A Mechanism for Sample-Efficient In-Context Learning for Sparse Retrieval Tasks

Jacob Abernethy   ABERNETHYJ@GOOGLE.COM
Google Research & Georgia Institute of Technology

Alekh Agarwal   ALEKHAGARWAL@GOOGLE.COM
Google Research

Teodor V. Marinov   TVMARINOV@GOOGLE.COM
Google Research

Manfred K. Warmuth   MANFRED@GOOGLE.COM
Google Research

Abstract

We study the phenomenon of *in-context learning* (ICL) exhibited by large language models, where they can adapt to a new learning task, given a handful of labeled examples, without any explicit parameter optimization. Our goal is to explain how a pre-trained transformer model is able to perform ICL under reasonable assumptions on the pre-training process and the downstream tasks. We posit a mechanism whereby a transformer can achieve the following: (a) receive an i.i.d. sequence of examples which have been converted into a prompt using potentially-ambiguous delimiters, (b) correctly segment the prompt into examples and labels, (c) infer from the data a *sparse linear regressor* hypothesis, and finally (d) apply this hypothesis on the given test example and return a predicted label. We establish that this entire procedure is implementable using the transformer mechanism, and we give sample complexity guarantees for this learning framework. Our empirical findings validate the challenge of segmentation, and we show a correspondence between our posited mechanisms and observed attention maps for step (c).

Keywords: in-context learning, transformers

1. Introduction

In-context learning has emerged as a powerful and novel paradigm where, starting with Brown et al. (2020), it has been observed that a pre-trained language model can "learn" simply through prompting with a handful of desired input-output pairs from a new task. Strikingly, the model is able to perform well on future input queries from the same task by simply conditioning on this prompt, without updating any model parameters, and using a surprisingly small number of examples in the prompt to learn a target task. While model fine-tuning for few shot learning can be explained in terms of the vast literature on transfer learning and domain adaptation, ICL eludes an easy explanation for its sample-efficiency and versatility. In this paper, we study the question: *What are plausible mechanisms to explain ICL for some representative tasks and what is their sample complexity?*

Before discussing potential answers, we note why the ICL capability is surprising, and merits a careful study. Typical few-shot learning settings consist of a family of related tasks among which transfer is expected. On the other hand, the pre-training task of predicting the next token for language models appears largely disconnected from the variety of downstream tasks ranging

from composing verses to answering math and analogy questions or writing code, that they are later prompted for. More importantly, a typical prompt consists of a sequence of unrelated inputs for a problem, followed by the desired outputs. This should constitute a very unlikely sequence for the model, since its training data seldom contains input, output pairs for a single concept occurring together. Due to this abruptness of example boundaries, recognizing such a prompt as a sequence of independent examples to learn across is an impressive feat in of itself. Once the model *segments* this prompt, it still needs to learn a *consistent hypothesis across examples* in the prompt to map inputs to desired outputs, and then apply this learned hypothesis to fresh query inputs. Given that this capability has been primarily observed in transformer based models, we investigate if there are aspects of self-attention which are particularly well-suited to addressing the aforementioned challenges?

In this paper, we study all the questions mentioned above. A formal investigation of ICL was pioneered in Garg et al. (2022), who trained transformer models (Vaswani et al., 2017) from scratch that can learn linear regression via ICL. Within this model, gradient descent or closed form ridge regression approaches to map the prompt to a hypothesis were put forth in von Oswald et al. (2022) and Akyürek et al. (2022). These works show that the posited mechanisms can be implemented using a transformer, given an appropriate formatting and tokenization of the inputs. From a statistical perspective, Xie et al. (2021) and Zhang et al. (2022) cast ICL as posterior inference, with the former studying a mixture of HMM models and the latter analyzing more general exchangeable sequences. These works do not, however, provide mechanisms to implement the desired learning procedures using a transformer. Olsson et al. (2022) give some evidence that ICL might arise from a mechanism called induction heads, but do not discuss the sample complexity aspects or describe how induction heads might be leveraged to address a variety of learning tasks. More recently, Zhang et al. (2023); Ahn et al. (2023) further the closed form ridge regression perspective by showing convergence to a transformer that implements this formula in pre-training, but use a linearized self-attention mechanism. Concurrently to our work, Bai et al. (2023) show constructions to implement L steps of gradient descent using $O(L)$ transformer layers on regularized loss objectives, and bound the sample complexity of ICL for many tasks including sparse regression which we study here. Most related to our results on learning sparse retrieval tasks is the work of Edelman et al. (2022). The authors study the inductive bias of self-attention layers and demonstrate that self-attention layers are able to represent sparse functions efficiently. Further, a careful union bound on the class of bounded norm self-attention mechanisms implies that transformers can learn sparse functions efficiently. However, Edelman et al. (2022) do not show how the self-attention mechanism can implement a learning rule for the class of sparse functions. We defer a more detailed discussion relative to these works, as well as connections with the broader literature on uses of the ICL capability to Appendix A, and highlight some of the key departures in our study from prior work below.

Our Contributions. Our work studies the ICL process in an end-to-end manner. Unlike most prior works on ICL, which either require pre-training task to be identical to the downstream task (Garg et al., 2022; von Oswald et al., 2022; Akyürek et al., 2022), or comprised of some mixture of downstream tasks (Xie et al., 2021; Wies et al., 2023), we abstract the details of this procedure by representing it as a fixed and given *prior* distribution over sequences. Our results include:

- **Prompt segmentation:** We propose a segmentation mechanism for the prompt, which maximizes the likelihood of a proposed segmentation under the prior learned during pre-training. The mechanism crucially leverages aspect of the attention architecture to *learn the segmentation* with few examples. The sample complexity scales logarithmically with the number

of candidate delimiters and inversely in a gap parameter between the prior likelihoods of correctly and incorrectly segmented sequences.

- **Inferring consistent hypothesis:** We then take the segmentation of the prompt and illustrate how to infer a consistent hypothesis which explains all the (input, output) pairs in the prompt using a transformer model. For this part, we specialize to a family of *sparse retrieval tasks*, where the output is simply a token of the input, or the sum of a subset of input tokens. *This family is a useful abstraction of token extraction and manipulation tasks in practical ICL settings.* We show how attention can naturally leverage correlations to identify a consistent hypothesis on such tasks. The proposed mechanism finds an ϵ accurate hypothesis from a class $\mathcal{F}$ using $O\left(\frac{1}{\epsilon}\ln|\mathcal{F}|\right)$ examples.

- **Inference with the learned hypothesis:** We also show that the attention mechanism is well-suited to carry the learned hypothesis from the prompt and apply it to subsequent query inputs.

- **Empirical validation:** Finally, we validate some of our theoretical findings through experiments, showing the dependence of ICL on easily identifiable delimiters. For hypothesis learning, we show that transformer models can be indeed trained to solve the sparse retrieval tasks studied here, and that the attention outputs correspond to the key steps identified in our theoretical mechanisms.

Some of the key differentiators in our work from prior studies are that we specify the input vectors x and labels y by tokenizing each scalar value to a vector, which is more representative of how transformers process text, rather than giving covariates x as the d-dimensional input vector to the transformer. None of the prior works mentioned above consider this tokenized input representation. With this representation, we show that the self-attention mechanism provides a particularly convenient abstraction for ICL of functions that are sparse combinations of input tokens. The family of sparse functions is attractive as extracting some subset of tokens from specified text is perhaps the simplest text analysis task, which we abstract into a regression setup. Unlike many prior works, we use the properties of self-attention as an integral mechanism in our construction. In contrast, many of the arguments of ICL as gradient descent or closed form ridge regression are often simplified by adopting a linear or other simplified self-attention maps, that do not apply softmax (Zhang et al., 2023; Ahn et al., 2023; Bai et al., 2023), meaning that the self-attention mechanism is a hindrance rather than a helpful feature in these works. Taken together, we find that the change in our task setup and analysis sheds novel insights from a differernt perspective on ICL, than merely explaining it via gradient descent, and offer an interesting direction for further research.

2. Problem Setting and Notation

A language model $\mathbb{LM}$ is an oracle that takes as input elements of a *language* $\mathcal{V}^*$, sequences of *tokens* from a vocabulary $\mathcal{V}$, with $V := |\mathcal{V}|$. A typical language model is autoregressive: it aims to predict the next sequence of tokens from an prefix. To complete the phrase "I came, I saw", we construct `prompt = [<begin>,I,<space>,came,<comma>,<space>,I,<space>,saw]`, and input `prompt` $\rightarrow \mathbb{LM} \rightarrow$ `output`, and we expect that
`output = [<comma>,<space>,I,<space>,conquered,<end> ]`.

2.1. The Transformer Architecture

We describe the design of a language model using the architecture known as the decoder-only *transformer* (Vaswani et al., 2017). In short, transformers are models that process arbitrary-length token sequences by passing them through a sequence of layers in order to obtain a distribution over the next token.

For the following definition, we will need some special operators, which we describe here. The operation $\mathsf{softmax}(M)$ returns a matrix the same shape as M whose i, j entry is $\frac{\exp(M_{i,j})}{\sum_{j'} \exp(M_{i,j'})}$. The $\mathsf{concat}(M_1, M_2)$ operation stacks the matrices vertically. The operation $\mathsf{mask}(M)$ takes a square matrix M and returns M' such that $M'_{i,j} = M_{i,j}$ for $i \leq j$ and $M'_{i,j} = -\infty$ otherwise. (The $-\infty$ is converted to a 0 after the $\mathsf{softmax}$ operation.) The GeLu operation is the Gaussian Error Linear Unit.

Definition 1 *Let d, d_{att}, κ be arbitrary positive integers. A* transformer layer *is a function $\mathbb{T}_\Pi$ parameterized by matrices $\Pi := \{Q_k, K_k, V_k \in \mathbb{R}^{d_{att} \times d} \text{ for } k \in [\kappa], W_O \in \mathbb{R}^{d \times \kappa \cdot d_{att}}\}$, that maps, for any length N sequence, $\mathbb{R}^{d \times N} \to \mathbb{R}^{d \times N}$ using the following procedure:*

> **Input:** $X \in \mathbb{R}^{d \times N}$, **Set:** $A_k \leftarrow \mathsf{softmax} \circ \mathsf{mask}(d_{att}^{-1/2} X^\top Q_k^\top K_k X) \quad \forall k \in [\kappa]$
>
> **Set:** $X' \leftarrow W_O \, \mathsf{concat}(V_1 X A_1, \ldots, V_\kappa X A_\kappa)$, **Output:** $X + \mathsf{GeLu}(X') \in \mathbb{R}^{d \times N}$.

We omit the layer normalization present in implementations (Vaswani et al., 2017) for ease of presentation. A convenient aspect of transformer layers is their composability. Assume we have L transformer layers, where the ℓ-th layer is parameterized by $\Pi^\ell := \{W_O^\ell \in \mathbb{R}^{d \times \kappa \cdot d_{\text{att}}}; Q_k^\ell, K_k^\ell, V_k^\ell \in \mathbb{R}^{d_{\text{att}} \times d}, k \in [\kappa]\}$. The remaining piece we need for the full transformer model is the token embedding layer, which is parameterized by a matrix $W_E \in \mathbb{R}^{d \times |\mathcal{V}|}$. If we take a prefix $x \in \mathcal{V}^*$ with N tokens, and write it using one-hot encoding as a matrix $Z \in \{0, 1\}^{|\mathcal{V}| \times N}$, then $W_E Z$ is referred to as the "embedded" tokens. Once these embedded tokens are passed through one or more transformer layers to obtain Z', we can convert back to vocab space by $W_E^\top Z'$. Here $Z_{i,j}$ represents the model's estimated probability that the $j + 1^{\text{th}}$ token will be token i given the first j tokens in the sequence. We need these embeddings to be reasonably distinct.

Definition 2 *A (decoder-only) L-layer* transformer *is a parameterized function that maps $\mathbb{R}^{|\mathcal{V}| \times N} \to \mathbb{R}^{|\mathcal{V}| \times N}$ for any sequence length N where the input X is a one-hot encoding of a token sequence x in $\mathcal{V}^*$, and the output is a column-stochastic matrix Z. The parameters are given by the matrix W_E and the sequence $\Pi_1, \ldots, \Pi_L$. The full map is defined as the composition,*

$$X \mapsto Z = \mathsf{softmax}(W_E^\top \cdot \mathbb{T}_{\Pi_L} \circ \cdots \circ \mathbb{T}_{\Pi_1}(W_E \cdot X)).$$

The one-hot encoding can be replaced with other (possibly learned) encodings when $\mathcal{V}$ is large or infinite. Of much interest in this work is to understand what operations can be implemented using a transformer. To establish our results, we often show that certain operations $\mathbb{R}^{d \times N} \overset{\phi}{\to} \mathbb{R}^{d \times N}$ on embedded token sequences can be implemented using a transformer layer parameterized by Π. When there is a Π such that $\mathbb{T}_\Pi \equiv \phi$ for all N, then ϕ *can be implemented as a transformer layer*.

2.2. In-Context Learning

Let us now imagine that we hope to solve the following learning problem. We are given an input space $\mathcal{X}$ and output space $\mathcal{Y}$. We assume that $\mathcal{X}, \mathcal{Y} \subset \mathcal{V}^*$ for simplicity –i.e., we are able to express inputs/outputs in the given language. Assume we have a set $\mathcal{F}$ of functions $f : \mathcal{X} \to \mathcal{Y}$. A *task* in this setting is a pair $f, \mathcal{D}$, with $f \in \mathcal{F}$ a function and $\mathcal{D} \in \Delta(\mathcal{X})$ a distribution on inputs $x \in \mathcal{X}$. A *sample* S_n from this task is a collection of n labelled examples $\{(x_1, y_1), \ldots, (x_n, y_n)\} \subset \mathcal{X} \times \mathcal{Y}$ where the x_i's are samples i.i..d. from $\mathcal{D}$ and $y_i = f(x_i)$ for every $i \in [n]$. When viewed as a typical supervised learning setting, we would design a *learning algorithm* $\mathcal{A}$ that is able to estimate $\hat{f} \in \mathcal{F}$ from a sample S_n, $\{(x_1, y_1), \ldots, (x_n, y_n)\} \quad \to \quad \mathcal{A} \quad \to \quad \hat{f}_n$. The goal of $\mathcal{A}$ is to minimize expected loss $\mathbb{E}_{x \sim \mathcal{D}}[\mathrm{loss}(f(x), \hat{f}_n(x))]$ with respect to a typical sample $x \sim \mathcal{D}$ and (unknown) function f.

The in-context learning framework poses the idea that perhaps for a large family of tasks we do not need to design such an algorithm $\mathcal{A}$ and instead we can leverage a pre-trained language model in order to solve a large family of learning tasks. That is, for a sample above and a test point $x \sim \mathcal{D}$, we have the following setup

$$\texttt{Encoding}(\{(x_1, y_1), \ldots, (x_n, y_n)\}, x) \quad \to \quad \mathbb{LM} \quad \to \quad \texttt{output},$$

and, if the ICL process succeeds, we expect that $\texttt{output} = f(x)\texttt{<end>}$.

Since there is no canonical procedure to encode a set of example-label pairs guaranteed to be understood by $\mathbb{LM}$, the choice of $\texttt{Encoding}$ influences its output behavior. Empirical works use typical *delimiters*—special tokens that are typically used to give structure to documents by *segmenting* text into lists, relations, etc.—for this task. As part of our language definition, we assume that there is a set of special tokens $\mathcal{V}_{\texttt{delims}} \subset \mathcal{V}$ including, e.g. punctuations (`<comma>`, `<colon>`, `<semicolon>`), or spacing characters (`<space>`, `<newline>`, `<tabspace>`). We assume that the user has selected one delimiter that separates the n examples, which we will call `<esep>`, and another that distinguishes between x_i and y_i, which we will call `<lsep>`. The only requirement is that `<esep>` and `<lsep>` are distinct elements of $\mathcal{V}_{\texttt{delims}}$, and that `<esep>` and `<lsep>` do not occur in any x, y examples generated in the task. With this in mind, we define $\texttt{Encoding}(\{(x_1, y_1), \ldots, (x_n, y_n)\}, x)$ as

$$\texttt{<begin>}x_1\texttt{<lsep>}y_1\texttt{<esep>}x_2\texttt{<lsep>}y_2\texttt{<esep>} \ldots \texttt{<esep>}x_n\texttt{<lsep>}y_n\texttt{<esep>}x\texttt{<lsep>}. \tag{1}$$

We note that the x's and y's have variable length and are being concatenated above.

3. An Overview of Results

We now survey the core results of the paper on segmenting the input sequence through delimiter identification, and the subsequent hypothesis learning.

3.1. Segmenting an input sequence

Suppose we have an underlying distribution $p_0(\cdot)$ on $\mathcal{V}^*$, which measures the typical likelihood of sequences observed "in the wild", and this distribution is encoded in a transformer through the pre-training process. The goal of the segmentation mechanism is to identify a pair of separators `<lsep>`, `<esep>` $\in \mathcal{V}_{\texttt{delims}} \times \mathcal{V}_{\texttt{delims}}$, such that the input z can be *reasonably* decomposed as:

$$z = \texttt{<begin>}x_1\texttt{<lsep>}y_1\texttt{<esep>} \ldots \texttt{<esep>}x_k\texttt{<lsep>}y_k\texttt{<esep>}x_*\texttt{<lsep>}.$$

To formalize a reasonable decomposition of z obtained using delimiters `<lsep>`,`<esep>`, we define its likelihood by leveraging the base model p_0 and then follow a *maximum likelihood segmentation*:

$$\widehat{\texttt{<lsep>}}, \widehat{\texttt{<esep>}} = \underset{\substack{\texttt{<lsep>}\in\mathcal{V}_{\texttt{delims}} \\ \texttt{<esep>}\in\mathcal{V}_{\texttt{delims}}}}{\text{argmax}} \quad p_0(x_*) \prod_{i=1}^{k} p_0(\texttt{<begin>}x_i\texttt{<end>}) p_0(\texttt{<begin>}y_i\texttt{<end>}),$$

(2)

We note that the number k of examples as well as the sequences x_i and y_i identified depend on the separators used, and hence are all functions of the optimization variables `<lsep>`,`<esep>` in the objective (2). This is a natural objective to decompose an input sequence, as it posits that the individual x, y sequences in ICL should be plausible under the base distribution. Crucially, if the true label separator `<lsep>`* is very unlikely to occur in a natural sequence, then a wrong segmentation which mistakenly includes `<lsep>`* as part of some x_i or y_i will be very unlikely under p_0.

The first result of our paper is that the objective (2) can be implemented using a transformer.

Theorem 3 (Transformers can segment) *There exists a transformer with $O(1)$ layers and $O(\mathcal{V}_{\texttt{delims}} \times \mathcal{V}_{\texttt{delims}})$ heads per layer which computes $\widehat{\texttt{<lsep>}}, \widehat{\texttt{<esep>}}$ according to (2).*

Next, we evaluate the sample requirements to learn an accurate segmentation.

Theorem 4 (Sample complexity of segmentation, informal) *Let c measure how much more likely a correctly segmented sequence is than an incorrectly segmented one under the task distribution $\mathcal{D}$. Given a minimum probability parameter ν, maximum likelihood segmentation (2) returns the correct label and example separators with probability $1 - \delta$ after seeing $n = \Omega\left(\frac{(\log(1/\nu))^2 \log \frac{|\mathcal{V}_{\texttt{delims}}|}{\delta}}{c^2}\right)$.*

In practice, the example and label separators are chosen so as to make the segmentation fairly unambiguous, ensuring that c is large, and the sample cost of learning segmentation is quite small. We can further enhance the objective (2) to include priors over `<lsep>` and `<esep>` being delimiters to zoom in on typical choices faster. We omit this extension to convey the basic ideas clearly. We note that among the various prior works that study ICL, the task of segmenting an input sequence is only considered to any degree in Xie et al. (2021) in an HMM-like model, while we offer a much more general explanation.

3.2. Learning a consistent hypothesis

Having generated a segmentation, the next step in ICL is to take the inputs $(x_i, y_i)_{i=1}^{n}$ identified above and generate a hypothesis $\widehat{f}$ such that $\widehat{f}(x) \approx f^\star(x)$, where $y_i = f^\star(x_i)$. To formalize the hypothesis learning setup, we focus on a specific family of learning problems that we define next.

