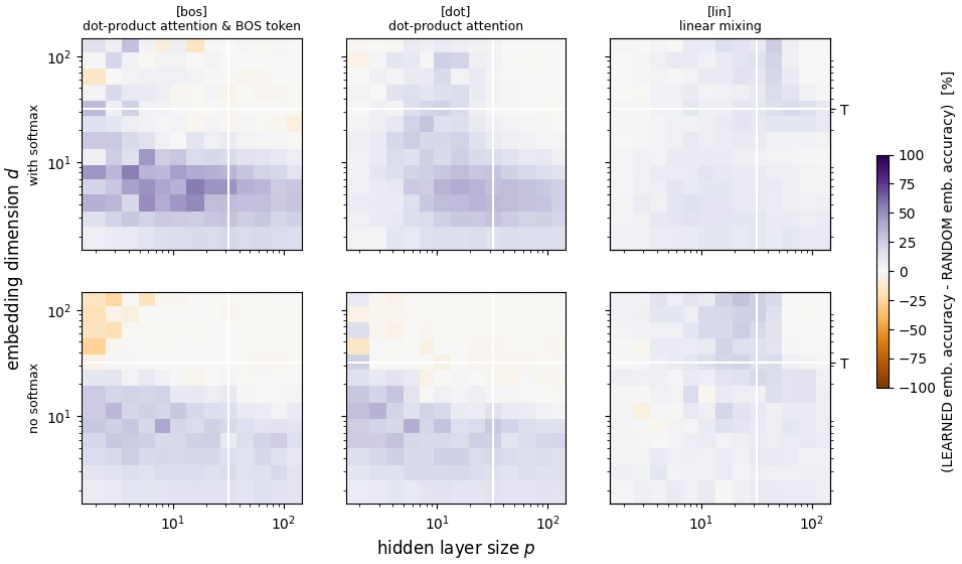


*Figure 11.* The difference between learned and random embeddings for $T = 32$. Orange indicates that the random embeddings perform better on average. Purple indicates that the learned embeddings perform better on average. Experimental settings as in Fig. 1.

### D.4. Model with alternative $T = 64$

We repeat the experiments presented in Fig. 1 for $T = 64$ in Fig. 12, leading to the same phenomenology, in line with our hypothesis that indeed the number of tokens $T$ determines the relevant transition point.

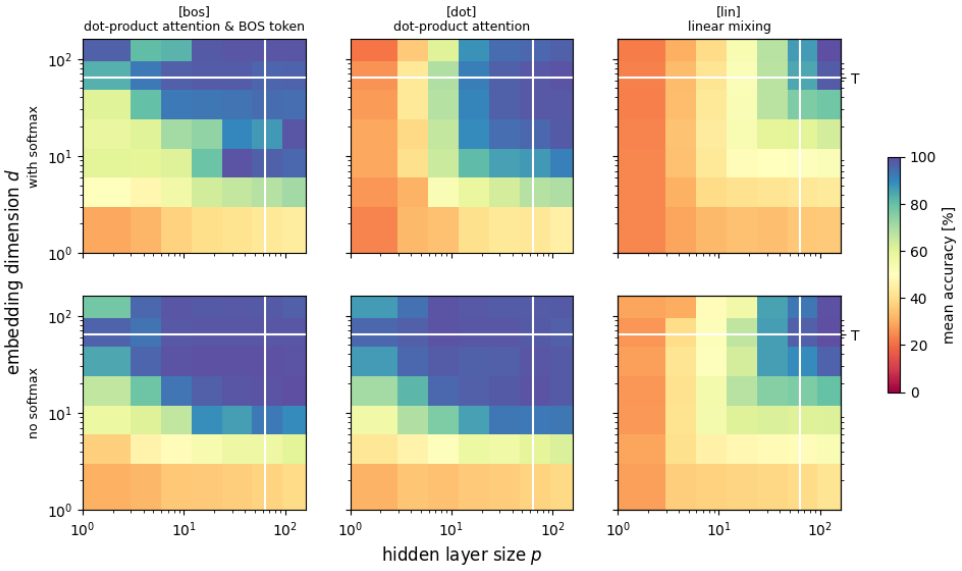


*Figure 12.* Phase diagrams for varying $p$ and $d$ for fixed $T = 64$ and $L = 10$ for `lin`, `dot` and `bos`. *(Top)* Models without softmax; *(Bottom)* Models with softmax. With five runs for every $d, p \in \{2, 4, 8, 16, 32, 64, 128\}$.