Definition 5 (Tokenized sparse regression) *Fix an input space $\mathcal{X}$, an output space $\mathcal{Y}$, a basis map $\psi(x) : \mathbb{R}^{\dim(\mathcal{X})} \to \mathbb{R}^m$, and distribution $\mathcal{D}$ over $\mathcal{X}$. Given $s \leq m$, an s-sparse tokenized regression problem is defined by weights $\beta_1, \ldots, \beta_m \in \{0, 1\}^m$ such that $|\{j : |\beta_j| = 1\}| = s$ and $y = \sum_{j=1}^{m} \beta_j \psi(x)_j$.*

In words, a tokenized sparse regression problem maps from inputs to outputs by taking sparse linear combinations of the inputs under some fixed basis transformation. We call the task tokenized due to the way in which the transformer processes the input x during ICL, accepting it one coordinate at a time as we will see momentarily. This is to distinguish from the regression tasks studied in prior works (Garg et al., 2022; Akyürek et al., 2022; von Oswald et al., 2022; Ahn et al., 2023; Bai et al., 2023; Zhang et al., 2023), which consider each vector x to be a token.

In particular, we study problems where the basis ψ is fixed across contexts and only the coefficients β_i vary across tasks. Since ψ is fixed, it can be assumed to be known from pre-training and we focus on the case of $\mathcal{X} = \mathbb{R}^m$ and $\psi(x)_i = x_i$, that is the basis is just the standard basis and the task is sparse linear regression. We focus on these tasks because *extracting and manipulating a few input tokens is emblematic of many text processing tasks where ICL is often used in practice.*

We analyze the following estimator for all $i \in [n]$:

$$f_i \in \big\{ (j_1, \ldots, j_s) \in [m]^s \; : \; \max_{t=1,2,\ldots,i} |x_{t,j_1} + \ldots + x_{t,j_s} - y_t| \le \epsilon \big\}. \tag{3}$$

The optimization problem (3) can be implemented with a transformer, as stated next.

Theorem 6 (Transformers find a consistent hypothesis) *There exists a transformer with $O(m)$ layers and 1 head per layer which computes an f_i according to (3) after reading example x_i.*

The estimator above is natural for the task, as it finds a solution with a zero loss on the training samples. Implementing this estimator is particularly natural with a transformer using properties of the attention mechanism, that is well suited to extracting coordinates of x which are highly correlated with the label y. In fact, the actual mechanism computes a weighting over $\{1, 2, \ldots, m\}$ as candidate solutions from each example, and then returns f_i as the coordinate with the largest cumulative weight (across examples) after each example i. This provides further robustness in case of label noise. For this procedure, we provide the following sample complexity guarantee.

Theorem 7 (Sample complexity of hypothesis learning, informal) *For any $\epsilon > 0$, suppose the initial token embeddings are such that tokens z_α, z_γ with $|z_\alpha - z_\gamma| \ge \epsilon$ have nearly orthogonal embeddings. Let f_n be any hypothesis returned by Equation 3 after seeing n examples from the s-sparse token regression task. Then for $n = \Omega(s \log(m/\epsilon)/\epsilon)$ we have $\mathbb{E}[|f_n(x) - f^\star(x)|] \le 2\epsilon$.*

The condition on the token embeddings is natural, since aliased tokens with different values can be problematic for learning. Our sample complexity matches the typical guarantees in sparse regression, which scale with the $\frac{s \log m}{\epsilon}$. Note that the estimator (3) learns f_n from scratch, in that there is no use of the pre-training to bias the estimator towards certain functions. While this is also done in several prior works on ICL (Garg et al., 2022; Akyürek et al., 2022; von Oswald et al., 2022), in practice ICL is often used in settings where the correct y has a high probability under the base distribution p_0, given x. Improving the estimator (3) to prioritize among the consistent tokens j using the prior probabilities $p_0(x_{i,m+1}|x_{i,1}, \ldots, x_{i,m})$ is an easy modification to our construction and allows us to further benefit from an alignment between p_0 and $\mathcal{D}$.

We highlight that the estimator (3) is very different from closed form linear regression or gradient descent mechanisms offered in Akyürek et al. (2022); von Oswald et al. (2022), as well as proximal gradient for regularized objectives studied for sparse regression in Bai et al. (2023), and is much more tailored to the attention mechanism as mentioned above. This offers an alternative hypothesis for how ICL happens, which will be partially verified by some of our empirical findings.

```
Double, double toil and trouble: / Macbeth; To die, - To sleep, - To sleep!
/ Hamlet; This above all: to thine own self be true / Hamlet; A deed without a name /
```

Figure 1: An example of an ICL task: quotes from Shakespeare followed by the name of the play. Correct delimiters are $(\texttt{<lsep>}, \texttt{<esep>}) = (/, ;)$, yet the presence of other potential delimiters creates ambiguity.

4. Segmenting an ICL Instance

In this section, we provide more details about the segmentation mechanism and the underlying assumptions. Given a sequence of N tokens $z = z_1, \ldots, z_N$, and given a pair $\sigma = (\texttt{<lsep>}, \texttt{<esep>})$, we first segment the sequence into as many chunks as possible, separated by $\texttt{<esep>}$ tokens:

$$z = \texttt{<begin>}u_1\texttt{<esep>}u_2\texttt{<esep>}\ldots\texttt{<esep>}u_k.$$

Then, for each chunk we segment $u_i = x_i\texttt{<lsep>}y_i$ according to the occurrence of $\texttt{<lsep>}$. If we find multiple occurrences of $\texttt{<lsep>}$, we infer that this σ is infeasible, setting $p_\sigma(z) = \nu$ to be a minimum probability. If no $\texttt{<lsep>}$ is found, then we set $u_i = x_i$. With this segmentation in mind, we can now define our probability model p_σ as follows,

$$p_\sigma(z) = p_0(x_*) \prod_{i=1}^{k} p_0(\texttt{<begin>}x_i\texttt{<end>})p_0(\texttt{<begin>}y_i\texttt{<end>}). \tag{4}$$

To understand the meaning of this definition better, it is helpful to look at an example. In Figure 1, we show a concatenation of quotes from Shakespeare and the corresponding play names. Using candidate delimiter pairs $\sigma = (/, ;)$ and $\sigma' = (:, -)$, we get two different likelihood models:

$$p_\sigma(z) = p_0(\texttt{<begin>Double,...trouble:<end>})p_0(\texttt{<begin>Macbeth<end>})$$
$$\cdot p_0(\texttt{<begin>To die, <end>})...$$
$$p_{\sigma'}(z) = p_0(\texttt{<begin>Double,...trouble<end>})p_0(\texttt{<begin>/ Macbeth;...To die,<end>})$$
$$\cdot p_0(\texttt{<begin>To sleep,<end>})p_0(\texttt{<begin><end>})...$$

It is also important to note that, even though the model above provides a nicely-factored estimate of the sequence likelihood, we are still able to compute the probability of other segments. For any $1 \le i < j \le N$, $p_\sigma(z_i \cdots z_j | z_1 \cdots z_{i-1})$ can be evaluated from the model above, even if $[i:j]$ may cross delimiter boundaries. The objective (2) maximizes $p_\sigma(z)$ over σ to select the model.

Implementation using a transformer. We now describe a high-level sketch of a transformer that can implement the objective (2). As mentioned in Theorem 3, our construction uses one head for each candidate σ, where we use the head corresponding to σ to compute $p_\sigma(z_{1:i})$ at each token i. The maximum over delimiter pairs σ by looking across heads is subsequently taken by the output MLP layers. Here we focus on the operations within each head. For a fixed $\sigma = (\texttt{<lsep>}, \texttt{<esep>})$, the first two layers of the transformer identify the nearest occurrences of $\texttt{<esep>}$ and $\texttt{<lsep>}$ to each token i. This can be done by attending to the occurrence of these tokens with the largest index, which is natural with softmax and position embeddings. Next, we find the first occurrence of $\texttt{<lsep>}$ following each $\texttt{<esep>}$ and add the log probabilities of the subsequence from $\texttt{<esep>}$ to the token before $\texttt{<lsep>}$ and from the token after $\texttt{<lsep>}$ to z_i. Implementing these operations is again relatively straightforward using a composition of soft attention layers, assuming access to a certain conditional probability evaluation module from pre-training, that returns log of conditional probability of a token, conditioned on a prefix sequence. Adding all the log probabilities gives us the desired output. Details of this construction are provided in Appendix C.

Sample complexity of segmentation. Let $\sigma^\star$ be the true pair of delimiters used to generate the input z. Then we expect the procedure (2) to return $\sigma^\star$ only if the sequences u_i identified under $\sigma^\star$ are more probable than those identified under a different delimiter σ'. To facilitate this, we now make an assumption, before stating the formal sample complexity guarantee.

Assumption 1 *Let $\mathcal{D} \in \Delta(\mathcal{V}^*)$ and f define the task at hand, with the input segmentation determined by $\sigma^\star = (\texttt{<lsep>}, \texttt{<esep>})$. For an ICL sequence z, let $u_i = x_i\texttt{<lsep>}y_i$ be the i-th example/label chunk (under $\sigma^\star$). Consider the expected log-likelihood ratio of u_i conditioned on its prefix according to two probability models, one with the correct segmentation using $\sigma^\star$ and alternatively with some other delimiter pair σ', conditioned on $\textsf{prefix}_{i-1}$, which is the prefix of the sequence before u_i. Now, for some $c > 0$ we assume that for any $\sigma' \neq \sigma^\star$ and with probability 1 for any $\textsf{prefix}_{i-1}$*

$$\mathbb{E}_{\substack{x_i \sim \mathcal{D} \\ y_i = f(x_i)}} \left[\log \frac{p_{\sigma^\star}(u_i | \textsf{prefix}_{i-1})}{p_{\sigma'}(u_i | \textsf{prefix}_{i-1})} \mid \textsf{prefix}_{i-1} \right] \geq c.$$

Note that in both cases above, the segmented chunks, u_i, are always determined by $\sigma^\star$ even though their likelihood is evaluated on the "false" σ'. What does this assumption mean? It may appear to be highly technical, but it encodes something that we would very naturally expect: incorrectly segmented data should look very weird (unlikely) relative to correctly interpreted data. For instance, revisiting the example from Figure 1, consider the second chunk according to the true segmentation with $\sigma^\star = (/, ;)$, $u_i =$ "To die, - To sleep, - To sleep! / Hamlet;". When correctly segmented we obtain the pair $x_i = \texttt{To die, - To sleep, - To sleep!}$ and $y_i = \texttt{Hamlet}$, and the likelihood $p_{\sigma^\star}(u_i) = p_0(x_i)p_0(y_i)$. On the other hand, when we use the incorrect delimiters $\sigma' = (:, -)$ we get a much less natural segmentation, with model estimate

$p_{\sigma'}(u_i \mid \textsf{prefix}_{i-1}) = p_0(\texttt{To die,<end> | <begin>/ Macbeth;})$
$\cdot p_0(\texttt{<begin>To sleep,<end>}) \cdot p_0(\texttt{<begin><end>})p_0(\texttt{<begin>To sleep!/Hamlet}).$

Assumption 1 says that the model estimate for these chunks according to $p_{\sigma^\star}$ should be much higher, on average, than that for $p_{\sigma'}$. This is indeed the only needed assumption to obtain the following.

Theorem 8 *Let $\nu > 0$ be such that $p_{\sigma'}(u_i|\textsf{prefix}_{i-1}) \geq \nu$ where u_i is the i-th chunk under $\sigma^\star$, for all σ', u_i and $\textsf{prefix}_{i-1}$ almost surely. Under Assumption 1, the maximum likelihood segmentation algorithm (2) outputs the correct delimiters $\sigma^\star$ w.p. at least $1 - \delta$, as long as $n \geq \frac{16\left(\log \frac{1}{\nu}\right)^2 \log \frac{|\mathcal{V}_{\texttt{delims}}|}{\delta}}{c^2}$.*

5. Learning a Consistent Hypothesis for Tokenized Sparse Regression

We now formalize the results for extracting a consistent hypothesis for the tokenized sparse regression tasks described in Definition 5. For intuition, we begin with $s = 1$.

The 1-sparse tokenized regression task: Recall that when $s = 1$, we have $y_i = x_{i,f^\star}$ for each example $i \in [n]$, and $f^\star \in [m]$ is the coordinate of x_i being copied to y. The objective (3) simplifies to finding an index $f_i \in [m]$ such that $|x_{t,f_i} - y_t| \leq \epsilon$ for all examples $t \leq i$. We now describe the key elements of a transformer that implements such a procedure; details in Appendix D. The construction is depicted in Figure 2. Let $\bar{z}_\alpha$ be the initial token embedding, i.e. the α-th column of $W_E X$ in Definition 2.

$f_1(x_2)$

$x_{1,1}\ x_{1,2}\ x_{1,3}\ \ y_1\ \ x_{2,1}\ x_{2,2}\ x_{2,3}\ \ y_2$
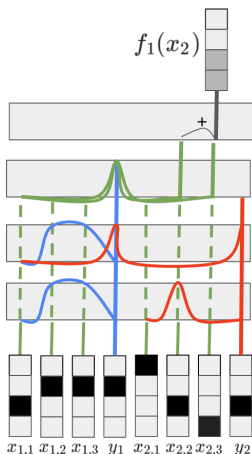
Figure 2: A transformer for 1-sparse tokenized regression with $n = 2$ examples and $m = 3$ tokens per example. The curved lines show attentions, with heights proportional to the attention. The blue and red attention lines show the attentions of y_1 and y_2 over the previous tokens. The green attention lines show the attentions of $x_{2,2}$ and $x_{2,3}$ over the previous tokens. In this case, $f^\star = 2$. After the first example, there is ambiguity between $f_1 \in \{2, 3\}$, hence the output $f_1(x_2)$ mixes these and is not correct. After the second example, the answer is uniquely determined, for inference on third example and beyond. In the first layer, each y_i attends to tokens $x_{i,j}$ from example i to find all consistent hypotheses in example i. By attending across previous y_t's, each y_i aggregates these hypotheses over all preceding inputs $t \leq i$. Example $i + 1$ attends to y_i to predict using the aggregated hypothesis in the final two layers.

Assumption 2 *Let $\bar{z}_\alpha$ be the input embedding of a token z_α. For any z_α, z_γ, such that $|z_\alpha - z_\gamma| \geq \epsilon$, we have $|\langle \bar{z}_\alpha, \bar{z}_\gamma \rangle| < 1/2\tau$, for some $\tau \geq 1$. If $|z_\alpha - z_\gamma| \leq \epsilon$, then $\langle \bar{z}_\alpha, \bar{z}_\gamma \rangle \geq 0$ and $\langle \bar{z}_\alpha, \bar{z}_\alpha \rangle = 1$.*

One embedding which satisfies the assumption maps all z_α, z_γ s.t. $|z_\alpha - z_\gamma| \geq \epsilon$ to orthogonal unit vectors in $\mathbb{R}^{\lceil (1/\epsilon) \rceil}$. In Appendix D we discuss ways to cut the dimension from $O(\frac{1}{\epsilon})$ to $\tilde{O}(\tau^2)$.

Given such an embedding, we can now use inner products between tokens to detect similarity, which is the key first step in our construction. This is also very natural to implement using attention. We define the first attention layer so that each $x_{i,j}$ only attends to itself and y_i only attends to $x_{i,1}, \ldots, x_{i,m}$ using the position embeddings. The attention weight between y_i and $x_{i,j}$ is defined using the inner product of their first layer embeddings. By Assumption 2, this inner product is large for $j = f^\star$ and small whenever $|x_{i,j} - y_i| > \epsilon$. In Appendix D, we define the query and key matrices to induce such an attention map. Under Assumption 2, this attention map identifies a good hypothesis for example i, as formalized below.

Lemma 9 *Given an example i, let $\mathcal{J}_i = \{j : |x_{i,j} - y_i| \leq \epsilon\}$, and let $f^\star \in [m]$ be such that $y_i = x_{i,f^\star}$. Under Assumption 2, for any $m \geq 2$, the output of the first layer at $f^\star$ is larger than the output at any $j \in [m] \setminus \mathcal{J}_i$ by at least $e/(4(m + 1))$.*

This key step in our construction gives us some consistent hypothesis from $\mathcal{J}_i$, for each example i *individually*. The second layer now finds a hypothesis consistent with all examples $(x_t, y_t), t \leq i$, and saves this hypothesis in token y_i^3, which is the input to the third layer. The remaining two layers implement appropriate copy and extraction mechanisms for the hypothesis f_i identified at y_i to be applied to the next input x_{i+1} by first extracting the value at x_{i+1,f_i} and then outputting this value at the token $x_{i+1,m}$ as the ICL prediction at the $(i + 1)$th example. We illustrate the key points of the construction in Figure 2. For more details, we refer the reader to Appendix D.

Prior work has focused on tasks where x_i is not decomposed into one token per coordinate but rather given to the transformer as a vector in $\mathbb{R}^m$ directly. In Appendix F we demonstrate how this setting can be reduced to the tokenized setting that we study here.

Iterative deflation for s-sparse token regression: For the more general case, it is tempting to directly apply the 1-sparse construction once more and hope that all the tokens in $f^\star$ will be identified simultaneously, since they all should have a high inner product with y_i under Assumption 2. However, suppose that $f^\star = (1, 2)$ and let us say that the hypothesis $(1, 3)$ is also consistent. Then an approach to learn 2 coordinates independently using the approach from the previous section can

10

result in the estimate $(2, 3)$ which might not be consistent. To avoid this, we identify coordinates one at a time, and deflate the target successively to remove the identified coordinates from further consideration. The deflation procedure works because of the following crucial lemma.

Lemma 10 *Under Assumption 2 with $\tau \geq 2s$, for any $\mathcal{C} \subseteq f^\star$ and, $j \in f^\star \setminus \mathcal{C}$, we have $\langle \bar{x}_{i,j}, \bar{y}_i - \sum_{j' \in \mathcal{C}} \bar{x}_{i,j'} \rangle \geq \frac{3}{4}$,[1] while if $|x_{i,j} - x_{i,j'}| \geq \epsilon$ for all $j' \in f^\star$ then we have $\langle \bar{x}_{i,j}, \bar{y}_i - \sum_{j' \in \mathcal{C}} \bar{x}_{i,j'} \rangle \leq \frac{1}{4}$.*

This lemma says that the deflated target favors unidentified coordinates in $f^\star$. Next, we show that the deflation procedure, to remove all previously identified tokens from each y_i, can be implemented using attention. We then stack $O(s)$ layers of the 1-sparse task, with a deflation layer after extracting each coordinate to identify a complete consistent hypothesis that optimizes the objective 3. Putting everything together, we have the following formal version of the earlier informal Theorem 7

Theorem 11 *For any $\epsilon > 0$, let f_n be an optimum of (3) after seeing n examples from the s-sparse token regression task. Then there exists an embedding of $x_{i,j}, y_i, \forall i \in [n], j \in [m]$ in $\mathbb{R}^{O(s/\epsilon)}$ satisfying Assumption 2, s.t. for any $n = \Omega(s \log(m/\epsilon)/\epsilon)$, w.p. $1 - \delta$, $\mathbb{E}[|f_n(x) - f^\star(x)|] \leq 2\epsilon$.*

More details for the construction and the proof of the sample complexity are in in Appendix E.

6. Empirical Results

In this section, we report some empirical findings on the 1-sparse tokenized regression task. For other experiments detailing the sensitivity of ICL to delimiters, we refer the readers to Appendix G.3.

We follow the experimental setup from Garg et al. (2022) and Akyürek et al. (2022), building on the implementation of Akyürek et al. (2022). We use transformers with 8 layers, 1 head per layer and embedding size 128. We experiment on the 1-sparse tokenized regression task with $(x_i, y_i)_{i \in [n]}$ generated in the following way. First a random hypothesis $f^\star$ is drawn uniformly at random from $[m]$, where $m = 5$. Next, $x_i \in \mathbb{R}^m$ is sampled i.i.d from a fixed distribution which is either a standard Gaussian ($x_i \sim \mathcal{N}(0, I_{5 \times 5})$), or uniform over $\{+1, -1\}^5$ ($x_i \sim Unif(\{+1, -1\}^5)$). y_i is always set to $f^\star(x_i)$. We refer to the first setting as the Gaussian setting and the second as the Rademacher setting. We train three different transformers, one for each of the two settings, and one where the samples come from a uniform mixture over both Gaussian and Rademacher settings. After pre-training we carry out the ICL experiments by generating 64 example sequences of length 5, either all from the Gaussian setting or the Rademacher setting. A randomly drawn $f^\star$ is sampled, fixed and shared across these 64 sequences, to allow averaging of results across multiple sequences.

In Figure 3 we show the averaged loss of a model pre-trained on the uniform mixture of the settings, attention weights at the final layer (8) and attention weights at layer 6 for both settings. Appendix G.1 shows results for models trained using Gaussian or Rademacher examples only and results for tasks with sparsity $s = 3$. For the loss plots, y-axis is the loss of the ICL inference at each example and x-axis is the number of in-context examples observed. *All examples are indexed from* 0. We see that the model reaches a loss of 0 in the Gaussian setting from a single sample, which is information theoretically optimal. In the Rademacher setting there are often multiple coordinates consistent with $f^\star$ on the first 3 or 4 examples, which results in higher loss compared to the Gaussian setting. In this setting $f^\star$ is typically learned as soon as the correct coordinate is disambiguated. In Figures 3 (b) and (e) we plot a heatmap of the attention of the tokens at which the model outputs its predicted label, corresponding to the last x token. The X-axis corresponds to example index $i \in [n]$

---

1. Here $\bar{x}_{i,j}$ and $\bar{y}_i$ refer to the input embeddings of these tokens satisfying Assumption 2.

(*a*) Loss (Gaussian), $y_i = x_{i,0}$

(*b*) Attention at layer 8 (Gaussian)

(*c*) Attention at layer 6 (Gaussian)

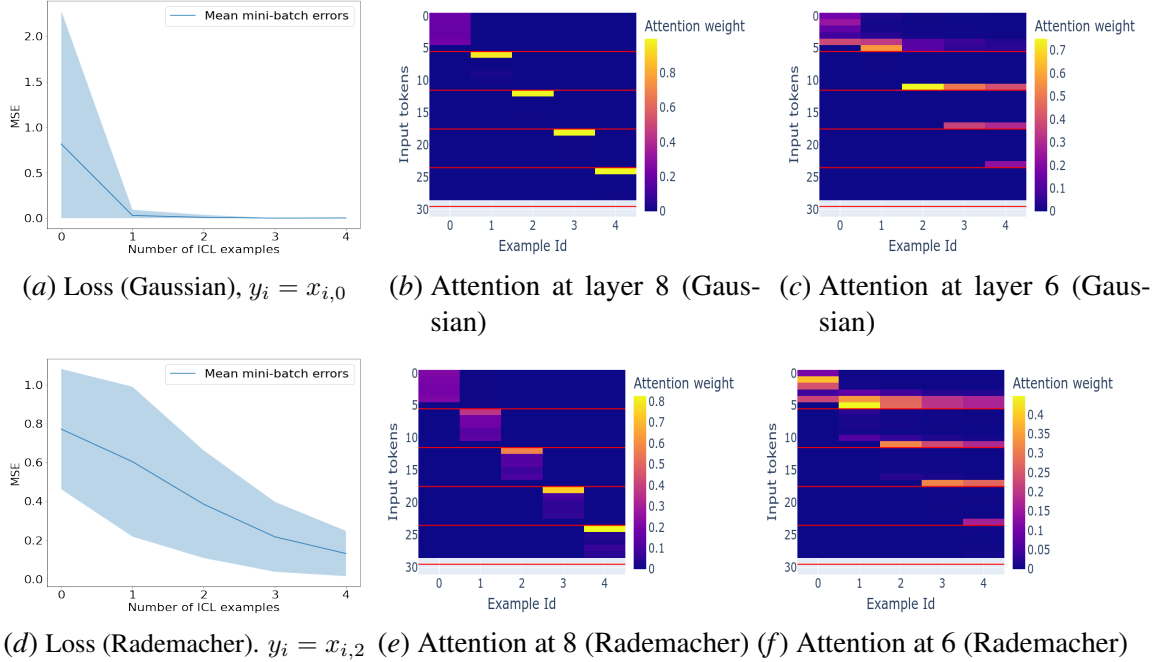(*d*) Loss (Rademacher). $y_i = x_{i,2}$ (*e*) Attention at 8 (Rademacher) (*f*) Attention at 6 (Rademacher)

Figure 3: Loss and attention plots for 1-sparse tokenized regression for Gaussian (top) and Rademacher (bottom) inputs. Loss drops to zero as soon as $f^\star$ is determined, and attentions follow the construction of Section 5. In panels (b) and (e), the prediction tokens for each example (on X-axis) attend to the index of $f^\star \equiv 0$ in x_i (on Y-axis) to predict y_i correctly. In panels (c) and (f), the prediction tokens attend to all previous labels (indices $5, 11, 17, \ldots$) to aggregate a consistent hypothesis across previous examples.

and the Y-axis shows all mn tokens across examples. The tokens for each example are separated by the red lines in the heatmap. For example i (starting at $i = 0$), the label is predicted at token index $6 * (i + 1) - 2$ in our sequence (corresponding to the input $x_{i,5}$), and we show the attention weights from this token over all tokens in the sequence. For these experiments $f^\star$ is fixed to be the 0 index. For the Gaussian setting (b), we see that the attention weights, starting at example $i = 1$, are peaked at $f^\star$. The prediction token for example 1 attends to token 6 accordingly to correctly predict y_1. In the Rademacher setting (e) we see that there is more variance, due to the fact that there are often multiple consistent hypothesis for the first few examples, however as the transformer processes more examples from the sequence the attention becomes more peaked at $f^\star$. This behavior is consistent with our theoretical construction. We note that our theoretical construction from Section 5 implies that attention weights in the last layer should be split uniformly over all j which are consistent with $f^\star$ up to example i. In Appendix G we also empirically demonstrate that this is the case by looking at the attention on a single randomly sampled sequence. In Figure 3 (c) and (f) we plot the attention of the prediction tokens but at layer 6 of the transformer, that is the third layer counting from the final layer. Surprisingly, we see that attentions are peaked at the tokens holding the labels for $f^\star$, that is $x_{i,5}$. This is analogous to the step where y_i attends to all previous y_s in our construction to aggregate across examples, and we expect its role is the same here. We emphasize that this behavior is *not* consistent across our experiments but was only observed for the 1-sparse 5-dimensional tasks.
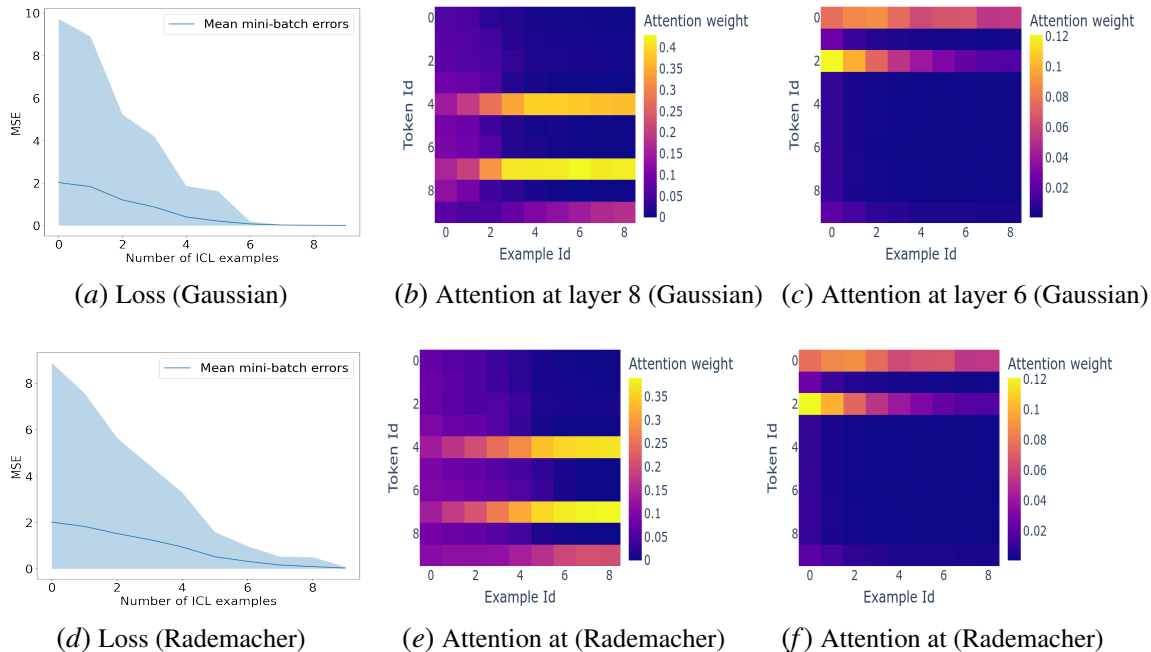
(a) Loss (Gaussian)   (b) Attention at layer 8 (Gaussian)   (c) Attention at layer 6 (Gaussian)

(d) Loss (Rademacher)   (e) Attention at (Rademacher)   (f) Attention at (Rademacher)

Figure 4: Loss and attention for $m = 10$, $s = 3$ for Gaussian and Rademacher inputs with $f^\star \equiv [4, 7, 9]$. The (j, i)-th entry of the heatmaps corresponds to the attention of the predictor token, $11(i + 1) - 2$, of the i-th example to the j-th token in the i-th example, $11i + j$. X-axis indexes the predictor tokens and Y-axis indexes the token ids within each example.

In Figure 4 we include experiments for the Gaussian and Rademacher token tasks with $m = 10$ and $s = 3$. In this experiment the target is $f^\star \equiv [4, 7, 9]$, that is $y_i = x_{i,4} + x_{i,7} + x_{i,9}$. We observe similar behavior to the $m = 5$, $s \in \{1, 3\}$ experiments for both the loss and attention at the final layer, however, the attention weights deviate slightly from the uniform distribution on $\{4, 7, 9\}$. The attention at layer 6, however, did not follow the pattern observed in Figure 3 and is seemingly arbitrary as can be observed from Figure 4 (c) and (f). In summary, we find that the attention maps at the final layer of the transformer in these experiments bear striking correspondence to our theory. We refer the reader to Appendix G for more results that validate this correspondence, including results for 3-sparse tasks.

7. Conclusion

In this paper, we take a fresh look at the in-context learning capability of transformers. We provide mechanisms that can implement sequence segmentation and hypothesis learning for a family of ICL tasks, and provide statistical guarantees showing that a fairly small number of examples indeed suffice to convey a target concept using ICL. More broadly, the ability of ICL to demonstrate information theoretically optimal learning in the types of tasks used both here and in prior works (Garg et al., 2022; von Oswald et al., 2022; Akyürek et al., 2022) is quite impressive. It would be interesting to understand if there are learning tasks where this optimality fails to hold in ICL, and determine the necessary scaling of model size as a function of problem size needed to achieve sample-optimal learning, when possible.

References

Kwangjun Ahn, Xiang Cheng, Hadi Daneshmand, and Suvrit Sra. Transformers learn to implement preconditioned gradient descent for in-context learning. *arXiv preprint arXiv:2306.00297*, 2023.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*, 2022.

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems*, 29, 2016.

Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. *arXiv preprint arXiv:2306.04637*, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Stephanie Chan, Adam Santoro, Andrew Lampinen, Jane Wang, Aaditya Singh, Pierre Richemond, James McClelland, and Felix Hill. Data distributional properties drive emergent in-context learning in transformers. *Advances in Neural Information Processing Systems*, 35:18878–18891, 2022.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Zhifang Sui, and Furu Wei. Why can gpt learn in-context? language models secretly perform gradient descent as meta optimizers. *arXiv preprint arXiv:2212.10559*, 2022.

Benjamin L Edelman, Surbhi Goel, Sham Kakade, and Cyril Zhang. Inductive biases and variable creation in self-attention mechanisms. In *International Conference on Machine Learning*, pages 5793–5831. PMLR, 2022.

N Elhage, N Nanda, C Olsson, T Henighan, N Joseph, B Mann, A Askell, Y Bai, A Chen, T Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. In *International Conference on Machine Learning*, pages 1920–1930. PMLR, 2019.

Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*, 2020.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022.

Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Hansen, Angelos Filos, Ethan Brooks, et al. In-context reinforcement learning with algorithm distillation. *arXiv preprint arXiv:2210.14215*, 2022.

Yingcong Li, M Emrullah Ildiz, Dimitris Papailiopoulos, and Samet Oymak. Transformers as algorithms: Generalization and implicit model selection in in-context learning. *arXiv preprint arXiv:2301.07067*, 2023.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

Ievgen Redko, Emilie Morvant, Amaury Habrard, Marc Sebban, and Younès Bennani. A survey on domain adaptation theory: learning bounds and theoretical guarantees. *arXiv preprint arXiv:2004.11829*, 2020.

Juergen Schmidhuber, Jieyu Zhao, and MA Wiering. Simple principles of metalearning. *Technical report IDSIA*, 69:1–23, 1996.

Seongjin Shin, Sang-Woo Lee, Hwijeen Ahn, Sungdong Kim, HyoungSeok Kim, Boseop Kim, Kyunghyun Cho, Gichang Lee, Woomyoung Park, Jung-Woo Ha, et al. On the effect of pretraining corpora on in-context learning by a large-scale language model. *arXiv preprint arXiv:2204.13509*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. *arXiv preprint arXiv:2212.07677*, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

Noam Wies, Yoav Levine, and Amnon Shashua. The learnability of in-context learning. *arXiv preprint arXiv:2303.07895*, 2023.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*, 2021.

Ruiqi Zhang, Spencer Frei, and Peter L Bartlett. Trained transformers learn linear models in-context. *arXiv preprint arXiv:2306.09927*, 2023.

Yufeng Zhang, Boyi Liu, Qi Cai, Lingxiao Wang, and Zhaoran Wang. An analysis of attention via the lens of exchangeability and latent variable models. *arXiv preprint arXiv:2212.14852*, 2022.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, pages 12697–12706. PMLR, 2021.

Contents

Appendix A. Related Work

There is a rich literature on fine-tuning a pre-trained model through various forms of meta-learning or bi-level optimization (Schmidhuber et al., 1996; Andrychowicz et al., 2016; Finn et al., 2017), which typically update all or some of the model's parameters to adapt it to a target task, with theoretical underpinnings either through direct analysis (Finn et al., 2019) or through the broader literature on transfer learning (see e.g. (Redko et al., 2020) and rerefences therein). Starting with the work of Brown et al. (2020), and with careful prompt design, this has become a predominant mechanism for adaptation in large language models (Zhao et al., 2021; Gao et al., 2020; Wei et al., 2022), including to complex scenarios such as learning reinforcement learning algorithms (Laskin et al., 2022). A consistently striking feature of these works is that it takes just a handful of examples from the target task to teach a new concept to the language model, once the inputs have been *carefully phrased and formatted*.

This has naturally motivated a number of studies on the mechanisms underlying ICL, as well as its sample complexity. A formal treatment of this topic is pioneered in Garg et al. (2022), who study ICL for linear regression problems, by pre-training a transformer architecture specifically for this task. That is, the model is trained on prompts consisting of (x, y) pairs from a linear regression instance, with the regression weights randomly chosen for each prompt. They showed that ICL can learn this task using a prompt of $O(d)$ examples when $x \in \mathbb{R}^d$. Subsequent work of von Oswald et al. (2022) proposes an explanation for ICL in this task by showing that self-attention can simulate gradient descent steps for linear regression, so that the model can effectively learn an optimizer during pre-training. Akyürek et al. (2022) further extend this by suggesting that given enough parameters, the model can compute a full linear regression solution. Both works present some empirical evidence as well suggesting that these operations correspond to the final outputs and some intermediate statistics of the transformer, when trained for this task. Dai et al. (2022) study the relationship between linear attention and gradient descent, and Li et al. (2023) study transformers as producing general purpose learning algorithms. Xie et al. (2021) and Zhang et al. (2022) cast ICL as posterior inference, with the former studying a mixture of HMM models and the latter analyzing more general exchangeable sequences. Wies et al. (2023) give PAC guarantees for the sample complexity of ICL, when pre-trained on a mixture of downstream tasks. Olsson et al. (2022) and Elhage et al. (2021) view ICL as an algorithm which copies concepts previously seen in a context example and then does inference by recalling these concepts when a new prompt matching previous examples occurs. Elhage et al. (2021) explain this behavior formally for transformers with a single attention head and two layers and Olsson et al. (2022) conduct an empirical study on a wider variety of tasks for larger transformers.

Despite this growing literature, many aspects of the ICL capability remain unexplained so far. First, only Li et al. (2023), Wies et al. (2023) and Zhang et al. (2022) provide any kind of sample complexity guarantees. Of these, the pre-training distribution in Wies et al. (2023) is too specific as the downstream task mixture, while Li et al. (2023) depend on an measure of algorithmic stability that is hard to quantify apriori. Secondly, all the works with the exception of Xie et al. (2021) require that the prompt has already been properly parsed into input and output examples, so as to facilitate the explanation of learning in terms of familiar algorithms, and the explanation of Xie et al. (2021) relies on a particular mixture of HMMs model. Further, we note that none of these works take into consideration the specifics of the transformer architecture and how self-attention can implement the proposed learning mechanisms.

The concurrent work of Ahn et al. (2023) and Zhang et al. (2023) study both the pre-training process and how it can lead to transformers implementing ICL as gradient descent. Ahn et al. (2023) study the landscape of the optimization problem for pre-training single attention layer linear transformers over random linear regression instances, and show that at the global minimum of the objective, the transformer is able to implement a step of pre-conditioned gradient descent as ICL. Zhang et al. (2023) study the dynamics of gradient flow in the same setting as Ahn et al. (2023) and show that the dynamics converge to a global optimum. At the optimum, the authors argue that the transformer can achieve low prediction error through approximating the LLS solution in the limit of infinite in-context examples.

The concurrent work of Bai et al. (2023) also studies ICL as a form of gradient descent. The authors show how a transformer can approximately implement L steps of gradient descent, using $O(L)$ self-attention layers for the ridge regression objective, the lasso objective, generalized linear models, and 2-layer feed-forward neural networks. The authors are able to show the statistical and computational efficiency of the approximate GD procedures. The authors also argue that transformers are able to model-select between the objectives by using part of the ICL examples as a validation set. As such, this is the only work, outside of our own, that we are aware of, which is able to recover efficient sparse linear regression results. The gradient descent view coupled with model selection is certainly a plausible explanation of the ability of transformers to learn sparse regression problems efficiently. However, the constructions in Bai et al. (2023) do not rely on the established self-attention mechanism to enable ICL but rather tailor the GD construction to fit the self-attention mechanism. Further, the authors use a relu-based self-attention mechanism instead of the softmax self-attention used in practice. Compared to the GD view, we see our construction as one which naturally leverages properties of the softmax attention with predictions which are borne out in the empirical study.

The work of Edelman et al. (2022) has the most similar construction in terms of how self-attention layers can approximate or exactly represent s-sparse functions. This work has two main contributions. First, the authors show careful capacity based union bounds on the class of bounded norm transformer blocks, that is a self-attention layer together with an MLP layer. Next, the authors demonstrate how a transformer block can represent s-sparse symmetric Boolean functions of $f : \{0,1\}^N \to \mathbb{R}$. The capacity bound (Theorem 4.7) is a bound on the ℓ_∞-covering number of the class of transformers with depth L, which scales logarithmically with the embedding dimension, d, and the context length, N. The bound also includes the norms of K, Q, V, W_E and scales exponentially with the depth as is standard for capacity bounds for DNNs. The s-sparse representation results (Proposition 5.1), demonstrate how transformers with bounded $\|Q\|_F \leq \log(sN), \|K\|_F, \|V\|_F, \|W_E\| \leq O(s)$ can approximate s-sparse functions. Together with the capacity bound this result implies that a transformer can learn the s-sparse regression functions we study with sample complexity similar to that of our own Theorem 11. While our construction for representing the functions is similar to that of Edelman et al. (2022), we emphasize that the authors do not study ICL, and do not provide a learning mechanism which explains how ICL might work, which is a main contribution of our work. Notably, while they provide a credible theory for learning a single sparse concept, when given a dataset generated according to it, they do not show how transformers can efficiently learn and represent a learning algorithm for the entire class of sparse linear functions, which we study in this paper.

While we do not study the properties of the pre-training process and data distribution in the ICL capability, these factors have been found to be crucial in empirical investigations (Chan et al., 2022;

Shin et al., 2022), and expanding the theoretical model to address such phenomena is an important direction for future research.