### D.5. Model with alternative $L = 15$

We repeat the experiments presented in Fig. 1 for $L = 15$ in Fig. 13, leading to the same phenomenology, in line with our hypothesis that indeed the number of tokens $T$ determines the relevant transition point, and not the sequence length $L$. However, the accuracy is comparatively worse when no high-accuracy solution is reached.

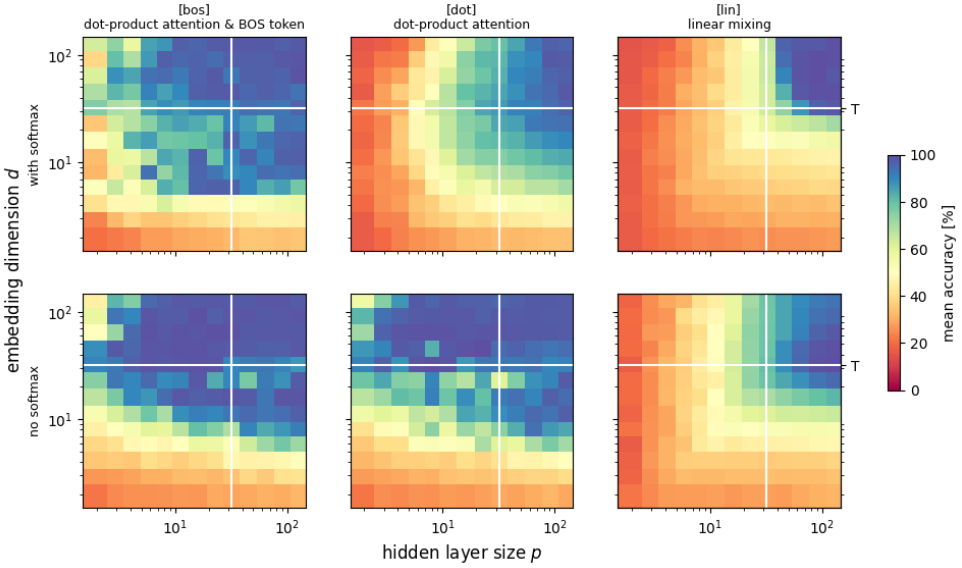


*Figure 13.* Experiments as in Fig. 1, but with sequence length fixed to $L = 15$.

### D.6. BOS mixing token

In Fig. 3 in the main, we describe how the $t_{\mathrm{BOS}}$ 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. 14.

### D.7. Overlaps

In Fig. 15 we show how the overlaps behave as a function of the dimension $d$ when $p$ is fixed.