Appendix B. Notation

We denote the input sequence as $z \in \mathcal{V}^N$ consisting of N tokens. Each sequence is assumed to contain up to n i.i.d. samples corresponding to a task, where sample i consists of an example $x_i \in \mathcal{V}^{m'}$ and label $y_i \in \mathcal{V}$, where $m' \leq m$ is the token length of example x_i. For majority of the learning tasks that we analyze, the label y_i consists of only 1 token, and x_i consists of up to m tokens. We use $x_{i,j} \in \mathcal{V}$ for $j \in [m]$ to denote the tokens of x_i. The ICL instance is encoded as we have already described, where each x_i and y_i is separated by $\texttt{<lsep>} \in \mathcal{V}_{\texttt{delims}}$, and $\texttt{<esep>} \in \mathcal{V}_{\texttt{delims}}$ is found between between each successive pair (x_i, y_i), where $\sigma = (\texttt{<lsep>}, \texttt{<esep>})$ are segmentation tokens. We will often refer to the "ground truth" segmentation token pair as $\sigma^\star$, where $\sigma \in \mathcal{V}_{\texttt{delims}} \times \mathcal{V}_{\texttt{delims}}$ is an arbitrary pair of delimiters. We also recall the notation $p_{\sigma^\star}(z) = \Pi_{i=1}^n p_0(x_i) p_0(y_i)$ for some underlying distribution p_0 on $\mathcal{V}^*$, where the segmentation is determined by σ^*, and we remind the reader that we can analogously define $p_\sigma(z)$ for any other delimiter, which segments z into another sequence of (x, y) pairs.

When it comes to constructing transformer architectures, we will generally follow the same notation laid out in Section 2.1, but with a few new symbols and additional conventions. When we write $\langle u, v \rangle_M$ for matrix M and vectors u, v we mean the inner product $\langle uM, v \rangle = \langle u, Mv \rangle$. For characters z, x, y, o, v, K, Q, V, which describe the algorithmic objects of the transformer architecture, a *numeric superscript*, as the 2 in $x_{i,j}^2$, shall be used to identify the *layer index* and should not be interpreted as an exponent.

We described the attention mechanism in Definition 1, where the operation performed at layer $\ell \in [L]$ and head $k \in [\kappa]$ is parameterized by the query, key, and value matrices Q_k^ℓ, K_k^ℓ and V_k^ℓ, respectively. The initial embeddings of the token sequence z is the matrix (equiv., list of vectors) $[z_1^1, \ldots, z_N^1] \in \mathbb{R}^{d \times N}$, which is the matrix $W_E X$ where $X \in \{0, 1\}^{|\mathcal{V}| \times N}$ is the one-hot encoding of the tokens z. In general, when a token variable is superscripted with 1, we mean the embedded token, i.e. after multiplying by W_E. So $x_{i,j}^1$ is the column of W_E corresponding to the token index of $x_{i,j}$.

The attention operation computes the matrix $A_k^\ell \in [0, 1]^{N \times N}$. Normally we index this matrix with token indices $i, j \in [N]$, but occasionally we will find it convenient to interpret A_k^ℓ is a function on pairs of embedded tokens z_i^ℓ, z_j^ℓ, meaning that overload notation by setting $A_k^\ell(z_i^\ell, z_j^\ell) := A_k^\ell(i, j)$. This is well defined as our token embeddings contain a positional encoding, and we note that this allows us to avoid determining the exact index of the embedded token $x_{i,j}$, which can be cumbersome to describe. Thus we have

$$A_k^\ell(z_i^\ell, z_j^\ell) = \frac{\exp\left(\left\langle z_i^\ell, z_j^\ell \right\rangle_{Q_k^\ell (K_k^\ell)^\top}\right)}{\sum_{j' \leq i} \exp\left(\left\langle z_i^\ell, z_{j'}^\ell \right\rangle_{Q_k^\ell (K_k^\ell)^\top}\right)}, \tag{5}$$

Finally, $v_{k,i}^\ell$ and $o_{k,i}^\ell$ refer to the "value vector" computed at position i for head k and layer ℓ, and the corresponding output vector:

$$v_{k,i}^\ell := V_k^\ell z_i^\ell \quad \text{and} \quad o_{k,i}^\ell := \sum_{j \leq i} A_k^\ell(z_i^\ell, z_j^\ell) v_{k,j}^\ell.$$

For each i we vertically concatenate all of the outputs $(o_{k,i}^\ell)_{k \in \kappa}$ to obtain a tall vector $o_i^\ell \in \mathbb{R}^{d_{\text{att}}\kappa}$, and the final output $z^{\ell+1} = [z_1^{\ell+1}, \ldots, z_N^{\ell+1}] \in \mathbb{R}^{d \times N}$ is defined for all $i \in [N]$ as

$$z_i^{\ell+1} := z_i^\ell + \mathsf{GeLU}(W_O^\ell o_i^\ell).$$

In our constructions in the remainder of this appendix, we make a number of simplifications for convenience. Let us state these here, and argue why these are acceptable without loss of generality.

1. We often assume that $\kappa = 1$ and we drop the reference to head k. Similarly, when ℓ is omitted it should be clear from context.

2. We often implicitly assume that either there is no "skip connection," thus $z_i^{\ell+1} = \mathsf{GeLU}(W_O^\ell o_i^\ell)$, and often go further and assume $z_i^{\ell+1} = o_i^\ell$, avoiding the GeLU transformation. While we did not use this feature in Definitions 1 and 2, in most transformer architectures there is an additional skip connection that makes this possible.

3. We occasionally refer to the dimension of the embedding d as being different between input and output, i.e. d_{in} and d_{out}. This is for notational ease, and we may do this by padding earlier or later embedding dimensions with 0's.

Appendix C. Proofs of the segmentation results

In this section we will use infinite weights in the query and key matrices to set some attention weights to 0. It is possible to instead use finite weights, which would scale only poly-logarithmically with the context length as we do in the construction for learning a consistent hypothesis in Section **??** We first give a proof of Theorem 8, and then give the details of the transformer construction from Theorem 3.

Proof Let the distribution $U_{\sigma^\star}^{(\sigma',i)} := p_{\sigma'}(u_i|\mathsf{prefix}_{i-1})$ where u_i is the ith chunk as parsed by the correct segmentation $\sigma^\star$. Let σ be the correct choice of delimiters (we drop the $\star$ superscript for ease of reading). Our goal will be to show that, if we construct the ICL sequence z by sampling i.i.d. $x_1, \ldots, x_n \sim \mathcal{D}$, computing $y_1, \ldots, y_n$ by applying $y_i = f(x_i)$, and assembling these example/label pairs into a sequence with σ, then the model estimate $p_\sigma(z)$ is very likely to be much larger than $p_{\sigma'}(z)$ for every alternative $\sigma' \neq \sigma$. What we analyze is the log ratio

$$\log \frac{p_\sigma(z)}{p_{\sigma'}(z)} = \log \frac{\prod_{i=1}^n U_\sigma^{(\sigma,i)}}{\prod_{i=1}^n U_\sigma^{(\sigma',i)}} = \sum_{i=1}^n \log \frac{U_\sigma^{(\sigma,i)}}{U_\sigma^{(\sigma',i)}},$$

which we aim to show is very likely to be positive. We do this by converting the above to a martingale sequence. Let $\mu_i := \mathbb{E}_{x_i \sim \mathcal{D}}\left[\log \frac{U_\sigma^{(\sigma,i)}}{U_\sigma^{(\sigma',i)}} \,|\, \mathsf{prefix}_{i-1}\right]$, and observe that $\mu_i > c$ according to Assumption 1. Now we have that the sequence $\xi_j := \sum_{i=1}^j \left(\log \frac{U_\sigma^{(\sigma,i)}}{U_\sigma^{(\sigma',i)}} - \mu_i\right)$ is a martingale.

We note that, since we have a lower bound ϵ on the model probabilities, we have that $\log \frac{U_\sigma^{(\sigma,i)}}{U_\sigma^{(\sigma',i)}}$ falls within the range $[\log(\nu), \log(1/\nu)]$. We can then apply Azuma's Inequality to see that

$$
\begin{aligned}
\mathbb{P}_{x_{1:n} \sim \mathcal{D}} \left(\sum_{i=1}^n \log \frac{U_\sigma^{(\sigma,i)}}{U_\sigma^{(\sigma',i)}} < 0 \right) &= \mathbb{P}\left(\xi_n < -\sum_{i=1}^n \mu_i \right) \\
&\leq \mathbb{P}\left(\xi_n < -nc \right) \\
&\leq \exp\left(\frac{-nc^2}{8 \log^2(1/\nu)} \right).
\end{aligned}
$$

Setting n as in the Theorem statement ensures that the right hand side above is smaller than $\frac{\delta}{|\mathcal{V}_{\texttt{delims}}|^2}$. Now if we take a union bound over all possible choices of $\sigma' \neq \sigma$ ensures that

$$
P(\exists \sigma' \neq \sigma : p_{\sigma'}(z) > p_\sigma(z)) < \delta
$$

and thus we are done. ∎

C.1. Segmenting example and label delimiters

We show how to identify example and label delimiters using one head per combination of an example delimiter δ_e and label delimiter δ_l in $\mathcal{V}_{\texttt{delims}} \times \mathcal{V}_{\texttt{delims}}$. To simplify notation and avoid subscripts, we first focus on one such head for a fixed delimiter pair (δ_e, δ_l). We assume that the input to the transformer consists of the vector $z_i^1 = \begin{pmatrix} \tilde{z}_i^1 \\ i \\ 1 - \mathbf{1}(z_i = \delta_e) \\ 1 - \mathbf{1}(z_i = \delta_l) \end{pmatrix}$, where $\tilde{z}_i^1 \in \mathbb{R}^d$ is the (pre-trained) encoding of z_i which we augment for convenience.

First transformer layer: The goal of the first layer is to take the input at index i and map it to an output $o_i^1 = \begin{pmatrix} z_i^1 \\ i \\ m_e(i) \\ 1 - \mathbf{1}(z_i = \delta_l) \end{pmatrix}$, where $m_e(i)$ is the largest index $j < i$ such that $z_j = \delta_e$. Let Q^1 and K^1 matrices in this head be such $Q^1(K^1)^\top$ is 0 on the first d coordinates and 0 on the last coordinate. The remaining 2×2 coordinates are specified as sending $\langle \theta, \theta' \rangle_{Q^1(K^1)^\top(d+1:d+2, d+1:d+2)} = \gamma(\theta'(1)\theta(2) - \theta(1)\theta'(2))$ for any $\theta, \theta' \in \mathbb{R}^2$ for $\gamma \to \infty$. That is the attention weights act as a selector of the $m_e(i)$ as

$$
\langle z_i^1, z_j^1 \rangle_{Q^1(K^1)^\top} = \gamma(j(1 - \mathbf{1}(z_i = \delta_e)) - i(1 - \mathbf{1}(z_j = \delta_e))).
$$

Consequently the soft attention weights act like hard attention and we get that $A^1(z_i, z_j) = \mathbf{1}(j = m_e(i))$. We further define the value tokens to be $v_j^1 = j$, so that $o_i^1 = m_e(i)$. Using the skip connection, we further augment the input to the second layer as $z_i^2 = \begin{pmatrix} z_i^1 \\ i \\ m_e(i) \\ 1 - \mathbf{1}(z_i = \delta_l) \end{pmatrix}$.

Second transformer layer: The second transformer layer is constructed in a similar way as the first layer, however, it's goal is to extract $m_l(i)$. We are going to assume that each example token can also be treated as a label token (e.g. by adding the indicator of example delimiter to the indicator of label delimiter in the previous layer), so that in the case that there are no label tokens between two example tokens we have $m_e(i) = m_l(i)$. Further, for any x_i it also holds that $m_e(i) = m_l(i)$ and for any y_i it will hold that $m_l(i) > m_e(i)$. This will allow us to distinguish between x_i tokens and y_i tokens, which is important for computing the necessary probabilities for the proof of Theorem 8. We also extend the input z_i^3 to contain the following

$$z_i^3 = \begin{pmatrix} z_i^1 \\ i \\ m_e(i) \\ m_l(i) \\ m_l(i)m_e(i) \\ m_l^2(i) \\ m_e^2(i) \\ m_l^2(i)m_e(i) \\ m_e^2(i)m_l(i) \\ 0 \end{pmatrix}.$$

This is easily accomplished by the MLP at in the second layer, following the attention.

Third attention layer: In the third attention layer we are going to check for consistency of the positions on all separator tokens. Suppose that we have tokens z_i^3 and $z_{i'}^3$ s.t. $i \leq i'$. Then the following must be satisfied:

$$m_e(i) = m_e(i') \implies m_l(i) = m_l(i') \text{ or } m_l(i) = m_e(i), \tag{6}$$

that is there can not be more than one label token between two example tokens. To check this consistency we use an attention head for $m_l(i) \leq m_l(i')$. The attention between z_i, z_i' is computed as

$$\langle z_{i'}^3, z_i^3 \rangle_{Q^3(K^3)^\top} = \gamma(m_l(i') - m_l(i))\left(m_e(i) - m_e(i') + \frac{1}{2}\right)(m_l(i) - m_e(i))$$
$$+ \frac{\gamma}{n}(i - i')$$

for $\gamma \to \infty$, where n is the max sequence length. Note that by construction it holds that $m_e(i') \geq m_e(i)$ so that this inner product is only ∞ if the following hold together $m_l(i') > m_l(i)$, $m_l(i) > m_e(i)$ and $m_e(i) = m_e(i')$. $m_e(i) = m_e(i')$ implies that i and i' are part of the same example, $m_l(i) > m_e(i)$ implies that i is part of the answer sequence for that example and $m_l(i') > m_l(i)$ implies that there is `<lsep>` between token $z_{i'}$ and token z_i. The value vectors $v_i^3 \in \mathbb{R}^{d+9}$ are set as $v_i^3 = ie_{d+9}$. The resulting output of the attention layer is now $o_{i'}^3 = i'e_{d+9}$ iff the condition in Equation 6 are met, otherwise $o_{i'}^3 = ie_{d+9}$ for some i where the condition is violated. Next, we

describe how the MLP acts on $o_{i'}^3 + z_{i'}^3$ (obtained using skip connection) as follows

$$
o_{i'}^3 + z_{i'}^3 = \begin{pmatrix} z_{i'}^1 \\ i' \\ m_e(i') \\ m_l(i') \\ m_l(i')m_e(i') \\ (m_l(i'))^2 \\ (m_i(i'))^2 \\ (m_l(i'))^2 m_e(i') \\ (m_e(i'))^2 m_l(i') \\ \iota \end{pmatrix} \rightarrow \begin{pmatrix} z_{i'}^1 \\ i' \\ \max(m_e(i'), m_l(i')) \\ \mathbf{1}(\iota - i' \geq 0) \end{pmatrix} =: z_{i'}^4
$$

Fourth transformer layer: The input to the third layer effectively gives a mapping from the current token to the proposed start of an example and most recent label under the delimiters being considered. Since our objective (2) evaluates candidate delimiters in terms of the probabilities of the segmented sequences under a pre-trained distribution p_0, we need to map the tokens z_i^4 to the appropriate conditional probabilities related to the objective. In fact, we assume that the pre-training process provides us with a transformer which can do this mapping, as we state formally below.

Assumption 3 (Conditional probability transformer) *There exists a transformer which takes as input a token sequence $\zeta_i, \ldots, \zeta_T$, with each token $\zeta_i = \left(z_i^1, i, j, \delta_{err}\right)$ for $z^1 \in \mathbb{R}^d, i, j \in [T], \delta_{err}, \delta_{delim} \in \{0, 1\}$. It produces, $\ln p_i$, for each ζ_i s.t. $\delta_{err} = 0$, where*

$$
\ln p_i = \begin{cases} \ln p_0(z_i | z_{i-1}, \ldots, z_{j+1}) \text{ if } j < i - 1 \\ \ln p_0(z_i) \text{ if } j = i - 1 \\ 0 \text{ if } j = i. \end{cases}
$$

Further if $\delta_{err} = 1$ it returns $\ln \nu$.

We note that since the basic language models are trained to do next word prediction through log likelihood maximization, this is a very reasonable abstraction to assume from pre-training. As a result, we assume that the fourth layer produces $z_i^5 = \ln p_i$.

Assumption 3 allows us to compute conditional probabilities of sequences according to their segmentation in the following way. Consider the sequence

<begin>z_1, z_2, <lsep>, z_3, z_4<esep>, z_5, <lsep><end>.

What we feed to the transformer from Assumption 3 is

$$
\begin{pmatrix} z_1^1 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} z_2^1 \\ 2 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \text{<lsep>} \\ 3 \\ 3 \\ 0 \end{pmatrix}, \begin{pmatrix} z_3^1 \\ 4 \\ 3 \\ 0 \end{pmatrix}, \begin{pmatrix} \text{<esep>} \\ 5 \\ 5 \\ 0 \end{pmatrix}, \begin{pmatrix} z_5^1 \\ 6 \\ 5 \\ 0 \end{pmatrix}, \begin{pmatrix} \text{<lsep>} \\ 7 \\ 7 \\ 0 \end{pmatrix}.
$$

The transformer, respectively, computes

$$
p_1 = p_0(z_1), p_2 = p_0(z_2 | z_1), p_3 = 1, p_4 = p_0(z_3), p_5 = 1, p_6 = p_0(z_5), p_7 = 1.
$$

23

Fifth transformer layer: Note that so far we have acquired the conditional probabilities of individual tokens, when conditioned on a prefix. Further, these conditional probabilities have the following properties. If i is a delimiter then $p_i = 1$. If i is such that $i - 1$ is a delimiter then $p_i = p_0(z_i)$, that is the marginal of z_i is returned and finally $p_i = \nu$ for some small $\nu > 0$ if there is some inconsistency in the token segmentation. Note that this ensures that inconsistent token segmentations will have small probability. The fifth transformer layer just assigns uniform attention with the 0 matrix for QK^T and use $v_j = \begin{pmatrix} \ln p_j \\ 1 \end{pmatrix}$. This results in an output $\sum_{j=1}^{i}(\ln p_j)/i$ at token i and $z_i^6 = \begin{pmatrix} \sum_{j=1}^{i}(\ln p_j)/i \\ i \end{pmatrix}$. Finally, we use two MLP layers to send $z_i^6 \to z_i^6(1) \times z_i^6(2) = \sum_{j=1}^{i}(\ln p_j)$.

Lemma 12 *For any token pair σ the output at token k of the transformer at attention head associated with σ satisfies the factorization in Equation 4.*

The above lemma is a direct consequence of the construction of the transformer. We have already seen how the fourth layer acts on the sequence

$$\texttt{<begin>} z_1, z_2, \texttt{<lsep>}, z_3, z_4 \texttt{<esep>}, z_5, \texttt{<lsep><end>}.$$

The final layer of the transformer will now output for token 7 the sum

$$\sum_{i=1}^{7} \ln p_i = \ln \Pi_{i=1}^{7} p_i = \ln \left(p_0(z_1) \times p_0(z_2|z_1) \times p_0(z_3) \times p_0(z_4|z_3) \times p_0(z_5) \right)$$
$$= \ln(p_0(z_1, z_2) p_0(z_3, z_4) p_0(z_5)),$$

which is precisely the factorization in Equation 4 which is also used in the proof of Theorem 8.

Final layer to select across delimiters: Finally, the network implements the objective (2) for a particular delimiter in one head. By using the MLP to implement maximization across heads from the concatenated output values, we can identify the optimal delimiter.

Appendix D. Learning the the 1-sparse tokenized regression task

Here we give the construction of a transformer mechanism and sample complexity for the 1-sparse tokenized regression task, to build intuition for general the s-sparse case. We start with the mechanism before giving the sample complexity result.

D.1. Transformer mechanism for 1-sparse tokenized regression

Recall that the 1-sparse tokenized regression task is defined by a vector in $x \in \mathbb{R}^m$ and the hypothesis class $\mathcal{F}$ consists of all basis vectors $\{e_i\}_{i=1}^{m}$ in $\mathbb{R}^m$. Each instance of the task is defined by fixing a vector $e_f \in \{e_i\}_{i=1}^{m}$. The labels for the task are $y_i = \langle x_i, e_f^\star \rangle$ for some unknown $f^\star \in \mathcal{F}$. The i-th element of the sequence given to the transformer for the task is $(x_{i,1}, \ldots, x_{i,m}, y_i)$, where $x_{i,j}$ denotes the j-th coordinate of the vector x_i. We now describe how ICL can learn the above task, by using 1 head per layer and 4 attention layers.

We begin by stating a useful lemma which will allow us to set attention weights between any two tokens z_i, z_j to 0 by only using positional embedding dependent transformations.

Lemma 13 *For any given token embeddings $\{\bar{z}_\alpha\}_{\alpha \in [mn]} \in \mathbb{R}^d$, there exist embeddings $\{z_\alpha^\ell\}_{\alpha \in [mn]} \in \mathbb{R}^{(d+1)mn}$ which only depend on the positions $i \in [mn]$ such that for any two tokens $z_\alpha^\ell, z_\beta^\ell$ we have*
$$\langle z_\alpha^\ell, z_\beta^\ell \rangle_{Q_k^\ell (K_k^\ell)^\top} = c_{\alpha,\beta}^1 + c_{\alpha,\beta}^2 \langle \bar{z}_\alpha, \bar{z}_\beta \rangle \in \mathbb{R}, \forall \alpha, \beta \in [nm].$$

Proof The first mn coordinates of z_α^ℓ are set to
$$z_\alpha^\ell(t) = \mathbf{1}(\alpha = t), \ t \leq mn.$$

We partition the remaining dmn coordinates into mn blocks, each consisting of d coordinates. The α-th block equals $\bar{z}_\alpha$ and the remaining blocks are equal to 0, that is:
$$z_\alpha^\ell(tmn + 1 : tmn) = \mathbf{1}(\alpha = t)\bar{z}_\alpha, \ d - 1 \geq t \geq 1.$$

We now describe $Q_k^\ell \in \mathbb{R}^{(d+1)(mn) \times (d+1)(mn)}$. The first mn columns act on z_i^ℓ by sending it to the vector with β-th coordinate $c_{\alpha,\beta}^1$. This is done by setting $Q_k^\ell(\alpha, \beta) = c_{\alpha,\beta}^1$, where $\alpha, \beta \leq mn$ and $Q_k^\ell(\alpha, \beta) = 0$ if $\alpha \leq mn < \beta$. The remaining dmn columns are partitioned into mn blocks of size d, similarly to how the embedding of z_α^ℓ is handled. The d entries in the α-th row block and the β-th column block are constant and equal $c_{\alpha,\beta}^2$, that is
$$Q_k^\ell(s, t) = c_{\alpha,t}^2, \ s \in [\alpha mn + 1, (\alpha + 1)mn], \ t \in [\beta mn + 1, (\beta + 1)mn].$$

We now have that the vector $(z_\alpha^\ell)^\top Q_k^\ell \in \mathbb{R}^{(d+1)(mn)}$ has coordinates in its β-th block equal to $c_{\alpha,\beta}^2 \bar{z}_\alpha$, that is
$$(z_\alpha^\ell)^\top Q_k^\ell(\beta mn + 1 : \beta mn) = c_{\alpha,\beta}^2 \bar{z}_\alpha.$$

The key matrix $K_k^\ell \in \mathbb{R}^{(d+1)(mn) \times (d+1)(mn)}$ now only needs to select the appropriate entries from our query embedding. That is the first mn coordinates of $(K_k^\ell)^\top z_\beta^\ell(1 : mn) = e_\beta$ and on the remaining dmn coordinates the key matrix acts as the identity. ∎

Notation for embedded tokens. Following the notation in Lemma 13, we use the notation $\bar{x}_{i,j}$ and $\bar{y}_i$ to denote the input embeddings of $x_{i,j}$ and y_i respectively, such that these embeddings satisfy Assumption 2. This helps in clearly distinguishing the input embeddings from the representations $x_{i,j}^\ell$ and y_i^ℓ which are subsequently constructed at layer ℓ of the transformer architecture.

Recall that Assumption 2 gives approximate orthogonality of the embeddings in the first layer of the transformer. We note that the assumption is not hard to satisfy, e.g., one can use the "bucket" construction described right after the assumption in Section 5. Other possible embeddings include a Random Fourier Feature approximation to a Gaussian kernel with appropriate bandwidth. In fact, since the lemma only requires only approximate orthogonality, we can further assign each bucket in the bucket embedding to a random Gaussian vector in $O(\tau^2)$ dimensions and obtain the same result with dimension $\tilde{O}(\tau^2)$ with high probability. We now give the construction of a transformer for this learning task, assuming the availability of such an embedding satisfying Assumption 2 in R^{d_ϵ} for some d_ϵ.

Construction of the first transformer layer. We now specify the first transformer layer, by specifying the value vectors and the attention weights. We begin with the following result for the attention weights:

Lemma 14 *There exists a setting for the query and key matrices such that the following holds for any $\bar{\epsilon} > 0$.*

$$\left| A^1(y_i, x_{i,j}) - \frac{\exp(\langle \bar{y}_i, \bar{x}_{i,j} \rangle)}{\sum_{s=1}^m \exp(\langle \bar{y}_i, \bar{x}_{i,s} \rangle)} \right| \leq \bar{\epsilon} \tag{7}$$

$$A^1(y_i, x_{i',j}) \leq \frac{\bar{\epsilon}}{mn}, \quad A^1(y_i, y_{i'}) \leq \frac{\bar{\epsilon}}{mn} \; \forall i \; and$$

$$\left| A^1(x_{i,j}, x_{i,j}) - 1 \right| \leq \bar{\epsilon}, \quad A^1(x_{i,j}, x_{i',j'}) \leq \frac{\bar{\epsilon}}{mn} \forall (i,j) \neq (i',j').$$

Further, the entries of the query and key matrices do not exceed $O(\log((mn)/\bar{\epsilon}))$.

Proof Recall that WLOG we take the norm of any $\bar{z}_\alpha$ from Lemma 13 to be $\|\bar{z}_\alpha\| \leq 1, \alpha \in [mn]$. Further, from the construction in Lemma 13, because we add the index of the embedding as a standard basis vector, it holds that $\|z_\alpha^1\| \leq \sqrt{2}$. We use Lemma 13 with $c^2 \equiv 1$. We also set $c^1 \equiv \log(\bar{\epsilon}/(mn)) - 2$ for all pairs $(y_i, y_{i'})$, $(y_i, x_{i',j})$ and $(x_{i,j}, x_{i',j'})$ where $i' \neq i, j' \neq j$, and zero otherwise. Since $|\langle y_i^1, x_{i',j}^1 \rangle_{Q^1(K^1)^\top}| \leq \|y_i\|\|x_{i',j}\| \leq 2$ it follows that $\langle y_i^1, x_{i',j}^1 \rangle_{Q_k^1(K_k^1)^\top} \leq \log(\bar{\epsilon}/(mn))$. This implies $A^1(y_i, x_{i',j}) \leq \frac{\bar{\epsilon}}{mn}$ and a similar argument shows that $A^1(y_i, y_{i'}) \leq \frac{\bar{\epsilon}}{mn}$. Further, we have

$$\left| A^1(y_i, x_{i,j}) - \frac{\exp(\langle \bar{y}_i, \bar{x}_{i,j} \rangle)}{\sum_{s=1}^m \exp(\langle \bar{y}_i, \bar{x}_{i,s} \rangle)} \right|$$

$$= \frac{\exp(\langle \bar{y}_i, \bar{x}_{i,j} \rangle)}{\sum_{s=1}^m \exp(\langle \bar{y}_i, x_{i,s} \rangle)} - A^1(y_i, x_{i,j})$$

$$\leq \frac{\exp(\langle \bar{y}_i, \bar{x}_{i,j} \rangle)}{\sum_{s=1}^m \exp(\langle \bar{y}_i, \bar{x}_{i,s} \rangle)} - \frac{\exp(\langle \bar{y}_i, \bar{x}_{i,j} \rangle)}{\sum_{s=1}^m \exp(\langle \bar{y}_i, \bar{x}_{i,s} \rangle) + \bar{\epsilon}}$$

$$\leq \bar{\epsilon}.$$

The final set of inequalities for $A^1(x_{i,j}, x_{i',j'})$ and $A^1(x_{i,j}, x_{i,j})$ follow in the same way as above. ∎

Lemma 14 shows how we can have attention weights that only depend on inner products between tokens within an example, and each $x_{i,j}$ token only attends to itself, while the y_i attends to all the tokens in x_i with different weights. Next, the value transformation is chosen as the map which sends any $x_{i,j}^1 \in \mathbb{R}^{d_{\text{in}}}$ to $v_{i,j}^1 \in \mathbb{R}^{d_{\text{in}}+m}$, where the first d_{in} coordinates of $v_{i,j}^1$ are equal to $x_{i,j}^1$ and the remaining m coordinates are equal to the basis vector $e_j \in \mathbb{R}^m$. We will shortly see how ICL learns a hypothesis consistent with all examples but the main idea is that the consistent hypothesis will have the highest weight within the last m coordinates of the vector $\sum_{i=1}^n \sum_{j=1}^m A^1(y_i, x_{i,j}) v_{i,j}^1$, that is the sum of the output vectors associated with each of the answer tokens $y_i, i \in [n]$, after the attention layer has been applied.

Let $v^1_{i,m+1} = 0$. Now the outputs $o^1_{i,j} \in \mathbb{R}^{d_{\text{in}}+m}$ of self-attention at the first layer satisfy for any choice of norm $\|\cdot\|$

$$\|o^1_{i,j} - v^1_{i,j}\| \leq \bar{\epsilon}, \qquad \left\| o^1_{i,m+1} - \sum_{j=1}^{m} A^1(y_i, x_{i,j}) v^1_{i,j} \right\| \leq \bar{\epsilon}. \tag{8}$$

We now argue that output from the y_i tokens identifies all positions j such that $|x_{i,j} - y_i| \leq \epsilon$.

Lemma 15 (Generalization of Lemma 9) *Given an example i, let $\mathcal{J}_i = \{j \; : \; |x_{i,j} - y_i| \leq \epsilon\}$, and let $f^\star \in [m]$ be such that $y_i = x_{i,f^\star}$. Under Assumption 2, for any $m \geq 2$, the output of the first layer satisfies that for any $i \in [n]$:*

$$o^1_{i,m+1}(d_{\text{in}} + f^\star) \geq \max_{j \in [m] \setminus \mathcal{J}_i} o^1_{i,m+1}(d_{\text{in}} + j) + \frac{1}{4(m+1)},$$

if $\bar{\epsilon}$ in Equation 7 satisfies $\bar{\epsilon} \leq 0.07/(m+1)$. Further, it holds that

$$o^1_{i,m+1}(d_{\text{in}} + f^\star) \geq \max_{j \in \mathcal{J}_i} o^1_{i,m+1}(d_{\text{in}} + j) - \bar{\epsilon}.$$

Proof By Equation 8, we know that under Assumption 2, for any sample i and index $j \notin \mathcal{J}_i$, $A^1(y^1_i, x^1_{i,j}) \leq \bar{\epsilon} + e^{1/(2\tau)}/Z_i \leq \bar{\epsilon} + e^{1/2}/Z_i$, where Z_i is denominator in Equation 7. On the other hand, $A^1(y_i, y_i) = A^1(y_i, x_{i,f^\star}) \geq e/Z - \bar{\epsilon}$, under Assumption 2. Hence, we see that

$$o^1_{i,m+1}(d_{\text{in}} + f^\star) - \max_{j \in [m] \setminus \mathcal{J}_i} o^1_{i,m+1}(d_{\text{in}} + j) \geq \frac{e - e^{1/2}}{Z_i} - 2\bar{\epsilon} \geq \frac{0.39e}{Z_i} - 2\bar{\epsilon},$$

where the second inequality uses $m \geq 2$. Since $Z_i \leq e(m+1)$ under our definition of the attention weights, we further get

$$o^1_{i,m+1}(d_{\text{in}} + f^\star) - \max_{j \in [m] \setminus \mathcal{J}_i} o^1_{i,m+1}(d_{\text{in}} + j) \geq \frac{0.39}{m+1} - 2\bar{\epsilon} \geq \frac{1}{4(m+1)},$$

for any $\bar{\epsilon} \leq 0.07/(m+1)$.

For the second claim of the lemma we note Lemma 14 implies that

$$\begin{aligned}
o^1_{i,m+1}(d_{\text{in}} + f^\star) = A^1(y_i, x_{i,f^\star}) &\geq \frac{\exp(\langle \bar{y}_i, \bar{x}_{i,f^\star} \rangle)}{\sum_{s=1}^{m} \exp(\langle \bar{y}_i, \bar{x}_{i,s} \rangle)} - \bar{\epsilon} \\
&= \frac{e}{\sum_{s=1}^{m} \exp(\langle \bar{y}_i, \bar{x}_{i,s} \rangle)} - \bar{\epsilon}.
\end{aligned}$$

Further, for any $j \in \mathcal{J}_i$ we have

$$\begin{aligned}
o^1_{i,m+1}(d_{\text{in}} + j) = A^1(y_i, x_{i,j}) &\leq \frac{\exp(\langle \bar{y}_i, \bar{x}_{i,j} \rangle)}{\sum_{s=1}^{m} \exp(\langle \bar{y}_i, \bar{x}_{i,s} \rangle)} \\
&\leq \frac{e}{\sum_{s=1}^{m} \exp(\langle \bar{y}_i, \bar{x}_{i,s} \rangle)}.
\end{aligned}$$

Combining the two inequalities finishes the proof. $\blacksquare$

We assume that the following MLP layer acts as the identity and $x^2_{i,j} = o^1_{i,j}, \forall j \in [m + 1]$, where we take $x^2_{i,m+1} \equiv y^2_i$. Notice that the embedding dimension changes from d_{in} in the first layer to $d_{\text{in}} + m$ for the second layer.

Second transformer layer. For inference we set the attention weights in the second attention head in the following way, which is permitted by Lemma 13:

$$|A^2(x_{i,j}^2, x_{i,j}^2) - 1| \le \bar{\epsilon}, A^2(x_{i,j}^2, y_{i'}^2) \le \frac{\bar{\epsilon}}{mn}, A^2(y_i^2, x_{i',j}^2) \le \frac{\bar{\epsilon}}{4(mn)^2} \forall i' \le i$$

$$|A^2(y_i^2, y_t^2) - 1/i| \le \frac{\bar{\epsilon}}{2mn}, \forall t \le i. \tag{9}$$

In words, the $x_{i,j}$ tokens only attend to themselves again, while y_i attends uniformly to all the previous labels, including itself. This is achieved by setting c^2 to be one for all $(x_{i,j}, x_{i,j})$ pairs and zero everywhere else. c^1 is set to zero for $(x_{i,j}, x_{i,j})$ pairs and (y_i, y_t) pairs with $t \le i$, and $\log(\bar{\epsilon}/mn) - 2$ otherwise. We set $v_{i,j}^2 = o_{i,j}^1$ to be the output of the first transformer layer. This construction yields the following lemma on the second layer outputs, which effectively aggregate across all examples in positions corresponding to the labels y_i.

Lemma 16 *The construction of the first transformer layer and the attention weights in Equation 9 imply*

$$\left\| o_{i,j}^2 - \begin{pmatrix} x_{i,j}^1 \\ e_j \end{pmatrix} \right\| \le 2\bar{\epsilon}, j \in [m],$$

$$\left\| o_{i,m+1}^2 - \frac{1}{i} \sum_{t=1}^{i} o_{t,m+1}^1 \right\| \le \bar{\epsilon}. \tag{10}$$

Proof For the first bound we note that

$$\|o_{i,j}^2 - v_{i,j}^2\| \le \bar{\epsilon} \quad \text{and} \quad \left\| o_{i,j}^1 - \begin{pmatrix} x_{i,j}^1 \\ e_j \end{pmatrix} \right\| \le \bar{\epsilon},$$

where the first inequality follows from Equation 9 and the second inequality follows from Equation 8. A triangle inequality now shows the first part of Equation 10. For the second part we have

$$o_{i,m+1}^2 = \sum_{t \le i, j \le m} A^2(y_i^2, x_{t,j}^2) v_{t,j}^2 + \sum_{t \le i} A^2(y_i^2, y_t^2) v_{t,m+1}^2$$

$$= \sum_{t \le i, j \le m} A^2(y_i^2, x_{t,j}^2) v_{t,j}^2 + \sum_{t \le i} A^2(y_i^2, y_t^2) o_{t,m+1}^1$$

This allows us to further conclude that

$$\left\| o_{i,m+1}^2 - \frac{1}{i} \sum_{t=1}^{i} o_{t,m+1}^1 \right\| \le \sum_{t \le i} \left| \frac{1}{i} - A^2(y_i^2, y_t^2) \right| \|o^{t,m+1}\|$$

$$+ \sum_{t \le i, j \le m} A^2(y_i^2, x_{t,j}^2) \|o_{t,j}^1\| \le \frac{i\sqrt{2\bar{\epsilon}}}{2mn} + \frac{\sqrt{2\bar{\epsilon}}}{4mn} \le \bar{\epsilon}$$

where in the last inequality we have used the fact that $\|x_{i,j}^1\| \le 1, \|y_i^1\| \le 1, \forall i, j$ and hence $\|o_{i,j}^1\| \le \sqrt{2}, \forall i, j$. ∎

We assume that the following MLP acts as the identity mapping on the first d coordinates of the value vectors and then sends the remaining m coordinates to the basis vector corresponding to

the index with highest value. That is, $\mathrm{MLP}^2 o^2_{i,m+1} = (v \; e_{f_i})$, with $v \in \mathbb{R}^{d_{\text{in}}}$ equal to the first d_{in} coordinates of $o^2_{i,m+1}$, and $f_i = \operatorname{argmax}_{j \in [m]} o^2_{i,m+1}(j)$. Ties are broken arbitrarily but consistently.

In particular, together with the construction of the attention weights at the previous layer we have the following

Lemma 17 *For $\bar{\epsilon} \leq \frac{e}{16(m+1)n}$ it holds that*

$$\left\| x^3_{i,j} - \begin{pmatrix} x^1_{i,j} \\ e_j \end{pmatrix} \right\| \leq 2\bar{\epsilon}, j \in [m], \quad y^3_{1,i}(d+1:d+m) = e_{f_i} \in \mathbb{R}^m, \quad f_i \in \mathcal{J}_t, \forall t \leq i. \quad (11)$$

Proof First, $\left\| x^3_{i,j} - \begin{pmatrix} x^1_{i,j} \\ e_j \end{pmatrix} \right\| \leq 2\bar{\epsilon}, j \in [m]$ follows from Equation 10. To show $f_i \in \mathcal{J}_t, \forall t \leq i$ we first argue that there is a sufficient gap between $\frac{1}{i} \sum_{t=1}^{i} o^1_{t,m+1}(f_i)$ and $\frac{1}{i} \sum_{t=1}^{i} o^1_{t,m+1}(j)$ for any j for which there exists $t \leq i$ s.t. $j \notin \mathcal{J}_t$. Assume that there exists $t \leq i$ s.t. $f_i \notin \mathcal{J}_t$. Due to the first part of Lemma 15, it holds that $o^1_{t,m+1}(d_{in} + f^\star) \geq o^1_{t,m+1}(d_{in} + f_i) + \frac{e}{4(m+1)}$. Further for all $t \leq i$ it holds that $o^1_{t,m+1}(d_{in} + f^\star) \geq o^1_{t,m+1}(d_{in} + f_i) - \bar{\epsilon}$ due to the second part of Lemma 15. This implies that for our choice of $\bar{\epsilon}$

$$\sum_{t \leq i} o^1_{t,m+1}(d_{in} + f^\star) > \sum_{t \leq i} o^1_{t,m+1}(d_{in} + f_i) + \frac{e}{4(m+1)} - i\bar{\epsilon}$$

$$\geq \sum_{t \leq i} o^1_{t,m+1}(d_{in} + f_i) + \frac{e}{8(m+1)},$$

for any $m \geq 2, n \geq 2$. Since $\| \cdot \|_\infty \leq \| \cdot \|$ in Equation 10, we further get

$$o^2_{i,m+1}(d_{in} + f^\star) \geq \frac{1}{i} \sum_{t \leq i} o^1_{t,m+1}(d_{in} + f^\star) - \bar{\epsilon}$$

$$\geq \frac{1}{i} \sum_{t \leq i} o^1_{t,m+1}(d_{in} + f_i) + \frac{e}{i8(m+1)} - \bar{\epsilon}$$

$$\geq o^2_{i,m+1}(d_{in} + f_i) + \frac{e}{i8(m+1)} - 2\bar{\epsilon}$$

$$\geq o^2_{i,m+1}(d_{in} + f_i),$$

where the last inequality follows form the setting of $\bar{\epsilon}$. This contradicts the maximality of f_i for example i, which completes the proof. ∎

In the next section, we show that this aggregation across examples followed by the maximization selects some coordinate such that $|x_{i,j} - y_i| \leq \epsilon$ for all examples i in the context, and that any such hypothesis has a small prediction error on future examples in this task. That is, the first two layers identify an approximately correct hypothesis for the task.