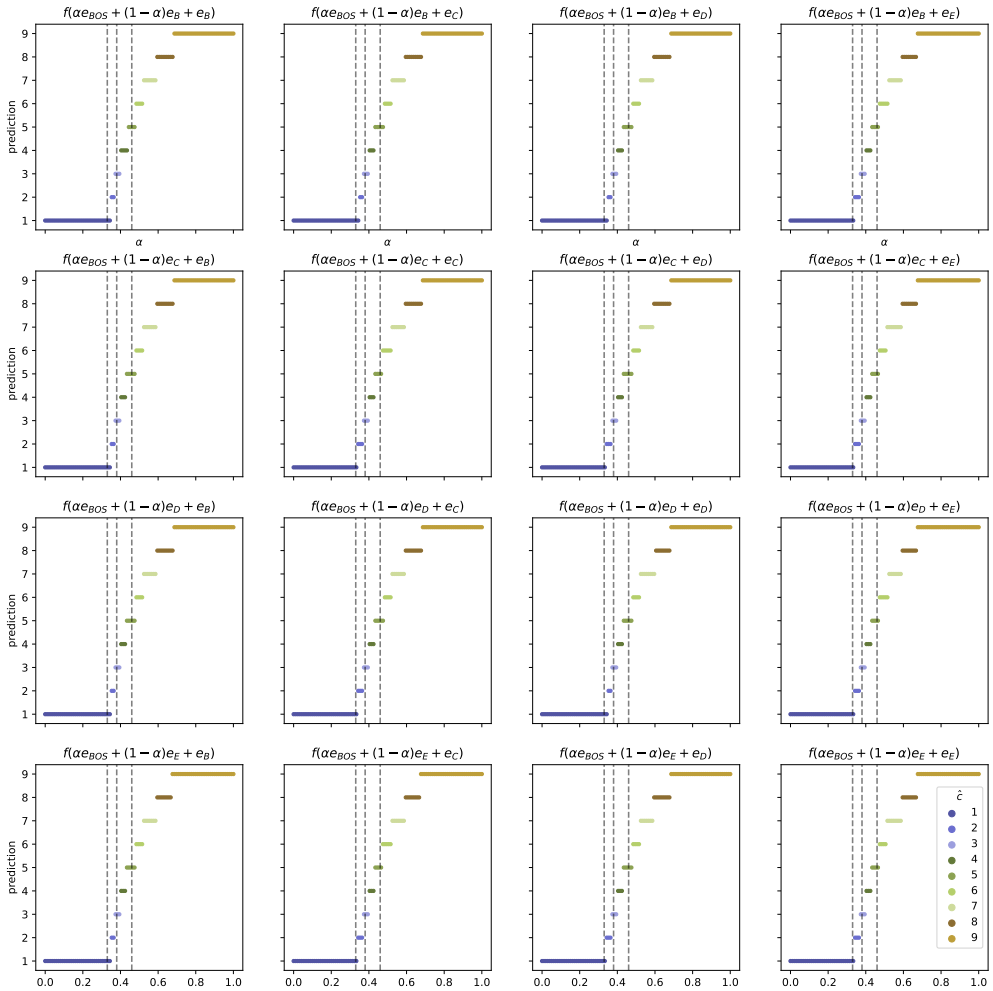


*Figure 14.* For the same model as in Fig. 3, we vary the inputs to the feature transformation $f$ to show it is independent on the precise input sequence, but only depends on the prevalence of $t_{\text{BOS}}$. We vary the inputs between the learned tokens $[B, C, D, E]$.

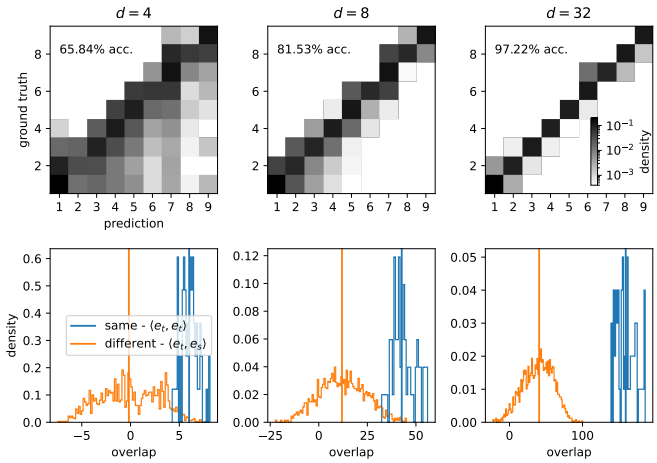


*Figure 15. Introspecting the Regime with Entangled Embeddings with* `dot` *($T = 64, L = 10, p = 128$).* We show examples of `dot` for $T = 64, L = 10, p = 128$ for varying the model dimension $d$. *(Top row)* The top row shows the confusion matrix of ground truth and predicted counts. *(Middle row)* Below is the overlap distribution between same and different token embeddings.