Inference with learned hypothesis. Finally we explain how to apply the returned hypothesis f_i to the next example. This will also describe how to do inference with the hypothesis f_n on the $n + 1$-st example. The attention pattern required here is a bit different in that each $x_{i,j}$ only attends

to the previous label y_i and itself. $Q_1^3(K_1^3)^\top$ only acts on coordinates $[d+1 : d+m]$ as the identity and sends everything else to 0. In particular, the unnormalized attention weights are

$$\exp(\langle x_{i+1,j}^3, y_i^3 \rangle_{Q^3(K^3)^\top}) = \exp(\langle e_j, e_{f_i} \rangle) = \exp(\mathbf{1}(f_i = j))$$
$$\exp(\langle x_{i+1,j}^3, x_{i+1,j}^3 \rangle_{Q^3(K^3)^\top}) = \exp(\langle e_j, e_j \rangle) = e$$
$$\exp(\langle x_{i+1,j}^3, x_{i+1,j'}^3 \rangle_{Q^3(K^3)^\top}) \leq \frac{\bar{\epsilon}}{mn}$$

This results in attention values

$$\left| A^3(x_{i+1,j}^3, y_i^3) - \mathbf{1}(f_i = j)\frac{1}{2} - (1 - \mathbf{1}(f_i = j))\frac{1}{1+e} \right| \leq \bar{\epsilon}, \qquad (12)$$

that is we approximately have

$$A^3(x_{i+1,j}^3, y_i^3) \approx \begin{cases} \frac{1}{2} & f_i = j \\ \frac{1}{1+e} < \frac{1}{2} & \text{otherwise} \end{cases}$$

The remaining attention outputs are not used and hence not specified here. The value vectors in $\mathbb{R}^2$ are set as

$$v_{i,j}^3 = 2 \begin{pmatrix} x_{i,j} \\ 0 \end{pmatrix}, v_{i,m+1}^3 = 2 \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Notice that this requires access to the raw input token, which can be done by either providing a skip connection from the inputs, or by carrying the input token as part of the embedding through all the layers at the cost of one extra embedding dimension. As a result, for the index f_i selected at the end of example i, we have that

$$\left\| o_{i+1,f_i}^3 - (x_{i+1,f_i}, 1) \right\| \leq \bar{\epsilon},$$

and for any other $j \in [m]$ we have $o_{i+1,j}^3(1) \leq \frac{1}{1+e} + \bar{\epsilon}$ in the second coordinate. The next MLP layer thresholds the second coordinate of $o_{i+1,j}^3$ so that

$$x_{i+1,j}^4 = \left(x_{i+1,j}, \mathbf{1}(j \text{ is consistent with } f^\star \text{ up to example } i) \right).$$

The above is possible as long as $2\bar{\epsilon} \leq \frac{1}{2} - \frac{1}{1+e}$. The final attention and MLP layers are used to copy the $x_{i+1,j}^4$ token with 1 in the second coordinate to the m-th token of the $i + 1$-st example, so that the transformer outputs the prediction at the end of each example sequence. This can be done by using the `mov` function described in Section 3.1 of Akyürek et al. (2022).

A summary of this construction can be found in Figure 2.

D.2. Sample complexity

Let the target hypothesis be $f^\star$, that is we assume, $y_i = f^\star(x_i), \forall i \in [n]$. We are going to analyze the error of the hypothesis returned by ICL after m examples. From Lemma 15, we know that the true hypothesis $f^\star$ has a large value in the output of the first layer, in the coordinate $d + f^\star$, at each example i. Suppose our construction identifies the hypothesis to make the prediction with,

after seeing i examples. Then if f_i makes an incorrect prediction (that is $|y_{i'} - f_i(x_{i'})| \geq \epsilon$ for some $i' \leq i$) on even one of these i examples, the output in coordinate $d + f_i$ is guaranteed to be smaller than in $d + f^\star$ by Lemma 15. Consequently, the hypothesis f_n returned after n examples is guaranteed to have an error at most ϵ on each of the n examples in context. We now show that this implies a risk bound on the hypothesis f_n.

Lemma 18 *Let $p_n = \mathbb{P}(|f_n(x) - f^\star(x)| \leq \epsilon)$ be the probability of the returned hypothesis deviating from $f^\star$ by more than ϵ on any example. Then with probability $1 - \delta$ it holds that $p_n \geq 1 - \frac{20 \log(m/\delta)}{3n}$.*

Proof Fix any hypothesis f. Let X_i denote the Bernoulli random variable indicating the event that $|f(x_i) - f^\star(x_i)| \leq \epsilon$ and $p_f = \mathbb{P}(|f(x) - f^\star(x)| \leq \epsilon)$. We compute the probability that this hypothesis is potentially returned by the transformer which is equivalent to the event that $\sum_{i=1}^n X_i \geq n$.

$$\mathbb{P}(\sum_{i=1}^n X_i \geq n) = \mathbb{P}\left(\sum_{i=1}^n X_i \geq np_f + 2\sqrt{np_f(1 - p_f)\log(1/\delta)} + \frac{4}{3}\log(1/\delta) \right.$$

$$\left. + n(1 - p_f) - 2\sqrt{np_f(1 - p_f)\log(1/\delta)} - \frac{4}{3}\log(1/\delta) \right) \leq \delta,$$

as long as $n(1 - p_f) - 2\sqrt{np_f(1 - p_f)\log(1/\delta)} - \frac{4}{3}\log(1/\delta) \geq 0$. We note that Cauchy-Schwartz implies

$$n(1 - p_f) - 2\sqrt{np_f(1 - p_f)\log(1/\delta)} - \frac{4}{3}\log(1/\delta) \geq \frac{n(1 - p_f)}{2} - \frac{4}{3}\log(1/\delta) - 2p_f \log(1/\delta)$$

$$\geq \frac{n(1 - p_f)}{2} - \frac{10}{3}\log(1/\delta)$$

Finally, we note that $p_f \leq 1 - \frac{20 \log(1/\delta)}{3n}$ implies $\frac{n(1 - p_f)}{2} - \frac{10}{3n}\log(1/\delta) \geq 0$. Taking a union bound over all possible f and applying with $f = f_n$, so that $p_{f_n} = p_n$ completes the proof. ∎

Theorem 19 *For any $\epsilon > 0$ there exists an embedding of $x_{i,j}, y_i, \forall i \in [n], j \in [m]$ in $\mathbb{R}^{O(1/\epsilon)}$ such that for $n = \Omega(\log(m/\epsilon)/\epsilon)$ it holds that $\mathbb{E}[|f_n(x) - f^\star(x)|] \leq 2\epsilon$, where f_n is the hypothesis returned by ICL.*

Proof We use the embedding into $\lceil 1/\epsilon \rceil$ buckets, as mentioned in the previous section together with the construction of the transformer to satisfy the conditions of Lemma 18. Conditioning on the good event, A, in Lemma 18 implies that $\mathbb{P}(|f_n(x) - f^\star(x)| > \epsilon|A) \leq \epsilon$ and so under A, we have

$$\mathbb{E}\,|f_n(x) - f^\star(x)| \leq \epsilon p_n + 1 - p_n \leq 2\epsilon,$$

where the second inequality follows from our condition on n. ∎

Appendix E. s-sparse Tokenized Regression

In this section we study the general s-sparse case defined in Definition 5. Recall that the hypothesis class now consists of $f = (j_1, \ldots, j_s) \in [m]^s$, that is each hypothesis selects s out of the m coordinates of x. We begin by making the following simple observation under Assumption 2: if $j \in f^\star$ then for any i it holds that $\langle \bar{x}_{i,j}, \bar{y}_i \rangle \geq 3/4$, while if j is not part of a consistent policy then we have $\langle \bar{x}_{i,j}, \bar{y}_i \rangle \leq 1/4$, where we recall that $\bar{x}_{i,j}, \bar{y}_i$ are the input embeddings satisying Assumption 2 as defined in the previous section.

Lemma 20 *Under Assumption 2 with $\tau \geq 2s$ we have that for any $\mathcal{C} \subseteq f^\star$ and, $j \in f^\star \setminus \mathcal{C}$, we have $\langle \bar{x}_{i,j}, \bar{y}_i - \sum_{j' \in \mathcal{C}} \bar{x}_{i,j'} \rangle \geq \frac{3}{4}$, while if $|x_{i,j} - x_{i,j'}| \geq \epsilon$ for all $j' \in f^\star$ then we have $\langle \bar{x}_{i,j}, \bar{y}_i - \sum_{j' \in \mathcal{C}} \bar{x}_{i,j'} \rangle \leq \frac{1}{4}$.*

Proof Since $\bar{y}_i = \sum_{j \in f^\star} \bar{x}_{i,j}$, for any $j \in f^\star \setminus \mathcal{C}$, we have

$$\left\langle \bar{x}_{i,j}, \sum_{j \in f^\star \setminus \mathcal{C}} \bar{x}_{i,j} \right\rangle \geq \langle \bar{x}_{i,j}, \bar{x}_{i,j} \rangle - \frac{s}{2\tau} \geq \frac{3}{4},$$

where the first inequality follows from Assumption 2, since any token which does not have an inner product of 1 with $x_{i,j}$ has the inner product at least $-1/(2\tau)$. The last equality follows from the precondition $\tau \geq 2s$ in the lemma. On the other hand, for any token j which is not ϵ-close to any token in $f^\star$, the inner product is at most $s/(2\tau)$ by a similar argument, which completes the proof. ∎

We proceed to give a construction which will use $O(m)$ layers with one head per layer. The idea behind the construction is to learn each coordinate of a single consistent hypothesis in $\mathcal{F}$. We note that it is not possible to directly take the approach in the 1-sparse task to learn each coordinate in $f^\star$ independently now, unless there is a unique consistent hypothesis with high probability. As described in Section 5, we follow an iterative deflation approach to avoid this issue.

Let $\mathcal{C}_i^\ell$ denote the subset of coordinates of a consistent hypothesis identified up to layer ℓ and example i. We embed $y_i^\ell, x_{i,j}^\ell \in \mathbb{R}^{(d+m+1)mn}$ in the following way. First we take the embeddings $\bar{x}_{i,j}, \bar{y}_i \in \mathbb{R}^d$ and extend them to $\bar{x}_{i,j}, \bar{y}_i \in \mathbb{R}^{d+m}$ so that the first d coordinates again equal the original embeddings $\bar{x}_{i,j}, \bar{y}_i \in \mathbb{R}^d$ from Assumption 2 and the remaining coordinates are as follows, if coordinate $j \in \mathcal{C}_i^\ell$, then $\bar{y}_i(d + j) = 0$, otherwise $\bar{y}_i(d + j) = -C_{\bar{\epsilon}}$, where $C_{\bar{\epsilon}}$ is a constant depending only poly-logarithmically on an accuracy parameter $\bar{\epsilon}$. Finally $\bar{x}_{i,j}(d + 1 : m) = e_j$. Now we take these $d + m$ dimensional vectors and apply the same transformation in the proof of Lemma 13 to them, which results in a $(d + m + 1)mn$ dimensional embedding.

The query and key matrices are set so that $\langle y_i^\ell, x_{i,j}^\ell \rangle_{Q^\ell(K^\ell)^\top} = \langle \bar{y}_i(d + 1 : m), \bar{x}_{i,j}(d + 1 : m) \rangle$ and $\langle y_i^\ell, y_i^\ell \rangle_{Q^\ell(K^\ell)^\top} = 0$. The value vectors are set to $v_{i,j}^\ell = \begin{pmatrix} -\bar{x}_{i,j}(1 : d) \\ -1 \end{pmatrix}$ for $j \leq m$ and $v_{i,m+1}^\ell = \begin{pmatrix} \bar{y}_i(1 : d) \\ 1 \end{pmatrix}$.

Notational remarks. For the remainder of this section, we use the notation $\bar{x}_{i,j}, \bar{y}_i$ to refer to the $d + m$ dimensional vectors described above, as opposed to just the embedding provided by Assumption 2 for convenience. We also define the embedding map $\mathrm{emb}(v, i, j)$ which takes a

vector $v \in \mathbb{R}^{d+m}$ and applies the transformation of Lemma 13 for the block $\alpha = (i,j)$ to this vector. That is, it pads the vector with an mn dimensional indicator of the position (i,j) and also places the vector v in the (i,j)-th block.

Lemma 21 *There exists a setting for the query, key and value matrices at layer ℓ so that given the embeddings y_i^ℓ and $x_{i,j}^\ell$, $i \in [n], j \in [m]$ it holds that*

$$\left\| o_{i,m+1}^\ell - \begin{pmatrix} \frac{1}{|\mathcal{C}_i^\ell|+1} \left(\bar{y}_i(1:d) - \sum_{j \in \mathcal{C}_i^\ell} \bar{x}_{i,j}(1:d) \right) \\ \frac{1}{|\mathcal{C}_i^\ell|+1} \end{pmatrix} \right\|_\infty \leq \frac{\bar{\epsilon}}{d}, \quad o_{i,m+1}^\ell \in \mathbb{R}^{d+1}$$

$$\left\| o_{i,j}^\ell - \begin{pmatrix} -\bar{x}_{i,j}(1:d) \\ -1 \end{pmatrix} \right\|_\infty \leq \frac{\bar{\epsilon}}{d}, \quad o_{i,j}^\ell \in \mathbb{R}^{d+1}.$$

Proof To show the claim of the lemma we only need to compute the attention weights from the ℓ-th attention layer. First, using Lemma 13 we can set $A^\ell(x_{i,j}^\ell, x_{i,j}^\ell) \geq 1 - \frac{\bar{\epsilon}}{d}$, which, together with the value vector choice, shows the second inequality of the lemma. If $j \notin \mathcal{C}_i^\ell$ then the construction implies

$$\langle y_i^\ell, x_{i,j}^\ell \rangle_{Q^\ell(K^\ell)^\top} = \langle \bar{y}_i, \bar{x}_{i,j} \rangle \leq s - C_{\bar{\epsilon}},$$

where for the inequality we use $\langle \bar{y}_i(1:d), \bar{x}_{i,j}(1:d) \rangle \leq s$ Further, using Lemma 13, we can also set

$$\langle y_i^\ell, y_i^\ell \rangle_{Q^\ell(K^\ell)^\top} = 0.$$

Finally, we want to ensure (nearly) uniform weights for all consistent examples in $\mathcal{C}_i^\ell$ and so we enforce $\langle y_i^\ell, x_{i,j}^\ell \rangle_{Q^\ell(K^\ell)^\top} = 0$ by using Lemma 13. We set $C_{\bar{\epsilon}} = s + \log(dmn/\bar{\epsilon} + s)$ and this implies $A^\ell(y_i^\ell, x_{i,j}^\ell) \leq \frac{\bar{\epsilon}}{dmn}$. Thus for any $j \in \mathcal{C}_i^\ell$ we have

$$\left| A^\ell(y_i^\ell, x_{i,j}^\ell) - \frac{1}{|\mathcal{C}_i^\ell|+1} \right| = \left| A^\ell(y_i^\ell, x_{i,j}^\ell) - \frac{\exp(0)}{\sum_{j \in \mathcal{C}_i^\ell} \exp(0) + \exp(\langle y_i^\ell, y_i^\ell \rangle_{Q^\ell(K^\ell)^\top})} \right| \leq \frac{\bar{\epsilon}}{d},$$

which completes the claim of the lemma. ■

Lemma 21 shows that we can "deflate" y_i by subtracting all consistent coordinates which have been identified so far. Next, we are going to use the construction for the 1-sparse task on i-th example $y_i^{\ell+1} \approx y_i^1 - \sum_{j \in \mathcal{C}_i^\ell} x_{i,j}^1$ and $x_{i,j}^{\ell+1} = x_{i,j}^1$. We make a slight modification to the outputs of the ℓ-th attention layer by setting

$$o_{i,m+1}^\ell = \begin{pmatrix} o_{i,m+1}^\ell \\ \bar{y}_i(d+1:m) \end{pmatrix}$$

$$o_{i,j}^\ell = \begin{pmatrix} o_{i,m+1}^\ell \\ \bar{x}_{i,j}(d+1:m) \end{pmatrix}.$$

This can be achieved using the skip-connection and appropriate padding of $o_{i,m+1}$. However, to simplify the argument we avoid describing this operation. We assume that the MLP layer after the

ℓ-th attention layer acts on $o_{i,j}^\ell$ in the following way, it sends $o_{i,j}^\ell \to \frac{1}{o_{i,j}^\ell(d+1)}o_{i,j}^\ell$. Further, it acts on the coordinates corresponding to $\bar{y}_i(d+1:m)$ by sending $-C_{\bar{\epsilon}}$ to 0 and 0 to $-C_{\bar{\epsilon}}$. This can be done by first adding 1 (as seen from the proof of Lemma 21 $C_{\bar{\epsilon}} > 1$) to all coordinates, then using a relu to clip all remaining $-C_{\bar{\epsilon}}$ to 0, and finally multiply the remaining positive coordinates by $-C_{\bar{\epsilon}}$ again. This operation is needed to take the complement of $\mathcal{C}_i^\ell$ so that all consistent coordinates which have already been added to $\mathcal{C}_i^\ell$ can be removed from consideration. We note that both these operations actually need a 2-layer MLP, however, for simplicity we assume that these are implementable by the MLP layer following the attention layer. We now describe the inputs $x_{i,j}^{\ell+1}$ and $y_i^{\ell+1}$ to the $\ell+1$-st transformer layer:

$$x_{i,j}^{\ell+1} = \mathrm{emb}(\bar{x}_{i,j}), \quad \text{and} \quad y_i^{\ell+1} = \mathrm{emb}(w_i), \quad \text{where}$$

$$\left\| w_i(1:d) - \bar{y}_i(1:d) + \sum_{j \in \mathcal{C}_i^\ell} \bar{x}_{i,j}(1:d) \right\|_\infty \leq \frac{\bar{\epsilon}}{d} \tag{13}$$

$$w_i(j) = -C_{\bar{\epsilon}}\mathbf{1}(j \in \mathcal{C}_i^\ell), \quad d+1 \leq j \leq d+m.$$

Next, we choose $Q^{\ell+1}, K^{\ell+1}$ so that we have $\langle x_{i,j}^{\ell+1}, y_i^{\ell+1} \rangle_{Q^{\ell+1}(K^{\ell+1})^\top} = \langle \bar{x}_{i,j}, \bar{y}_i \rangle$ This implies for all $j \in \mathcal{C}_i^\ell$ $\langle x_{i,j}^{\ell+1}, y_i^{\ell+1} \rangle_{Q^{\ell+1}(K^{\ell+1})^\top} \leq s - C_{\bar{\epsilon}}$ and otherwise $|\langle x_{i,j}^{\ell+1}, y_i^{\ell+1} \rangle_{Q^{\ell+1}(K^{\ell+1})^\top} - \langle \bar{x}_{i,j}, \bar{y}_i - \sum_{j \in \mathcal{C}_i^\ell} \bar{x}_{i,j} \rangle| \leq \bar{\epsilon}$. Finding a consistent coordinate is now equivalent to recovering a consistent hypothesis for the 1-sparse task, which we know how to do using exactly two attention layers as described previously.

Lemma 22 *Applying the first two layers of the 1-sparse task from Section D.1 to $x_{i,j}^{\ell+1}, j \in [m], y_i^{\ell+1}$ as defined in Equation 13 yields:*

$$\left\| o_{i,j}^{\ell+3} - \begin{pmatrix} x_{i,j}^{\ell+1} \\ e_j \end{pmatrix} \right\|_\infty \leq \frac{2\bar{\epsilon}}{d}$$

$$\left\| o_{i,m+1}^{\ell+3} - \frac{1}{i}\sum_{t=1}^{i}\sum_{j \in [m]\setminus\mathcal{C}_t^\ell} A^{\ell+1}(y_t^{\ell+1}, x_{t,j}^{\ell+1}) \begin{pmatrix} x_{t,j}^{\ell+1} \\ e_j \end{pmatrix} \right\|_\infty \leq \frac{2\bar{\epsilon}}{d}$$

Proof We argue the first inequality in a similar way to Equation 10. For the second inequality we can first argue that

$$\left\| o_{i,m+1}^{\ell+3} - \frac{1}{i}\sum_{t=1}^{i}\sum_{j=1}^{m} A^{\ell+1}(y_t^{\ell+1}, x_{t,j}^{\ell+1}) \begin{pmatrix} x_{t,j}^{\ell+1} \\ e_j \end{pmatrix} \right\|_\infty \leq \frac{\bar{\epsilon}}{d}$$

in a similar way to how we argued for Equation 10, by using Lemma 13 to set $A^{\ell+1}(y_t^{\ell_1}, x_{t',j}^{\ell+1}) \leq \frac{\bar{\epsilon}}{d((m+1)n)^2}, \forall t' < t, j \in [m+1]$. Further, using the setting of $C_{\bar{\epsilon}}$ from the proof of Lemma 21 we have that $A^{\ell+1}(y_t^{\ell+1}, x_{t,j}^{\ell+1}) \leq \frac{\bar{\epsilon}}{d(m+1)n}, \forall j \in \mathcal{C}_i^\ell$. This now implies

$$\left\| \frac{1}{i}\sum_{t=1}^{i}\sum_{j \in [m]\setminus\mathcal{C}_t^\ell} A^{\ell+1}(y_t^{\ell+1}, x_{t,j}^{\ell+1}) \begin{pmatrix} x_{t,j}^{\ell+1} \\ e_j \end{pmatrix} - \frac{1}{i}\sum_{t=1}^{i}\sum_{j=1}^{m} A^{\ell+1}(y_t^{\ell+1}, x_{t,j}^{\ell+1}) \begin{pmatrix} x_{t,j}^{\ell+1} \\ e_j \end{pmatrix} \right\|_\infty \leq \frac{\bar{\epsilon}}{d}.$$

34

Combining with the above inequality for $o_{i,m+1}^{\ell+3}$ finishes the proof. ∎

Using Lemma 15 we have that for every j selected by some consistent hypothesis $A^{\ell+1}(y_t^{\ell+1}, x_{t,j}^{\ell+1})$ will exceed $A^{\ell+1}(y_t^{\ell+1}, x_{t,j'}^{\ell+1})$, where j' is not selected by any consistent hypothesis. Hence, this two layer module identifies a consistent coordinate across the first i examples at example i by Lemma 20. Furthermore, this coordinate cannot be one of the previously identified coordinates in $\mathcal{C}_i^\ell$ by Lemma 22.

Let this new consistent coordinate be j^ℓ. We would like to add j^ℓ to $\mathcal{C}_i^\ell$. This can be done using two MLP layers and skip connections. We give a sketch of the construction below. Lemma 22 has the consistent coordinate as the largest magnitude coordinate among $[(d+m+1)mn+1, (d+m+1)mn+1]$ of $o_{i,m+1}^{\ell+3}$. Similarly to the 1-sparse construction the MLP can transform the last $[(d+m+1)mn+1, (d+m+1)mn+1]$ coordinates of $o_{i,m+1}^{\ell+3}$ so that the MLP returns the vector $-e_{j^\ell} \in \mathbb{R}^m$. A skip connection from $y_i^{\ell+1} = \mathrm{emb}(w_i)$ now allows us to form the vector $\mathrm{emb}\left(w_i + \begin{pmatrix} \mathbf{0} \\ -e_{j^\ell} \end{pmatrix}\right)$. We note that the last m coordinates of $w_i + \begin{pmatrix} \mathbf{0} \\ -e_{j^\ell} \end{pmatrix} \in \mathbb{R}^{d+m}$ are such that

$$\left(w_i + \begin{pmatrix} \mathbf{0} \\ -e_{j^\ell} \end{pmatrix}\right)(d+j^\ell) = -1$$

$$\left(w_i + \begin{pmatrix} \mathbf{0} \\ -e_{j^\ell} \end{pmatrix}\right)(d+j) = -C_{\bar{\epsilon}}, j \in \mathcal{C}_i^\ell$$

$$\left(w_i + \begin{pmatrix} \mathbf{0} \\ -e_{j^\ell} \end{pmatrix}\right)(d+j) = 0, j \in [m] \setminus \left(\mathcal{C}_i^\ell \bigcup\{j^\ell\}\right).$$

Add $1/2$ to the coordinates in $[d+1, d+m]$. Next, we use another relu activation on each coordinate. The resulting vector already satisfies that every coordinate $j \in \mathcal{C}_i^{\ell+4}$ is equal to 0, and every coordinate outside of the set is $\frac{1}{2}$. It remains to multiply the resulting vector by $-2C_{\bar{\epsilon}}$ and add $\bar{y}_i$ to the first d coordinates using a skip connection. All of the above can be done using one additional attention layer, together with an MLP. Since skip connections in the original transformer architecture are only in between consecutive attention layers, we can implement the above by extending the embedding of each $y_i^{\ell+1}, \ldots, y_i^{\ell+4}$ to have an additional $d+m$ coordinates in which to store $\bar{y}_i$.

Applying the learned hypothesis. The above construction implies that after $L = O(s)$ layers the resulting y_i^L will contain exactly a set $\mathcal{C}_i^\ell$ of cardinality s which contains only consistent coordinates. Further, using the deflation construction, we can show the following.

Lemma 23 *After $L = O(s)$ layers it holds that $|\mathcal{C}_i^L| = s$ and further, there exists a bijection b_i from $f^\star$ to $\mathcal{C}_i^L$ such that for any $j \in f^\star$, $|x_{t,j} - x_{t,b_i(j)}| \le \epsilon, t \le i$. The output $y_i^L \in \mathbb{R}^{(d+m+1)mn}$ is such that $y_i^L = \mathrm{emb}(w_i)$ with $w_i(d+j) = 0, \forall j \in \mathcal{C}_i^L$ and $w_i(d+j) = -C_{\bar{\epsilon}}, \forall j \in [m] \setminus \mathcal{C}_i^L$.*

Proof For the first part of the lemma we begin by showing that for any $j' \in \mathcal{C}_i^L$, there exists a $j \in f^\star$ such that $|x_{i,j} - x_{i,j'}| < \epsilon$. Suppose that this does not hold true, i.e., there is some j' such that for all $j \in f^\star$, for which $|x_{t,j} - x_{t,j'}| \ge \epsilon$ for some $t \le i$. Lemma 20 implies that $\langle \bar{x}_{t,j'}(1:d), \bar{y}_t(1:d) \rangle \le \frac{1}{4}$. On the other hand if $j' \in \mathcal{C}_i^L$ then the construction implies that at some layer $\ell' \le L$ it must have been the case that $\langle \bar{x}_{t,j'}(1:d), \bar{y}_t(1:d) - \sum_{j \in \mathcal{C}_t^{\ell'}} \bar{x}_{t,j}(1:d) \rangle \ge 3/4$ for

all t, otherwise j' can not be added to $\mathcal{C}_i^{\ell'}$ as it is not consistent with $f^\star$ on some round t and so it would not be part of $\mathcal{C}_i^L$ as $\mathcal{C}_i^{\ell'} \subseteq \mathcal{C}_i^L$. This is now a contradiction as it implies

$$\frac{3}{4} \le \langle \bar{x}_{t,j'}(1:d), \bar{y}_t(1:d) - \sum_{j \in \mathcal{C}_t^{\ell'}} \bar{x}_{t,j}(1:d) \rangle \le \langle \bar{x}_{t,j'}(1:d), \bar{y}_t(1:d) \rangle + \frac{s}{2\tau} \le \frac{1}{2},$$

where the second inequality follows from Assumption 2 as $\langle \bar{x}_{i,j'}(1:d), \bar{x}_{i,j}(1:d) \rangle > -\frac{1}{2s}$. This shows that we can never add a coordinate which is not similar to some coordinate in $f^\star$ across all the examples till i.

We show that the map is injective as follows. Let j_{ℓ_0} some coordinate for which we have already established the mapping $j_{\ell_0} \to j \in f^\star$ at layer ℓ_0. Consider another candidate j_{ℓ_1}, for $\ell_1 > \ell_0$ such that $|x_{t,j_{\ell_1}} - x_{t,j}| \le \epsilon$, that is j_{ℓ_1} can potentially be mapped to j as well on round t. We consider two cases, first for $j' \in f^\star$ s.t. $j' \ne j$ we have $|x_{t,j_{\ell_1}} - x_{t,j'}| > \epsilon$ or $j' \in \mathcal{C}_t^{\ell_1 - 1}$ already. In this case we show that $\bar{x}_{t,j_{\ell_1}}(1:d)$ is nearly orthogonal to $\bar{y}_t(1:d) - \sum_{j \in \mathcal{C}_t^{\ell_1 - 1}} \bar{x}_{t,j}(1:d)$ so that $x_{t,j_{\ell_1}}$ can not be added at any layer after $x_{t,j'}$ has been added:

$$\langle \bar{x}_{t,j_{\ell_1}}(1:d), \bar{y}_t(1:d) - \sum_{j \in \mathcal{C}_t^{\ell_1 - 1}} \bar{x}_{t,j} \rangle = \sum_{w \in \mathcal{C}_t^{\ell_1 - 1}} \langle \bar{x}_{t,j_{\ell_1}}, \bar{x}_{t,w} \rangle \le \frac{s}{\tau},$$

where the last inequality follows as before together with the assumption $|x_{t,j_{\ell_1}} - x_{t,j'}| > \epsilon$ outside of $\mathcal{C}_t^{\ell_1 - 1}$. Next, if there exists some $j' \in f^\star, j' \notin \mathcal{C}_t^{\ell_1 - 1}$ such that $|x_{t,j_{\ell_1}} - x_{t,j'}| < \epsilon$ we can map $j_{\ell_1} \to j'$ and add j' to $\mathcal{C}_t^{\ell_1}$ as long as the consistency property holds for all $t' \le t$. Otherwise, there exists a round t where $|x_{t',j_{\ell_1}} - x_{t',j'}| > \epsilon, \forall j' \in \mathcal{C}_t^{\ell_1 - 1}$ and the argument above can be repeated.

Further, we note that the construction can add at least every $j \in f^\star$ to $\mathcal{C}_i^L$ as the following is always satisfied:

$$\langle \bar{x}_{i,j}(1:d), \bar{y}_i(1:d) - \sum_{s \in S} \bar{x}_{i,s}(1:d) \rangle \ge \frac{3}{4}, \forall j \in f^\star, \forall S \subsetneq f^\star,$$

unless S contains some coordinate j' such that $|x_{i,j} - x_{i,j'}| \le \epsilon$ for all i. That is, every $j \in f^\star$ is mapped to at least one coordinate in $\mathcal{C}_i^L$. Taken together, each $j \in \mathcal{C}_i^L$ is mapped to *exactly one* element of $f^\star$ and each element of $f^\star$ is mapped to some element of $\mathcal{C}_i^L$. This establishes the claim for the bijection. The second claim of the lemma follows just from the construction of the transformer. ∎

To use the returned y_i^ℓ guaranteed by Lemma 23 for inference we first modify it in the following way. We add the vector consisting of all 1s and then apply a relu on each coordinate. The resulting vector now contains a consistent hypothesis in $w_i(d+1:d+m)$. To apply the hypothesis we can follow a similar construction to that of the final three layers from the 1-sparse token task.

E.1. Proof of Theorem 11

We treat $f^\star$ and f_n as two subsets of $[m]$ with cardinality s. Lemma 23 implies that for every example $i \in [n]$, there is a bijection b_n between f_n and $f^\star$ which maps any $j \in f^\star$ to a $j' \in f_n$ such that $|x_{i,j} - x_{i,j'}| \le \epsilon, i \in [n]$. The same argument as in Lemma 18 shows the following.

Lemma 24 *For any $x \in \mathbb{R}^m, j \in f^\star$ let $p_{n,j} = \mathbb{P}(|x_j - x_{b_n(j')}| \leq \epsilon)$. Then with probability $1 - \delta$ it holds that $p_n \geq 1 - \frac{20s \log(m/\delta)}{3n}$.*

Using the above lemma we can show the equivalent to the sample complexity bound for the index token task.

Proof [Proof of Theorem 11] The same argument as in Theorem 19 can be used to show that for the bijection guaranteed by Lemma 23 and the setting of n we have $\mathbb{E}[|x_j - x_{b_n(x_j)}|] \leq 2\epsilon, \forall j \in f^\star$. This implies the result of the theorem as

$$\mathbb{E}[|f_n(x) - f^\star(x)|] = \mathbb{E}[|\sum_{j \in f^\star} x_{b_n(j)} - x_j|] \leq \sum_{j \in f^\star} \mathbb{E}[|x_{b_n(j)} - x_j|] \leq 2s\epsilon.$$

Redefining $\epsilon \to \epsilon/s$ completes the proof. ∎

Appendix F. Vector 1-sparse regression task

We now quickly discuss how to solve the vector version of the 1-sparse regression task, where the transformer's input is a sequence of examples $(x_i, y_i)_{i \in [n]}$, however, now $x_i \in \mathbb{R}^m$ is a single token, rather than being split into m tokens. The idea is to learn each bit of a consistent hypothesis sequentially using a total of $O(\log(m))$ attention layers. To do so we focus on recovering learning a consistent hypothesis for example i as done in the first attention layer in the 1-sparse token task. The remainder of the construction follows the ideas from the 1-sparse token task.

First attention layer. Unlike in the 1-sparse tokenized regression task, we can not represent a single hypothesis by the respective token (even though it does still correspond to a coordinate in x). Instead we assume that the value vector $v_{i,1}^1$, for x_i, in the first layer, contains 0 in its first m coordinates and the following vector $\beta_{i,1}^1 \in \mathbb{R}^m$ in the next m coordinates

$$\beta_{i,1}^1(j) = \mathbf{1}(\text{bit 1 of } j \text{ equals 1}).$$

The value vector $v_{i,2}^1$ for y_i is constructed similarly, with the first m coordinates equal to 0 again and the second m coordinates equaling $\beta_{i,2}^1 \in \mathbb{R}^m$ which is the complement of $\beta_{i,1}^1$ in $\{0,1\}^m$. The embeddings in the first layer are as follows. $x_i^1 \in \mathbb{R}^{(d+1)m}$ contains the embedding of $\bar{x}_{i,j}$ from Assumption 2 in coordinates $x_{i,1}^1(d(j-1) + 1 : dj)$. The remaining m coordinated are all set to 1. $y_i^1 \in \mathbb{R}^{(d+1)m}$ is constructed similarly, where the first dm coordinates contain the embedding $\bar{y}_i$ from Assumption 2, repeated d times. The last m coordinates equal the last m coordinates of $v_{i,2}^1$, that is $y_i^1(dm + 1 : (d+1)m) = v_{i,2}^1(m + 1 : 2m)$. The query and key matrices Q^1, K^1 now implement the following linear operation:

$$\langle y_i^1, x_i^1 \rangle_{Q^1(K^1)^\top} = \gamma \sum_{j=1}^m \beta_{i,1}^1(j) \langle y_i^1(d(j-1) + 1 : dj), x_i^1(d(j-1) + 1 : dj) \rangle,$$

$$\langle y_i^1, y_i^1 \rangle_{Q^1(K^1)^\top} = \frac{\gamma}{2} \sum_{j=1}^m \beta_{i,1}^1(j) \langle y_i^1(d(j-1) + 1 : dj), y_i^1(d(j-1) + 1 : dj) \rangle.$$

This is implemented in the following way, the query matrix Q is a diagonal matrix with $Q(d(j - 1) + 1 : dj) = \beta_{i,1}^1(j)I_{d \times d}$. In the above $\gamma = \Theta(\log(m/\epsilon))$ is a threshold parameter which will turn the softmax into an approximate max. We do not specify the inner product $\langle x_i^1, \cdot \rangle_{Q^1(K^1)^\top}$ as the second layer embedding x_i^2 will be independent of the first layer.

Lemma 25 *The inner product $\langle y_i^1, x_i^1 \rangle_{Q^1(K^1)^\top} \geq \gamma(1 - \frac{m}{\tau})$ iff there exists at least one consistent with $f^\star$ hypothesis with first bit equal to 1. Further, if there is no such hypothesis then $\langle y_i^1, x_i^1 \rangle_{Q^1(K^1)^\top} \leq \gamma \frac{m}{\tau}$.*

Proof Using the definition of the embeddings we have

$$\langle y_i^1, x_i^1 \rangle_{Q^1(K^1)^\top} = \gamma \sum_{j=1}^m \beta_{i,1}^1(j)\langle \tilde{y}_i, \tilde{x}_{i,j} \rangle,$$

where $\tilde{x}_{i,j}, \tilde{y}_i \in \mathbb{R}^d$ are the initial embeddings from the 1-sparse task. From Assumption 2 we have that $\langle \tilde{y}_i^1, \tilde{x}_{i,j}^1 \rangle \geq 1$ if $j = f^\star$, $\left\langle y_i^1, x_{i,j}^1 \right\rangle \geq 0$ if j is some other coordinate consistent with $f^\star$ and $\beta_{i,j}^1(j) = 1$ iff the first bit of j equals 1. Hence we get an inner product of at least γ from $f^\star$, at least 0 from any other consistent coordinate, and at least $-1/\tau$ from any inconsistent coordinates. This implies the first claim of the lemma. For the second part we note that if there is no consistent hypothesis with first bit equal to 1 then $\langle \tilde{y}_i, \tilde{x}_{i,j} \rangle \leq \frac{1}{\tau}$ according to Assumption 2. $\blacksquare$

To keep the argument clean, we assume that the softmax acts as an argmax. As we have pointed out, this can be achieved up to ϵ when setting $\gamma = \Theta(\log(m/\epsilon))$. The output for the i-th answer token, $o_{i,2}^1$, now contains in its last m coordinates an indicator of which hypotheses are consistent, when restricted to the value of the first bit. In particular, if there exists a consistent hypothesis then $o_{i,2}^1(m + 1 : 2m) = \beta_{i,1}^1$ and otherwise $o_{i,2}^1(m + 1 : 2m) = \beta_{i,2}^1$.

Lemma 26 *Let $z \in \mathbb{R}^m$ be some vector such that $\|z\|_\infty \leq c < \infty$ and $\beta \in \{0, 1\}^m$. Let $z \odot \beta$ denote the element-wise product of the two vectors. Then the operation $z \odot \beta$ can be implemented by a Relu MLP layer.*

Proof Let $\mathbf{e} \in \mathbb{R}^m$ be the all ones vector. The MLP applies the following operation $\text{Relu}(z + c(\mathbf{e} - \beta) - c\beta) - c(\mathbf{e} - \beta)$. $\blacksquare$

Using Lemma 26 the MLP acts on $o_{i,2}^1$ by setting $y_i^2(d(j - 1) + 1 : dj) := o_{i,2}^1(j)o_{i,2}^1(d(j - 1) + 1 : dj)$, so that the first dm entries of y_i^2 only contain coordinates which are consistent with the recovered bit in the first layer.

Second attention layer. In this layer we demonstrate how to learn the second bit of a consistent hypothesis for example i, conditioned on the first bit contained in $y_i^2(m + 1 : 2m)$. The value vectors are defined similarly to the first layer, using

$$\beta_{i,1}^2(j) = \mathbf{1}(\text{bit 2 of } j \text{ equals 1}),$$

and its complement $\beta_{i,2}^2 \in \mathbb{R}^m$. For the embeddings, $x_i^2 = x_i^1$, and y_i^2 is as described above. Finally we set $K^2 = K^1$ and Q^2 is defined to act similarly to Q^1, however, with respect to $\beta_{i,1}^2$, that is:

$$\langle y_i^2, x_i^2 \rangle_{Q^2(K^2)^\top} = \gamma \sum_{j=1}^m \beta_{i,1}^2(j)\langle y_i^2(d(j - 1) + 1 : dj), x_i^2(d(j - 1) + 1 : dj) \rangle.$$

A result similar to Lemma 25 can now be shown, where the attention weight $A^2(y_i, x_i) \approx 1$ if there exists a consistent hypothesis with first bit set according to $o_{i,2}^1$ and second bit equal to 1, otherwise $A^2(y_i, y_i) \approx 1$ and there exists a consistent hypothesis with first bit set according to $o_{i,2}^1$ and second bit equal to 0. Finally, we describe how the MLP is applied. First, we add $o_{i,2}^1 + y_i^2(dm + 1 : (d + 1)m)$ using the skip connection. This results in the following (assuming a max, instead of a soft-max).

Lemma 27 *The j-th coordinate of $o_{i,2}^1 + y_i^2(dm + 1 : (d + 1)m)$ satisfies $o_{i,2}^1 + y_i^2(dm + 1 : (d + 1)m) \geq 2$ iff the j-th hypothesis is consistent with $f^\star$ on the i-th example.*

Proof WLOG assume that $A^1(y_i, x_i) \approx 1$ and $A^2(y_i, y_i) \approx 1$, so that the inner product in the second layer has shown that there exists a consistent hypothesis with first two bits equal to 10. From the construction it holds that $o_{i,2}^1$ indexes all hypotheses with second bit set to 0. Further, $o_{i,2}^1$ indexes all hypothesis with first bit set to 1, and so under the assumption $o_{i,2}^1 + y_i^2(dm + 1 : (d + 1)m) = o_{i,2}^1 + o_{i,2}^2$ will have j-th coordinate greater than 2 if the j-th hypothesis is consistent on the i-th example and has first two bits equal to 10. ∎

We apply the following operation to $o_{i,2}^2 + y_i^2(dm + 1 : (d + 1)m)$. First we subtract some threshold $2 > c > 1$. Then we clip each negative coordinate to 0 and each positive coordinate to 1. Lemma 27 implies that the resulting vector indexes exactly all consistent hypotheses with first and second bit set according to $o_{i,2}^1$ and $o_{i,2}^2$ respectively. Let this vector be $y_i^3(dm + 1 : (d + 1)m) \in \mathbb{R}^m$. We now want to apply $y_i^3(dm + 1 : (d + 1)m) \in \mathbb{R}^m$ to $y_i^1(1 : dm)$, similarly to how the first layer MLP applied the consistent hypothesis to $y_i^1(1 : dm)$ as well. To do so we require an extra MLP layer. Note that this can be achieved by adding another attention layer to carry out this operation.

Further layers. Replicating the construction for the second layer, but focusing on the b-th bit of the hypothesis we can show the following

Lemma 28 *After $M = O(\log(m))$ layers it holds that $y_i^M(dm + 1 : (d + 1)m)(j) = \mathbf{1}(j$ is consistent with $f^\star)$.*

One can now use the same type of construction as in the index token task to learn a hypothesis which is consistent on all i examples seen so far and further use this hypothesis to do inference. The sample complexity bound for this approach are similar to the one in Theorem 19. We also note that this construction can be extended to handle the s-sparse index vector task as well, but we will not go into details as the constructions required should not demonstrate any new ideas.

Appendix G. Experiments

G.1. 1-sparse tokenized regression experiments

We experiment with two settings for 1-sparse tokenized regression. In both settings the dimensionality of the problem is $m = 5$, that is each example x_i consists of 5 tokens $(x_{i,1}, \ldots, x_{i,5})$ together with the answer token y_i. The transformer architecture is the same for both tasks. We use 8 attention layers, with masking future tokens, that is the only non-zero attention weights are $A^i(x_{i,j}, x_{i',j'})$ for $i \geq i', j \geq j'$. Each attention layer is follows by a layer-norm normalization and an MLP layer with GeLU activation. Further, skip connections are used between the input to the attention layers and the output of the layer-norm and MLP layers. The hidden size for the embeddings is $d = 128$, and we use a single attention head per layer. Positional embeddings are learned. Predictions are done by

a final MLP layer mapping the d-dimensional embeddings to a scalar. The training for both settings uses the same hyper-parameters and optimizer. We use Adam as optimizer with the schedule used in (Akyürek et al., 2022) and initial step-size set to $1e - 4$. Initializing the network parameters also follows Akyürek et al. (2022).

Training in both settings proceeds by generating example sequences $(x_i, y_i)_{i \in [n]}$, by first selecting a fixed hypothesis $f^\star$, sampled uniformly at random from $[m]$ and then sampling x_i's i.i.d. from fixed distributions which we describe momentarily. The sequence for a single pre-training iteration is then $(x_i, f^\star(x_i))_{i \in [n]}$. Pre-training proceeds in mini-batches of size 64, that is each mini-batch has 64 sequences $(x_i, f^\star(x_i))_{i \in [n]}$ sampled independently as we described above. We use mean-squared loss over the sequence for pre-training in the following way:

$$\mathcal{L}((x_i, f^\star(x_i))_{i \in [n]}; \theta) = \frac{1}{n} \sum_{i=1}^{n} (x_{i,5}^8 - f^\star(x_i))^2,$$

where θ denotes the parameters of the transformer and $x_{i,j}^8$ is the output of the final MLP layer of the transformer (in accordance with our index token task notation), that is we use the transformers output after seeing the 5-th token in each example as the prediction, and the loss is taken as the squared difference between this prediction and the 6-th token in each example. Finally, we note that while $f^\star$ is fixed for a sequence $(x_i, y_i)_{i \in [n]}$, we sample a fresh $f^\star$ for each new sequence in the mini-batch. This setup is identical to both Garg et al. (2022) and Akyürek et al. (2022). We train for 8000 epochs, where each epoch consists of 100 iterations, each one on a mini-batch of size 64.

The two settings we consider are in terms of the distribution over $(x_i, y_i)_{i \in [n]}$. In the first setting $x_i \sim \mathcal{N}(0, I_{5 \times 5})$ and in the second setting $x_i \sim Unif(\{+1, -1\}^5)$. These settings are complementary to each other in the following way. In the Gaussian setting it is possible to learn the hypothesis $f^\star$ after a single ICL example almost surely. In the uniform over $\{-1, 1\}^m$ setting, which refer to as the Rademacher setting, one needs to see $\Omega(\log(m))$ examples before $f^\star$ can be identified with high probability.

We plot the squared error at every example for a given sequence, attention at the last layer and attention at layer 6. Plots are averaged over the mini-batch of size 64. For averaging we fix the same $f^\star$ over the full mini-batch. Results can be found in Figure 5, Figure 6, Figure 3. The figures depicting attention in these plots are slightly different from the heatmaps in Section 6, however, they represent the same results. On the x-axis we enumerate all tokens in the sequence and on the y-axis we show the attention weight. We plot $A(x_{i,6*(i+1)-2}, x_{i',j'})$ for $i \in [m], i' \le i, j \in [n]$, that is the attention of each inference token for example i to all other tokens preceding it. For the attention plots at layer 8, the final layer, we also plot what the target indices are to have a reference on how well the transformer tracks $f^\star$.

The transformer pre-trained on the Rademacher task only, exhibits very similar properties to the mixed model discussed in Section 6. Perhaps, surprisingly, the model is able to achieve the same performance on the Gaussian inference task as the mixed model, even though it has never seen Gaussian examples.

The transformer pre-trained on the Gaussian only task, still retains the ability to learn from a single example as demonstrated by Figure 5. However, the attention at layer 6 are less interpretable compared to the mixed model and the Rademacher only model. The attention weights in the last layer retain the nice properties from the other two models. The Gaussian model, however, performs poorly on the Rademacher task as seen in Figure 6. We note that the attention weights at the last
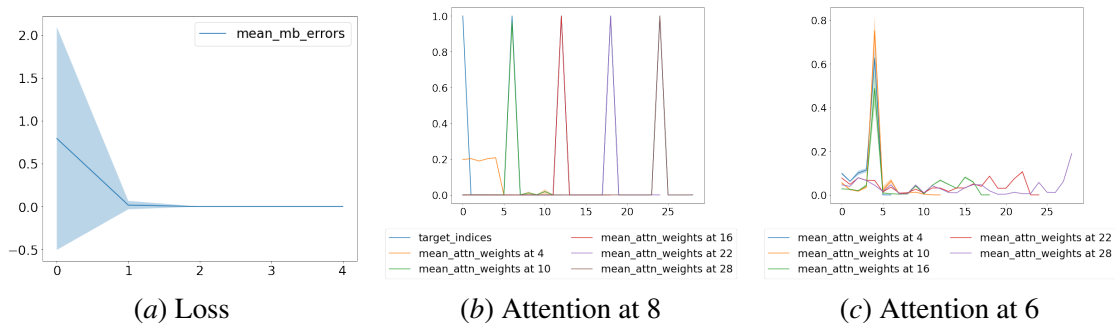
(*a*) Loss   (*b*) Attention at 8   (*c*) Attention at 6

Figure 5: Train Gaussian, inference Gaussian.

(*a*) Loss   (*b*) Attention at 8   (*c*) Attention at 6
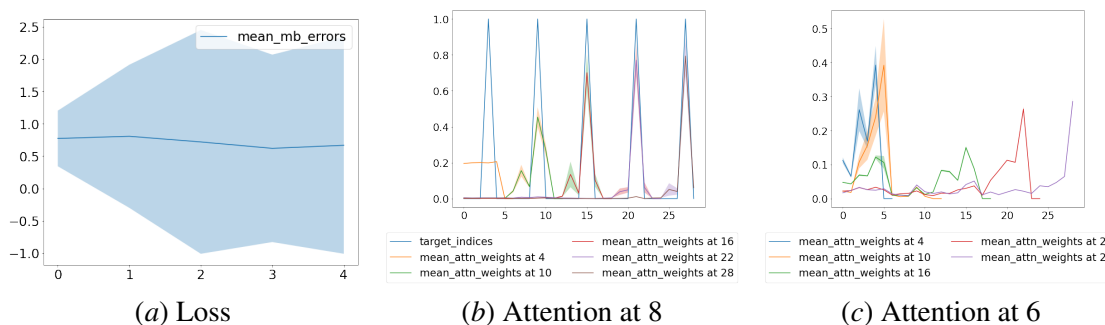
Figure 6: Train Gaussian, inference Rademacher.

(*a*) Loss   (*b*) Attention at 8   (*c*) Attention at 6
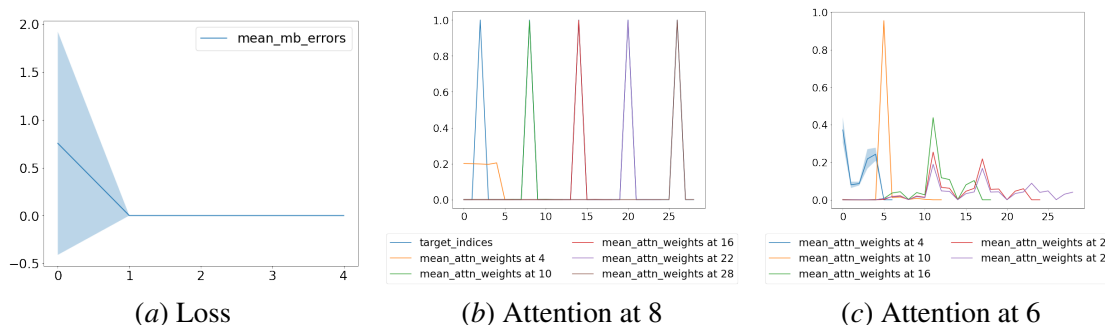
Figure 7: Train Rademacher, inference Gaussian.

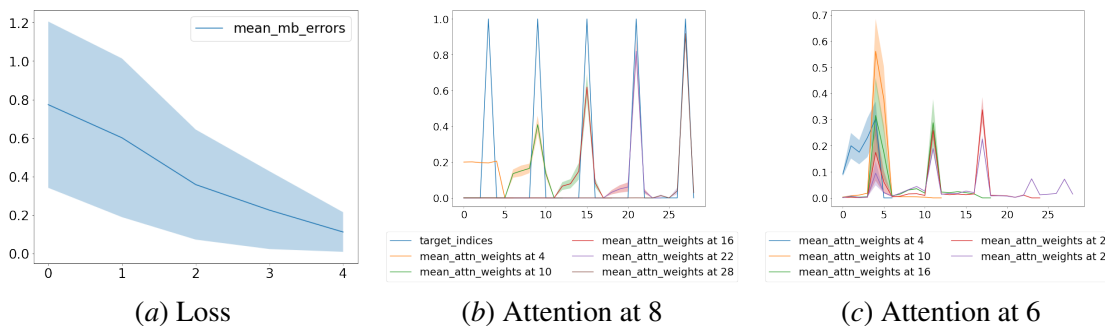(*a*) Loss   (*b*) Attention at 8   (*c*) Attention at 6

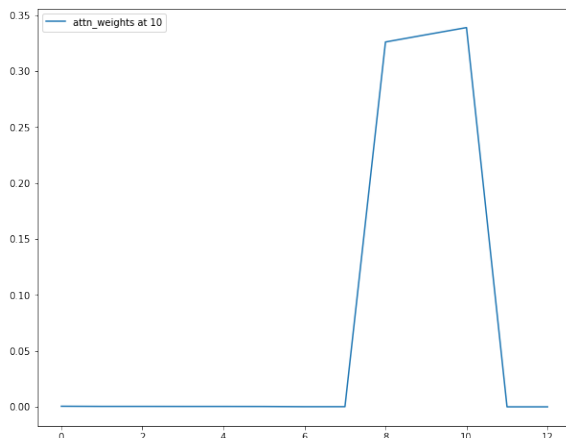Figure 8: Train Rademacher, inference Rademacher.

Figure 9: Rademacher task attention spread

layer still behave similarly to the attention weights of the mixed model and the Rademacher model, suggesting that the Gaussian model can still distinguish $f^\star$. We conjecture that the reason for the poor performance is due to how the learned hypothesis is applied to examples for inference. In particular, we expect that the Gaussian model, during pre-training, has learned to apply the inferred hypothesis after the first example, however, this would be detrimental for the Rademacher setting, as it is very unlikely that $f^\star$ is identifiable after only a single example.

Finally in Figure 9 we show the behavior of the mixed model on a single Rademacher sequence. The first 6 elements of the sequence from the figure are $x_{1,1} = 1, x_{1,2} = 2, x_{1,3} = -1, x_{1,4} = -1, x_{1,5} = -1, y_1 = -1$. At inference time for the second example, we show the attention weights for example $x_{1,4}$, which is token z_{10} in the sequence, spreads its attention uniformly on all consistent hypothesis $j \in \{3, 4, 5\}$ corresponding to tokens z_8, z_9, z_{10}. This is again consistent with our construction for inference. In our experiments we have observed that the attention is put on $f^\star$ at the earliest example i where the identification is possible, and this is why the averaged attention plots at layer 8 are peaked, with some variance, at $f^\star$.

G.2. 3-sparse token task

The setting for the 3-sparse task is the same as for the 1-sparse task in terms of all parameters, outside of sparsity. Further, training is done in the same way as the 1-sparse mixed task, with equal mix of Gaussian and Rademacher examples. Loss and attentions are plotted in Figure 10. We only show the attention and heatmap plots for the final layer. Unfortunately, there is no layer with similar behavior to Layer 6 in the 1-sparse plots, which was consistent with our construction. All observations for the final layer attention carry over from the 1-sparse task.

G.3. Segmentation

We make the following empirical observation: the performance of ICL is sensitive to the choice of delimiter. In Figure 11 we show the quality of ICL using OpenAI's GPT-3 model (known as *text-davinci-003*) on a family of relational tasks and using a range of different delimiters. The tasks in question are relational tasks, usually covering a type of "trivia" question. But we vary the two delimiters and consider performance of the completion. Below are three example queries we

(*a*) Loss (Gaussian), $y_i = x_{i,0}$

(*b*) Attention at layer 8 (Gaussian)

(*c*) Heatmap at 8 (Gaussian)

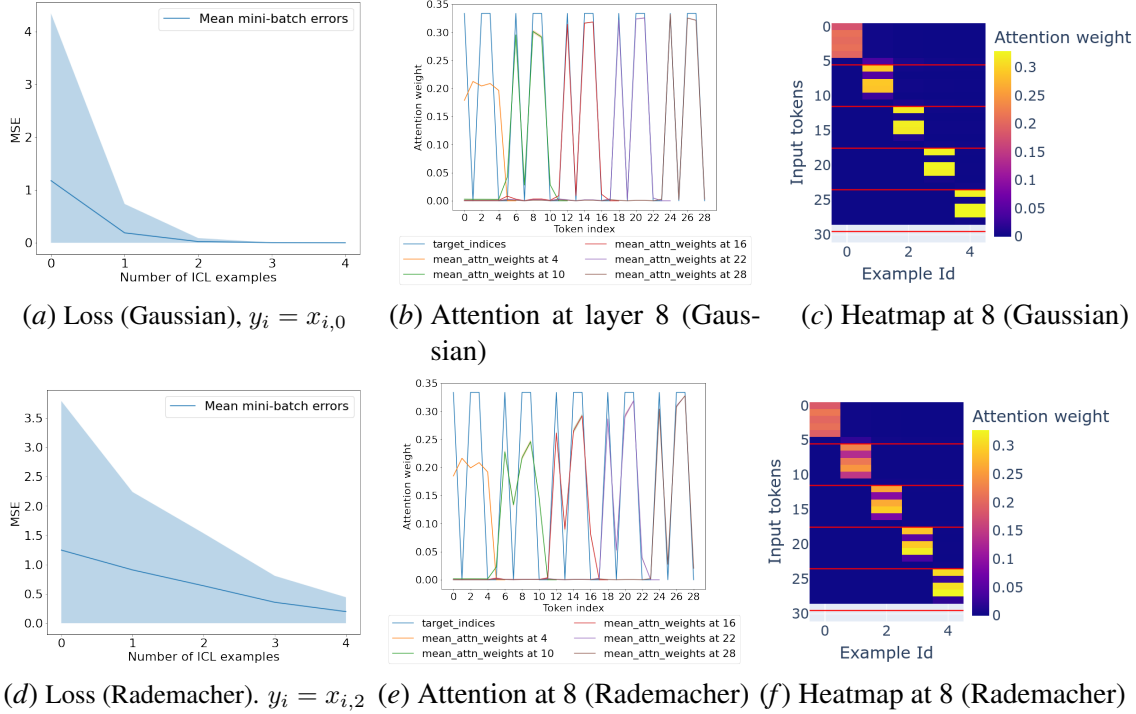(*d*) Loss (Rademacher). $y_i = x_{i,2}$ (*e*) Attention at 8 (Rademacher) (*f*) Heatmap at 8 (Rademacher)

Figure 10: Loss and attention plots for 3-sparse tokenized regression for Gaussian (top) and Rademacher (bottom) inputs. Loss drops to zero as soon as $f^\star$ is determined, and attentions follow the construction of Section 5. Indices $4, 10, 16, \ldots$ are tokens where the label is predicted. In panels (b) and (e), these indices attend to the index of $f^\star$ in x_i to predict y_i correctly. The target indices line (blue) in panel (b) perfectly overlaps with the attention spikes at tokens $x_{i,0}$. In panel (d), the attention spikes largely overlap with target indices, but there is some noise (see text). In panels (c) and (f), we plot a heatmap with the same indexing as in Figure 3. These indices attend to all previous labels (indices $5, 11, 17, \ldots$) to aggregate a consistent hypothesis across previous examples.
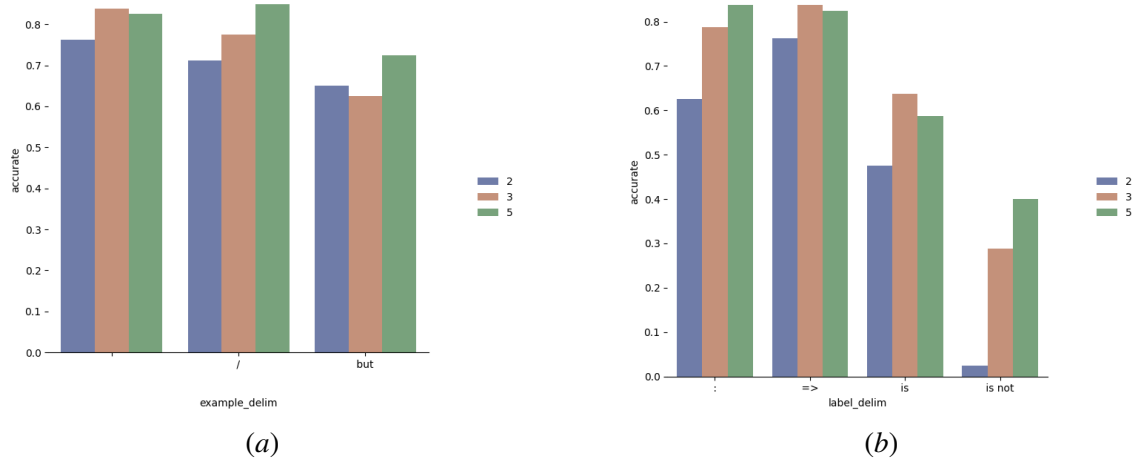
(a) (b)

Figure 11: The accuracy of ICL over a range of tasks when we vary the choice of delimiter. On the left figure, we vary the delimiter used to separate examples among $\{\backslash n, /, \texttt{but}\}$, and on the right we vary the delimiter used to separate x from y among $\{:, =>, \texttt{is}, \texttt{is not}\}$. The performance is computed across four association tasks, we run each task 10 times (across different "training" example sets), and for all the example delimiter tasks we use the label delimiter $:$, and for all the label delimiter tasks we use the example delimiter $\backslash n$.

provide to the model, and we consider the answer *correct* if the correct answer occurs within the first 3 tokens of the response.

```
// scientist year of death
Albert Einstein => 1955 \n Isaac Newton => 1727 \n Johannes Kepler => _____
// famous actor year of birth
Leonardo DiCaprio is 1974 but Meryl Streep is 1949 but Dustin Hoffman is _____
// baseball team last won world series
Houston Astros is not 2017 / St. Louis Cardinals is not 2011 /
    Boston Red Sox is not _____
```